

Décompositions tensorielles et factorisations de calculs intensifs appliquées à l'identification de modèles de comportement non linéaire

Clément Olivier

▶ **To cite this version:**

Clément Olivier. Décompositions tensorielles et factorisations de calculs intensifs appliquées à l'identification de modèles de comportement non linéaire. Matériaux. Université Paris sciences et lettres, 2017. Français. NNT : 2017PSLEM040 . tel-01783950

HAL Id: tel-01783950
https://pastel.hal.science/tel-01783950

Submitted on 2 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à MINES ParisTech

Décompositions tensorielles et factorisations de calculs intensifs appliquées à l'identification de modèles de comportement non linéaire

École doctorale n°432

SCIENCES DES MÉTIERS DE L'INGÉNIEUR

Spécialité MÉCANIQUE

Soutenue par **Clément OLIVIER**
le 14 décembre 2017

Dirigée par **David RYCKELYNCK**

COMPOSITION DU JURY :

M. Anthony NOUY, Président du jury
École Centrale de Nantes

M. Laurent GALLIMARD, Rapporteur
Université de Paris Nanterre

M. David NÉRON, Rapporteur
ENS Cachan

M^{me} Mathilde CHEVREUIL, Examinateur
Université de Nantes

M. Julien JARAVEL, Examinateur
Safran Helicopter Engines

M. Balázs KÉGL, Examinateur
CNRS LAL IN2P3

M. Christian REY, Examinateur
SafranTech

M. David RYCKELYNCK, Examinateur
Mines ParisTech

M. Julien CORTIAL, Invité
SafranTech

MINES ParisTech | PSL RESEARCH UNIVERSITY PARIS

« Dans notre connaissance des choses de l'Univers (qu'elles soient mathématiques ou autres), le pouvoir rénovateur en nous n'est autre que l'innocence. »

Alexandre GROTHENDIECK, *Récoltes et semailles*

Remerciement

Ces trois années de thèse ont été pour moi une formidable aventure autant intellectuelle qu'humaine. Tout d'abord, je souhaite remercier Antony Nouy de m'avoir fait l'honneur de présider mon jury de thèse. Je tiens également à remercier mes rapporteurs David Néron et Laurent Gallimard pour leur lecture particulièrement attentive ainsi que la pertinence de leurs remarques. À leur tour, je remercie mes examinateurs Julien Jaravel, Mathilde Chevreuil et Balázs Kélg qui, de par leur domaine d'expertise varié, m'ont permis d'aborder ma thèse sous des regards différents lors de la soutenance.

Je souhaiterais exprimer ma gratitude envers Frédéric Feyel qui m'a fait confiance en me donnant la possibilité de travailler sur ce sujet et qui a mis tout en œuvre pour qu'elle se déroule dans les meilleures conditions. Un grand merci à Christian Rey pour la liberté qu'il m'a donnée au cours de ces trois ans, mais également pour sa disponibilité et son énergie qu'il a su me transmettre. Je souhaite remercier sincèrement Julien Cortial pour son encadrement au jour le jour et pour l'esprit de rigueur et de pragmatisme qu'il a su m'enseigner. Je voudrais à présent remercier chaleureusement mon directeur de thèse David Ryckelynck pour m'avoir partagé sa passion pour la recherche et l'innovation. Son inspiration débordante, ses conseils précieux et sa bienveillance m'ont véritablement guidé au cours de ces trois années.

Je tiens à remercier l'ensemble de mes collègues du Centre des Matériaux et de Safran Tech. Leur accueil chaleureux et leur disponibilité m'ont permis de réaliser ma thèse dans un cadre autant agréable que productif. Parmi eux, je souhaite remercier tout particulièrement Fabien Casenave pour son ouverture d'esprit et les discussions passionnantes que nous avons échangées et enfin Felipe Bordeu pour m'avoir appris d'innombrables choses dans tant de domaines. Je lui exprime ici toute ma reconnaissance pour sa disponibilité, sa gaieté et son amitié.

Pour finir, je souhaite remercier du fond du cœur l'ensemble de mes proches. Merci à mes parents pour m'avoir toujours fait confiance et encouragé dans mes choix ainsi que pour toutes les valeurs qu'ils m'ont transmises et qui me constituent aujourd'hui. Merci à mes amis pour tous les moments de bonheur et de rire partagés qui ont rendu ces trois ans inoubliables. Je terminerai par remercier infiniment Marine pour son soutien inconditionnel et sa générosité.

Apprendre et découvrir sont probablement les choses qui me sont les plus chères, c'est donc un immense merci que j'adresse à toutes ces personnes.

Table des matières

Remerciement **v**

Table des illustrations **xiii**

I Introduction **1**
- I.1 Contexte général . . . 1
- I.2 Objectifs . . . 3
- I.3 Contexte scientifique . . . 3
- I.4 Contributions . . . 6
- I.5 Organisation du mémoire . . . 7
- I.6 Communications scientifiques . . . 7

II Contexte scientifique et technique **9**
- II.1 Calibration des lois matériaux . . . 10
  - II.1.a Modélisation des lois matériaux . . . 10
  - II.1.b Expérimentation . . . 11
    - II.1.b.1 Définition des essais . . . 11
    - II.1.b.2 Réalisation des essais . . . 12
  - II.1.c Identification . . . 14
    - II.1.c.1 Difficulté à définir une mesure d'écart . . . 15
    - II.1.c.2 Espace paramétrique . . . 16
    - II.1.c.3 Travail collaboratif . . . 17
    - II.1.c.4 Illustration sur la loi Polystar . . . 17
  - II.1.d Bilan . . . 18
- II.2 Modèles de substitution . . . 19

III État de l'art **23**
- III.1 Méthodes de réduction de modèle par projection . . . 25
  - III.1.a Décomposition orthogonale aux valeurs propres . . . 26
  - III.1.b Méthode des bases réduites . . . 28
- III.2 Méthodes de réduction de la complexité . . . 30
  - III.2.a Méthodes d'interpolation empirique . . . 30

III.2.a.1 Squelette fonctionnel . . . 33
III.2.a.2 Méthode d'interpolation empirique généralisée . . . 34
III.2.a.3 Méthode d'interpolation empirique discrète . . . 35
III.2.a.4 Best Points Interpolation Method . . . 37
III.2.b Méthodes de projection lacunaire . . . 39
III.2.b.1 Gappy POD . . . 39
III.2.b.2 Hyper-réduction . . . 40
III.2.b.3 Missing Point Estimation . . . 41
III.3 Décompositions matricielles approchées . . . 42
III.3.a Notations matricielles . . . 43
III.3.b Décomposition en valeurs singulières . . . 44
III.3.c Sélection de colonnes/lignes . . . 46
III.3.c.1 Maxvol . . . 47
III.3.c.2 Décomposition QR avec pivotage de colonnes . . . 48
III.3.c.3 Q-DEIM . . . 49
III.3.d Décomposition skeleton et pseudo-skeleton . . . 49
III.3.d.1 Méthode de construction . . . 50
III.3.e Approximation gappy . . . 53
III.4 Décompositions tensorielles . . . 53
III.4.a Notations tensorielles . . . 54
III.4.b Décomposition canonique . . . 59
III.4.c Décomposition de Tucker . . . 60
III.4.d Décomposition de Tucker hiérarchique . . . 60
III.4.e Décomposition en train de tenseurs . . . 61
III.4.e.1 Définition . . . 61
III.4.e.2 Propriétés . . . 63
III.5 Méthodes d'approximations tensorielles de faible rang . . . 64
III.5.a Alternating least square . . . 65
III.5.b High Order Singular Value Decomposition . . . 67
III.5.c TT-SVD . . . 67
III.5.d TT-cross . . . 69
III.5.e DMRG . . . 72
III.5.f Autres algorithmes associés au format TT . . . 75
III.5.g Proper Generalized Decomposition . . . 75

IV Cadre générique d'approximation par décomposition en train de tenseurs **79**
IV.1 Approximation de fonctions multivariées . . . 80

IV.1.a Représentation tensorielle 80
IV.1.b Représentation discrète 80
IV.1.c Représentation en train de tenseurs 81
IV.1.d Interpolation multilinéaire par morceaux 82
IV.2 Algorithme générique de décomposition TT par approximations matricielles successives 84
IV.2.a Description de l'algorithme 84
IV.2.b Erreur d'approximation 85
IV.2.c Algorithme de refactorisation 89
IV.3 Particularisation de l'algorithme de décomposition TT par approximations matricielles successives 91

V Spécialisation de l'algorithme générique **97**
V.1 Approximation matricielle 98
V.1.a Formulation particulière 98
V.2 Spécialisations préliminaires 99
V.2.a TT-SVD 99
V.2.b TT-POD 100
V.2.c TT-PSD 101
V.2.c.1 Décomposition matricielle 101
V.2.c.2 Décomposition tensorielle 102
V.3 TT-gappy 102
V.3.a Décomposition matricielle 103
V.3.b Décomposition tensorielle 105
V.3.c Détails algorithmiques 107
V.4 Application pratique 109
V.5 Interprétation en termes de projecteur 111
V.5.a Matrice de projection 111
V.5.b Formulation en termes de projecteurs 112
V.5.c Principe général d'approximation 113
V.6 Intérêt de la procédure de refactorisation 114

VI Décomposition en trains de tenseurs hétérogènes **115**
VI.1 Cadre général des modèles à approcher 117
VI.1.a Modèle de référence 117
VI.1.b Représentations tensorielles d'un modèle de référence 119
VI.2 Algorithme de décomposition en trains de tenseurs hétérogènes 121
VI.3 Détails algorithmiques 122
VI.3.a Sélection de colonnes 122

VI.3.b Sélection de lignes . . . 126
VI.3.c Calcul de la SVD . . . 127
VI.4 Détails relatifs aux applications . . . 128
VI.4.a Définition des sorties du modèle . . . 129
VI.4.a.1 Domaine de définition des sorties dépendant du paramétrage . . . 129
VI.4.a.2 Influence de l'agrégation des quantités d'intérêt . . . 130
VI.4.b Définition des entrées du modèle . . . 131
VI.4.c Accumulation des erreurs . . . 133
VI.5 Applications théoriques . . . 133
VI.5.a Agrégation . . . 134
VI.5.b Raffinement du rang de troncature . . . 136
VI.6 Améliorations possibles . . . 137
VI.6.a SVD incrémentale . . . 137
VI.6.b Autres perspectives d'amélioration . . . 138

VII Application de la méthodologie **141**
VII.1 Généralités . . . 142
VII.2 Loi élasto-viscoplastique non linéaire . . . 142
VII.2.a Modèle physique . . . 143
VII.2.a.1 Système d'équations . . . 143
VII.2.a.2 Sollicitations et conditions initiales . . . 144
VII.2.a.3 Abstraction tensorielle . . . 144
VII.2.b Résultats . . . 146
VII.2.b.1 Indicateurs de performance . . . 146
VII.2.b.2 Erreurs d'approximation . . . 147
VII.2.b.3 Convergence de l'erreur vis-à-vis de la tolérance de troncature . . . 147
VII.2.b.4 Estimateur de cohérence en ligne . . . 150
VII.2.b.5 Exemples de courbes . . . 152
VII.3 Premier cas industriel : Polystar . . . 153
VII.3.a Modèle physique . . . 153
VII.3.a.1 Équations physiques . . . 153
VII.3.a.2 Sollicitations et conditions initiales . . . 155
VII.3.a.3 Abstraction tensorielle . . . 155
VII.3.b Résultats . . . 156
VII.3.b.1 Indicateurs de performance . . . 156
VII.3.b.2 Erreurs d'approximation . . . 158
VII.3.c Identification . . . 159
VII.4 Second cas industriel : RaftX . . . 160

VII.4.a Définition du modèle physique de référence . . . 160
VII.4.b Résultats . . . 162
VII.4.b.1 Indicateurs de performance . . . 162
VII.4.b.2 Erreurs d'approximation . . . 162
VII.4.c Identification . . . 164
VII.5 Couplage avec l'hyper-réduction . . . 164
VII.5.a Cas d'étude . . . 165
VII.5.b Description de la méthodologie de couplage . . . 166
VII.5.c Construction du modèle hyper-réduit . . . 167
VII.5.d Sélection des points d'apprentissage pour l'hyper-réduction . . . 167

VIII Implémentation algorithmique et outils numériques 171
VIII.1 Présentation générale de l'implémentation des modèles . . . 172
VIII.2 Interfaçage du modèle numérique . . . 172
VIII.2.a Mise en place d'un calcul par le solveur numérique . . . 173
VIII.2.b Interface du solveur numérique . . . 173
VIII.2.b.1 Méthode d'évaluation parallèle . . . 176
VIII.2.c Interface d'un modèle de référence . . . 177
VIII.2.d Définition du domaine paramétrique . . . 179
VIII.3 Modèle de substitution basé sur la décomposition TT . . . 180
VIII.3.a Méthode de construction . . . 181
VIII.3.b Méthode d'évaluation . . . 181
VIII.3.b.1 Factorisation de calcul . . . 182
VIII.4 Exploitation des modèles . . . 183
VIII.4.a Retour sur le processus de calibration . . . 184
VIII.4.b Outils d'aide à la décision . . . 184
VIII.4.b.1 Outil de visualisation interactif . . . 185
VIII.4.b.2 Approches collaboratives . . . 186

IX Conclusions et perspectives 187
IX.1 Conclusions . . . 187
IX.2 Perspectives . . . 188

A Résultats d'algèbre linéaire et tensorielle 191

B Démonstrations des propositions 195

Références 207

Table des illustrations

Figures

I.1 Hélicoptère bimoteur en vol. . . . 1
I.2 Turbine haute-pression dans un moteur Ardiden 3. . . . 2
I.3 Disque de turbine haute-pression. . . . 2
I.4 Aube d'une turbine haute-pression. . . . 2

II.1 Courbes de fluage expérimentales et numériques. . . . 18

III.1 Domaine d'intégration réduit utilisé en hyper-réduction. . . . 41
III.2 Tenseur d'ordre 3 56
III.3 Premier déploiement matriciel d'un tenseur d'ordre 3. . . . 57
III.4 Second déploiement matriciel d'un tenseur d'ordre 3. . . . 57
III.5 Second déploiement modal d'un tenseur d'ordre 3. . . . 57
III.6 Diagrammes tensoriels pour un vecteur, une matrice et un tenseur. . . . 58
III.7 Diagramme tensoriel d'un produit matriciel. . . . 59
III.8 Diagramme tensoriel de la décomposition de Tucker. . . . 60
III.9 Diagramme tensoriel de la décomposition de Tucker hiérarchique. . . . 61
III.10 Évaluation d'un train de tenseurs 63
III.11 Diagramme tensoriel d'un train de tenseurs d'ordre 7. . . . 63

IV.1 Le tenseur $\boldsymbol{\mathcal{H}}_k$ et la matrice H_k. . . . 85
IV.2 Sélection de lignes d'un déploiement matriciel. . . . 95

V.1 Sorties du modèle de Rosenbrock et du modèle de substitution associé. . . . 111

VI.1 Sélection de colonnes impliquée dans l'algorithme de décomposition 12. . . . 122
VI.2 Illustration de la normalisation et discrétisation temporelle. . . . 129
VI.3 Comparaison de sorties entre les métamodèles couplé et découplé. . . . 136
VI.4 Comparaison de sorties entre les métamodèles simple et raffiné. . . . 137

VII.1 Déformation imposée dans le temps de $\varepsilon^{11}(t)$. . . . 145
VII.2 Distribution empirique pour e_ε. . . . 148

VII.3 Distribution empirique pour $e_{\varepsilon_{\text{vp}}}$. 148
VII.4 Distribution empirique pour e_{σ}. 148
VII.5 Distribution empirique pour e_{p}. 148
VII.6 Distribution empirique de e_{ε} en fonction de ϵ_{tol}. 149
VII.7 Distribution empirique de $e_{\varepsilon_{\text{vp}}}$ en fonction de ϵ_{tol}. 149
VII.8 Distribution empirique de e_{σ} en fonction de ϵ_{tol}. 149
VII.9 Distribution empirique de e_{p} en fonction de ϵ_{tol}. 149
VII.10 Notations des utilisées pour les diagrammes en boîte. 150
VII.11 Dépendance du nombre d'appels aux modèles de référence par rapport à ϵ_{tol}. 150
VII.12 Dépendance du nombre d'éléments stockés pour les modèles de substitution par rapport à ϵ_{tol}. 150
VII.13 Effectivité de l'estimateur de cohérence η_{σ} en fonction de l'erreur relative e_{σ}. 151
VII.14 Courbes $\epsilon^{11}(\sigma^{11})$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents. 152
VII.15 $p(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents. 152
VII.16 $\epsilon^{11}_{\text{vp}}(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents. 152
VII.17 Distribution empirique pour e^i_T. 158
VII.18 Distribution empirique pour e^i_{p}. 158
VII.19 Courbes $p(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents ainsi que les données expérimentales. 159
VII.20 Distribution empirique pour e^i_T. 163
VII.21 Distribution empirique pour e^i_{p}. 164
VII.22 Courbes $p(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents ainsi que les données expérimentales. 165
VII.23 Éprouvette d'étude 166
VII.24 Sélection de multi-indices emboités obtenue par l'algorithme 12. 168
VII.25 Points d'échantillonnage $\widehat{\mathcal{I}}_k$. 169
VII.26 Classification par k-means des points d'échantillonnages. 170
VII.27 Coordonnées parallèles des points d'apprentissage sélectionnés pour l'hyper-réduction. 170

VIII.1 Dossier du modèle paramétrique. 173
VIII.2 Exemple de fichier `material.inp.tpl`. 174
VIII.3 Exemple de fichier `computation.inp.tpl`. 175

VIII.4 Fichiers temporaires produits au cours des simulations. 177
VIII.5 Exploration préliminaire du domaine paramétrique. 179
VIII.6 Outil de visualisation pour la loi elasto-viscoplastique. 185
VIII.7 Illustration d'un outil de visualisation collaboratif. 186

Tableaux

V.1 Tableau d'équivalence entre approximation matricielle et projection. . . . 113

VII.1 Intervalles de variation des paramètres du modèle élasto-viscoplastique. . 143
VII.2 Tailles de stockages des décompositions TT associés au modèle élasto-viscoplastique. 146
VII.3 Intervalles de variation des paramètres associés au modèle Polystar. 153
VII.4 Tailles de stockages des décompositions TT pour le modèle simple. 157
VII.5 Tailles de stockages des décompositions TT pour le modèle raffiné. 157
VII.6 Tailles de stockages des décompositions TT pour le modèle raffiné avant refactorisation. 157
VII.7 Intervalles de variation des paramètres associés au modèle RaftX. 161
VII.8 Tailles de stockages des décompositions TT associées au modèle RaftX. . 162

Algorithmes

1 EIM 32
2 Version matricielle de l'EIM 36
3 Maxvol 47
4 Décomposition QR de Householder avec pivotage de colonnes 48
5 Q-DEIM 49
6 ACA 52
7 HOSVD 67
8 TT-SVD 68
9 PGD 77

10 Décomposition TT par approximations matricielles successives 86
11 Décomposition TT par approximations matricielles successives spécifiques . . 92

12 Décomposition en trains de tenseurs hétérogènes 123

Chapitre I

Introduction

I.1 Contexte général

Les hélicoptéristes réclament selon leurs applications des moteurs toujours plus puissants, légers et économiques en termes de consommation. Par ailleurs, les contraintes de sécurité imposées par la réglementation de l'Agence Européenne de la Sécurité Aérienne tendent à s'intensifier. Pour répondre à ces exigences, un des leviers majeurs mis en œuvre par les équipementiers aéronautiques comme le groupe Safran, consiste à concevoir des matériaux technologiquement performants (légers, résistants, isolants, etc.).

À titre d'illustration, la conception et le dimensionnement des moteurs d'hélicoptère nécessitent de garantir leur bon fonctionnement lors de scénarios types de vol. En particulier, ils doivent être opérationnels en dehors de leur plage nominale d'utilisation sur de courtes durées. Par exemple pour des hélicoptères bimoteurs, lors d'incident de *One Engine Inoperative* où l'un des moteurs cesse de fonctionner, le second doit être capable de supporter une montée en puissance pour maintenir l'appareil en vol.



FIGURE I.1 – L'appareil doit être maintenu en vol lorsqu'un des moteurs cesse de fonctionner[1].

Lors de ces épisodes d'urgence, les composants du moteur, notamment les aubes de turbines haute-pression (HP), sont soumis à des contraintes thermomécaniques violentes.

1. Crédit Safran Helicopter Engines

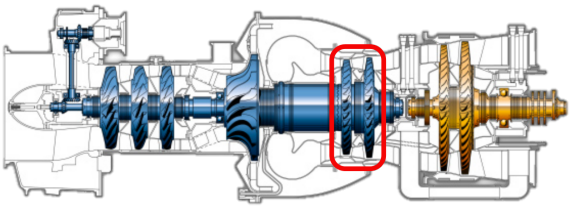


FIGURE I.2 – Localisation de la turbine haute-pression dans une coupe de moteur Ardiden 3[2].



FIGURE I.3 – Disque de turbine haute-pression[2] .



FIGURE I.4 – Aube d'une turbine haute-pression[2] .

Pour supporter de telles sollicitations l'utilisation de matériaux performants tels que les monocristaux de superalliage base nickel, est fondamentale. En parallèle, la modélisation et la simulation numérique sont essentielles pour comprendre les phénomènes physiques qui interviennent. Dans l'exemple proposé, la prédiction du comportement anisotherme et de l'endommagement des aubes nécessite l'utilisation de lois matériaux viscoplastiques complexes. Les modèles physiques en science des matériaux sont généralement formulés à partir d'équations différentielles et peuvent faire apparaitre un grand nombre de coefficients matériaux. Ces coefficients a priori inconnus doivent être identifiés au cours d'une étape dite de calibration. Cette étape est capitale pour garantir la fidélité des modèles vis-à-vis des observations physiques.

Le processus de calibration d'un modèle physique consiste à identifier les valeurs des paramètres pour lesquelles les simulations numériques du modèle coïncident autant que possible avec les résultats des essais expérimentaux disponibles. L'identification des coefficients matériaux est donc réalisée sur une implémentation spécifique de la résolution du modèle physique qu'on appelle modèle numérique. Trois facteurs influencent de manière décisive l'exécution d'une identification pertinente :

— L'importance de l'implication des experts. Selon la complexité des modèles physiques, l'implication d'un ou plusieurs experts peut être fondamentale pour conduire une identification de manière correcte ;

— La capacité à quantifier de manière objective l'écart entre les résultats expérimentaux

2. Crédit Safran Helicopter Engines

et numériques. La définition d'une mesure adaptée est complexe, car les données à comparer sont multiples et entachées d'incertitudes ;

— La possibilité d'explorer de manière efficace l'espace paramétrique. Lorsque le nombre de paramètres est grand et que la résolution du modèle physique est chère, les coûts de calcul associés à l'exploration du domaine paramétrique peuvent fortement nuire à la recherche d'un paramétrage optimal.

C'est essentiellement la combinaison de ces trois aspects qui peut rendre les procédures de calibration particulièrement délicates.

I.2 Objectifs

Dans le cadre de modèles physiques complexes, les méthodologies de calibration actuelles ne sont pas toujours satisfaisantes. Un des objectifs applicatifs de cette thèse est de mettre à la disposition des experts de nouvelles méthodologies de travail qui permettent de rendre les procédures de calibration plus efficaces.

On estime d'une part que l'intervention des experts est essentielle au travail de calibration, et d'autre part que le problème de définition de mesure écart entre résultats numériques et expérimentaux est trop spécifique aux applications pour est résolu de manière générale. Vis-à-vis de ce constat, la stratégie qui reste offerte est d'améliorer l'efficacité d'évaluation des modèles.

L'approche proposée dans ce travail de thèse consiste à fonder l'identification non pas sur le modèle numérique, mais sur un modèle de substitution significativement plus rapide à évaluer et qui exprime de manière suffisamment précise la relation entre les paramètres à identifier des sorties d'intérêt issues des solutions du modèle numérique.

L'objectif scientifique de cette thèse est donc de développer une méthode générale de construction de modèles de substitution applicable à des modèles multiparamétriques tels que des lois de comportement matériaux.

I.3 Contexte scientifique

Pour certaines applications, il peut être particulièrement intéressant de sacrifier la précision d'un modèle si cela permet d'obtenir des gains importants en temps de calcul. Avoir à disposition des modèles extrêmement rapides à évaluer permet d'effectuer efficacement des études paramétriques nécessitant un grand nombre d'évaluations. La dégradation de la précision d'un modèle n'est pas nécessairement un problème, par exemple lorsque l'objectif est de comprendre les comportements globaux sans tenir compte de détails particuliers. L'utilisation de modèle de substitution s'inscrit dans ce contexte.

Deux phases fondamentalement distinctes se dégagent couramment dans les méthodes

associées aux modèles de substitution :

— La phase hors ligne de construction du modèle de substitution aussi appelé étape d'apprentissage. La mise en œuvre de la phase hors ligne diffère selon les approches, mais est essentiellement fondée sur l'agrégation d'un ensemble de résolutions du modèle physique de référence pour des valeurs de paramètres échantillonnées ;

— La phase en ligne d'exploitation du modèle de substitution. Lors de cette phase, il est possible d'obtenir une estimation des sorties d'intérêt du modèle de référence à des coûts de calcul négligeables et pour des valeurs de paramètres définis sur un espace paramétrique donné.

La méthode de construction de modèles de substitution proposée dans cette thèse s'inspire de deux types d'approches : les méthodes de modèles d'ordre réduit par projection (particulièrement développées en science de matériaux) et les méthodes de surfaces de réponses.

Les modèles d'ordre réduit par projection L'objectif des méthodes de modèles d'ordre réduit est de construire à partir de modèles physiques à « haute-fidélité », des modèles dégradés dont la résolution numérique est significativement plus rapide et permet d'obtenir des solutions approchées. La littérature relative à ces méthodes propose de nombreuses approches applicables à des modèles paramétriques formulés sous la forme d'équations différentielles ordinaires ou partielles.

L'objectif des méthodes de projection est d'approcher les solutions du modèle haute-fidélité dans un espace réduit. Plus précisément, les solutions sont recherchées comme des combinaisons linéaires de fonctions de bases (de projection) construites préalablement à partir du modèle haute-fidélité. La forme particulière des solutions approchées permet de reformuler les équations du problème initiales (phase hors ligne) en un modèle simplifié de dimension inférieure dont la résolution (phase en ligne) permet des gains en temps de calcul importants.

Les méthodes se distinguent en particulier, par la base de projection utilisée. On trouve classiquement les méthodes de bases réduites (Reduced Basis) et les méthodes POD (Proper Orthogonal Decomposition). Une large gamme de techniques existe pour permettre la mise en œuvre de ces méthodes et s'accommoder des différentes spécificités relatives aux modèles physiques. En science des matériaux, ce type de méthodes est favorisé par rapport aux méthodes de surfaces de réponses, car des résultats de convergences relatifs aux approximations existent. Comme la phase en ligne implique la résolution d'équations, ces méthodes peuvent être exploitées sur un espace paramétrique plus large que celui qui a été exploré lors de la phase hors ligne. En contrepartie, la résolution d'équations ne permet généralement pas d'obtenir des gains de calcul en ligne suffisants pour approcher les solutions en temps réel. Ce constat est particulièrement valide en plasticité où les équations

sont hautement non linéaires.

Les surfaces de réponses Les méthodes de surfaces de réponses (ou de métamodèles) visent à construire une représentation approchée de la relation entre des paramètres d'un modèle et des sorties d'intérêt. Les sorties d'intérêt correspondent aux solutions du modèle de référence ou à des grandeurs dérivées des solutions (typiquement des quantités d'intérêt dépendantes du temps).

Une surface de réponses peut s'interpréter donc comme une application définie sur l'espace paramétrique à valeurs fonctionnelles pour laquelle on dispose d'une procédure d'évaluation efficace. En général, mais pas nécessairement, les méthodes de construction de surfaces de réponses sont non intrusives. Cela signifie que le modèle de référence est vu comme une boite noire mettant en relation des paramètres et des sorties. La particularité de ces méthodes est que l'évaluation de la surface de réponses n'implique pas de résolution d'équations physiques.

Parmi ces méthodes, on trouve les régressions polynomiales, les régressions par fonctions de base radiales ou encore le Krigeage. Elles englobent également toutes les méthodes d'apprentissages automatiques (*supervised machine learning*) de régression telles que les réseaux de neurones (*neural network*) ou les machines à vecteurs de support (*support vector machine*).

Une sous-catégorie de méthodes qui nous intéresse particulièrement dans cette thèse concerne les approximations par décompositions tensorielles, où la notion de tenseurs fait ici référence à des tableaux multidimensionnels. En introduisant une discrétisation des espaces associés au modèle de référence, il est possible de donner une représentation tensorielle, dite de référence, de la surface de réponses en rassemblant l'ensemble des sorties calculable pour toutes les combinaisons de valeurs de paramètres possibles. Dû au fléau de la dimension, une approche de force brute, consistant à calculer et stocker tous les éléments des tenseurs de référence, est inenvisageable lorsque le nombre de dimensions croît.

Ces méthodes consistent à construire explicitement des approximations des tenseurs de référence basées sur des décompositions tensorielles. Les principaux enjeux concernent d'une part le coût de calcul de la phase d'apprentissage et d'autre part la place mémoire requise pour stocker les tenseurs approchés. Lorsque les tenseurs possèdent des structures sous-jacentes particulières, un certain nombre de décompositions tensorielles permettent de réduire significativement l'espace de stockage mémoire nécessaire pour les représenter exactement. Les principaux formats sont : la décomposition Canonique Polyadic, la décomposition de Tucker, les décompositions de Tucker hiérarchiques et la décomposition en train de tenseurs. Pour chaque format, des méthodes de construction sont proposées dans la littérature. Ces méthodes sont principalement non-intrusive au sens où l'origine des tenseurs de référence n'est pas exploitée.

La Proper Generalized Decomposition (PGD) propose une approche originale et novatrice qui consiste à approcher des solutions d'équations aux dérivées partielles (EDP) par des représentations tensorielles. Elle s'apparente à une méthode de projection puisque les solutions sont recherchées comme des éléments de produit d'espace vectoriel. La phase hors ligne consiste à construire un tenseur approché donné au format canonique via la résolution d'une reformulation des équations physiques. On rattache la PGD aux méthodes de surfaces de réponse puisque la phase en ligne est basée sur l'exploitation d'un tenseur explicitement stocké en mémoire permettant d'accéder instantanément à une estimation des solutions du modèle de référence sur un domaine paramétrique défini a priori. La PGD a montrée son efficacité en termes de précision et temps de calcul dans le cadre de lois matériaux linéaires décrites par des EDP. Cette méthode établit l'intérêt des représentations tensorielles pour approcher des solutions d'équations différentielles. Cependant, les non-linéarités qui apparaissent dans les applications qui nous intéressent ne permettent pas de reformuler aisément les équations physiques en supposant une séparation entre temps et paramètres et donc d'appliquer la PGD.

On s'intéresse dans cette thèse à une décomposition tensorielle particulière introduite relativement récemment : la décomposition en train de tenseurs. Ce format permet d'une part une « compression » importante de l'information pour des tenseurs de référence de faibles rangs et d'autre part un accès quasi temps réel à l'ensemble de ses éléments. La méthode d'approximation de tenseur TT-cross a démontré son efficacité pour construire des approximations précises et ne souffre pas du fléau de la dimension car repose sur un échantillonnage parcimonieux des éléments du tenseur de référence.

I.4 Contributions

Cette thèse propose une nouvelle approche de construction de modèles de substitution applicable à des modèles physiques multiparamétriques. Ces modèles de substitution consistent à approcher la relation entre des paramètres et des quantités d'intérêt dérivées des solutions de modèles physiques et sont basés sur des décompositions en train de tenseurs. La construction exploite les avantages des approches non intrusives permettant de ne pas avoir à reformuler la résolution numérique des modèles physiques considérés. Une exploration parcimonieuse des domaines paramétriques est adoptée permettant de limiter le nombre de résolutions des modèles physiques et donc les temps de calculs hors ligne.

Les principales contributions de ce travail de thèse correspondent :

— À l'introduction d'un cadre d'algorithmique générique pour la construction de décompositions en train de tenseurs (Chapitre IV) permettant d'approcher des tenseurs multidimensionnels quelconques et généralisant un certain nombre de procédures existantes (Chapitre V) ;

— Au développement d'une variante de l'algorithme générique (Chapitre VI) – l'algorithme de décomposition en trains de tenseurs hétérogènes – basée sur l'utilisation de l'approximation matricielle gappy POD et permettant la construction simultanée d'approximations d'un nombre arbitraire de tenseurs représentant des quantités hétérogènes issues de la résolution d'un modèle physique ;
— L'application de la méthodologie pour la construction de modèles de substitution sur des lois de comportement et d'endommagement élasto-viscoplastiques hautement non linéaires (Chapitre VII) ;
— L'implémentation d'un code de calcul modulaire applicable à tout type de modèle physique et la mise à disposition d'outils d'exploitation de données élémentaires qui illustrent l'intérêt de la méthodologie en science des matériaux (Chapitre VIII).

I.5 Organisation du mémoire

Le chapitre II de la thèse permet d'introduire plus en détail le contexte de l'identification de lois de comportement ainsi que le concept de modèle de substitution. Le chapitre III correspond à un état de l'art général sur les méthodes d'approximation utilisées en mécanique, en science des matériaux, mais aussi dans la communauté de décomposition tensorielle. Le chapitre IV décrit en détail l'algorithme générique de construction d'approximation de fonctions et tenseurs multidimensionnels basée sur le format en train de tenseurs. Le chapitre V est dédié aux spécialisations possibles de l'algorithme générique et se concentre particulièrement sur une nouvelle approche reposant sur l'approximation gappy POD. Dans le chapitre VI, on présente une méthodologie pour adapter l'algorithme générique pour la construction de modèle de substitution pour approcher des modèles physiques. Des applications variées sont proposées dans le chapitre VII et en particulier des lois de comportement et d'endommagement. Finalement, le chapitre VIII expose les détails d'implémentation des méthodes et propose des outils de visualisation interactifs destinés aux experts pour la calibration de lois matériaux.

I.6 Communications scientifiques

Certains travaux présentés dans cette thèse ont également fait l'objet de communications scientifiques, en particulier :

Article soumis dans une revue à comité de lecture :

Clément Olivier, David Ryckelynck, and Julien Cortial. Tensor-train approximation of parametric constitutive equations in elasto-viscoplasticity. Soumis pour publication dans le journal Computer Methods in Applied Mechanics and Engineering, April 2017, https://hal.archives-ouvertes.fr/hal-01590194

Contribution à des actes de conférence :

Clément Olivier, David Ryckelynck, Julien Cortial, and Christian Rey. Méthode de décomposition tensorielle pour la calibration de lois de comportement en sciences des matériaux. In Actes du 13ème colloque national en calcul des structures, May 2017, https://csma2017.sciencesconf.org/128154

Chapitre II

Contexte scientifique et technique

Table des matières

II.1 Calibration des lois matériaux 10
II.1.a Modélisation des lois matériaux 10
II.1.b Expérimentation 11
II.1.c Identification 14
II.1.d Bilan 18
II.2 Modèles de substitution 19

II.1 Calibration des lois matériaux

La simulation numérique, visant à prédire les comportements thermomécaniques des matériaux, nécessite la mise en place de modèles adaptés. La modélisation physique s'attache à formaliser mathématiquement des observations expérimentales dans le but d'être capable de les reproduire numériquement.

Les modèles sont élaborés pour reproduire les observations spécifiques qui intéressent le modélisateur dans le cadre d'applications données. La modélisation implique donc des choix. Par exemple, la modélisation du comportement des aubes de turbine d'un moteur d'hélicoptère doit représenter le plus fidèlement possible les déformations relatives à des mécanismes de fluage, de plasticité, d'endommagement et de transformations métallurgiques, pour des températures variant dans un intervalle extrêmement large. La précision des prévisions de durée de vie dépend au premier ordre de la prévision de ces déformations et des contraintes de Cauchy associées.

Dans les modèles physiques en mécanique, on distingue deux types d'équations particulières. Les premières sont liées aux lois de la physique. Elles rassemblent, les équations d'équilibre, les équations de compatibilité et les conditions limites. Ces équations sont générales au sens où elles sont valides, quel que soit le matériau. Le second type d'équations permet de caractériser spécifiquement les matériaux. Elles constituent ce qu'on appelle les lois matériaux ou de lois de comportement. Elles sont généralement formulées localement en espace et l'histoire des transformations irréversibles est décrite à l'aide de variables internes.

Les étapes principales de mise en place d'un modèle de comportement mécanique sont les suivantes :

- Caractérisation expérimentale du phénomène étudié ;
- Définition du modèle le mieux adapté pour reproduire les comportements que l'on cherche à comprendre ;
- Définition des essais et des mesures expérimentales qui permettront d'ajuster et de valider le modèle ;
- Calibration effective du modèle à partir des résultats expérimentaux et des simulations numériques.

Ces étapes qui se succèdent chronologiquement peuvent être particulièrement couteuses en temps. Compte tenu de la nature très exploratoire des procédures et questions posées, ces étapes impliquent nécessairement l'intervention d'experts du domaine.

II.1.a Modélisation des lois matériaux

Les lois matériaux sont usuellement formulées sous la forme d'équations différentielles algébriques faisant intervenir les variables mécaniques classiques (tenseur des contraintes

et tenseur des déformations) et dans le cas de lois de comportement plus élaborées des variables mécaniques internes (telles que des variables d'écrouissage, d'endommagement ...).

Les lois de comportement se divisent en deux catégories. La première catégorie est constituée de lois formalisées à partir de considérations microscopiques. Plus précisément, elles visent à prédire des comportements par la modélisation de phénomènes qui se déroulent à l'échelle microscopique. Cette approche fournit des résultats précis, mais n'est pas applicable à tous les comportements. D'autre part, le passage à l'échelle peut introduire des effets difficilement prévisibles. La seconde catégorie rassemble les lois dites phénoménologiques. Ces lois sont formulées empiriquement pour reproduire des observations expérimentales sans faire appel à des équations de bilan formulées à l'échelle microscopique. L'approche phénoménologique est la plus ancienne et toujours la plus utilisée actuellement.

Lorsque les phénomènes mis en jeu sont complexes, il peut être nécessaire de construire des lois par assemblage de modèles plus élémentaires. Dans le cadre de la mécanique des matériaux, on renvoie le lecteur aux références [13, 76] pour des descriptions plus précises sur les méthodes d'élaboration de lois de comportement.

Ces modèles font généralement apparaitre un certain nombre de paramètres appelés coefficients matériaux. Lors de l'élaboration théorique des modèles, ces coefficients n'ont pas de valeur numérique. L'un des enjeux principaux de la mise en place d'un modèle de lois de comportement pour un matériau donné consiste à déterminer la valeur de ces paramètres. C'est ce qu'on appelle l'étape d'identification ou de calibration du modèle.

En tant que modèle, les lois de comportement sont par définition imparfaites. La validation d'une loi consiste à certifier qu'elle soit capable de reproduire les observations physiques avec un niveau d'erreur jugé acceptable par les experts du domaine.

Les étapes de validation et calibration sont donc intimement liées puisqu'un modèle ne pourra pas être qualifié de valide tant qu'un vecteur de paramètres admissible n'aura pas été trouvé.

II.1.b Expérimentation

II.1.b.1 Définition des essais

Une fois qu'un modèle matériau a été théoriquement formulé, le travail de l'expert consiste à choisir les essais expérimentaux à réaliser pour pouvoir le calibrer. La définition des essais qui permettront d'ajuster correctement le modèle physique avec les résultats observés est un problème complexe. La contrainte principale vient du fait que la réalisation d'essais peut être particulièrement longue et couteuse. Un des objectifs est donc à minimiser le nombre d'essais. Pour assurer une calibration robuste, ils doivent être suffisamment complexes pour révéler l'ensemble des comportements que le modèle est censé reproduire.

Par ailleurs, ils doivent être suffisamment simples pour pouvoir mesurer et identifier les différents comportements qui se déroulent simultanément. Typiquement, certains comportements peuvent avoir des effets similaires, il s'agit donc de choisir des essais fournissant des résultats qui permettent de les distinguer.

Puisque les lois matériaux sont des relations locales, l'hypothèse fondamentale qui est faite dans la plupart des procédures est que les résultats correspondant à des essais sur des géométries simples sont suffisamment riches pour permettre d'identifier les lois. Il est très rare pour des raisons de coût, de procéder à des identifications sur des géométries plus représentatives des applications réelles. En général, les essais sont réalisés sur des échantillons en effectuant des mesures localisées en des points précis afin d'obtenir le moins d'incertitudes de mesure possible.

La définition des essais coïncide chronologiquement avec la mise en place d'une procédure d'identification qui associe les essais aux coefficients matériaux qu'ils permettent d'identifier. Le travail de l'expert consiste à distinguer les différents comportements physiques mis en jeu dans les essais et à les associer aux coefficients matériaux influents. En général, l'ordre qu'a retenu l'expert pour l'identification des paramètres a une importance puisqu'il permet de contrôler les couplages entre paramètres.

II.1.b.2 Réalisation des essais

Un essai expérimental consiste typiquement à soumettre une éprouvette constituée du matériau étudié à des sollicitations mécaniques (déformations ou contraintes imposées) et/ou thermiques et à mesurer simultanément l'évolution de variables mécaniques observables. Dans le cadre d'étude d'aubes de turbine, on réalise deux types d'essais [42].

Le premier consiste à imposer un effort constant dans une direction et à observer l'évolution des déformations ainsi que le moment de la rupture de l'échantillon. Ce type d'essais, dit de fluage, peut être réalisé de manière isotherme ou anisotherme afin de simuler les sollicitations subites en condition réelle. Ces essais permettent de mettre en évidence et de mesurer le caractère viscoplastique du matériau et donc d'identifier les coefficients associés.

Des essais de fatigue sont d'autre part effectués. Ils consistent à soumettre l'éprouvette à un cyclage uniaxial pour des déformations imposées. L'objectif de ce type d'essais est de caractériser la capacité du matériau à résister à une série de sollicitations répétées représentatives de celles subies lors de la vie de la pièce. Ces essais permettent d'identifier l'évolution de l'endommagement et donc d'identifier les coefficients associés.

Dispersion expérimentale Du point de vue de la calibration, une des problématiques principales est la non-répétabilité des essais qui se traduit par la variabilité des résultats expérimentaux mesurés pour des essais théoriquement similaires. Selon les applications,

l'importance de ce phénomène est extrêmement variable. Par exemple en science des matériaux, on tolère des variations sur les déformations de l'ordre de 0,1% mais lorsqu'il s'agit de mesurer des temps de rupture sur des essais de fatigue un facteur 2 peut-être acceptable dans certains cas.

On qualifie les variations mesurées entre essais de *dispersion expérimentale*. Cette dispersion découle d'origines multiples et s'accumule aux différentes étapes des campagnes expérimentales. Les principales causes sont liées d'une part aux étapes de conception des essais et d'autre part aux étapes d'acquisition des données.

Dans les applications qui nous intéressent, les essais sont réalisés sur des éprouvettes typiquement obtenues par fonderie. Les différences, même très faibles, de compositions des coulées mères utilisées entrainent des propriétés matériaux différentes. La solidification des éprouvettes induites par leur refroidissement entraine une germination d'un ou de multiples cristaux métalliques. Le processus de croissance des cristaux a une importance sur les comportements mécaniques, or les conditions expérimentales (état initial, évolution et homogénéité de la température ...) qui guident le processus sont difficilement contrôlables. La conception des éprouvettes implique d'autres processus qu'il est difficile de maîtriser précisément par exemple les traitements de surface. Les structures microscopiques des éprouvettes sortant d'une même ligne de production étant différentes, elles présenteront nécessairement des variations dans les comportements mécaniques observés.

La seconde source de dispersion expérimentale est liée à la mise en place des essais et aux mesures. Les machines mécaniques qui permettent d'appliquer des efforts ou des déformations aux échantillons ne sont pas parfaites. Par conséquent, les sollicitations sont connues uniquement avec des marges d'erreur. Enfin, toute mesure est nécessairement affectée par une incertitude liée à la précision de l'appareil de mesure. Cela implique à nouveau l'introduction d'incertitudes sur les résultats des campagnes expérimentales.

La dispersion expérimentale est donc généralement inévitable et extrêmement difficile à quantifier a priori. La seule manière d'estimer cette dispersion est de réaliser des séries d'essais dans des conditions expérimentales identiques. Les résultats de mesures permettent alors de construire pour chaque variable mécanique non pas de simplement des courbes d'évolution, mais des faisceaux de courbes. Typiquement, le calcul des évolutions moyennes et des écarts types permet de donner une estimation empirique des évolutions pour des conditions expérimentales idéales. Ce type d'étude permet d'apprécier la fiabilité des essais et donc de quantifier le degré de confiance associé aux résultats obtenus.

Même si la dispersion expérimentale est incontournable et peut avoir des conséquences importantes sur les résultats, elle n'est malheureusement pas systématiquement prise en compte lors de la mise en place des modèles. Une raison prédominante vient du fait que la quantification de la dispersion expérimentale représente un coût important puisqu'elle nécessite de répéter des essais dont le coût unitaire est déjà élevé. La mesure de la dispersion

expérimentale est d'autre part particulièrement intéressante lorsqu'il est possible de la prendre en compte dans la définition des modèles physiques. Or les analyses de propagation d'incertitudes dans les équations différentielles [41] ainsi que les analyses de sensibilité [95] sont des techniques qui demandent une certaine expertise. C'est une seconde raison pour laquelle la dispersion n'est généralement pas prise en compte.

II.1.c Identification

Une fois que le type de modèle a été déterminé et que les résultats des campagnes d'essais sont disponibles, il reste à identifier effectivement les coefficients matériaux. L'étape de calibration est capitale pour assurer la fidélité des prédictions numériques de modèle vis-à-vis des comportements physiques effectivement observés. La procédure d'identification consiste à résoudre le modèle numérique pour différentes valeurs de coefficients matériaux et à comparer les résultats numériques avec les résultats expérimentaux afin de déterminer le jeu de valeurs donnant la meilleure correspondance entre les données. En pratique, de nombreux obstacles peuvent rendre la procédure d'identification délicate, longue et fastidieuse.

Deux approches complémentaires sont utilisées pour identifier les coefficients matériaux.

Dans l'approche manuelle, l'expert effectue une comparaison qualitative entre les résultats expérimentaux et numériques en essayant différentes valeurs de paramètres. La connaissance de l'influence des paramètres sur la réponse du modèle lui permet de faire les bons choix de paramètres et ainsi de converger pas à pas vers un jeu de paramètres pertinents. Le problème majeur de ce type d'approche est qu'elle nécessite une compréhension profonde du modèle par l'expert et dépend également de son expérience propre. Cette approche est utilisée majoritairement lorsque le nombre de paramètres est faible et les temps de calcul associés à une unique simulation sont négligeables.

Dans le cas contraire, il est indispensable d'avoir recours également à l'autre type d'approche. Celle-ci passe par l'utilisation d'algorithmes d'optimisation. Le problème d'identification revient alors à trouver le vecteur de paramètres qui minimise un critère. Ce critère doit être défini pour mesurer la ressemblance entre les résultats expérimentaux et numériques. La formalisation de cet objectif ainsi que la définition du domaine paramétrique admissible de recherche représentent des difficultés majeures.

En pratique, les deux approches sont utilisées conjointement et reposent essentiellement sur la compréhension des modèles par les experts. Il existe des outils permettant de guider le travail des experts. Les analyses de sensibilité génèrent des indicateurs (comme les indices de Sobol) qui permettent de quantifier l'influence de chaque paramètre sur la réponse du système. Ces techniques sont des aides indispensables aux experts, mais ne permettent pas de résoudre entièrement les problèmes soulevés précédemment. On détaille dans la suite de cette section les principaux obstacles à l'identification.

II.1.c.1 Difficulté à définir une mesure d'écart

La première difficulté est liée à la comparaison des résultats. Les données qu'on cherche à confronter sont usuellement multidimensionnelles et hétérogènes. Il s'agit typiquement de courbes d'évolution temporelle associées à différentes variables mécaniques.

Un expert est en général capable à l'œil de distinguer parmi deux courbes quelle est la plus proche de la solution expérimentale. Cependant, cette comparaison demande du temps. La capacité de l'expert à comparer des courbes est donc limitée. L'automatisation de cette tâche passe donc par la définition d'une fonction critère définie sur l'espace paramétrique qu'il s'agit de minimiser. Même si l'expert possède une certaine intuition, il est en général extrêmement difficile de la retranscrire dans un critère unique non biaisé.

Typiquement, une fois la fonction objectif définie on peut s'apercevoir que les valeurs de paramètres qui la minimisent ne sont en réalité pas celles associées aux résultats les plus proches des données expérimentales. Il est parfois indispensable d'affiner le critère de minimisation en ajoutant des poids sur certaines grandeurs ou en ajoutant des quantités comme la pente des courbes. L'élaboration d'un « bon » critère, au sens des experts, nécessite donc généralement plusieurs itérations.

En outre, les valeurs de paramètres autour de la solution théorique pourront également être considérées comme pertinentes. Plus généralement, un critère permet de définir des intervalles de variation admissible pour lesquels le modèle numérique est jugé valide.

Les lois matériaux étant complexes et les données à comparer multiples, les fonctions objectifs générées sont particulièrement complexes et engendre couramment des problèmes mal posés. En autre, la fonction peut posséder de nombreux minima locaux, rendant la résolution pratique du problème d'optimisation délicate et ce d'autant plus quand la dimension de l'espace de paramétrique est grande.

Sans considérer la dispersion expérimentale, la définition d'une mesure d'écart objective est déjà une tâche difficile. La présence inévitable d'une dispersion expérimentale a d'autre part été évoquée. Les incertitudes sur les résultats expérimentaux qui peuvent ou non être quantifiées rendent la définition d'un critère d'autant plus pénible. Cependant plus la dispersion est importante et moins la précision de la calibration importe. En effet, lorsque les résultats expérimentaux présentent des différences de plus de 10%, on tolère nécessairement une marge d'erreur plus grande pour la calibration.

Pour ces raisons, la procédure de calibration doit nécessairement impliquer une expertise humaine pour définir correctement les critères et guider la recherche d'un minimum qui a un sens physique.

II.1.c.2 Espace paramétrique

Quelle que soit l'approche mise en pratique, la recherche d'un paramétrage optimal nécessite de résoudre le modèle numérique pour de nombreuses valeurs de paramètres. Cette exploration du domaine paramétrique pose des problèmes d'ordre différent.

Une première difficulté apparait lors de la définition du domaine paramétrique. La formulation des modèles est faite de manière théorique et par conséquent ne fournit pas de définition naturelle pour l'espace paramétrique. Il s'agit donc de trouver un domaine admissible au sens numérique et physique. C'est-à-dire un domaine tel que la résolution numérique soit possible et fournisse des solutions physiquement admissibles. Ce travail de définition repose sur la connaissance des experts. En général, on considère des domaines paramétriques rectangulaires. C'est-à-dire des produits tensoriels d'intervalles de réel correspondant à chaque paramètre.

La complexité de l'identification est intimement liée aux dépendances entre les paramètres du modèle. Ces dépendances empêchent d'identifier les paramètres les uns après les autres et contraignent à les considérer collectivement. Or, le nombre de combinaison de valeurs de paramètres croît exponentiellement avec le nombre de paramètres. C'est ce qu'on appelle la malédiction de la dimension. Ainsi, lorsque le nombre de paramètres à identifier est élevé, l'exploration exhaustive de l'espace paramétrique n'est plus envisageable et les procédures d'optimisation peuvent devenir inefficaces.

L'exploration du domaine paramétrique est d'autre part soumise à des contraintes de temps de calcul. Dans le cadre des études paramétriques, dont la calibration fait partie, le coût de calcul pour une unique simulation est une donnée essentielle. Les lois de comportement sont le plus souvent données sous la forme d'un système d'équations différentielles. Les temps de résolution peuvent être importants et ce d'autant plus que les non-linéarités sont grandes. Ainsi, lorsque le nombre de paramètres augmente et que le temps d'une simulation numérique n'est pas négligeable, il devient inenvisageable de tester l'ensemble de combinaisons possibles.

À titre d'illustration, pour un modèle caractérisé par 6 paramètres, 20^6 simulations sont nécessaires si on souhaite évaluer chacun d'entre eux en 20 valeurs différentes. Si une simulation numérique dure 1s, le temps nécessaire pour effectuer tous les calculs est d'environ 2 ans. Au problème de temps de calcul s'ajoute un problème d'espace de stockage.

Enfin, les modèles conçus pour reproduire des phénomènes physiques observés ne sont pas toujours aptes à le faire en pratique. En particulier, lorsque les couplages entre les équations sont forts, il n'est pas toujours évident de prédire les comportements. Lorsque l'identification pose problème, il n'est généralement pas possible de savoir s'il existe un paramétrage permettant l'identification ou si le modèle n'est simplement pas capable de reproduire les comportements attendus.

II.1.c.3 Travail collaboratif

Une dernière difficulté peut apparaitre lorsque la calibration des modèles requiert l'intervention de plusieurs experts. Certains modèles complexes sont définis à partir de couplages entre des modèles de base. Les modèles de base peuvent être fondés sur des formalismes divers de sorte que leur compréhension nécessite des compétences différentes. Dans ces circonstances, l'identification des coefficients matériaux peut nécessiter des discussions entre experts afin de choisir les valeurs de paramètres les plus pertinentes selon leurs expériences dans leur domaine d'expertise respectif. Or l'évaluation de modèle, nécessaire comme socle de discussion, peut être longue et la disponibilité des experts est souvent limitée. Les procédures d'identification sont donc considérablement pénalisées par le fait que les méthodologies ne permettent pas de collaborations efficaces entre experts.

II.1.c.4 Illustration sur la loi Polystar

On illustre quelques difficultés liées à l'identification sur un exemple concret. Polystar [30, 43, 42, 75] est une loi de comportement anisotrope formulée selon la théorie de la plasticité cristalline. Elle décrit l'évolution de la microstructure et modélise l'endommagement de fluage et de cyclage thermique. Cette loi matériau a été développée pour décrire en particulier le CMSX-4 [45] qui compose les aubes de turbine HP utilisées dans les moteurs d'hélicoptères. Polystar est paramétré par une trentaine de coefficients matériaux. Comme un certain nombre de coefficients est associé à des comportements découplés, il a été possible, dans le cadre de la thèse de Rémi Giraud [45], d'en identifier la plus grande partie. Cependant, l'identification de 8 paramètres associés à l'endommagement et au fluage tertiaire n'a pas abouti.

En réalisant une optimisation uniquement sur ces 8 paramètres, on obtient la courbe donnée de la figure II.1. L'optimisation a été effectuée par un algorithme de recuit simulé en utilisant un critère de minimisation moindres carrés entre la courbe numérique et expérimentale. Les intervalles de variation des paramètres ont été choisis suite à des discussions avec des experts.

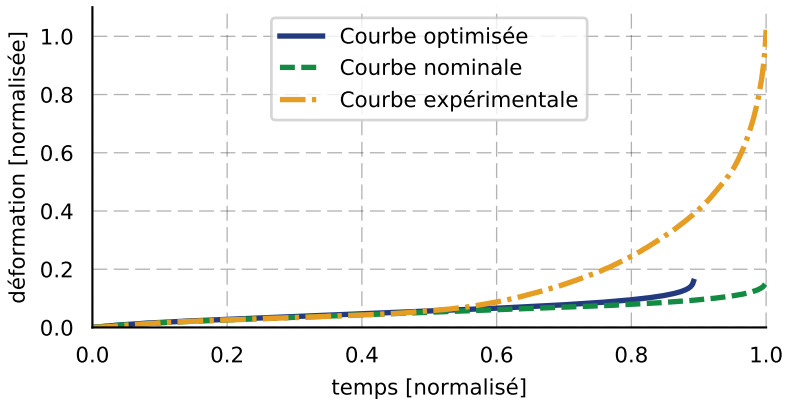


FIGURE II.1 – Résultat d'une identification via minimisation d'un critère moindres carrés.

Cet exemple d'identification infructueuse soulève des questions qui illustrent les difficultés liées à la calibration. Les intervalles de variation de paramètres ont-ils été choisis de manière suffisamment étendue ? Le critère de minimisation est-il pertinent pour permettre de trouver une courbe qui s'approche de la courbe expérimentale ? Est-il possible d'identifier indépendamment les coefficients ? Le modèle est-il capable de reproduire le comportement mesuré ?

II.1.d Bilan

Les conclusions principales à tirer de la discussion précédente sont les suivantes :

— les procédures de calibration sont insatisfaisantes lorsque le nombre de paramètres dépendants est grand ;

— l'expertise intervenant à différentes étapes est indispensable, quelle que soit l'approche suivie ;

— l'exploration du domaine paramétrique qui est inévitable peut être couteuse.

On constate d'une part qu'un verrou majeur est le temps de calcul pour obtenir une solution, et d'autre part que la précision des résultats numériques n'est pas un critère essentiel. Cela conduit à proposer une nouvelle approche pour effectuer le travail de calibration. On propose de réaliser la calibration non pas directement sur le modèle numérique à haut niveau de fidélité, mais sur un modèle de substitution permettant d'obtenir en temps réel une approximation de quantités d'intérêt dérivées de la résolution du modèle numérique pour n'importe quelles valeurs de paramètres au prix d'une relative dégradation de la précision, qui reste toutefois acceptable.

Cette nouvelle approche permet de mieux valoriser les résultats des simulations du modèle physique chers à obtenir. Dans les approches classiques, il est courant d'abandonner les données de simulation, une fois la comparaison avec les données expérimentales effectuée. Dans l'approche proposée, les résultats de simulation sont valorisés de manière systématique

puisqu'ils servent à construire le modèle de substitution.

II.2 Modèles de substitution

Un modèle de substitution correspond à une méthode numérique permettant d'approcher à coût réduit les résultats produits par un modèle de référence. La notion de modèle de référence peut correspondre à des modèles physiques paramétriques ou plus généralement à des applications mettant en correspondance des entrées (typiquement des paramètres dont on souhaite connaitre l'influence) avec des sorties (typiquement des quantités d'intérêt relatives à l'application en question). Ce concept très général rassemblant des approches variées se caractérise essentiellement par le découpage de la procédure d'approximation en deux phases distinctes.

La phase *hors ligne*, dite aussi d'apprentissage, correspond à la construction du modèle de substitution, c'est-à-dire à l'établissement de la procédure d'évaluation approché du modèle de référence. La formulation du modèle de substitution peut être extrêmement différente selon l'approche mise en œuvre. Néanmoins, elle repose toujours sur l'agrégation d'une grande quantité de résultats issus du modèle de référence. Les calculs de la phase hors ligne peuvent être particulièrement intensifs mais leur comptabilité n'a pas de pertinence puisqu'ils n'ont pas d'incidence en termes de « temps de disponibilité humaine ».

La phase *en ligne* correspond à l'exploitation du modèle de substitution. Lors de cette phase, l'évaluation du modèle de substitution pour des valeurs de paramètres choisies par l'utilisation permet de donner une estimation des sorties du modèle de référence à des coûts de calcul négligeables.

On trouve dans [38] une classification des différentes approches de construction de modèles de substitution. Elle contient 3 catégories : les modèles hiérarchiques (*hierarchical models*), les modèles d'ordre réduit (*reduced-order models*) et les modèles d'ajustement de données (*data-fit models*). Les modèles hiérarchiques, correspondant aux modèles physiques basses-fidélités typiquement obtenus via dégradation des modèles à hautes-fidélités, ne seront pas abordés dans cette thèse. Les sections III.1 et III.2 présentent un ensemble de méthodes de projection, pour construire des modèles d'ordre réduit, largement développées en science des matériaux et en mécanique. Il est essentiel de remarquer que la phase en ligne de ce type de modèle de substitution reste basée sur la physique. On donne dans cette section quelques éléments généraux relatifs aux modèles d'ajustement de données [40, 24] qu'on appelle aussi surfaces de réponses.

Lorsque les domaines relatifs au modèle de référence ont été discrétisés, une surface de réponse peut s'interpréter comme l'approximation $\widehat{f}$ d'une fonction multivariée à valeurs

vectorielles

$$f : (x_1, \dots, x_d) \in \mathcal{D} \mapsto f(\boldsymbol{x}) \in \mathbb{R}^N$$

mettant en relation des paramètres $(x_1, \dots, x_d)$ et quantités d'intérêt $f(\boldsymbol{x})$ du modèle de référence.

Les méthodes de surfaces de réponses se caractérisent principalement par le fait que la phase en ligne ne fait pas appel à des considérations physiques. En d'autres termes, la procédure d'évaluation de $\widehat{f}$ n'est pas basée sur la résolution d'équations physiques. Cette caractéristique fondamentale rend la certification de ce type de modèle de substitution vis-à-vis des prédictions physique, difficile. Néanmoins, ces méthodes connaissent actuellement un regain d'intérêt en science des matériaux [28].

Pour la grande majorité des méthodes de surfaces de réponses, la phase d'apprentissage est non-intrusive. Cela signifie que la fonction f à approcher est uniquement vue comme une boite noire. En particulier, à aucun moment la méthode ne fait intervenir de reformulation des équations physiques qui constituent le modèle de référence. Ce type de méthode inclut les régressions polynomiales, les régressions par fonctions de base radiales ou encore le Krigeage. C'est également le cas de toutes les méthodes d'apprentissages automatiques (*supervised machine learning*) de régression telles que les réseaux de neurones (*neural network*) ou les machines à vecteurs de support (*support vector machine*).

Les méthodes d'approximation tensorielles constituent une sous-catégorie de méthodes de surfaces de réponses qui nous intéressent particulièrement dans le cadre de cette thèse. L'objet de la section III.5 est d'en présenter un aperçu. Parmi ces méthodes, la PGD présentée dans la section III.5.g propose une approche de construction intrusive impliquant une reformulation des équations du modèle physique.

Dans l'ensemble des méthodologies, la construction d'un modèle de substitution nécessite une exploration de l'espace paramétrique sur lequel sont définies les entrées du modèle de référence. Ces points spécifiques du domaine paramétrique sont appelés *points d'apprentissage* et constituent *l'ensemble d'apprentissage*.

Deux types de méthodes se distinguent en fonction de l'ensemble d'apprentissage utilisé.

Type 1 : Ces méthodes sont capables d'intégrer un nombre arbitraire de points d'apprentissage qui ont été choisis ou imposés a priori. C'est le cas des méthodes de régressions linéaires, de régressions polynomiales, mais aussi de méthodes plus élaborées telles que la complétion de tenseurs [71, 114] ainsi que la majorité des méthodes utilisées en apprentissage automatique (*Machine learning*).

Type 2 : Ces méthodes déterminent de manière adaptative les points d'apprentissage nécessaires lors de la phase hors ligne. Il est indispensable dans ce cas de pouvoir évaluer au cours de l'algorithme le modèle de référence pour n'importe quel paramétrage. En général,

ces méthodes requièrent malgré tout de définir initialement une partie de l'ensemble d'apprentissage. Parmi ces méthodes, on retrouve typiquement le Krigeage et un certain nombre de méthodes d'approximation tensorielle présentées dans la section III.5.

Ensemble d'apprentissage Lorsqu'une partie ou la totalité de l'ensemble d'apprentissage doit être déterminée a priori pour appliquer les méthodes de modèle de substitution, le problème du choix le plus pertinent apparait. Le modèle de référence étant par définition couteux, le choix d'un ensemble d'apprentissage est soumis à un compromis entre sa taille (nombre de points) et sa représentativité du domaine complet. Sans connaissance préliminaire sur les relations entre entrées et sorties d'un modèle de référence, il n'est pas possible de garantir qu'un ensemble d'apprentissage permette de représenter l'ensemble des comportements potentiels. L'objectif est donc de choisir un ensemble de points limité permettant de représenter au mieux les comportements du modèle de référence.

La définition d'ensemble d'apprentissage peut être effectuée via l'utilisation de méthodes de plan d'expérience (*Design of experiment*) [77]. Ces méthodes s'intéressent à sélectionner des points dans un domaine prédéfini de manière à assurer une répartition pertinente.

Dans l'ensemble de cette thèse, on considère des domaines $\mathcal{D}$ définis comme des produits cartésiens d'intervalles de $\mathbb{R}$:

$$\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_d \quad \text{avec} \quad \mathcal{D}_k \subset \mathbb{R}$$

La méthode naturelle qui consiste à définir sur cet espace une grille de discrétisation n'est pas envisageable en grande dimension puisque le nombre de points est soumis au fléau de la dimension.

La méthode aléatoire de Monte Carlo, très simple à mettre en œuvre, est largement utilisée pour échantillonner des domaines. Elle consiste à tirer aléatoirement des points pour chaque intervalle selon une loi uniforme. Le défaut de cette sélection est que l'échantillonnage est réalisé indépendamment sur chaque intervalle et n'assure donc pas de répartition suffisamment homogène dans le domaine complet. D'autre part, on peut montrer que lorsque la dimension augmente les points déterminés par cette méthode ont tendance à se trouver proches du bord du domaine.

La méthode de l'hypercube latin permet de repartir les points de manière uniforme sur chaque intervalle, cependant comme pour la méthode de Monte Carlo l'échantillonnage de chaque intervalle est indépendant et donc la méthode ne garantit pas une répartition uniforme dans le domaine complet. Des méthodes plus avancées ajoutant des contraintes de répartition permettent d'obtenir des échantillonnages plus efficaces. Par exemple la méthode Orthogonale Array-based Latin hypercube [117] ou la méthode (t,m,s)-net [77].

Dans les applications (Chapitre VII), on utilise principalement les suites de Halton

[57] définies en dimension arbitraire. Ces suites pseudo-aléatoires à faible discrépance permettent de répartir des points dans un domaine de manière particulièrement uniforme tout en gardant un aspect pseudo-aléatoire.

Chapitre III

État de l'art

Ce chapitre présente un état de l'art d'un ensemble de méthode de construction de modèle de substitution. On commence par décrire les méthodes classiques utilisées en science des matériaux et en mécanique. On présente dans un second temps la notion d'approximation matricielle qui constitue un outil essentiel dans l'ensemble des méthodes de construction de modèle de substitution. On introduit dans la suite les décompositions tensorielles ainsi que les méthodes de construction approchées associées.

Table des matières

III.1 Méthodes de réduction de modèle par projection 25
III.1.a Décomposition orthogonale aux valeurs propres 26
III.1.b Méthode des bases réduites 28
III.2 Méthodes de réduction de la complexité 30
III.2.a Méthodes d'interpolation empirique 30
III.2.b Méthodes de projection lacunaire 39
III.3 Décompositions matricielles approchées 42
III.3.a Notations matricielles 43
III.3.b Décomposition en valeurs singulières 44
III.3.c Sélection de colonnes/lignes 46
III.3.d Décomposition skeleton et pseudo-skeleton 49
III.3.e Approximation gappy 53
III.4 Décompositions tensorielles 53
III.4.a Notations tensorielles 54
III.4.b Décomposition canonique 59
III.4.c Décomposition de Tucker 60
III.4.d Décomposition de Tucker hiérarchique 60
III.4.e Décomposition en train de tenseurs 61
III.5 Méthodes d'approximations tensorielles de faible rang 64
III.5.a Alternating least square 65
III.5.b High Order Singular Value Decomposition 67

III.5.c TT-SVD 67
III.5.d TT-cross 69
III.5.e DMRG 72
III.5.f Autres algorithmes associés au format TT 75
III.5.g Proper Generalized Decomposition 75

III.1 Méthodes de réduction de modèle par projection

Les méthodes de réduction de modèle par projection correspondent aux méthodes les plus utilisées actuellement en mécanique et en sciences des matériaux pour construire des modèles de substitution. On considère un problème paramétrique $\mathcal{P}_{\boldsymbol{\mu}}$ dépendant du paramètre $\boldsymbol{\mu} \in \mathcal{D}$, typiquement formulé par un système d'EDO ou d'EDP, dont la solution $f_{\boldsymbol{\mu}}$ est définie pour tout $\boldsymbol{x} \in \Omega$. Ω et $\mathcal{D}$ représentent respectivement le domaine de définition et le domaine paramétrique admissible. Selon le contexte, une coordonnée de temps $t \in [0, T]$ peut être incluse dans Ω ou $\mathcal{D}$. On note l'espace des solutions $\mathcal{M} = \{f_{\boldsymbol{\mu}} \mid \forall\, \boldsymbol{\mu} \in \mathcal{D}, f \text{ soit solution du problème } \mathcal{P}_{\boldsymbol{\mu}}\}$.

Un des principes des méthodes de réduction de modèles par projection est de rechercher les solutions comme des combinaisons linéaires de fonctions φ_k définies sur Ω :

$$f_{\boldsymbol{\mu}} = \sum_{k=1}^{K} \alpha_k(\boldsymbol{\mu})\varphi_k \tag{III.1}$$

où l'ensemble $\{\varphi_k\}_{k=1}^{K}$ constitue une base de fonctions qui génèrent un espace d'approximation de $\mathcal{M}$ de dimension K. Implicitement, l'hypothèse sous-jacente qui est faite est que l'épaisseur de Kolmogorov de l'espace $\mathcal{M}$ est suffisamment faible pour espérer avoir des approximations précises avec un rang égal à K. La substitution d'une solution écrite sous la forme (III.1) dans le problème $\mathcal{P}_{\boldsymbol{\mu}}$ engendre un nouveau système moins couteux à résoudre.

L'intérêt principal des modèles réduits par projection provient de la décomposition du travail en deux phases distinctes.

La phase hors ligne, qui représente en général des coûts de calcul importants, correspond à une étape d'apprentissage du modèle numérique. Deux activités qui peuvent être menées simultanément permettent de préparer le modèle réduit :

- La résolution répétée du modèle numérique pour des valeurs échantillonnées dans le domaine paramétrique ;
- L'exploitation des données pour construire les bases d'approximations et formuler le modèle réduit par projection des équations physiques.

La phase en ligne consiste à résoudre pour n'importe quelles valeurs de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ le système d'équations physiques dégradées du modèle réduit afin d'obtenir une approximation de la solution du modèle initial. Si K est suffisamment petit, la résolution des équations est significativement plus rapide et permet une exploitation efficace du modèle.

Remarque 1. *Les méthodes de réduction de modèles par projection permettent de calculer lors de la phase en ligne des solutions correspondant à des paramétrages hors du*

domaine paramétrique initialement défini. Même si dans ce cas aucune certification sur la qualité de la solution ne peut être établie, cela peut donner une estimation de la forme des solutions en particulier sur le bord du domaine admissible.

On présente dans cette section les deux méthodes par projection les plus notoires : la décomposition orthogonale aux valeurs propres et la méthode des bases réduites. On renvoie le lecteur intéressé aux références [11, 96] pour une description détaillée de ces méthodes et quelques-unes de leurs applications.

Dans le cas de systèmes non linéaires, des termes couteux à calculer en ligne restent présents après la projection. Nous verrons dans la section suivante qu'il existe des techniques pour résoudre ces difficultés.

III.1.a Décomposition orthogonale aux valeurs propres

La décomposition orthogonale aux valeurs propres dont on fait usuellement référence par son acronyme anglais POD (*Proper Orthogonal Decomposition*) constitue la méthode emblématique des méthodes de réduction par projection. On associe l'introduction de cette méthode à [80] pour modéliser des écoulements turbulents en mécanique des fluides. Depuis, cette méthode a été abondamment étudiée dans le cadre de problèmes paramétriques et pour des applications variées. On présente dans cette section les idées essentielles mises en jeux dans cette méthodologie et le lecteur pourra trouver des descriptions plus détaillées dans [112, 33].

Rigoureusement, la POD fait référence à la décomposition particulière qui est utilisée pour représenter les solutions. La méthode consiste à chercher les solutions comme des combinaisons linéaires (Éq. (III.1)) en utilisant une base dite POD. Cette base est constituée de l'ensemble des fonctions orthonormées $\{\varphi_k\}_{k=1}^{K}$ qui minimise le problème :

$$\min \int_{\boldsymbol{\mu}\in\mathcal{D}} \left\| f_{\boldsymbol{\mu}} - \sum_{k=1}^{K} \langle f_{\boldsymbol{\mu}}, \varphi_k \rangle \, \varphi_k \right\|_2^2 d\boldsymbol{\mu} \tag{III.2}$$

où $\langle . \rangle$ et $\|.\|_2$ définissent respectivement le produit scalaire et la norme associée pour l'espace de Hilbert auquel appartiennent les solutions $f_{\boldsymbol{\mu}}$. On ne donne pas ici les détails mathématiques relatifs aux définitions rigoureuses des espaces. La base POD est dite « optimale » car elle permet de représenter le plus précisément possible pour un rang fixé l'ensemble des solutions du problème au sens de la moyenne des erreurs quadratiques. Le problème donné à l'équation (III.2) admet théoriquement [59] une solution qui correspond à l'ensemble des fonctions propres (associées à des valeurs propres σ_k) d'un opérateur $\mathcal{R}$ qui dépend uniquement de l'ensemble $\mathcal{M}$. En général, la valeur de K qui permet d'obtenir des résidus nuls est trop importante pour permettre une résolution efficace lors de la phase en ligne. La réduction de la complexité est réalisée en choisissant des valeurs de K plus

petites pour former ce qu'on appelle des bases tronquées. Comme la valeur du rang K influence d'une part la précision des approximations et d'autre part le coût de la phase en ligne, son choix est déterminant .

Une manière de choisir le rang consiste à résoudre le problème donné à l'équation (III.2) de manière exacte puis de sélectionner les maximums de modes, estimés comme significatifs, en fonction de l'évolution des valeurs propres σ_k. Le rang de troncature K peut d'autre part être défini via la résolution du problème dual donné à l'équation(III.3) en choisissant a priori une tolérance d'approximation ϵ_{tol} : K est défini comme le plus petit rang vérifiant pour toute solution $f_{\boldsymbol{\mu}} \in \mathcal{M}$:

$$\left\| f_{\boldsymbol{\mu}} - \sum_{k=1}^{K} \langle f_{\boldsymbol{\mu}}, \varphi_k \rangle \, \varphi_k \right\|_2 \leq \epsilon_{\text{tol}} \tag{III.3}$$

La phase hors ligne de la méthode POD consiste donc à construire la base réduite puis procéder à la projection des équations (étape non présentée dans le cadre général). On illustre une utilisation de la décomposition POD dans la méthode de Galerkin. Si on considère un problème de mécanique linéaire typiquement donnée par une formulation faible, la méthode consiste dans un premier temps à construire l'espace d'approximation de dimension K qu'engendre la base POD. L'écriture des solutions sur l'espace d'approximation puis la projection des équations sur ce même espace permet d'obtenir un système d'équations de dimension K. C'est ce nouveau système qui constitue le modèle réduit.

En pratique pour des problèmes non linéaires, la projection des équations ne permet pas de rendre la complexité de la résolution en ligne indépendante de la dimension du problème initial discrétisé. Des méthodes complémentaires pour traiter les non-linéarités ont été développées et seront présentées dans la section III.2.

Construction de la base POD La POD est un problème qui a été étudié dans de nombreuses communautés scientifiques. Selon les domaines, on y fait référence entre autres sous les appellations suivantes :

— Karhunen-Loeve Decomposition (KLD) [79, 66] en analyse de données ;
— Principal Component Analysis (PCA) [65] en analyse statistique ;
— Empirical Eigenvector Basis [72] en dynamique non linéaire des structures.

La résolution exacte du problème donné à l'équation (III.2) n'est en général pas envisageable puisqu'elle suppose que l'on dispose de l'ensemble des solutions $f_{\boldsymbol{\mu}} \in \mathcal{M}$ afin de réaliser l'intégration. La méthode de *Snapshots POD* introduite par [112] permet de résoudre empiriquement ce problème. La méthode consiste à ne réaliser que partiellement l'intégration sur $\mathcal{D}$. Plus spécifiquement, l'intégration est réduite à un nombre limité de solutions appelées *snapshots* associées aux éléments de l'ensemble $D = \left\{ \boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(m)} \right\} \subset \mathcal{D}$. Les éléments $\boldsymbol{\mu}^{(j)}$ sont appelés les points d'apprentissage.

Le problème donné à l'équation (III.2) devient : trouver une base orthonormée $\{\varphi_k\}_{k=1}^K$ qui minimise :

$$\min \sum_{j=1}^{m} \left\| f_{\boldsymbol{\mu}^{(j)}} - \sum_{k=1}^{K} \left\langle f_{\boldsymbol{\mu}^{(j)}}, \varphi_k \right\rangle \varphi_k \right\|_2^2 \tag{III.4}$$

En pratique, la résolution numérique implique une discrétisation $\overline{\Omega} = \left\{\overline{\boldsymbol{x}}^{(1)}, \ldots, \overline{\boldsymbol{x}}^{(n)}\right\}$ de l'espace de définition Ω. On définit la matrice des snapshots $A \in \mathbb{R}^{n \times m}$ et la matrice de base réduite $U \in \mathbb{R}^{n \times K}$ telles que :

$$A(i,j) = f_{\boldsymbol{\mu}^{(j)}}(\overline{\boldsymbol{x}}^{(i)}) \quad \text{et} \quad U(i,k) = \varphi_k(\overline{\boldsymbol{x}}^{(i)})$$

Le problème donné à l'équation (III.4) devient donc : trouver une matrice orthogonale $U \in \mathbb{R}^{n \times K}$ qui minimise :

$$\min \left\| A - UU^T A \right\| \tag{III.5}$$

On peut montrer que ce problème de minimisation équivaut au calcul des vecteurs singuliers à gauche de la matrice A. Le problème revient donc au calcul de la décomposition en valeurs singulières (*Singular Value Decomposition* (SVD)) de A. Une présentation détaillée de la SVD est faite dans la section III.3.b.

Une difficulté soulevée par cette méthode est la sélection des snapshots. Puisque la base d'approximation est calculée uniquement à partir des snapshots, ces derniers doivent présenter des comportements suffisamment variés pour être certain de pouvoir reconstruire une approximation fidèle des solutions lors de la phase en ligne. En pratique, la connaissance de la physique et de l'influence de chaque paramètre peut guider ce choix, mais des échantillonnages pseudo-aléatoires réguliers sont souvent choisis. On évoque dans la section II.2 des alternatives d'échantillonnage de domaine.

III.1.b Méthode des bases réduites

La méthode des bases réduites [98] en anglais *Reduced Basis Method* (RBM) englobe une autre grande classe de méthodes de projection permettant de résoudre des problèmes paramétriques. Dans cette méthode, les fonctions de base sont construites de manière gloutonne. Contrairement à la méthode POD, elles sont égales, à réorthonormalisation près, à des solutions du modèle numérique associées à des valeurs de paramètres déterminées de manière adaptative.

La méthode des bases réduites a été introduite pour s'appliquer à des problèmes linéaires elliptiques donnés sous la forme de formulations variationnelles du type : trouver la solution

$f_{\boldsymbol{\mu}}$ dans un espace suffisamment régulier pour $\boldsymbol{\mu} \in \mathcal{D}$ tel que :

$$a_{\boldsymbol{\mu}}(f_{\boldsymbol{\mu}}, f^\star) = b_{\boldsymbol{\mu}}(f^\star), \quad \forall\, f^\star \text{ dans l'espace des fonctions test}$$

avec $a_{\boldsymbol{\mu}}$ une forme bilinéaire et coercive et $b_{\boldsymbol{\mu}}$ une forme linéaire. D'autres variantes ont été développées pour s'appliquer à des problèmes non linéaires [123, 32].

La méthode des bases réduites consiste à calculer de manière itérative (k faisant référence à l'itération courante) des solutions $f_{\boldsymbol{\mu}^{(k)}}$ associée au paramètre $\boldsymbol{\mu}^{(k)}$ choisi spécifiquement. À l'itération k, on définit une nouvelle fonction de base $\varphi_k = \frac{f_{\boldsymbol{\mu}^{(k)}}}{\left\| f_{\boldsymbol{\mu}^{(k)}} \right\|_2}$ puis l'algorithme de Gram-Schmidt est appliqué à l'ensemble des fonctions de base définies préalablement. L'efficacité de la construction de la base est fondée sur l'existence d'un estimateur d'erreur qui permet de choisir le paramètre $\boldsymbol{\mu}^{(k)}$ associé à la solution qui maximise l'erreur globale, c'est-à-dire la moins bien représentée par la base construite à l'itération k.

L'estimateur d'erreur noté $\Delta_k(\boldsymbol{\mu})$ est une fonction scalaire définie sur l'espace paramétrique $\mathcal{D}$ qui dépend de l'itération courante k. À l'itération k, trouver le vecteur de paramètres $\boldsymbol{\mu}^{(k)}$ associé à la prochaine fonction de base consiste à résoudre le problème :

$$\boldsymbol{\mu}^{(k)} = \underset{\boldsymbol{\mu} \in \mathcal{D}}{\operatorname{argmax}}\, \Delta_{k-1}(\boldsymbol{\mu})$$

En pratique, l'espace de recherche est restreint à un sous-domaine discret D du domaine paramétrique $\mathcal{D}$. La résolution des problèmes d'optimisation successifs est rendue possible par le fait que l'estimateur d'erreur soit calculable en complexité indépendante de N (où N est la dimension du problème initial). Les itérations se terminent lorsque la tolérance d'approximation ϵ_{tol} préalablement choisie est atteinte, c'est-à-dire lorsque $\Delta_K(\boldsymbol{\mu}) \leq \epsilon_{\text{tol}}$.

Le principal avantage de cette méthode est d'assurer que les solutions calculées lors de la phase en ligne pour des valeurs de paramètres inclus dans le sous-domaine paramétrique discret D admettent des erreurs inférieures à une tolérance ϵ_{tol} fixée lors de la phase hors ligne. Toutefois, cette méthode peut être difficile à mettre en œuvre :

— La méthodologie n'est pas applicable directement lorsque la dépendance aux paramètres des formes $a_{\boldsymbol{\mu}}$ et $b_{\boldsymbol{\mu}}$ est non affine. En effet, dans ce cadre il n'est pas possible de réduire les calculs d'intégration. La phase en ligne implique le calcul de termes dont la complexité dépend de la taille du problème non réduit. Des méthodes complémentaires existent néanmoins comme l'EIM (Section III.2) pour permettre d'étendre la méthodologie aux problèmes non affines.
— La construction d'un estimateur d'erreur peut être extrêmement délicate et nécessiter des développements mathématiques lourds.

III.2 Méthodes de réduction de la complexité

Les méthodes POD et RBM sont très performantes lorsqu'il s'agit de problèmes linéaires paramétriques de grande taille. Cependant, lorsque des termes non linéaires sont présents, les coûts de calcul associés (par exemple pour des lois de comportement non linéaires, les coûts associés à la mise à jour des opérateurs tangents pendant les itérations de Newton-Raphson) peuvent remettre en cause l'intérêt pratique des méthodes de réduction de modèle par projection. Des méthodes de réduction complémentaires, nommées parfois méthodes d'hyper-réduction (à distinguer de la méthode d'hyper-réduction introduite par [100]) ont été élaborées afin d'étendre leur cadre d'application. Les nouveaux modèles sont parfois appelés modèles hyper-réduits. L'idée principale exploitée par ces méthodes pour réduire la complexité de calcul est d'estimer les termes non linéaires sur des sous-domaines déterminés lors de la phase hors ligne. La mise en œuvre de cette idée se décline en différentes techniques qu'on rassemble en deux catégories : les méthodes interpolantes et les méthodes lacunaires. L'objet de cette section est de présenter quelques-unes des approches classiques.

III.2.a Méthodes d'interpolation empirique

On désigne par méthodes d'interpolation empirique les méthodes qui permettent de construire des approximations interpolantes de termes (non linéaires en général) intervenant dans les équations.

La Méthode d'Interpolation Empirique (EIM : *Empirical Interpolation Method*) a été introduite et développée par [7, 53, 84] afin d'étendre le cadre d'application de la méthode des bases réduites à des formes non affines par rapport aux paramètres. Dans le cadre de la RBM, l'idée de l'EIM est de donner une représentation séparée des termes non linéaires qui apparaissent en argument des intégrales. Cela permet lors de la phase hors ligne d'approcher les intégrales par des décompositions en combinaisons linéaires dont l'évaluation est indépendante de la taille du problème initial.

On ne présente pas ici la construction de l'approximation des intégrales, mais uniquement deux aspects de l'EIM telle qu'elle a été introduite par [7] :

— La représentation séparée pour approcher des fonctions de deux variables ;

— La procédure de construction de cette représentation.

Définition 1. *Soit une fonction* $f : \boldsymbol{x} \in \mathcal{D} \mapsto f(\boldsymbol{x}) \in \mathbb{R}$ *définie sur un domaine* $\mathcal{D} \subset \mathbb{R}^n$. *L'approximation interpolante de* f, *notée* $I_K[f]$, *associée aux points d'interpolation* $\left\{\boldsymbol{x}^{(k)}\right\}_{k=1}^{K} \subset \mathcal{D}$ *et à la base* $\{\varphi_k\}_{k=1}^{K}$ *est égale à :*

$$I_K[f] = \sum_{k=1}^{K} \alpha_k \varphi_k \qquad \text{avec} \quad \{\alpha_k\}_{k=1}^{K} \in \mathbb{R}^K$$

et telle que (propriété d'interpolation) :

$$I_K[f](\boldsymbol{x}^{(i)}) = f(\boldsymbol{x}^{(i)}) \quad \forall\, i \in [\![1, K]\!]$$

On appelle $I_K[.]$ l'opérateur d'approximation interpolante.

Pour que les objets mathématiques soient bien définis, on suppose que les fonctions f et φ_k sont suffisamment régulières et donc en particulier qu'elles sont définies pour tout $\boldsymbol{x}^{(k)}$.

Le calcul de l'approximation interpolante $I_K[f]$ consiste à trouver $\{\alpha_k\}_{k=1}^K$ solution du système linéaire donné à l'équation (III.6) :

$$\sum_{k=1}^{K} \alpha_k \varphi_k(\boldsymbol{x}^{(i)}) = f(\boldsymbol{x}^{(i)}) \quad \forall\, i \in [\![1, K]\!] \tag{III.6}$$

Remarque 2. *L'approximation interpolante $I_K[f]$ d'une fonction f dépend en réalité uniquement de l'espace vectoriel engendré par $\{\varphi_k\}_{k=1}^K$ et non spécifiquement de la base $\{\varphi_k\}_{k=1}^K$. Seuls les coefficients $\{\alpha_k\}_{k=1}^K$ dépendent de la base.*

Soit une fonction scalaire F définie sur un domaine de $\mathbb{R}^n$ dont les variables ont été séparées en deux groupes :

$$\begin{array}{rccc} F & : & \Omega \times \mathcal{D} & \longrightarrow & \mathbb{R} \\ & & (\boldsymbol{x}, \boldsymbol{\mu}) & \longmapsto & F(\boldsymbol{x}, \boldsymbol{\mu}) \end{array} \tag{III.7}$$

Et l'ensemble de fonctions associé :

$$\mathcal{F} = \{f_{\boldsymbol{\mu}} : \boldsymbol{x} \in \mathcal{D} \mapsto F(\boldsymbol{x}, \boldsymbol{\mu}) \in \mathbb{R} \mid \forall\, \boldsymbol{\mu} \in \mathcal{D}\} \tag{III.8}$$

Si on dispose d'une base d'approximation $\{\varphi_k\}_{k=1}^K$ de $\mathcal{F}$, l'approximation interpolante (Définition 1) permet de construire une approximation $\widehat{f_{\boldsymbol{\mu}}}$ pour toutes fonctions $f_{\boldsymbol{\mu}} \in \mathcal{F}$ uniquement à partir de la connaissance de f aux points $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(K)}\right\} \subset \mathcal{D}$.

Sous condition que le système de l'équation (III.6) soit soluble, n'importe quel choix de base et points d'interpolation permet de définir une approximation interpolante. Toutefois, certains choix permettent de construire des approximations plus précises.

Étant donné un ensemble $\mathcal{F}$ (Éq. (III.8)), l'algorithme EIM (Algorithme 1) est une heuristique gloutonne permettant de construire simultanément une suite de fonctions de base $\{\varphi_1, \ldots, \varphi_k\}$ et une suite de points d'interpolation $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}\right\}$ à partir desquels il est possible d'approcher les éléments de $\mathcal{F}$ par des approximations interpolantes précises.

Remarque 3. *Il est possible de substituer les normes utilisées pour sélectionner les points d'apprentissage et d'interpolation (par exemple choisir une norme infinie). Dans la*

Algorithme 1 : EIM

Données : Un ensemble de fonctions $\mathcal{F} = \{f_{\boldsymbol{\mu}} : \boldsymbol{x} \in \mathcal{D} \mapsto F(\boldsymbol{x}, \boldsymbol{\mu}) \in \mathbb{R} \mid \forall\, \boldsymbol{\mu} \in \mathcal{D}\}$ et un rang d'approximation K choisi a priori.

Résultat : Un ensemble de points d'interpolation $\left\{\boldsymbol{x}^{(k)}\right\}_{k=1}^{K}$ et une base de fonction $\{\varphi_k\}_{k=1}^{K}$.

Initialisation

Définir la première fonction génératrice $f_{\boldsymbol{\mu}^{(1)}}$ et le premier point d'interpolation $\boldsymbol{x}^{(1)}$ tel que :

$$\boldsymbol{\mu}^{(1)} = \underset{\boldsymbol{\mu}\in\mathcal{D}}{\operatorname{argmax}} \left\|f_{\boldsymbol{\mu}}\right\|_2 \quad \text{et} \quad \boldsymbol{x}^{(1)} = \underset{\boldsymbol{x}\in\Omega}{\operatorname{argmax}} \left|f_{\boldsymbol{\mu}^{(1)}}(\boldsymbol{x})\right|$$

Définir la première fonction de base :

$$\varphi_1 = \frac{f_{\boldsymbol{\mu}^{(1)}}}{f_{\boldsymbol{\mu}^{(1)}}(\boldsymbol{x}^{(1)})}$$

Pour $k \in [\![2, K]\!]$ **faire**

Définir la $k^{\text{ème}}$ fonction génératrice $f_{\boldsymbol{\mu}^{(k)}}$ et le $k^{\text{ème}}$ point d'interpolation $\boldsymbol{x}^{(k)}$ tel que :

$$\boldsymbol{\mu}^{(k)} = \underset{\boldsymbol{\mu}\in\mathcal{D}}{\operatorname{argmax}} \left\|f_{\boldsymbol{\mu}} - I_{k-1}\left[f_{\boldsymbol{\mu}}\right]\right\|_2 \quad \text{et} \quad \boldsymbol{x}^{(k)} = \underset{\boldsymbol{x}\in\Omega}{\operatorname{argmax}} \left|f_{\boldsymbol{\mu}^{(k)}} - I_{k-1}\left[f_{\boldsymbol{\mu}^{(k)}}\right]\right|$$

où $I_{k-1}[.]$ est l'opérateur d'interpolation associé à $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k-1)}\right\}$ et $\{\varphi_1, \ldots, \varphi_{k-1}\}$.

Définir la $k^{\text{ème}}$ fonction de base φ_k telle que :

$$\varphi_k = \frac{f_{\boldsymbol{\mu}^{(k)}} - I_{k-1}\left[f_{\boldsymbol{\mu}^{(k)}}\right]}{f_{\boldsymbol{\mu}^{(k)}}(\boldsymbol{x}^{(k)}) - I_{k-1}\left[f_{\boldsymbol{\mu}^{(k)}}\right](\boldsymbol{x}^{(k)})}$$

suite, par abus de langage, on fait référence à ces variantes toujours par le terme « algorithme EIM ».

L'erreur d'approximation peut être quantifiée [7] pour tout k :

$$\|f_{\boldsymbol{\mu}} - I_k[f_{\boldsymbol{\mu}}]\| \leq (1 + \Lambda_k) \inf_{w \in \mathcal{F}_k} \|f_{\boldsymbol{\mu}} - w\|$$

avec Λ_k la constante de Lebesgue telle que $\Lambda_k \leq 2^k - 1$ et $\mathcal{F}_k = \text{Vect}(\varphi_1, \ldots, \varphi_k)$. On peut noter de la borne de Lebesgue est particulièrement pessimiste, mais qu'en pratique la convergence du second terme permet de construire des approximations satisfaisantes.

En pratique, la résolution des problèmes de maximisation qui interviennent dans l'algorithme impose une discrétisation des domaines Ω et $\mathcal{D}$ notés respectivement $\overline{\Omega}$ et D. Notons que les étapes de recherche de maximum sont d'autant plus couteuses, mais simultanément plus précises que les grilles de discrétisation sont fines. Il s'agit donc de trouver un compromis de discrétisation selon le cas d'étude.

Remarque 4. *Les fonctions de base φ_k construites itérativement sont des combinaisons linéaires des fonctions $\left\{f_{\boldsymbol{\mu}^{(1)}}, \ldots, f_{\boldsymbol{\mu}^{(k)}}\right\}$. Par conséquent, d'après la remarque 2, l'opérateur d'approximation interpolante $I_k[.]$ à l'itération k est associé de manière équivalente à l'ensemble de fonctions $\left\{f_{\boldsymbol{\mu}^{(1)}}, \ldots, f_{\boldsymbol{\mu}^{(k)}}\right\}$. Les fonctions φ_k construites au cours de l'algorithme peuvent ainsi être vues uniquement comme des intermédiaires de calcul. Finalement, l'algorithme EIM permet essentiellement des construire une succession de points d'interpolation $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}\right\}$ et de points d'apprentissage $\left\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(k)}\right\}$ à partir desquels il est possible de définir une approximation interpolante pour n'importe quelle fonction de $\mathcal{F}$.*

L'EIM a donné lieu à de nombreuses variantes dont les principales (GEIM, DEIM, BPIM, Q-DEIM) seront présentées dans la suite.

III.2.a.1 Squelette fonctionnel

L'approximation interpolante de la définition 1 permet de construire des approximations particulières de fonctions de l'ensemble $\mathcal{F}$ défini à l'équation (III.8). Or, $\mathcal{F}$ est défini à partir de la fonction F (Éq. (III.7)). On peut donc formuler l'approximation interpolante de manière à approcher F [10, 23]. Cette formulation permet, entre autres, de faire apparaitre clairement le rôle symétrique des variables $\boldsymbol{x}$ et $\boldsymbol{\mu}$ évoqué en remarque 4.

Une fonction F de deux variables (Éq. (III.7)) peut être approchée par son squelette fonctionnel (*functional skeleton*) $\widehat{F}$ associé aux ensembles de points $\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{K} \subset \Omega$ et

$\left\{\boldsymbol{\mu}^{(i)}\right\}_{i=1}^{K} \subset \mathcal{D}$ en supposant que la matrice $\left[\mathcal{F}(\boldsymbol{x}^{(i)}, \boldsymbol{\mu}^{(j)})\right]_{i,j}$ soit inversible. On a :

$$\widehat{F} = \sum_{i,j=1}^{K} F(\boldsymbol{x}; \boldsymbol{\mu}^{(i)}) M(i,j) F(\boldsymbol{\mu}; \boldsymbol{x}^{(j)})$$

où la matrice $M \in \mathbb{R}^{K \times K}$ est égale à :

$$M = \begin{bmatrix} F(\boldsymbol{x}^{(1)}, \boldsymbol{\mu}^{(1)}) & \dots & F(\boldsymbol{x}^{(1)}, \boldsymbol{\mu}^{(K)}) \\ \vdots & & \vdots \\ F(\boldsymbol{x}^{(K)}, \boldsymbol{\mu}^{(1)}) & \dots & F(\boldsymbol{x}^{(K)}, \boldsymbol{\mu}^{(K)}) \end{bmatrix}^{-1}$$

On peut montrer que la fonction $\widehat{F}$ correspond exactement à la décomposition :

$$\widehat{F}(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{k=1}^{K} g_k(\boldsymbol{\mu}) h_k(\boldsymbol{x}) \quad \forall\, (\boldsymbol{x}, \boldsymbol{\mu}) \in \Omega \times \mathcal{D} \tag{III.9}$$

qui soit interpolante aux points $\left\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(K)}\right\}$ et $\left\{\boldsymbol{\mu}^{(1)}, \dots, \boldsymbol{\mu}^{(K)}\right\}$, c'est à dire :

$$\begin{aligned} \widehat{F}(\boldsymbol{x}^{(i)}, \boldsymbol{\mu}) &= F(\boldsymbol{x}^{(i)}, \boldsymbol{\mu}), & \forall\, (i, \boldsymbol{\mu}) \in [\![1, K]\!] \times \mathcal{D} \\ \widehat{F}(\boldsymbol{x}, \boldsymbol{\mu}^{(j)}) &= F(\boldsymbol{x}, \boldsymbol{\mu}^{(j)}), & \forall\, (j, \boldsymbol{x}) \in [\![1, K]\!] \times \Omega \end{aligned}$$

III.2.a.2 Méthode d'interpolation empirique généralisée

La méthode d'interpolation empirique généralisée : *Generalized Empirical Interpolation Method* (GEIM) introduite et développée par [81, 83, 82] est une généralisation possible de l'EIM qui consiste à substituer l'opérateur d'interpolation $I_k[.]$ par un opérateur plus général qui s'applique à des fonctions possiblement moins régulières.

Définition 2. *Soit une fonction $f : \boldsymbol{x} \in \mathcal{D} \mapsto f(\boldsymbol{x}) \in \mathbb{R}$ définie sur un domaine $\mathcal{D} \subset \mathbb{R}^n$. L'approximation interpolante généralisée de f, notée $\bar{I}_K[f]$, associée à la base $\{\varphi_k\}_{k=1}^K$ et aux formes linéaires $\{\lambda_k\}_{k=1}^K$ s'appliquant à f, est égale à :*

$$\bar{I}_K[f] = \sum_{k=1}^{K} \alpha_k \varphi_k \qquad \textit{avec} \quad \{\alpha_k\}_{k=1}^K \in \mathbb{R}^K$$

et telle que (propriété d'interpolation généralisée) :

$$\lambda_i \left(\bar{I}_K[f]\right) = \lambda_i(f) \quad \forall\, i \in [\![1, K]\!]$$

On appelle $\bar{I}_K[.]$ l'opérateur d'approximation interpolante généralisée.

Cette approximation généralise celle de la définition 1 et lui est équivalente lorsque les formes linéaires ont pour image la valeur de la fonction argument en un point donné.

Le calcul de l'approximation interpolante généralisée $\bar{I}_K[f]$ consiste à trouver $\{\alpha_k\}_{k=1}^K$ solution du système linéaire de l'équation (III.10) :

$$\sum_{k=1}^{K} \alpha_k \lambda_i(\varphi_k) = \sigma_i(f) \quad \forall\, i \in [\![1, K]\!] \tag{III.10}$$

Remarque 5. *La remarque 2 vaut aussi pour l'approximation interpolante généralisée.*

En reprenant les notations de la section précédente, on considère l'ensemble de fonctions $\mathcal{F}$. L'objectif de la GEIM est de construire une base d'approximation $\{\varphi_k\}_{k=1}^K$ ainsi qu'un ensemble $\{\lambda_k\}_{k=1}^K$ choisi parmi un dictionnaire de formes linéaires sur $\mathcal{F}$ à partir desquels il est possible de construire des approximations interpolantes généralisées.

L'algorithme GEIM reprend l'algorithme EIM en remplaçant la propriété d'interpolation par une propriété d'interpolation généralisée. Plus spécifiquement, les évaluations de fonctions $f_{\boldsymbol{\mu}} \in \mathcal{F}$ en des points $\boldsymbol{x}$ sont remplacées par l'application de formes λ sur les fonctions $f_{\boldsymbol{\mu}}$ et l'opérateur d'approximation interpolante est remplacé par l'opérateur d'approximation interpolante généralisé.

III.2.a.3 Méthode d'interpolation empirique discrète

La méthode d'interpolation empirique discrète : *Discrete Empirical Interpolation Method* (DEIM) a été introduite par [26] pour approcher les termes non linéaires qui apparaissent dans des systèmes d'équations différentielles ordinaires issus de la discrétisation temporelle d'équations différentielles partielles paramétriques.

Dans formulation de l'EIM, les domaines Ω et $\mathcal{D}$ qui apparaissent dans la définition de la fonction F (Éq. (III.7)) et de l'ensemble $\mathcal{F}$ (Éq. (III.8)) peuvent être continus ou discrets. $\mathcal{D}$ peut être en particulier discret de sorte que $\mathcal{F}$ soit constitué d'un nombre fini de fonctions $f_{\boldsymbol{\mu}}$.

L'algorithme DEIM [26, Algorithme 1] correspond à la version discrète de l'algorithme EIM 2 appliquée à un ensemble $\mathcal{F}$ constitué d'un nombre fini de fonctions linéairement indépendantes.

On définit l'équivalent de l'approximation interpolante pour les matrices à la définition 1.

Définition 3. *Soit une matrice $A \in \mathbb{R}^{n\times m}$. L'approximation interpolante de A, notée $I_K[A]$, associée aux indices de lignes $\left\{\mathcal{I}^{(k)}\right\}_{k=1}^K$ et à la matrice de base $V \in \mathbb{R}^{n\times K}$ est égale à :*

$$I_K[A] = VW \qquad \textit{avec} \quad W \in \mathbb{R}^{K\times m}$$

et telle que (propriété d'interpolation) :

$$I_K\left[A\right](\mathcal{I},:) = A(\mathcal{I},:)$$

On appelle $I_K[.]$ l'opérateur d'approximation interpolante.

L'approximation interpolante existe lorsque $V(\mathcal{I},:)$ est inversible et est donnée par :

$$I_k\left[A\right] = V\left[V(\mathcal{I},:)\right]^{-1} A(\mathcal{I},:) \tag{III.11}$$

Algorithme 2 : Version matricielle de l'EIM

Données : Une matrice $A \in \mathbb{R}^{n\times m}$.

Résultat : Une matrice de base $V = [v_1;\ldots;v_K] \in \mathbb{R}^{n\times K}$ et un ensemble de lignes d'interpolation $\mathcal{I} = \left\{\mathcal{I}^{(1)},\ldots,\mathcal{I}^{(K)}\right\}$.

Initialisation

Définir :

$$\mathcal{J}^{(1)} = \underset{j\in[\![1,m]\!]}{\operatorname{argmax}} \left\|A(:,j)\right\|_2 \quad \text{et} \quad \mathcal{I}^{(1)} = \underset{i\in[\![1,n]\!]}{\operatorname{argmax}} \left|A(i,\mathcal{J}^{(1)})\right|$$

Définir la première fonction de base :

$$v_1 = \frac{A(:,\mathcal{J}^{(1)})}{A(\mathcal{I}^{(1)}),\mathcal{J}^{(1)})}$$

Pour $k \in [\![2,K]\!]$ **faire**

Définir :

$$\mathcal{J}^{(k)} = \underset{j\in[\![1,m]\!]}{\operatorname{argmax}} \left\|A(:,j) - I_{k-1}\left[A\right](:,j)\right\|_2$$

et

$$\mathcal{I}^{(k)} = \underset{i\in[\![1,n]\!]}{\operatorname{argmax}} \left|A(i,\mathcal{J}^{(k)}) - I_{k-1}\left[A\right](i,\mathcal{J}^{(k)})\right|$$

où $I_{k-1}[.]$ est l'opérateur d'interpolation associé à $\left\{\mathcal{I}^{(1)},\ldots,\mathcal{I}^{(k-1)}\right\}$ et la matrice $[v_1;\ldots;v_k]$.

Finalement, définir la $k^{\text{ème}}$ fonction de base :

$$v_k = \frac{A(:,\mathcal{J}^{(k)}) - I_{k-1}\left[A\right](:,\mathcal{J}^{(k)})}{A(\mathcal{I}^{(k)},\mathcal{J}^{(k)}) - I_{k-1}\left[A\right](\mathcal{I}^{(k)},\mathcal{J}^{(k)})}$$

Notons qu'il est également possible de modifier la manière de sélectionner les points

d'apprentissage et d'interpolation par exemple en choisissant :

$$(\mathcal{I}^{(k)}, \mathcal{J}^{(k)}) = \underset{(i,j)\in[\![1,n]\!]\times[\![1,m]\!]}{\operatorname{argmax}} |A(i,j) - I_{k-1}[A](i,j)|$$

La DEIM telle qu'elle a été introduite fait référence rigoureusement à l'algorithme 2 appliqué à des matrices de rang plein, cependant dans [26] Chaturantabut et Sorensen proposent une méthodologie générale pour approcher des matrices en couplant les méthodes POD et EIM. On présente la méthodologie dans le cadre de l'approximation de fonctions. Par partir d'un ensemble de fonctions $\mathcal{F}$, l'idée est de construire dans un premier temps une base POD tronquée $\{\varphi_k\}_{k=1}^K$ permettant d'approcher $\mathcal{F}$. Dans cette approche, l'algorithme EIM est ensuite appliqué essentiellement pour sélectionner des points d'interpolation pertinents $\left\{\boldsymbol{x}^{(k)}\right\}_{k=1}^K$. Au terme de l'algorithme, on est capable d'approcher n'importe quelle fonction $f \in \mathcal{F}$ par son approximation interpolante 1 $I_K[f]$ associée à $\{\varphi_k\}_{k=1}^K$ et aux points $\left\{\boldsymbol{x}^{(k)}\right\}_{k=1}^K$.

Le passage à l'approximation matricielle se fait naturellement. En termes discrets, la méthodologie DEIM consiste à approcher une matrice $A \in \mathbb{R}^{n\times m}$ par son approximation interpolante $I_K[A]$ associée à $V \in \mathbb{R}^{n\times K}$ (constituée des K premiers vecteurs singuliers à gauche de A) et aux lignes d'interpolation $\mathcal{I}$ (sélectionnées en appliquant l'algorithme EIM à la matrice V).

III.2.a.4 Best Points Interpolation Method

La méthode *Best Points Interpolation Method* (BPIM) introduite dans [86] est une variante de l'EIM développée spécifiquement pour modifier la manière de sélectionner les points d'interpolation $\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^K$. Comme pour la DEIM, la méthode suppose que l'on dispose préalablement d'une base de fonction $\{\varphi_k\}_{k=1}^K$ qui approche $\mathcal{F}$, typiquement une base POD tronquée ou une base calculée à l'aide de l'algorithme EIM. La BPIM définit trois ensembles de points d'interpolation différents : *Optimal interpolation points, Best interpolation points* et *Hierarchical interpolation points*. Ces ensembles de points sont définis comme des solutions de problèmes de minimisation moindres carrés.

Points d'interpolation optimale Les points d'interpolation optimale réalisent le minimum du critère :

$$\min \int_{\boldsymbol{\mu}\in\Omega} \|f_{\boldsymbol{\mu}} - I_K[f_{\boldsymbol{\mu}}]\|^2 \, d\boldsymbol{\mu} \qquad \text{où } I_K[.]\text{est associée aux points } \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^K$$

L'intégration ne permet pas en pratique une résolution exacte, le problème doit donc être modifié en remplaçant l'intégrale par une somme sur un ensemble de fonctions linéairement

indépendantes associées aux points $\left\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(m)}\right\}$. Le critère de minimisation devient :

$$\min \sum_{j=1}^{m} \left\| f_{\boldsymbol{\mu}^{(j)}} - I_K \left[f_{\boldsymbol{\mu}^{(j)}} \right] \right\|^2 \quad \text{où } I_K[.] \text{ est associée aux points } \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{K} \tag{III.12}$$

La résolution de ce problème reste difficile en pratique, car le critère est non linéaire, non convexe et possède potentiellement de nombreux minima locaux. Pour cette raison, un autre ensemble de points qui approche les points d'interpolation optimale est défini.

Meilleurs points d'interpolation Pour passer au problème de meilleurs points d'interpolation, on transforme le problème de l'équation (III.12) en remplaçant les $f_{\boldsymbol{\mu}^{(j)}}$ par leur meilleure approximation $f^K_{\boldsymbol{\mu}^{(j)}}$ sur la base des $\{\varphi_k\}_{k=1}^K$ données par :

$$f^K_{\boldsymbol{\mu}^{(j)}} = \sum_{k=1}^{K} \beta_k^j \varphi_k \quad \text{où} \quad \beta_k^j \in \mathbb{R}$$

D'autre part, on note :

$$I_K \left[f_{\boldsymbol{\mu}^{(j)}} \right] = \sum_{k=1}^{K} \alpha_k^j \varphi_k \quad \text{où} \quad \alpha_k^j \in \mathbb{R}$$

Notons que les α_k^j dépendent des points d'interpolations $\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{K}$. Compte tenu de l'orthogonalité de la base, le problème de l'équation (III.12) devient :

$$\min \sum_{j=1}^{m} \sum_{k=1}^{K} \left(\alpha_k^j - \beta_k^j \right)^2 \tag{III.13}$$

La résolution de ce problème de minimisation moindres carrés devient envisageable et [86] propose une résolution à l'aide d'un algorithme d'optimisation classique du type Levenberg–Marquardt.

Hierachical interpolation points Dans le cas où le problème de minimisation de l'équation (III.13) reste trop couteux à résoudre, une autre heuristique de résolution est proposée. L'heuristique fournit un ensemble de points appelés Hierachical interpolation points. Le principe consiste à sélectionner les points de manière itérative en ajoutant une à une les fonctions de base. Le premier point $\boldsymbol{x}^{(1)}$ est choisi afin de minimiser le critère :

$$\min \sum_{j=1}^{m} \left(\alpha_1^j - \beta_1^j \right)^2$$

Ensuite, à l'étape k, le point $\boldsymbol{x}^{(k)}$ est choisi afin de minimiser le critère :

$$\sum_{j=1}^{m}\sum_{l=1}^{k}\left(\alpha_l^j - \beta_l^j\right)^2$$

La résolution des problèmes successifs n'impliquant qu'une variable devient très rapide.

III.2.b Méthodes de projection lacunaire

Les méthodes de projection lacunaire proposent un autre moyen pour réduire la complexité des calculs en ligne. Dans la méthode POD classique, le calcul en ligne implique une résolution des équations qui tient compte de l'information sur l'ensemble du domaine de définition. L'idée de fondamentale des méthodes de projection lacunaire est de formuler les équations uniquement sur un domaine d'intégration partiel, dit domaine d'intégration réduit. La phase en ligne permet d'obtenir une approximation des solutions sur ce domaine réduit. Finalement, la connaissance d'un espace d'approximation des solutions (espace vectoriel de dimension réduite) sur le domaine complet permet via une étape de projection d'approcher les solutions sur le domaine complet.

Plusieurs méthodes de projection lacunaire s'inspirant du concept de gappy POD ont été développées ces dernières années. On présente dans la suite l'hyper-réduction [99] et la Missing Point Estimation [4, 124] mais plusieurs autres méthodes existent.

La méthode *Gauss-Newton with Approximate Tensors* (GNAT) [20, 21] consiste à coupler la notion de gappy POD avec la méthode de Gauss-Newton. L'algorithme de Gauss-Newton est une méthode itérative de résolution de problème de minimisation de résidus dont le calcul fait intervenir des termes non linéaires. Le couplage avec la gappy POD permet de limiter les calculs pour l'évaluation des termes non linéaires et donc de réduire la complexité algorithmique.

III.2.b.1 Gappy POD

La gappy POD introduite par [39] s'intéresse au problème de reconstruction d'une estimation d'une fonction uniquement sur la base d'une connaissance de ses valeurs sur un domaine partiel de son domaine de définition. La méthode a été initialement appliquée à la restauration d'images de visages ayant été partiellement masqués. Le principe est de chercher à approcher une fonction $f : \boldsymbol{x} \in \Omega \mapsto f(\boldsymbol{x}) \in \mathbb{R}$ par une fonction $\widehat{f}$ qui appartient à un espace vectoriel $\mathcal{V}_K$ engendré par les fonctions $\{\varphi_k\}_{k=1}^K$ en connaissant les valeurs de f uniquement sur un domaine réduit noté Ω_r. L'approximation lacunaire est définie comme la fonction $\widehat{f} \in \mathcal{V}_K$ qui minimise le critère :

$$\int_{\Omega_r}\left[f(\boldsymbol{x}) - \widehat{f}(\boldsymbol{x})\right]^2 d\boldsymbol{x}$$

$\widehat{f}$ constitue donc l'approximation la plus proche de f sur le domaine Ω_r au sens des moindres carrés.

On montre dans [39] que le problème de minimisation admet une solution donnée par :

$$\widehat{f} = \sum_{k=1}^{K} \alpha_k \varphi_k$$

où les $\{\alpha_k\}_{k=1}^{K} \in \mathbb{R}^K$ sont solutions du système matriciel :

$$M\boldsymbol{\alpha} = \boldsymbol{f}$$

avec

$$M(i,j) = \int_{\Omega_r} \varphi_i \varphi_j d\boldsymbol{x}$$
$$\boldsymbol{f}_i = \int_{\Omega_r} f \varphi_i d\boldsymbol{x}$$

Dans la formulation présentée dans [39], on part du principe que le domaine réduit Ω_r est imposé. Ce domaine correspond dans [39] au domaine complémentaire des pixels masqués. La base de fonction $\{\varphi_k\}_{k=1}^{K}$ est quant à elle générée en appliquant la POD à un ensemble de fonctions (snapshots) représentatives de l'espace à approcher. Dans [39], ces snapshots sont constitués de différentes images de visages humains.

Lorsque le domaine réduit n'est pas imposé, une étape fondamentale pour utiliser la gappy POD consiste à définir ce domaine. L'objectif est alors de choisir un domaine permettant de minimiser les erreurs d'approximation. Sélectionner le meilleur domaine réduit est un problème combinatoire difficile à résoudre d'un point de vue temps de calcul même lorsque la taille du problème est faible. En pratique, il est possible de le définir à l'aide de méthodes de sélection de points telles que l'EIM [103], la *Missing Point Estimation* (MPE) (Section III.2.b.3) ou encore des améliorations de la MPE [124].

III.2.b.2 Hyper-réduction

L'hyper-réduction introduite par [99] est une méthode de projection lacunaire qui s'inspire de la gappy POD pour construire des modèles réduits basés sur les éléments finis. La méthode a été appliquée en thermique et en mécanique des structures pour différents types de lois de comportement : plasticité, viscoplasticité [104], frittage [107], plasticité cristalline [101] et endommagement ductile [105].

L'hyper-réduction permet de construire un modèle réduit formulé via une projection de Petrov-Galerkin en utilisant une base POD des déplacements. La base POD est calculée

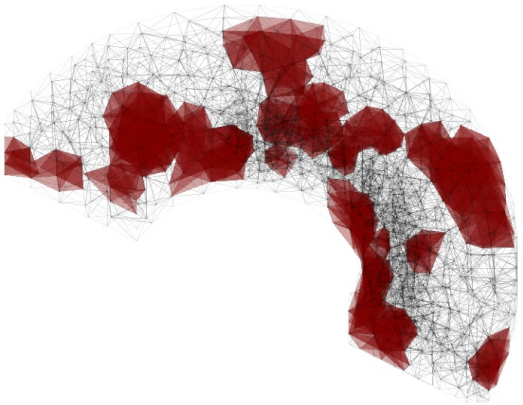


FIGURE III.1 – Exemple de domaine d'intégration réduit en rouge utilisé en hyper-réduction[3].

préalablement à l'aide de la Snapshot POD. La particularité de l'hyper-réduction est que les équations sont intégrées uniquement sur un domaine réduit. La formulation impose de compléter les conditions aux limites par des conditions de type Dirichlet [103] (à calculer) à l'interface entre le domaine réduit et le reste du domaine. Lors de la phase hors ligne, on construit d'autre part des bases POD pour toutes les quantités d'intérêts telles que les contraintes et les variables internes.

Lors de la phase en ligne, la résolution permet d'obtenir les solutions sur le domaine réduit. On obtient une approximation de toutes les variables mécaniques sur le domaine total en projetant les solutions du domaine réduit sur les bases POD. La résolution en ligne ne dépend plus de la taille du problème initial, mais uniquement de la taille du problème réduit.

La définition du domaine réduit est en général réalisée en agrégeant les points d'interpolation donnés par la DEIM appliquée aux bases POD des déformations et des contraintes. Il peut être complété en ajoutant des éléments où les chargements et déformations sont imposés. Il est d'usage d'ajouter les éléments qui entourent les éléments précédemment sélectionnés afin de rendre la résolution plus précise et robuste. Enfin, il est possible d'ajouter des éléments appartenant à une zone d'intérêt identifiée par un expert. On peut noter que le domaine réduit n'est pas nécessairement connexe.

La figure III.1 illustre un exemple de domaine réduit utilisé pour effectuer un calcul d'hyper-réduction sur un maillage éléments finis.

III.2.b.3 Missing Point Estimation

La Missing Point Estimation (MPE) proposé par [4] est appliquée pour construire des domaines réduits typiquement dans le cas de problèmes formulés par projection de Galerkin des équations sur une base POD. Lorsque les équations ont été discrétisées, la

3. Crédit David Ryckelynck

sélection d'un domaine réduit correspond à une sélection d'indices de lignes des vecteurs solutions. Pour une base POD associé à la matrice $V \in \mathbb{R}^{n\times K}$, [4] montre que l'erreur d'approximation gappy POD est contrôlée par la sensibilité à l'aliasing (*alias sensitivity*) qui correspond à :

$$\left\|(V(\mathcal{I},:)^T V(\mathcal{I},:))^{-1} - \mathbb{I}\right\|_2$$

où $\mathcal{I}$ correspond aux indices de lignes du domaine réduit.

Il est alors naturel de chercher l'ensemble d'indices $\mathcal{I}$ qui minimise cette quantité. Ce problème d'optimisation est combinatoire et donc sa résolution exacte n'est pas envisageable.

La MPE définit une heuristique gloutonne de sélection de lignes $\mathcal{I} = \left\{\mathcal{I}^{(1)}, \dots, \mathcal{I}^{(K)}\right\}$ visant à approcher cet optimum. Il est postulé dans [4] que la condition d'optimalité peut être relâchée en cherchant à minimiser à la place le conditionnement de $V(\mathcal{I},:)^T V(\mathcal{I},:)$. L'heuristique se déroule alors de la manière suivante. Considérons que l'ensemble de lignes $\mathcal{I}_{k-1} = \left\{\mathcal{I}^{(1)}, \dots, \mathcal{I}^{(k-1)}\right\}$ ait été sélectionné. La sélection du point suivant $\mathcal{I}^{(k)}$ consiste à chercher l'indice de la ligne parmi $[\![1, n]\!] \setminus \mathcal{I}_{k-1}$ dont qui minimise le conditionnement de $V\left(\mathcal{I}_{k-1} \cup \mathcal{I}^{(k)}, :\right)^T V\left(\mathcal{I}_{k-1} \cup \mathcal{I}^{(k)}, :\right)$. L'algorithme s'arrête lorsque le critère devient inférieur à un critère fixé préalablement. Cet algorithme reste couteux car de nombreuses décompositions SVD doivent être calculées à chaque itération [124]. Plus spécifiquement, la complexité de calcul est $\mathcal{O}(nsK^2)$ où s est le nombre d'indices sélectionnés. On peut trouver dans [124] une étude plus approfondie de l'algorithme MPE ainsi que des propositions d'améliorations.

III.3 Décompositions matricielles approchées

Les méthodes de réduction de modèle exploitent abondamment des outils d'algèbre linéaire. Dans cette section, on s'intéresse au problème d'approximation matricielle de rang faible ainsi qu'au problème très similaire de sélection de lignes et colonnes.

Soit une matrice $A \in \mathbb{R}^{n\times m}$ quelconque. Si cette matrice A possède une structure particulière, on peut s'attendre à ce qu'il suffise de stocker un nombre réduit d'éléments pour pouvoir la représenter exactement. Typiquement, si la matrice est de rang R, il est possible de l'écrire sous la forme :

$$A = BC \quad \text{avec} \quad B \in \mathbb{R}^{n\times R} \quad \text{et} \quad C \in \mathbb{R}^{R\times m} \tag{III.14}$$

Pour représenter la même information, la complexité de stockage qui était $\mathcal{O}(nm)$ pour la matrice A est réduite à $\mathcal{O}(nR + Rm)$. Si le rang $R \ll \max(n, m)$ cela permet un gain de mémoire important.

L'approximation matricielle de faible rang s'intéresse aux cas où le rang R de la matrice

n'est pas suffisamment petit pour qu'une représentation exacte de la forme donnée en équation (III.14) apporte une compression suffisante. L'objectif est d'approcher la matrice de référence par une matrice écrite sous la forme (Éq. (III.14)) en cherchant un compromis entre espace de stockage nécessaire et précision de l'approximation.

Ce problème classique a été largement étudié et un grand nombre de méthodes de construction existent. L'objectif de cette section est de présenter quelques-unes de ces méthodes en faisant le lien avec les approches de réduction de modèle.

III.3.a Notations matricielles

Soit la matrice $A \in \mathbb{R}^{n\times m}$. On définit la matrice de sélection de colonnes $Q \in \mathbb{R}^{m\times s}$ associée à l'ensemble d'indices de colonnes $\mathcal{J} = \left\{\mathcal{J}^{(1)}, \ldots, \mathcal{J}^{(s)}\right\} \subset [\![1, m]\!]$ de A est telle que :

$$Q(i,j) = \delta_{i,\mathcal{J}^{(j)}} \quad \forall\, (i,j) \in [\![1,m]\!] \times [\![1,s]\!] \tag{III.15}$$

avec δ le symbole de Kronecker.

On utilise la notation *deux-point* (telle qu'elle est utilisée en Matlab ou Python) [46, Sections 1.1.8 et 1.2.9] pour définir des extractions de sous-matrices. Le produit matriciel de A par Q donne la sous-matrice constituée des colonnes de A correspondant aux indices $\mathcal{J}$:

$$AQ = A(:, \mathcal{J})$$

De manière similaire, on définit la matrice de sélection de lignes $P \in \mathbb{R}^{n\times s}$ associée à l'ensemble d'indices de lignes $\mathcal{I} = \left\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(s)}\right\} \subset [\![1, n]\!]$ de A telle que :

$$P(i,j) = \delta_{i,\mathcal{I}^{(j)}} \quad \forall\, (i,j) \in [\![1,n]\!] \times [\![1,s]\!] \tag{III.16}$$

L'extraction de la sous-matrice constituée des lignes $\mathcal{I}$ est donnée par le produit de la transposée de P par A :

$$P^T A = A(\mathcal{I}, :)$$

La matrice identité de $\mathbb{R}^n$ est notée $\mathbb{I}_n$.

On donne une définition équivalente pour les matrices de sélection de lignes et colonnes :

$$Q = \mathbb{I}_m\,(:, \mathcal{J})$$
$$P = \mathbb{I}_n\,(:, \mathcal{I})$$

Remarque 6. *Les matrices de sélection de lignes et colonnes ont des colonnes orthogonales de norme* 2 *égale à* 1*.*

La norme de Frobenius est notée $\|.\|$ sans l'indice usuel $\mathcal{F}$ et la norme 2 est notée $\|.\|_2$. On rappelle que pour $A \in \mathbb{R}^{n\times m}$:

$$\|A\| = \sqrt{\sum_{i,j=1}^{n,m} A(i,j)^2} \quad \text{et} \quad \|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$

III.3.b Décomposition en valeurs singulières

La décomposition en valeurs singulières (*Singular Value Decomposition* : SVD) [46] d'une matrice $A \in \mathbb{R}^{n\times m}$ est définie comme la décomposition :

$$A = V\Sigma W^T$$

pour laquelle $V \in \mathbb{R}^{n\times n}$ est unitaire ($V^TV = VV^T = \mathbb{I}_n$), $W \in \mathbb{R}^{m\times m}$ est unitaire et $\Sigma \in \mathbb{R}^{n\times m}$ est diagonale composée d'éléments positifs σ_k.

Les éléments diagonaux de Σ correspondent aux valeurs singulières, notées σ_k, de A. En choisissant d'ordonner les σ_k de manière décroissante, Σ est définie de manière unique.

On appelle :

— V la matrice des vecteurs singuliers à gauche ;

— W la matrice des vecteurs singuliers à droite ;

— Σ la matrice des valeurs singulières.

Lorsque la matrice A de rang R on peut réécrire la décomposition SVD de manière exacte en tronquant les matrices. On redéfinit V et W en sélectionnant uniquement les R premières colonnes et Σ en sélectionnant la sous-matrice des R premières valeurs singulières. De cette manière les matrices V et W sont définies de manière unique au signe près de leur première composante. Notons que, $V^TV = W^TW = \mathbb{I}_R$ mais $VV^T \neq \mathbb{I}_n$ et $WW^T \neq \mathbb{I}_m$

Les valeurs singulières non nulles sont notées :

$$\sigma_1(A) \geq \cdots \geq \sigma_R(A) > 0$$

On note aussi $\sigma_{\max}(A)$ et $\sigma_{\min}(A)$ respectivement la plus grande et plus petite des valeurs singulières non nulles.

La norme de Frobenius et la norme 2 s'expriment d'autre part, en fonction des valeurs singulières :

$$\|A\| = \sqrt{\sum_{i=1}^{R} \sigma_i(A)^2} \quad \text{et} \quad \|A\|_2 = \sigma_{\max}(A)$$

La SVD est une méthode classique et robuste pour construire des approximations de faible rang. Une manière naturelle de formaliser le problème d'approximation de rang faible consiste à rechercher la matrice solution du problème :

$$A^r = \underset{X \in \mathbb{R}^{n\times m}, \mathrm{rg}(X)=r}{\operatorname{argmin}} \|A - X\|$$

La solution de ce problème existe et est donnée par la troncature de la SVD au rang r. Plus précisément, A^r est obtenue en tronquant les matrices V et W aux r premières colonnes et la matrice Σ aux r premières valeurs singulières :

$$A^r = V^r \Sigma^r (W^r)^T$$

avec $V^r = V(:, I^r)$, $W^r = W(:, I^r)$ et $\Sigma^r = \Sigma(I^r, I^r)$ où $I^r = [\![1, r]\!]$.

On définit la matrice d'erreur de troncature $E^r = A - A^r$ et l'erreur de troncature $\epsilon^r = \|E^r\|$. On peut montrer que l'erreur de troncature est donnée par :

$$\epsilon^r = \sqrt{\sum_{k=r+1}^{R} \sigma_k^2} \tag{III.17}$$

Notons que l'erreur de troncature fait parfois référence au rapport $\frac{\|E^r\|}{\|A\|}$.

Calcul de la SVD La SVD a été largement étudiée et des algorithmes performants sont implémentés dans toutes les librairies d'algèbre de référence. La complexité du calcul SVD étant $\mathcal{O}\left(\min(mn^2, m^2n)\right)$, son calcul pour des matrices de grande taille peut devenir irréalisable. D'autre part, le calcul d'une SVD tronquée donnant une approximation suffisamment fine demande essentiellement le même nombre d'opérations et donc n'est pas un moyen de réduire la complexité de calcul.

Dans certaines applications, on s'intéresse avant tout au calcul ou à l'estimation l'espace engendré par les colonnes de A. Or les colonnes de V constituent une base qui engendre cet espace et les valeurs singulières σ_k permettent de quantifier l'importance de chaque vecteur (voir erreur de troncature de l'équation (III.17)). Dans ces circonstances, on cherche donc avant tout à obtenir la matrice des vecteurs singuliers à gauche V et la matrice des valeurs singulières Σ (ou leur troncature).

Une manière de réduire la complexité du calcul de la SVD et d'obtenir une estimation des matrices V et Σ est d'appliquer l'équivalent de la Snapshot POD discutée dans la section III.1.a. La méthode consiste à calculer la SVD complète d'une sous-matrice $\widetilde{A}$ constituée d'un échantillonnage de colonnes de A. Soit $\tilde{\mathcal{J}} \subset [\![1, m]\!]$ une liste d'indices

correspondants à des colonnes de A, on définit par :

$$\widetilde{A} = A(:, \tilde{\mathcal{J}})$$

La SVD appliquée à $\widetilde{A}$ donne :

$$\widetilde{A} = \widetilde{V}\widetilde{\Sigma}\widetilde{W}^T$$

Si l'échantillonnage de colonnes est suffisamment riche tel qu'il permette de capturer suffisamment de modes de la matrice A, on peut montrer que les matrices $\widetilde{V}$ et $\widetilde{\Sigma}$ approchent les matrices V et Σ. Cette méthode permet ainsi d'obtenir une estimation de V et Σ en réduisant les coûts de calcul.

Finalement, une approximation de la matrice A est donnée par :

$$A \simeq \widetilde{V}\widetilde{V}^T A$$

En algèbre linéaire, les algorithmes aléatoires [85] (*Randomized algorithms*) constituent une classe d'algorithmes permettant entre autres de construire des approximations matricielles basées sur des parcours partiels aléatoires des matrices de référence. Des algorithmes [69] qui s'appuient sur ces concepts d'échantillonnages aléatoires permettent de calculer la SVD de matrices de grande taille.

III.3.c Sélection de colonnes/lignes

Soit une matrice $A \in \mathbb{R}^{n\times m}$. On considère le problème de sélection d'une sous-matrice $A(:, \mathcal{J})$ telle que l'espace engendré par les colonnes $\mathcal{J}$ approche le mieux possible l'espace engendré par les colonnes de A. On peut formuler de manière équivalente le problème en considérant des lignes. Ces problèmes, qui apparaissent naturellement dans le cadre de la construction d'approximation de faible rang et dans le cadre de l'estimation du rang d'une matrice, ont été amplement étudiés en algèbre linéaire (*Column subset selection problem*).

Sur des bases théoriques, [50] justifie qu'un bon candidat consiste à choisir la sous-matrice $A(:, \mathcal{J}) \in \mathbb{R}^{n\times n}$ de volume maximal [48]. Dans [25] pour quantifier la qualité de la sélection, 3 critères liés aux valeurs singulières de la sous-matrice sont comparés à la maximisation du volume de la sous-matrice $A(:, \mathcal{J})$:

— Minimisation de $\sigma_{\max}(A(:, \mathcal{J}))$;
— Maximisation de $\sigma_{\min}(A(:, \mathcal{J}))$;
— Minimisation du conditionnement $\frac{\sigma_{\max}(A(:,\mathcal{J}))}{\sigma_{\min}(A(:,\mathcal{J}))}$;

Il est montré que tous ces problèmes sont NP-difficiles.

Par conséquent, quel que soit le critère de sélection choisi, la méthode de résolution pratique ne pourra qu'être basée sur une heuristique visant à s'approcher de la solution

optimale. On présente ici quelques algorithmes déterministes permettant de réaliser ou d'améliorer une sélection de lignes/colonnes lorsque l'ensemble des éléments de la matrice sont accessibles.

Remarque 7. *Le problème de sélection de lignes/colonnes est intimement lié à la sélection de domaines réduits qui intervient en réduction de modèle. En effet, comme on peut le voir dans la section précédente, les méthodes EIM, DEIM, BPIM et MPE formulées de manière discrète consistent essentiellement en une sélection de lignes de matrices.*

III.3.c.1 Maxvol

L'algorithme 3 Maxvol est une heuristique proposée par [47] visant à trouver la sous-matrice de volume maximale. Le volume d'une matrice carrée correspond à la norme de son déterminant. On qualifie de dominante une sous-matrice $A(\mathcal{I},:)$ de rang plein telle que tous les éléments de $AA(\mathcal{I},:)^{-1}$ sont de module inférieur à 1. On peut montrer que la sous-matrice de volume maximale est dominante.

Le principe de l'algorithme Maxvol est de chercher une matrice dominante de manière itérative par substitution de lignes. La sous-matrice sélectionnée par l'algorithme dépend fortement de l'initialisation constituée d'une première sélection de lignes. Typiquement, cette initialisation pourra être faite via un autre algorithme de sélection de lignes/colonnes (voir algorithmes suivants). L'algorithme Maxvol peut donc être interprété comme un algorithme d'amélioration d'une sélection de lignes via une maximisation du volume de la sous-matrice.

Algorithme 3 : Maxvol

Données : Une matrice $A \in \mathbb{R}^{n\times r}$ de rang r.
Résultat : Un ensemble de lignes $\mathcal{I}$ tel que $A(\mathcal{I},:)$ soit dominante.
Initialisation
Choisir un ensemble de r lignes $\mathcal{I}$ tel que $A(\mathcal{I},:) \in \mathbb{R}^{r\times r}$ soit inversible. Soit $Z \in \mathbb{R}^{n\times r}$ tel que :

$$A = ZA(\mathcal{I},:)$$

et trouver $z_{\max}$ l'élément de module maximal de Z associé aux indices $(i_{\max}, j_{\max})$.
Tant que $|z_{\max}| > 1$ **faire**
Permutation : $\mathcal{I}^c(i_{\max}) \leftrightarrow \mathcal{I}(j_{\max})$ Soit $Z \in \mathbb{R}^{n\times r}$ tel que :

$$A = ZA(\mathcal{I},:)$$

et trouver $z_{\max}$ l'élément de module maximal de Z associé aux indices $(i_{\max}, j_{\max})$.

III.3.c.2 Décomposition QR avec pivotage de colonnes

La décomposition QR [46] est une factorisation matricielle qui s'écrit $A = QR$ où $Q \in \mathbb{R}^{n\times n}$ est une matrice orthogonale et $R \in \mathbb{R}^{n\times m}$ est une matrice triangulaire supérieure. La décomposition QR joue un rôle important pour de l'estimation de rang de matrices.

Des implémentations triviales de cette factorisation, comme l'algorithme de Gram-Schmidt, sont numériquement instables. Plusieurs alternatives stables existent [54] comme l'algorithme de Gram-Schmidt modifié ou la décomposition QR via l'utilisation des matrices de Householder.

Une autre alternative, l'algorithme 4 de décomposition QR avec pivotage de colonnes (*colonnes pivoting*) proposé par [19], peut être utilisée comme un algorithme de sélection de colonnes. En effet, les r premiers éléments de la liste de permutations piv qui apparait dans l'algorithme 4 correspondent à des colonnes linéairement indépendantes. D'autre part, la procédure peut être interprétée [54] comme un algorithme de sélection de colonnes cherchant à maximiser le volume de la sous-matrice associée.

Algorithme 4 : Décomposition QR de Householder avec pivotage de colonnes [46, Algorithme 5.4.1]

Données : Une matrice $A \in \mathbb{R}^{n\times m}$ de rang r.
Résultat : Une matrice orthogonale $Q \in \mathbb{R}^{n\times n}$, une matrice triangulaire supérieure $R \in \mathbb{R}^{n\times m}$ et une matrice de permutation $\Pi \in \mathbb{R}^{n\times n}$ telle que $\Pi A = QR$.

Initialisation

Soit $A^0 = A$.

Pour $k = [\![1, r]\!]$ **faire**

Soit $p = \text{argmax}\, A^{k-1}(:, p)^2$. Permutation des colonnes p et k :

$$A^{k-1}(:, p) \leftrightarrow A^{k-1}(:, k)$$

et $piv(k) = p$. Soit la matrice de Householder $H^k \in \mathbb{R}^{n\times n}$ telle que :

$$H^k = \begin{pmatrix} \mathbb{I}_{k-1} & 0 \\ 0 & \mathbb{I}_{n-k+1} - 2\dfrac{v_k v_k^T}{v_k^T v_k} \end{pmatrix}$$

avec $v_k = A^{k-1}(k : n, k) - \left\| A^{k-1}(k : n, k) \right\| e_1$.

Définir $A^k = H^k A^{k+1}$.

Finalisation

Soit $Q = (H^1)^T \dots (H^r)^T$, $R = A^r$ et Π la matrice de sélection de colonnes associée à piv. On a alors $\Pi A = QR$.

Remarque 8. *L'algorithme 5.4.1 dans [46] s'applique en principe à des matrices possédant plus de lignes que de colonnes, mais la stratégie de calcul s'applique de manière*

similaire à des matrices possédant plus de colonnes que de lignes.

III.3.c.3 Q-DEIM

On a montré dans la section III.2.a.3 que l'algorithme DEIM consistait à appliquer l'algorithme EIM à une matrice V de rang plein dans le but de sélectionner un ensemble de lignes afin de construire par la suite des approximations interpolantes.

La méthode introduite sous le nom de Q-DEIM [37] est une méthode alternative de sélection de points, présentés comme une procédure gloutonne de maximisation du volume. La supériorité de cette approche par rapport à la DEIM est établie vis-à-vis des erreurs d'approximations.

La Q-DEIM est définie explicitement comme l'algorithme 4 de décomposition QR avec pivotage de colonnes appliqué à la transposée de la matrice V.

Algorithme 5 : Q-DEIM [37]

Données : Une matrice $U \in \mathbb{R}^{n\times r}$ de rang plein avec $r \leq n$.
Résultat : Un ensemble $\mathcal{I} = \left\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(r)}\right\} \subset [1:n]$ r d'indices de lignes.
Définir $A = U^T$ et calculer la décomposition QR (Algorithme 4) de A. On obtient :

$$A\Pi = QR \quad \text{avec} \quad \Pi \in \mathbb{R}^{n\times n},\ Q \in \mathbb{R}^{r\times r} \text{ et } R \in \mathbb{R}^{r\times n}$$

où Π est une matrice de permutation, Q une matrice orthogonale et R une matrice triangulaire supérieure.
Définir la liste d'indices de colonnes $\mathcal{I}_\Pi$ telles que :

$$A\Pi = A(:, \mathcal{I}_\Pi)$$

Définir $\mathcal{I}$ comme l'ensemble des r premiers indices de la liste $\mathcal{I}_\Pi$:

$$\mathcal{I} = \mathcal{I}_\Pi(:r)$$

Notons que d'autres approches ont été développées telles que l'application d'algorithme génétique pour améliorer la sélection DEIM [108]. Enfin, dans d'autres communautés on peut trouver des problèmes de sélection de colonnes formalisés sous la forme de parcours de graphes et résolus par des algorithmes de recherche tels que l'heuristique A* (*A star*) [3].

III.3.d Décomposition skeleton et pseudo-skeleton

La décomposition skeleton, *Skeleton Decomposition* en anglais, est une décomposition matricielle introduite par [121, 49]. Le principe de base est qu'une matrice de rang R peut être représentée exactement en connaissant uniquement un sous-ensemble de R colonnes et R lignes à condition que la sous-matrice constituée de ces R lignes et R colonnes soit inversible.

Formellement, soit la matrice $A \in \mathbb{R}^{n\times m}$ de rang R. Soit $\mathcal{I} = \left\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(R)}\right\}$ une liste d'indices de lignes et $\mathcal{J} = \left\{\mathcal{J}^{(1)}, \ldots, \mathcal{J}^{(R)}\right\}$ une liste d'indices de colonnes tels que la sous-matrice $A(\mathcal{I}, \mathcal{J})$ soit inversible, alors la décomposition skeleton est donnée par :

$$A = A(:, \mathcal{J})\left[A(\mathcal{I}, \mathcal{J})\right]^{-1} A(\mathcal{I}, :)$$

La décomposition skeleton n'est pas unique puisqu'il existe potentiellement plusieurs ensembles $\mathcal{I}$ et $\mathcal{J}$ qui satisfont la propriété d'inversibilité.

Sous cette forme, la représentation est exacte, néanmoins il est possible d'utiliser cette décomposition pour construire des approximations matricielles en sélectionnant un nombre de lignes et colonnes inférieures au rang de la matrice A. Ce type d'approximation a été largement développé et est désigné de manière équivalente par les termes : décomposition pseudo-skeleton [121, 49] (*Pseudo-Skeleton Decomposition* : PSD), approximation croisée [8] (*cross-approximation*), décomposition CUR [115, 12, 36] (C pour *columns* et R pour *rows*) ou encore approximation Sparse Column-Row (SCR) [115, 12].

La décomposition PSD $\widetilde{A}$ associée aux lignes $\mathcal{I}$ et aux colonnes $\mathcal{J}$ est interpolante en ces lignes et colonnes, i.e :

$$\widetilde{A}(\mathcal{I}, :) = A(\mathcal{I}, :) \quad \text{et} \quad \widetilde{A}(:, \mathcal{J}) = A(:, \mathcal{J})$$

Remarque 9. *On montre que la PSD de rang r de A associée aux indices de lignes $\mathcal{I}$ et colonnes $\mathcal{J}$ correspond strictement à l'approximation interpolante matricielle (Définition 3) $I_r[A]$ associée aux lignes d'interpolation $\mathcal{I}$ et à la matrice $V = A(:, \mathcal{J})$.*

Dans un cas particulier de PSD, deux résultats [48, 109] donnent une borne sur l'erreur d'approximation. Pour une approximation pseudo-skeleton T_{psd} (Éq. (V.6)) associée aux r indices de lignes et colonnes $\mathcal{I}$ et $\mathcal{J}$ tels que le volume de la sous-matrice $A(\mathcal{I}, \mathcal{J})$ soit maximal on a :

$$\|A - T_{\text{psd}}\|_C \leq (r+1) \min_{\text{rg}\, X \leq r} \|A - X\|_2$$
$$\|A - T_{\text{psd}}\|_C \leq (r+1)^2 \min_{\text{rg}\, X \leq r} \|A - X\|_C$$

où la norme de Chebyshev est définie par $\|A\|_C = \max_{i,j} |A(i,j)|$.

Notons que $\min_{\text{rg}\, X \leq r} \|A - X\|_2 = \sigma_{r+1}(A)$.

III.3.d.1 Méthode de construction

Le calcul d'une représentation skeleton exacte d'une matrice de rang R est un problème facile. En effet, il s'agit simplement de sélectionner un ensemble de R colonnes $\mathcal{J}$ linéairement indépendantes de la matrice A puis de sélectionner un ensemble de R lignes $\mathcal{I}$

linéairement indépendantes de la matrice $A(:, \mathcal{J})$. La sous-matrice $A(\mathcal{I}, \mathcal{J})$ est inversible et permet de définir facilement la représentation skeleton exacte.

De manière équivalente, construire une approximation PSD arbitraire de rang r est évident. Cependant la recherche d'une approximation PSD pertinente consiste à rechercher les ensembles de lignes et colonnes qui minimisent la norme de l'erreur d'approximation. De la même manière que les problèmes de sélection de lignes ou de sélection de colonnes (Section III.3.c), ce problème est combinatoire, sa résolution exacte est donc inenvisageable.

Les références citées en début de section III.3.d ainsi que [36, 16, 113] proposent différentes heuristiques de résolution pour construire des approximations PSD. On évoque dans les paragraphes suivants deux méthodes alternatives.

Algorithme alterné Maxvol Dans [120] un algorithme exploitant le principe de maximisation du volume est proposé pour construire des approximations PSD de matrice $A \in \mathbb{R}^{n \times m}$. La première étape consiste à appliquer l'algorithme Maxvol (Algorithme 3) sur une sous-matrice $A(:, \mathcal{J})$ constituée de colonnes linéairement indépendantes afin d'obtenir un ensemble d'indices de lignes $\mathcal{I}$. La deuxième étape consiste à appliquer l'algorithme Maxvol sur la transposée de la sous-matrice $A(\mathcal{I}, :)$ afin de sélectionner un ensemble d'indices de colonnes $\mathcal{J}$. La répétition de ces deux étapes permet de sélectionner des sous-matrices $A(\mathcal{I}, \mathcal{J})$ dont le volume ne fait que diminuer. Avec un choix initial pertinent de colonnes, on peut espérer converger vers des sélections de lignes et colonne définissant une approximation PSD précise.

En général, le rang de la matrice A n'est pas connu. En choisissant initialement un nombre R de colonnes supérieur au rang de A, les opérations numériques, comme le calcul de $A(:, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}$, deviennent instables. L'algorithme [93, Algorithme 2] propose de calculer à la première étape la décomposition QR de $A(:, \mathcal{J}) = QR$ puis d'appliquer l'algorithme Maxvol sur la matrice $Q \in \mathbb{R}^{n \times R}$ pour sélectionner les indices de lignes $\mathcal{I}$. Le calcul de $A(:, \mathcal{J})A(\mathcal{I}, \mathcal{J})^{-1}$ revient donc au calcul de $QQ(\mathcal{I}, :)^{-1}$ qui peut être effectué de manière stable. On utilise la même idée pour la deuxième étape . Finalement, l'algorithme permet de construire des approximations de A sans avoir à connaitre initialement le rang de la matrice. Notons que lorsque le rang est initialement surestimé, l'approximation résultante ne correspond pas exactement à une décomposition pseudo-skeleton.

Adaptive cross approximation La remarque 9 fait le lien entre la PSD et l'approximation interpolante matricielle. Par conséquent, la version discrète de l'algorithme EIM 2 peut être interprétée comme une heuristique de construction d'approximation PSD.

Au cours de l'algorithme EIM, on doit calculer une approximation interpolante associée à la base et l'ensemble de lignes courant. En réalité, le passage de l'approximation de l'itération k à l'approximation de l'itération $k+1$ revient à une mise à jour de rang 1 de

l'approximation. La construction de la matrice d'interpolation $I_k[A]$ n'a donc pas à être recalculée entièrement à toutes les itérations.

Cette observation permet de réaliser que pour des fonctions appartenant à un espace vectoriel de dimension finie, l'algorithme EIM est essentiellement similaire [10] à l'algorithme 6 Adaptive Cross Approximation (ACA) introduit par [8, 9] en algèbre linéaire.

Algorithme 6 : ACA

Données : Une matrice $A \in \mathbb{R}^{n\times m}$ et une tolérance d'approximation ϵ_{tol}.
Résultat : Un ensemble d'indices de lignes $\mathcal{I}$ et de colonnes $\mathcal{J}$.
Initialisation
Définir $A^0 = A$ et $k = 1$.
Tant que $\left\|A^k\right\| > \epsilon_{\text{tol}}$ **faire**
Choisir un indice de colonne $\mathcal{J}^{(k)}$ tel que $A^{k-1}(:, \mathcal{J}^{(k)})$ soit non nul et l'indice de ligne $\mathcal{I}^{(k)}$ tel que :

$$\mathcal{I}^{(k)} = \underset{i\in[\![1,n]\!]}{\operatorname{argmax}} A^{k-1}(i, \mathcal{J}^{(k)}) \tag{III.18}$$

Approximation de rang k
Définir l'approximation de rang 1, $\widetilde{A}^k$ telle que :

$$\widetilde{A}^k = \frac{A^{k-1}(:, \mathcal{J}^{(k)})A^{k-1}(\mathcal{I}^{(k)}, :)}{A^{k-1}(\mathcal{I}^{(k)}, \mathcal{J}^{(k)})}$$

et définir la matrice de résidu $A^k = A^{k-1} - \widetilde{A}^k$.
Incrémentation $k := k + 1$.
Finalisation
L'approximation ACA est donnée par :

$$\widetilde{A} = \sum_{k=1}^{R} \frac{A^{k-1}(:, \mathcal{J}^{(k)})A^{k-1}(\mathcal{I}^{(k)}, :)}{A^{k-1}(\mathcal{I}^{(k)}, \mathcal{J}^{(k)})}$$

Des stratégies alternatives sont possibles pour la sélection de l'indice de lignes et l'indice de colonne. L'indice de colonne peut être choisi comme celui dont la colonne est de norme maximale :

$$\mathcal{J}^{(k)} = \underset{j\in[\![1,m]\!]}{\operatorname{argmax}} \left\|A^{k-1}(:, j)\right\|$$

Une autre possibilité consiste à sélectionner simultanément (comme dans l'algorithme EIM discrète) les indices de ligne et colonne tels que :

$$(\mathcal{I}^{(k)}, \mathcal{J}^{(k)}) = \underset{i,j\in[\![1,m]\!]\times[\![1,n]\!]}{\operatorname{argmax}} |A^{k-1}(i, j)|$$

III.3.e Approximation gappy

La notion d'approximation gappy POD a été présentée dans le cadre continu dans la section III.2.b.1. On introduit ici la notion d'approximation gappy qui étend l'approximation gappy POD dans le cadre discret.

Définition 4. *Soit une matrice $A \in \mathbb{R}^{n\times m}$. L'approximation gappy, notée $I^G[A]$, associée aux L indices de lignes $\mathcal{I} = \left\{\mathcal{I}^{(l)}\right\}_{l=1}^{L} \subset [1:n]$ et à la matrice de base $V \in \mathbb{R}^{n\times K}$ avec $K \leq L$ correspond à la solution du problème de minimisation moindres carrés suivant :*

$$I^G[A] = \underset{X=VW, W\in\mathbb{R}^{K\times m}}{\operatorname{argmin}} \left\| P^T A - P^T X \right\| \tag{III.19}$$

où $P \in \mathbb{R}^{n\times L}$ est la matrice de sélection associée aux L indices de lignes $\mathcal{I}$.

Si $V^T P P^T V$ est inversible, le problème admet une unique solution donnée explicitement par :

$$I^G[A] = V\left[V^T P P^T V\right]^{-1} V^T P P^T A$$

On peut donner une borne sur la norme de l'erreur d'approximation gappy [124] :

$$\left\| A - I^G[A] \right\| \leq \frac{1}{\sigma_{\min}(P^T V)} \left\| A - VV^T A \right\| \tag{III.20}$$

III.4 Décompositions tensorielles

Dans cette section, la notion de tenseurs fait uniquement référence à des tableaux multidimensionnels aussi appelés hypermatrices [78]. Lorsque le nombre de dimensions d'un tenseur est grand, le stockage de l'ensemble de ses éléments peut devenir inenvisageable. Les décompositions tensorielles généralisent aux tenseurs le concept de décompositions matricielles. Lorsque les tenseurs sont de rang faible, elles permettent de les représenter exactement dans des formats compacts. Ces factorisations permettent donc potentiellement de résoudre le problème de la complexité de stockage. Cette section permet d'introduire les quatre décompositions tensorielles les plus couramment utilisées, à savoir : la décomposition canonique, la décomposition de Tucker, la décomposition de Tucker hiérarchique, la décomposition en train de tenseurs. Chaque décomposition exploitant des structures de stockage différentes donnera selon les tenseurs, des performances de compression différentes.

L'évaluation d'un élément d'une décomposition tensorielle se fait non pas simplement par un accès mémoire, mais par une suite d'opérations élémentaires. Puisque les opérations impliquées sont simples et en nombre relativement réduit, les moyens de calcul actuels

permettent d'avoir dans les faits un accès temps réel aux éléments. Cette propriété est particulièrement importante lorsqu'il s'agit d'utiliser ces représentations dans le cadre de modèle de substitution.

On fait référence à [70, 52, 55] pour une revue détaillée des formats de décompositions tensorielles.

III.4.a Notations tensorielles

Un tenseur d'ordre d

$$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \tag{III.21}$$

(dénoté par une majuscule calligraphique grasse) fait référence à un tableau multidimensionnel (en anglais *multiway array*) possédant d axes (ou dimensions). La taille (*shape* en anglais) du tenseur $\boldsymbol{\mathcal{A}}$ désigne le d-uplet $(n_1, \ldots, n_d)$. On appelle n_i la taille de la dimension i.

L'élément du tenseur $\boldsymbol{\mathcal{A}}$ identifié par l'indice $(i_1, \ldots i_d) \in I_1 \times \cdots \times I_d$ est noté :

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) \in \mathbb{R}$$

où $I_k = [\![1, n_k]\!]$ pour $k \in [\![1, d]\!]$.

La notation *crochet* permet de définir un tenseur comme la collection de ses éléments de la manière suivante :

$$\boldsymbol{\mathcal{A}} = [\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d),\ \forall\, (i_1, \ldots, i_d) \in I_1 \times \cdots \times I_d]$$

Lorsque les intervalles de définition I_k sont évidents, on peut omettre de les indiquer et la notation devient :

$$\boldsymbol{\mathcal{A}} = [\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d)]$$

On peut de plus définir un tenseur en spécifiant les intervalles de variations associés à chaque dimension :

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}(I_1, \ldots, I_d)$$

Avec cette notation, on peut définir à partir de $\boldsymbol{\mathcal{A}}$, des extractions de sous-tenseurs par sélections d'indices associées aux différentes dimensions. Soit $\mathcal{I}_p \subset I_p$ une sélection de m_p indices de la dimension p. Le sous-tenseur composé des éléments de $\boldsymbol{\mathcal{A}}$ uniquement pour

les indices $\mathcal{I}_p$ est noté :

$$\boldsymbol{\mathcal{A}}(I_1, \ldots, I_{p-1}, \mathcal{I}_p, I_{p+1}, \ldots, I_d) \in \mathbb{R}^{n_1 \times \cdots \times n_{p-1} \times m_p \times n_{p+1} \times \cdots \times n_d}$$

On utilise également la notation *deux-point* pour signifier la sélection de l'ensemble des indices d'une dimension.

Il est intéressant en pratique de manipuler les tenseurs sous des représentations de tailles différentes. On définit l'opérateur de *réarrangement tensoriel* :

$$\underset{(n_1 \times \cdots \times n_d) \mapsto (m_1 \times \cdots \times m_{d'})}{\text{reshape}}$$

qui prend en entrée un tenseur $\boldsymbol{\mathcal{A}}$ de taille $(n_1 \times \cdots \times n_d)$ et retourne un tenseur $\boldsymbol{\mathcal{B}}$ de taille $(m_1 \times \cdots \times m_{d'})$ appelé *tenseur réarrangé*. Le tenseur réarrangé $\boldsymbol{\mathcal{B}}$ correspond à une réindexation du tenseur $\boldsymbol{\mathcal{A}}$ dont la correspondance entre les éléments est donnée par :

$$\boldsymbol{\mathcal{B}}(j_1, \ldots, j_{d'}) = \boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) \quad \text{avec} \quad \sum_{k=1}^{d} (i_k - 1) \prod_{l=1}^{k-1} n_l = \sum_{k'=1}^{d'} (j_{k'} - 1) \prod_{l'=1}^{k'-1} m_{l'} \tag{III.22}$$

L'opérateur $\text{reshape}_{(n_1 \times \cdots \times n_d) \mapsto (m_1 \times \cdots \times m_{d'})}$ est défini si et seulement si $\prod_{k=1}^{d} n_k = \prod_{k'=1}^{d'} m_{k'}$. Son opérateur réciproque est $\text{reshape}_{(m_1 \times \cdots \times m_{d'}) \mapsto (n_1 \times \cdots \times n_d)}$.

En l'absence d'ambiguïté et pour alléger les notations, on ne précisera pas toujours la taille d'entrée de sorte que l'opérateur $\text{reshape}_{(n_1 \times \cdots \times n_d) \mapsto (m_1 \times \cdots \times m_{d'})}$ sera noté $\text{reshape}_{m_1 \times \cdots \times m_{d'}}$.

Remarque 10. *L'équation* (III.22) *suggère que lors du passage d'un multi-indice [55] à un indice linéaire, on choisit la convention où l'indice le plus rapide est l'indice le plus à gauche. Il s'agit donc de la convention « Fortran » plutôt que la convention « C ».*

Un réarrangement particulier consiste à associer des dimensions consécutives. Pour un tenseur $\boldsymbol{\mathcal{A}}$ (Éq. (III.21)), l'association des q dimensions consécutives $p, (p+1), \ldots, (p+q-1)$ donne un tenseur d'ordre $(d - q + 1)$:

$$\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times \cdots \times n_{p-1} \times \overline{n}_p \times n_{p+q} \times \cdots \times n_d} \quad \text{avec} \quad \overline{n}_p = \prod_{k=p}^{p+q-1} n_k \tag{III.23}$$

La relation entre les éléments des deux tenseurs est donnée par :

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) = \boldsymbol{\mathcal{B}}\,(i_1, \ldots, i_{p-1}, \overline{\imath}_p, i_{p+q}, \ldots, i_d) \quad \text{avec} \quad \overline{\imath}_p - 1 = \sum_{k=p}^{p+q-1} (i_k - 1) \prod_{l=p}^{k-1} n_l$$

En utilisant la notation crochet, l'association des dimensions est indiquée par l'ajout de parenthèses autour des indices en question créant ainsi un multi-indice. Le tenseur $\boldsymbol{\mathcal{B}}$

(Éq. (III.23)) est ainsi défini par :

$$\boldsymbol{\mathcal{B}} = \left[\boldsymbol{\mathcal{A}}\left(i_1, \ldots, i_{p-1}, \left(i_p, \ldots, i_{(p+q-1)}\right), i_{(p+q)}, \ldots, i_d\right)\right] \tag{III.24}$$

Notons qu'il est possible d'associer simultanément plusieurs groupes d'indices.

Remarque 11. *On identifie les sélections d'indices du tenseur $\boldsymbol{\mathcal{B}}$ (Éq. (III.24)) avec les sélections de multi-indices du tenseur $\boldsymbol{\mathcal{A}}$ (Éq. (III.21)). Soit une sélection*

$$\mathcal{I}_p = \left\{\overline{i}_p^{(\alpha)} \mid \alpha \in [\![1, s]\!]\right\} \subset [\![1, n_p \ldots n_{p+q-1}]\!]$$

de s indices de la dimension p de $\boldsymbol{\mathcal{B}}$. On identifie $\mathcal{I}_p$ à la sélection

$$\mathcal{I}_p = \left\{\left(i_p^{(\alpha)}, \ldots, i_{(p+q-1)}^{(\alpha)}\right) \mid \alpha \in [\![1, s]\!]\right\} \subset I_p \times \cdots \times I_{p+q-1}$$

de s multi-indices associés aux dimensions $p, \ldots, (p+q-1)$ de $\boldsymbol{\mathcal{A}}$.

Le sous-tenseur de taille $(n_1, \ldots, n_{p-1}, s, n_{p+q}, \ldots, n_d)$ qui contient uniquement les éléments de $\mathcal{I}_p$ est défini par :

$$\boldsymbol{\mathcal{B}}(I_1, \ldots, I_{p-1}, \mathcal{I}_p, I_{p+q}, \ldots, I_d) = \boldsymbol{\mathcal{A}}(I_1, \ldots, I_{p-1}, \mathcal{I}_p, I_{p+q}, \ldots, I_d)$$

L'opération de *déploiement matriciel* (ou simplement déploiement) (*matricization* ou *matrix unfolding* en anglais) consiste à associer les dimensions consécutives d'un tenseur en deux groupes afin de l'interpréter comme une matrice (un tenseur d'ordre 2). À l'aide de la notation crochet, on définit le $q^{\text{ème}}$ déploiement du tenseur $\boldsymbol{\mathcal{A}}$, noté $\langle\boldsymbol{\mathcal{A}}\rangle_q$ par :

$$\langle\boldsymbol{\mathcal{A}}\rangle_q = [\boldsymbol{\mathcal{A}}((i_1, \ldots, i_q), (i_{q+1}, \ldots, i_d))] \in \mathbb{R}^{(n_1 \ldots n_q) \times (n_{q+1} \ldots n_d)}$$

Les q premières dimensions et $(d-q)$ dernières dimensions du tenseur $\boldsymbol{\mathcal{A}}$ correspondent à des multi-indices qui énumèrent respectivement les lignes et colonnes de la matrice $\langle\boldsymbol{\mathcal{A}}\rangle_q$.

Les figures III.3 et III.4 illustrent respectivement le premier et second déploiement matriciel du tenseur d'ordre 3 donné en figure III.2.

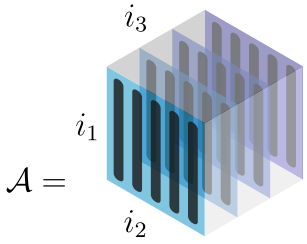


FIGURE III.2 – Illustration d'un tenseur d'ordre 3.

On définit le $q^{\text{ème}}$ *déploiement modal* $\{\boldsymbol{\mathcal{A}}\}_k \in \mathbb{R}^{n_q \times n_1 \ldots n_{q-1} n_{q+1} \ldots n_d}$ d'un tenseur $\boldsymbol{\mathcal{A}}$ (Éq.

$$\langle \mathcal{A} \rangle_1 = [\mathcal{A}(i_1, (i_2, i_3))] = \overset{(i_2,i_3)}{\underset{i_1}{\boxed{}}}$$

FIGURE III.3 – Premier déploiement matriciel pour le tenseur de la figure III.2.

$$\langle \mathcal{A} \rangle_2 = [\mathcal{A}((i_1, i_2), i_3)] = \overset{i_3}{\underset{(i_1,i_2)}{\boxed{}}}$$

FIGURE III.4 – Second déploiement matriciel pour le tenseur de la figure III.2.

(III.21)) comme la matrice dont les lignes sont énumérées par l'indice i_q et les colonnes sont énumérées par le multi-indice $(i_1, \ldots, i_{q-1}, i_{q+1}, \ldots, i_d)$. Plus précisément, on a :

$$\{\boldsymbol{\mathcal{A}}\}_q (j_1, j_2) = \boldsymbol{\mathcal{A}}(i_1, \ldots, i_d)$$

avec

$$\begin{aligned} j_1 &= i_q \\ j_2 - 1 &= \sum_{k=1, k\neq q}^{d} (i_k - 1) \prod_{l=1, l\neq q}^{k-1} n_l \end{aligned}$$

Le premier déploiement modal et le premier déploiement matriciel correspondent exactement : $\langle \boldsymbol{\mathcal{A}} \rangle_1 = \{\boldsymbol{\mathcal{A}}\}_1$. Notons que dans la littérature on fait parfois référence au déploiement modal simplement par le terme déploiement de tenseur. La figure III.5 illustre le second déploiement modal du tenseur donné en figure III.2.

$$\{\mathcal{A}\}_2 = \overset{(i_1,i_3)}{\underset{i_2}{\boxed{}}}$$

FIGURE III.5 – Second déploiement modal pour le tenseur de la figure III.2.

On définit l'opération de *squeeze* qui s'applique à un tenseur $\boldsymbol{\mathcal{A}}$ (Éq. (III.21)) et retourne un tenseur noté $\boldsymbol{\mathcal{A}}^\star$ pour lequel toutes les dimensions de taille 1 ont été supprimées. Comme l'opération est conceptuellement simple, mais lourde à écrire rigoureusement, on préfère donner un exemple. Le squeeze appliqué à $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{1\times 10\times 10\times 1}$ donne le tenseur $\boldsymbol{\mathcal{A}}^\star \in \mathbb{R}^{10\times 10}$ tel que :

$$\boldsymbol{\mathcal{A}}^\star(i,j) = \boldsymbol{\mathcal{A}}(1,i,j,1) \quad \forall\,(i,j) \in [1:10]^2$$

On rappelle la norme de Frobenius pour un tenseur $\boldsymbol{\mathcal{A}}$ (Éq. (III.21)) :

$$\|\boldsymbol{\mathcal{A}}\| = \sqrt{\sum_{i_1,\ldots,i_d \in I_1\times\cdots\times I_d} \boldsymbol{\mathcal{A}}(i_1,\ldots,i_d)^2}$$

On note que la norme de Frobenius est invariante par réarrangement du tenseur. En particulier, tous les déploiements modaux et réarrangements tensoriels d'un tenseur ont la même norme de Frobenius.

On définit l'opérateur de contraction $\bullet$ tel que pour $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times\cdots\times n_p\times r}$ et $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{r\times m_1\times\cdots\times m_q}$ les éléments du tenseur $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \bullet \boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1\times\cdots\times n_p\times m_1\times\cdots\times m_q}$ soient données par :

$$\boldsymbol{\mathcal{C}}(i_1,\ldots,i_p,j_1,\ldots,j_q) = \sum_{\alpha=1}^{r} \boldsymbol{\mathcal{A}}(i_1,\ldots,i_p,\alpha)\boldsymbol{\mathcal{B}}(\alpha,j_1,\ldots,j_p)$$

Diagrammes tensoriels Les diagrammes tensoriels (*Tensor network diagram*) introduits par [94] sont des schémas qui permettent de donner une représentation visuelle simple des décompositions tensorielles.

Un tenseur d'ordre d est représenté par un point à partir duquel part d branches. La figure III.6 donne la représentation d'un vecteur (tenseur d'ordre 1), d'une matrice (tenseur d'ordre 2), et d'un tenseur d'ordre 6.

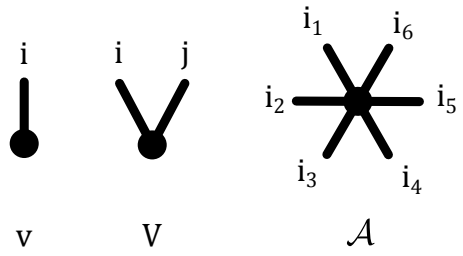


FIGURE III.6 – Diagrammes tensoriels pour un vecteur v, une matrice V et un tenseur $\boldsymbol{\mathcal{A}}$.

Un schéma peut s'interpréter par le fait que l'évaluation d'un élément est obtenue par un accès mémoire indexé par autant d'indices que de branches.

Ces schémas permettent, d'autre part, de représenter des opérations de contraction entre indices de différents tenseurs. Plus précisément, une branche reliant deux points

représente la contraction entre les indices respectifs des deux tenseurs. Par exemple, le produit matriciel est représenté par la figure III.7.

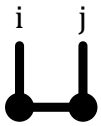


FIGURE III.7 – Diagramme tensoriel d'un produit matriciel.

Ces schémas expriment de manière condensée les opérations élémentaires entre tenseurs. Or, une décomposition tensorielle se définit fondamentalement par les opérations entre facteurs nécessaires à l'évaluation de ses éléments. Ainsi, les diagrammes tensoriels sont particulièrement adaptés pour représenter de manière compacte les décompositions tensorielles.

III.4.b Décomposition canonique

La décomposition canonique [22, 58] aussi appelée décomposition CP pour *Canonical decomposition/Parallel factors* s'écrit pour le tenseur $\boldsymbol{\mathcal{A}}$:

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) = \sum_{k=1}^{K} A_1(i_1, k) A_2(i_2, k) \ldots A_d(i_d, k) \quad \forall\, (i_1, \ldots, i_d) \in I \tag{III.25}$$

avec $A_j \in \mathbb{R}^{n_j \times K}$ pour $j \in [\![1, d]\!]$.

Il est possible de formuler la décomposition CP en imposant des conditions d'orthogonalités sur les colonnes des facteurs A_j ($i \in [\![1, d]\!]$). Dans ce cas, des coefficients α_k apparaissent et on a pour tout $\forall\, (i_1, \ldots, i_d) \in I$:

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) = \sum_{k=1}^{K} \alpha_k \bigotimes_{j=1}^{d} A_i(i_j, k)$$

Cette décomposition possède deux défauts majeurs. D'une part, pour un tenseur de rang K il n'existe pas d'unicité de la décomposition CP. D'autre part pour un tenseur arbitraire, le calcul du rang est un problème NP-difficile [62]. Lorsque ce format est utilisé pour construire des approximations tensorielles, les algorithmes vont typiquement avoir tendance à converger vers des minima locaux. En outre, le rang associé à des approximations de tenseurs complexes est en pratique souvent important et peut ne pas permettre de résoudre le problème de complexité de stockage. On trouve dans [70, 46] d'autres problèmes liés au rang qui rendent l'utilisation pratique de ce format difficile.

III.4.c Décomposition de Tucker

La décomposition de Tucker, introduite par [119], est une généralisation possible de la décomposition canonique faisant intervenir un tenseur de corps noté $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{K_1,\dots,K_d}$. Elle est donnée sous la forme :

$$\boldsymbol{\mathcal{A}}(i_1,\dots,i_d) = \sum_{k_1,\dots,k_d=1}^{K_1,\dots,K_d} \boldsymbol{\mathcal{C}}(k_1,\dots,k_d) A_1(i_1,k_1) A_2(i_2,k_2) \dots A_d(i_d,k_d) \quad \forall\, (i_1,\dots,i_d) \in I$$

où les K_i sont appelés multi-rangs.

Le format de Tucker se réduit au format CP lorsque le tenseur de corps est diagonal, i.e :

$$\boldsymbol{\mathcal{C}}(k_1,\dots,k_d) = \begin{cases} 1 & \text{si} \quad k_1 = \dots = k_d \\ 0 & \text{sinon} \end{cases}$$

Le diagramme tensoriel d'une décomposition de Tucker d'un tenseur d'ordre 5 est donné en figure III.8. Notons que le format CP et le format de Tucker d'un tenseur de même ordre

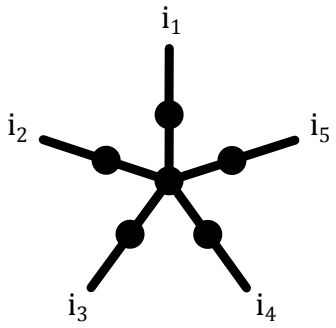


FIGURE III.8 – Diagramme tensoriel de la décomposition de Tucker.

sont représentés par des diagrammes tensoriels identiques. La principale difficulté posée par ce tenseur est qu'il reste soumis à la malédiction de la dimension. En effet, le tenseur de corps $\boldsymbol{\mathcal{C}}$ est de dimension d par conséquent même lorsque d est faible la complexité de stockage peut être très grande.

III.4.d Décomposition de Tucker hiérarchique

La décomposition de Tucker hiérarchique [52, 6, 5] introduite par [56, 51] pour réduire significativement la complexité de stockage du format de Tucker. Cette représentation consiste à scinder les dimensions de manière itérative pour obtenir un arbre au sens des diagrammes tensoriels. L'écriture mathématique générale de ce format est particulièrement lourde et n'a ici pas d'intérêt. On donne en figure III.9 un exemple de diagramme tensoriel d'un format hiérarchique de Tucker d'ordre 4.

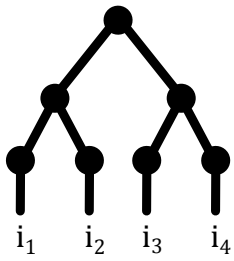


FIGURE III.9 – Diagramme tensoriel de la décomposition de Tucker hiérarchique.

La structure complexe d'arbre permet de réduire la complexité de stockage par rapport au format de Tucker et donc de surmonter le fléau de la dimension. Les facteurs de la décomposition (noeuds du diagramme) correspondent en effet ici au plus à des tenseurs d'ordre 3 dont le stockage est envisageable. Plus généralement, la structure ne correspond pas nécessairement à un arbre binaire et les facteurs peuvent être des tenseurs d'ordre supérieur à 3.

La difficulté de construction de ce type de tenseur vient du fait que la structure doit, en général, être déterminée à l'avance. Or trouver a priori la structure la mieux adaptée n'est pas un problème trivial. Pour une structure déterminée, il existe des algorithmes de construction qui opèrent par succession de décompositions SVD sur les différents déploiements modaux du tenseur de référence.

III.4.e Décomposition en train de tenseurs

Parmi les multiples formats de décompositions tensorielles existants, le choix a été d'étudier dans le cadre de cette thèse plus particulièrement le format en train de tenseurs (*tensor train format*) introduit par [93]. Ce choix est guidé par des résultats pratiques mettant en évidence son efficacité en termes de compression de données pour représenter des tenseurs de grande dimension.

III.4.e.1 Définition

Un tenseur $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ s'écrit comme une décomposition en train de tenseurs lorsque ses éléments sont obtenus par la suite de produits matriciels suivante :

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d) \quad \forall\, (i_1, \ldots, i_d) \in I_1 \times \cdots \times I_d \tag{III.26}$$

où pour tout $k \in [\![1, d]\!]$, $I_k = [\![1, n_k]\!]$ et les facteurs G_k (en anglais *tensor carriages* ou *core tensors*) du train de tenseurs sont des fonctions à valeurs matricielles définies sur I_k :

$$\begin{array}{rcccc} G_k & : & I_k & \to & \mathbb{R}^{r_{k-1} \times r_k} \\ & & i_k & \mapsto & G_k(i_k) \end{array}$$

L'acronyme TT fait référence à l'expression « Train de Tenseurs », on parle ainsi de décomposition TT ou encore de format TT.

Dans le format TT tel qu'il est présenté dans [93], les facteurs extrémaux $G_1(i_1)$ et $G_d(i_d)$ évalués aux indices i_1 et i_d sont respectivement des vecteurs ligne et colonne. On choisit la convention $r_0 = r_d = 1$ de sorte que les facteurs $G_1(i_1)$ et $G_d(i_d)$ de l'équation (III.26) puissent être interprétés indifféremment comme des matrices ou des vecteurs.

Selon le contexte, il est utile de manipuler des formulations alternatives de la décomposition TT. Les facteurs du train peuvent être interprétés comme des tenseurs d'ordre 3 notés $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ tels que :

$$G_k(i_k)(p,q) = \boldsymbol{\mathcal{G}}_k(p, i_k, q) \quad \forall\, (i_k, p, q) \in I_k \times [\![1, r_{k-1}]\!] \times [\![1, r_k]\!] \tag{III.27}$$

Une formulation équivalente de la décomposition TT qui exprime explicitement les sommations s'écrit :

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \boldsymbol{\mathcal{G}}_1^{\star}(i_1, \alpha_1) \boldsymbol{\mathcal{G}}_2(\alpha_1, i_2, \alpha_2) \ldots \boldsymbol{\mathcal{G}}_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) \boldsymbol{\mathcal{G}}_d^{\star}(\alpha_{d-1}, i_d) \tag{III.28}$$

où $\star$ correspond à l'opérateur squeeze défini dans la section III.4.a.

En utilisant l'opérateur de contraction $\bullet$ introduit dans la section III.4.a, l'équation (III.28) s'écrit de manière plus compacte :

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}}_1^{\star} \bullet \boldsymbol{\mathcal{G}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{G}}_{d-1} \bullet \boldsymbol{\mathcal{G}}_d^{\star} \tag{III.29}$$

On introduit la notion de *tailles de stockage* du train de tenseurs qui désigne le $(d-1)$-uplet $(r_1, \ldots, r_{d-1})$. Pour un même tenseur, il existe une infinité de décompositions. En effet, il est toujours possible de décomposer pour tout $(i_k, i_{k+1}) \in I_k \times I_{k+1}$ le produit $G_k(i_k)G_{k+1}(i_{k+1})$ de manière différente pour fournir une nouvelle décomposition égale en termes d'évaluation. Par conséquent, les tailles de stockage d'une décomposition TT ne sont pas uniques.

Représentations graphiques La figure III.10 donne une représentation visuelle d'un train de tenseurs en dimension arbitraire d. Elle illustre l'ensemble des matrices $G_k(i_k)$ à stocker ainsi que la suite de multiplications matricielles requises pour l'évaluation d'un élément du train de tenseurs.

La figure III.11 donne le diagramme tensoriel d'un train de tenseurs en dimension 7.

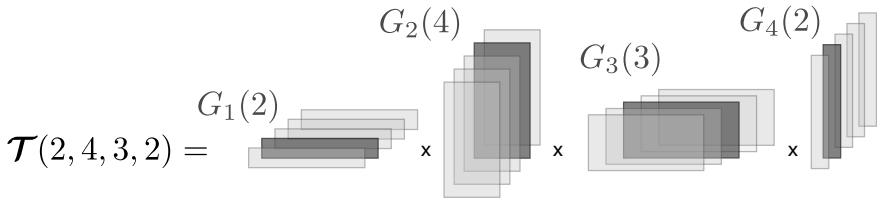


FIGURE III.10 – Illustration de l'évaluation d'un train de tenseurs d'ordre 4. L'élément $\mathcal{T}(2,4,3,2) \in \mathbb{R}$ est obtenu en effectuant la suite de multiplications matricielles entre les termes $G_1(2) \in \mathbb{R}^{r_1}$, $G_2(4) \in \mathbb{R}^{r_1 \times r_2}$, $G_3(3) \in \mathbb{R}^{r_2 \times r_3}$ et $G_4(2) \in \mathbb{R}^{r_3}$ identifiés en gris foncé.

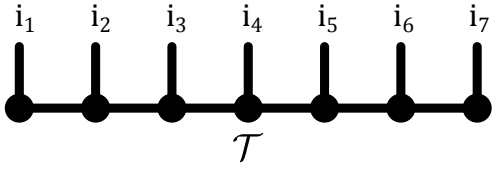


FIGURE III.11 – Diagramme tensoriel d'un train de tenseurs d'ordre 7.

III.4.e.2 Propriétés

Existence et unicité Pour un tenseur quelconque A, il existe toujours une représentation *exacte* au format TT, c'est-à-dire une représentation TT dont l'évaluation de chaque élément donne un résultat identique. Ce résultat se démontre de manière constructive via l'existence de l'algorithme TT-SVD (Algorithme 8).

Une décomposition TT (Éq. (III.26)) est dite *minimale* ou *de rangs pleins* [61] lorsque les facteurs $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ sont de rangs pleins à gauche et à droite. C'est-à-dire que les matrices $[\boldsymbol{\mathcal{G}}_k((\alpha_{k-1}, i_k), \alpha_k)]$ et $[\boldsymbol{\mathcal{G}}_k(\alpha_{k-1}, (i_k, \alpha_k))]$ sont de rangs pleins respectivement égaux à r_k et r_{k-1}.

On définit les *rangs de compression* ou *rangs TT* (*TT-ranks*) d'un tenseur comme les tailles de stockage $(r_1, \ldots, r_{d-1})$ d'une représentation TT minimale de ce tenseur. Ce concept est bien défini en vertu du théorème [61, Théorème 3.1] qui établit que les rangs de compression $(r_1, \ldots, r_{d-1})$ sont uniques.

On appelle *rangs de compression* d'un tenseur les rangs de ses différents déploiements matriciels. La référence [61] montre que les rangs de compression sont égaux aux rangs de séparation d'un tenseur.

On peut montrer [93] que le rang canonique R d'un tenseur est une borne supérieure des rangs TT :

$$r_k \leq R \quad \forall k [\![1, d-1]\!]$$

On observe en pratique [93] que les rangs de compression sont en réalité nettement inférieurs. Cela rend le format TT significativement plus efficace en termes de compression que le format canonique.

Les rangs de compression d'un tenseur ne sont pas invariants par permutation des dimensions [61, Remarque 3.2]. Par conséquent, l'ordre des dimensions, autrement dit la manière dont les éléments sont organisés dans un tenseur, a une influence sur la complexité de stockage des représentations TT.

En ajoutant des conditions orthogonalités sur les facteurs $\boldsymbol{\mathcal{G}}_k$, il est également possible de montrer [61, Théorème 3.1] que la décomposition minimale est unique.

Évaluation La complexité du calcul séquentiel qui intervient lors de l'évaluation d'un élément du train de tenseurs est $\mathcal{O}\left(dr^2\right)$, où $r = \max(r_1, \ldots, r_{d-1})$. En supposant que r est suffisamment petit, le coût de calcul pour accéder à un élément reste faible même en grande dimension. D'autre part, si on cherche à évaluer les éléments correspondant à des indices définis de manière tensorielle (par exemple pour les indices $\left\{i_1^{(1)}, i_1^{(2)}\right\} \times \left\{i_2^{(1)}, i_2^{(2)}, i_2^{(2)}\right\} \times \left\{i_3^{(1)}\right\}$ d'un tenseur d'ordre 3), la structure particulière du format TT permet de factoriser les calculs de manière à ce que le nombre d'opérations nécessaires soit significativement inférieur à la somme des opérations nécessaires pour obtenir les éléments individuellement. Le faible coût de calcul permet une évaluation du tenseur en temps réel. En termes d'exploitation en ligne, cette représentation est donc conforme aux spécifications requises pour un modèle de substitution.

Stockage Pour un tenseur arbitraire $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, la complexité de stockage de l'intégralité des éléments est $\mathcal{O}(n^d)$ où $n = \max(n_1, \ldots, n_d)$. La dépendance exponentielle à l'ordre d du tenseur empêche son stockage complet en mémoire même pour d petit. L'utilisation du format TT permet des gains significatifs en termes d'espace mémoire lorsque la structure sous-jacente du tenseur le permet, c'est-à-dire lorsque les rangs de compression du tenseur sont suffisamment faibles. La complexité de stockage du format TT est de $\mathcal{O}(nr^2d)$. La dépendance à l'ordre d du tenseur devient linéaire ce qui en fait un format particulièrement adapté pour des tenseurs de grandes dimensions. Dans de nombreuses applications d'intérêt [93], on observe des rangs TT petits ce qui permet de vaincre en pratique le fléau de la dimension. Il est possible de compresser davantage [91] une représentation TT en décomposant les facteurs du train de tenseurs au format de Tucker.

III.5 Méthodes d'approximations tensorielles de faible rang

Comme dans le cas des matrices, la structure des tenseurs qu'on manipule en pratique ne permet généralement pas de stocker en mémoire des décompositions tensorielles exactes. Il est alors intéressant d'en construire des approximations donc le stockage en mémoire est envisageable.

Les méthodes d'approximations tensorielles de faible rang consistent à approcher des tenseurs de dimension arbitraire par des décompositions tensorielles approchées. Ces méthodes ont fait l'objet de nombreux développement et la littérature [70, 67, 52, 14, 46, 55, 5] propose pour chaque format de tenseur des approches variées. Dans [29], on trouve une revue détaillée sur les liens entre les différents formats tensoriels ainsi que des méthodes de construction développées dans plusieurs communautés.

Dans cette section, on présente les algorithmes majeurs applicables à des tenseurs définis de manière *implicite*, c'est-à-dire dont tous les éléments peuvent être calculés via une procédure connue, cependant ni le calcul ni le stockage de l'intégralité des éléments est possible, car impliquant des temps de calcul et des espaces mémoire trop importants.

D'autres méthodes, dites de complétion de tenseurs [114] et basées pour certaines sur les mêmes types de procédures, s'appliquent lorsque le tenseur à approcher est connu uniquement partiellement.

Les méthodes d'approximations tensorielles de rang faible ont été d'autre part adaptées dans de nombreuses applications par exemple pour approcher des fonctions multivariées ou des solutions d'équations différentielles.

En particulier, la décomposition en train de tenseurs sur laquel se fonde la méthodologie développée dans cette thèse a été exploitée pour résoudre des problèmes d'équations intégrales (typiquement issus de la reformulation d'EDP) en dimension 3 [31], des systèmes d'EDP elliptiques paramétriques [68, 15] ou encore des systèmes d'EDP linéaires paramétriques [118]. [5] propose une méthode de résolution d'EDP paramétrique basée sur le format hiérarchique de Tucker. Enfin dans [35], une méthode de construction non intrusive basée sur le format canonique est appliquée à des modèles physiques typiquement donnés sous la forme d'EDO ou EDP . D'autres applications existent et on fait référence à [52] qui passe en revue les applications les plus marquantes.

III.5.a Alternating least square

L'algorithme *Alternating Least Square* (ALS) est un des premiers algorithmes de décomposition tensorielle [22, 58]. Il a été initialement introduit pour construire des décompositions CP, mais est en réalité un concept général qui peut s'appliquer pour construire différents types de décompositions tensorielles.

On présente l'ALS (de [70]) pour construire une approximation de rang R fixé au format CP du tenseur $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. La recherche de la meilleure approximation est formalisée par le problème de minimisation suivant :

$$\min_{\mathcal{T}} \|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \tag{III.30}$$

Avec $\boldsymbol{\mathcal{T}}$ au format CP, i.e :

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = \sum_{k=1}^{R} \prod_{j=1}^{d} T_j(i_j, k)$$

Le principe de l'ALS, comme son nom l'indique, est de résoudre successivement des problèmes de moindres carrés. Pour une décomposition CP, l'idée est de fixer à toutes les matrices T_j à l'exception d'une seule et de résoudre le problème (Éq. (III.30)). La norme de Frobenius étant invariante par déploiement modal, le problème de minimisation se réécrit pour tout $j \in [\![1, d]\!]$:

$$\min_{T_j \in \mathbb{R}^{n_j \times R}} \left\| \{\boldsymbol{\mathcal{A}}\}_j - \{\boldsymbol{\mathcal{T}}\}_j \right\|$$

Ce qui est équivalent au problème :

$$\min_{T_j \in \mathbb{R}^{n_j \times R}} \left\| \{\boldsymbol{\mathcal{A}}\}_j - T_j \{\boldsymbol{\mathcal{T}}_{-j}\}_j \right\| \tag{III.31}$$

où

$$\boldsymbol{\mathcal{T}}_{-j}(i_1, \ldots, i_d) = \prod_{k=1, k \neq j}^{d} T_k(i_k, i_j)$$

La solution du problème (Éq. (III.31)) est donnée par :

$$T_j = \{\boldsymbol{\mathcal{A}}\}_j \left[\{\boldsymbol{\mathcal{T}}_{-j}\}_j \right]^{\dagger}$$

La structure particulière de $\{\boldsymbol{\mathcal{T}}_{-j}\}_j$ permet de reformuler simplement l'expression de sa pseudo-inverse [70]. Le calcul de la pseudo-inverse se réduit alors au calcul de la pseudo-inverse d'une matrice de forme (R, R). L'algorithme ALS consiste à résoudre les problèmes d'optimisation de manière itérative sur toutes les dimensions puis de répéter le processus jusqu'à obtenir une approximation satisfaisante de $\boldsymbol{\mathcal{A}}$.

L'application de cet algorithme pose plusieurs problèmes. Il requiert de fixer le rang R or il n'existe pas de méthode satisfaisante pour déterminer a priori le rang CP d'un tenseur. L'algorithme demande d'autre part d'initialiser les matrices T_j Cela peut être fait de manière aléatoire ou en choisissant les T_j comme les matrices des valeurs singulières à gauche tronquées au rang R de la SVD de $\{\boldsymbol{\mathcal{A}}\}_j$. En pratique, le calcul des SVD n'est pas envisageable en grande dimension puisque les matrices déployées contiennent un nombre trop important d'éléments.

Pour résumer, la méthode ALS est particulièrement simple à comprendre et à implémenter, mais son application reste difficile. En effet, elle prend de nombreuses itérations,

dépend fortement de l'initialisation et ne garantit pas une convergence vers un minimum global [70].

III.5.b High Order Singular Value Decomposition

La méthode High Order Singular Value Decomposition (HOSVD) [74] permet de construire des approximations au format de Tucker de rangs $(R_1, \ldots, R_d)$. Elle peut être vue comme une généralisation de la SVD en dimension quelconque. L'algorithme 7 HOSVD vise à résoudre le problème de minimisation similaire au problème de l'ALS (Éq. (III.30)) en cherchant $\boldsymbol{\mathcal{T}}$ au format de Tucker :

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = \sum_{k_1,\ldots,k_d=1}^{R_1,\ldots,R_d} \boldsymbol{\mathcal{C}}(k_1, \ldots, k_d) \prod_{j=1}^{d} G_j(i_j, k_j)$$

L'algorithme consiste essentiellement à calculer une série de décompositions SVD associées respectivement à chaque dimension et indépendantes les unes des autres, puis de recomposer les résultats pour définir le tenseur approché. Comme les itérations sont indépendantes, leur ordre d'application n'a pas d'importance sur le résultat.

Algorithme 7 : HOSVD

Données : Un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.
Résultat : Le tenseur de corps $\boldsymbol{\mathcal{C}}$ et les facteurs de la décomposition $\{G_1, \ldots, G_d\}$.
Initialisation
Soit $\boldsymbol{\mathcal{C}}_0 = \boldsymbol{\mathcal{A}}$.
Pour $k \in [\![1, d]\!]$ **faire**
Définir A_k comme la matrice des valeurs principales à gauche tronquées au rang R_k de la SVD de $\{\boldsymbol{\mathcal{A}}\}_k$. Définir $\boldsymbol{\mathcal{C}}_k \in \mathbb{R}^{R_1 \times R_k \times n_{k+1} \cdots \times n_d}$ tel que :

$$\{\boldsymbol{\mathcal{C}}_k\}_k = A_k^T \{\boldsymbol{\mathcal{C}}_{k-1}\}_k$$

Finalisation
Définir $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{C}}_d$.

Les calculs de SVD qui interviennent dans l'algorithme peuvent poser problème en pratique de la même manière que pour l'algorithme ALS. Des variantes efficaces ont été développées telles que la HOOI *High-Order Orthogonal Iteration* [74] pour diminuer la complexité de calcul associé aux SVD.

III.5.c TT-SVD

L'algorithme 8 TT-SVD introduit par [93] décrit une procédure pour construire une décomposition en train de tenseurs $\boldsymbol{\mathcal{T}}$ à partir du tenseur $\boldsymbol{\mathcal{A}}$ de dimension d. Par extension, on appelle *décomposition TT-SVD*, une décomposition en train de tenseurs issue de l'algorithme TT-SVD.

Algorithme 8 : TT-SVD

Données : Un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.
Résultat : Les facteurs de la décomposition en train de tenseurs $\{\boldsymbol{\mathcal{G}}_1, \ldots, \boldsymbol{\mathcal{G}}_d\}$.
Initialisation
Définir la matrice $A_1 = \text{reshape}_{n_1 \times n_2 \ldots n_d}(\boldsymbol{\mathcal{A}})$ et $r_0 = 1$.
Pour $k \in [\![1, d-1]\!]$ **faire**
Calculer la SVD de A_k :

$$A_k = V_k \Sigma_k W_k^T \tag{III.32}$$

Définir r_k comme le rang de V_k puis définir $\boldsymbol{\mathcal{G}}_k$ tel que :

$$\boldsymbol{\mathcal{G}}_k = \underset{r_{k-1} \times n_k \times r_k}{\text{reshape}}\ (V_k)$$

Définir le réarrangement A_{k+1} tel que :

$$A_{k+1} = \underset{r_k n_{k+1} \times n_{k+2} \ldots n_d}{\text{reshape}} \left(\Sigma_k W_k^T\right)$$

Finalisation
Définir le tenseur

$$\boldsymbol{\mathcal{G}}_d = \underset{r_{d-1} \times n_d \times r_d}{\text{reshape}}\ (A_d)$$

avec $r_d = 1$.

On peut montrer [93] que la décomposition TT (Éq. (III.29)) associée aux facteurs $\boldsymbol{\mathcal{G}}_k$ est exacte, prouvant ainsi que tout tenseur peut s'exprimer sous la forme d'une décomposition TT.

Le théorème [61, Theorème 3.1] établit que pour tout tenseur, la décomposition TT minimale dont les matrices $\langle \boldsymbol{\mathcal{G}}_k \rangle_2$ possèdent des colonnes orthogonales est définie de manière unique (à insertion de matrices orthogonales près) et que d'autre part cette décomposition correspond à la TT-SVD.

Il est possible de modifier l'algorithme 8 afin construire des représentations TT approchées. La modification consiste à substituer la décomposition SVD (Éq. (III.32)) de chaque itération k par une décomposition SVD tronquée.

La proposition 1 [93, Théorème 2.2] donne une borne supérieure sur la norme de l'erreur entre un tenseur de référence et une approximation TT construite via la TT-SVD tronquée.

Proposition 1. *Soit un tenseur $\boldsymbol{\mathcal{A}}$, la TT-SVD tronquée aux rangs r_k, donne une approximation $\boldsymbol{\mathcal{T}}$ au format TT tel que :*

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sqrt{\sum_{k=1}^{d-1} \epsilon_k^2}$$

où ϵ_k est défini par :

$$\epsilon_k = \min_{\mathrm{rg}(X) \leq r_k} \|\langle \boldsymbol{\mathcal{A}} \rangle_k - X\|$$

Il est donc théoriquement possible pour n'importe quel tenseur de référence de construire des approximations au format TT aussi précises que l'on désire. La construction de telles représentations, que ce soit avec la TT-SVD ou sa version tronquée, requiert l'accès effectif à tous les éléments du tenseur de référence. En pratique, cela n'est pas envisageable en termes de temps de calcul en grande dimension.

III.5.d TT-cross

Une approche plus réaliste pour effectivement construire des représentations TT s'inspire de la décomposition matricielle skeleton (ou pseudo-skeleton, notée PSD). L'algorithme TT-cross, proposé par [93], est une procédure itérative sur les dimensions permettant de construire un format TT en se basant uniquement sur un nombre réduit d'éléments du tenseur de référence. L'idée fondamentale de la TT-cross est de substituer la décomposition SVD par une décomposition PSD. On décrit l'algorithme TT-cross, tel qu'il a été introduit initialement dans [93], c'est-à-dire pour construire des représentations exactes.

Soit un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. La première étape consiste à définir le premier déploiement $A_1 = \langle \boldsymbol{\mathcal{A}} \rangle_1$ de rang r_1. On suppose qu'on dispose d'un ensemble de r_1 colonnes $\mathcal{J}_1$

linéairement indépendantes. Choisir r_1 lignes de $A_1(:, \mathcal{J}_1)$, permettant d'écrire A_1 sous la forme d'une décomposition PSD :

$$A_1 = A_1(:, \mathcal{J}_1) \left[A(\mathcal{I}_1, \mathcal{J}_1)\right]^{-1} A_1(\mathcal{I}_1, :) \tag{III.33}$$

et définir $G_1 = A_1(:, \mathcal{J}_1) \left[A_1(\mathcal{I}_1, \mathcal{J}_1)\right]^{-1}$.

Définir à présent le tenseur $\boldsymbol{\mathcal{A}}^{(2)} \in \mathbb{R}^{r_1 \times n_2 \times \cdots \times n_d}$ en inversant la relation suivante :

$$\left\langle \boldsymbol{\mathcal{A}}^{(2)} \right\rangle_1 = A_1(\mathcal{I}_1, :)$$

La définition de $\boldsymbol{\mathcal{A}}^{(2)}$ consiste simplement à *dissocier* les colonnes de la matrice A_1 qui avaient été associées à la première étape de l'algorithme.

À l'itération $k \in [\![2, d-1]\!]$, on considère la matrice $A_k \in \mathbb{R}^{r_{k-1} n_k \times n_{k+1} \ldots n_d}$ de rang r_k comme le second déploiement du tenseur $\boldsymbol{\mathcal{A}}^{(k)}$:

$$A_k = \left\langle \boldsymbol{\mathcal{A}}^{(k)} \right\rangle_2 \tag{III.34}$$

À nouveau, on suppose qu'on dispose d'un ensemble de r_k colonnes $\mathcal{J}_k$ de A_k linéairement indépendantes. Choisir r_k lignes de $A_k(:, \mathcal{J}_k)$ pour écrire une décomposition PSD de A_k :

$$A_k = A_k(:, \mathcal{J}_k) \left[A_k(\mathcal{I}_k, \mathcal{J}_k)\right]^{-1} A_k(\mathcal{I}_k, :)$$

et définir $G_k = A_k(:, \mathcal{J}_k) \left[A_k(\mathcal{I}_k, \mathcal{J}_k)\right]^{-1}$. L'étape k s'achève en définissant le tenseur $\boldsymbol{\mathcal{A}}^{(k+1)} \in \mathbb{R}^{r_k \times n_{k+1} \times \cdots \times n_d}$ en inversant la relation suivante :

$$\left\langle \boldsymbol{\mathcal{A}}^{(k+1)} \right\rangle_1 = A_k(\mathcal{I}_k, :)$$

La définition de $\boldsymbol{\mathcal{A}}^{(k+1)}$ consiste à *dissocier* les colonnes de A_k qui avaient été associées lors de sa définition (Éq. (III.34)).

La procédure se termine ($k = d$) en définissant G_d telle que :

$$G_d = \boldsymbol{\mathcal{A}}^{(d)} \in \mathbb{R}^{r_{d-1} \times n_d}$$

Par souci d'exhaustivité, on définit la matrice $A_d = \left\langle \boldsymbol{\mathcal{A}}^{(d)} \right\rangle_2$.

Finalement, on montre [93, Thérorème 3.1] que les facteurs $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ définis via l'équation (III.27) (en imposant $r_0 = r_d = 1$) permettent d'obtenir une représentation exacte de $\boldsymbol{\mathcal{A}}$ par un train de tenseurs :

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}}_1^\star \bullet \cdots \bullet \boldsymbol{\mathcal{G}}_d^\star$$

La TT-cross définit une procédure théorique permettant de construire des décompo-

sitions TT exactes uniquement à partir d'un nombre limité d'éléments du tenseur de référence si celui-ci possède des rangs de compression faibles. Par extension, on appelle décomposition TT-cross [109] toute décomposition TT qui s'exprime de la même manière que celle donnée par l'algorithme TT-cross.

Lorsque les rangs de compression $(r_1, \ldots, r_{d-1})$ du tenseur sont connus, à chaque itération k tout choix de r_k colonnes et r_k lignes linéairement indépendantes suffit pour définir une décomposition TT-cross exacte. En pratique, l'application de l'algorithme TT-cross peut poser problème, car les rangs du tenseur de référence sont en général inconnus.

À l'instar de la TT-SVD, l'algorithme TT-cross permet de construire des représentations approchées en remplaçant la décomposition skeleton par une décomposition pseudo-skeleton. La construction d'une représentation approchée soulève une difficulté pratique supplémentaire liée au choix des meilleures sélections de lignes/colonnes. Dans [93] une heuristique, basée sur le principe de maximisation du volume des sous-matrices est proposée. La méthode repose sur une généralisation de l'algorithme alterné Maxvol (Section III.3.d.1). L'heuristique consiste dans un premier temps à appliquer l'algorithme TT-cross en sélectionnant un nombre de colonnes qui surestime le rang. Cette phase est appelée balayage de gauche à droite (*left-to-right sweep*). Dans un second temps, on applique l'algorithme TT-cross en inversant l'ordre des dimensions et en choisissant pour colonnes les indices qui correspondent aux indices de lignes $\{I_k\}_{k=1}^{d-1}$ sélectionnées lors du balayage de gauche à droite précédent. Cette phase est appelée balayage de droite à gauche (*right-to-left sweep*). Ces deux étapes sont répétées jusqu'à « stagnation », en mesurant la distance en termes de norme de Frobenius entre deux approximations successives.

Comme pour l'algorithme alterné Maxvol (Section III.3.d.1), les rangs du tenseur ne sont en général pas connus à l'avance et doivent être surestimés. Par conséquent le calcul numérique des $[A_k(\mathcal{I}_k, \mathcal{J}_k)]^{-1}$ devient numériquement instable.

Pour remédier à ce problème, [93] propose d'utiliser une idée déjà mise en œuvre pour l'algorithme alterné Maxvol. Au début de chaque itération on calcule la décomposition QR de la matrice $A_k(:, \mathcal{J}_k) = Q_k R_k$ et on sélectionne des lignes sur la matrice Q_k. On peut finalement définir les matrices G_k par :

$$G_k = Q_k Q_k(\mathcal{I}_k, :)^{-1}$$

Dans cette approche, les rangs de la décomposition ont été initialement surestimés, cela signifie qu'il existe théoriquement des représentations TT plus compactes et équivalentes en termes d'évaluation. L'application de l'algorithme de recompression proposé dans [91] permet d'approcher la décomposition initiale par une décomposition TT dont les tailles de stockage $(r_1, \ldots, r_{d-1})$ sont plus faibles. Une taille de stockage r_k qui se trouve être

égale à la valeur R_k initialement choisie pour surestimer le rang de compression, indique très probablement que la valeur de R_k n'a pas été prise suffisamment grande. La solution proposée dans ce cas consiste à recommencer le processus à partir de zéro en choisissant des rangs plus importants.

La procédure TT-cross itérative montre des résultats de convergence intéressants sur des fonctions complexes mais demande de pouvoir accéder aux éléments du tenseur de référence un très grand nombre de fois. D'autre part, il peut être difficile en pratique de surestimer les rangs de compression d'un tenseur sans choisir des valeurs trop importantes rendant le coût de construction prohibitif. Dans les applications qui nous intéressent, ce type d'approche n'est pas envisageable.

III.5.e DMRG

Récemment, il a été identifié que le format TT était similaire [64] à la décomposition *Matrix Product State* MPS [63, 111] introduite en chimie quantique comme un cas particulier des réseaux de tenseurs (*Tensor Network*) [18]. L'algorithme DMRG (*Density Matrix Renormalization Group*) [111] est une heuristique de construction associée introduite par [122]. La procédure peut être formulée à la manière de l'ALS (SectionIII.5.a) comme la résolution successive de problèmes d'optimisation.

Le problème d'approximation se formalise de la manière suivante : Pour un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, on cherche $\boldsymbol{\mathcal{T}}$ solution du problème de minimisation :

$$\min_{\boldsymbol{\mathcal{T}}} \|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \tag{III.35}$$

où $\boldsymbol{\mathcal{T}}$ est donné sous la forme d'un train de tenseurs :

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}}_1^\star \bullet \cdots \bullet \boldsymbol{\mathcal{G}}_d^\star \quad \text{(Éq. (III.29))}$$

Le concept d'ALS peut être étendu naturellement à la résolution du problème (Éq. (III.35)). L'idée est de résoudre le problème de manière itérative en fixant successivement (pour $k \in [\![1, d]\!]$) tous les facteurs du train de tenseurs à l'exception du facteur $\boldsymbol{\mathcal{G}}_k$. En itérant de manière répétée sur les dimensions k, on peut atteindre un critère de convergence généralement associé à la norme de la différence entre deux approximations successives. L'algorithme ALS présente une bonne convergence [60, 110] en pratique, toutefois il possède un inconvénient majeur : les rangs des facteurs du train doivent être définis a priori et ne peuvent pas être modifiés au cours du calcul.

L'algorithme DMRG permet de surmonter cette difficulté en apportant une légère modification à l'algorithme ALS. Contrairement à l'ALS, l'étape k de la DMRG consiste à calculer simultanément G_k et G_{k+1} dans un premier temps en cherchant à optimiser

le produit $W_k(i_k, i_{k+1}) = G_k(i_k)G_{k+1}(i_{k+1})$ puis dans un second temps en décomposant $W_k(i_k, i_{k+1})$ via la SVD pour retrouver les deux facteurs. L'intérêt de cette méthode est qu'elle ne fait pas apparaitre le rang courant r_k dans le problème d'optimisation. Celui-ci peut donc être déterminé de manière adaptative lors de la décomposition de $W_k(i_k, i_{k+1})$.

Supposons que l'on dispose d'une approximation TT grossière $\boldsymbol{\mathcal{T}}$ du tenseur de référence $\boldsymbol{\mathcal{A}}$, l'objectif de la DMRG est d'enrichir itérativement cette décomposition pour obtenir une approximation précise. Pour tout $k \in [\![1, d-1]\!]$, on peut réécrire le tenseur $\boldsymbol{\mathcal{T}}$ sous la forme :

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{U}}_k \bullet \boldsymbol{\mathcal{W}}_k \bullet \boldsymbol{\mathcal{V}}_k$$

où

$$\begin{aligned}
\boldsymbol{\mathcal{U}}_k &= \boldsymbol{\mathcal{G}}_1^\star \bullet \cdots \bullet \boldsymbol{\mathcal{G}}_{k-1} \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times r_{k-1}} \\
\boldsymbol{\mathcal{W}}_k &= \boldsymbol{\mathcal{G}}_k \bullet \boldsymbol{\mathcal{G}}_{k+1} \in \mathbb{R}^{r_{k-1} \times n_k \times n_{k+1} \times r_{k+1}} \\
\boldsymbol{\mathcal{V}}_k &= \boldsymbol{\mathcal{G}}_{k+2} \bullet \cdots \bullet \boldsymbol{\mathcal{G}}_d^\star \in \mathbb{R}^{r_{k+1} \times n_{k+2} \times \cdots \times n_d}
\end{aligned}$$

On définit

$$\begin{aligned}
U_k &= \langle \boldsymbol{\mathcal{U}} \rangle_k \in \mathbb{R}^{(n_1 \ldots n_{k-1}) \times r_{k-1}} \\
V_k &= \langle \boldsymbol{\mathcal{V}}_k \rangle_1 \in \mathbb{R}^{r_{k+1} \times (n_{k+2} \ldots n_d)} \\
\boldsymbol{\mathcal{A}}_k &= \underset{(n_1 \ldots n_{k-1}) \times n_k \times n_{k+1} \times (n_{k+2} \ldots n_d)}{\text{reshape}} (\boldsymbol{\mathcal{A}})
\end{aligned}$$

et les ensembles de matrices $W_k = \left\{ W_k^{(i_k, i_{k+1})} \right\}_{(i_k, i_{k+1}) = (1,1)}^{(n_k, n_{k+1})}$ et $\left\{ A_k^{(i_k, i_{k+1})} \right\}_{(i_k, i_{k+1}) = (1,1)}^{(n_k, n_{k+1})}$ telles que :

$$\begin{aligned}
W_k^{(i_k, i_{k+1})}(\alpha, \beta) &= \boldsymbol{\mathcal{W}}_k(\alpha, i_k, i_{k+1}, \beta) \\
A_k^{(i_k, i_{k+1})}(\alpha, \beta) &= \boldsymbol{\mathcal{A}}_k(\alpha, i_k, i_{k+1}, \beta)
\end{aligned}$$

Avec ces notations, on montre que le problème (Éq. (III.35)) avec uniquement un facteur libre s'écrit :

$$\min_{W_k^{(i_k, i_{k+1})}} \left\| A_k^{(i_k, i_{k+1})} - U_k W_k^{(i_k, i_{k+1})} V_k \right\| \quad \forall\, (i_k, i_{k+1}) \in [\![1, n_k]\!] \times [\![1, n_{k+1}]\!]$$

La solution du problème s'écrit explicitement :

$$W_k^{(i_k,i_{k+1})} = U_k^{\dagger} A_k^{(i_k,i_{k+1})} V_k^{\dagger} \tag{III.36}$$

Une fois les $W_k^{(i_k,i_{k+1})}$ calculées, leur décomposition via la SVD permet de définir $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r'_k}$ et $\boldsymbol{\mathcal{G}}_{k+1} \in \mathbb{R}^{r'_k \times n_{k+1} \times r_{k+1}}$ tels que $\boldsymbol{\mathcal{W}}_k = \boldsymbol{\mathcal{G}}_k \bullet \boldsymbol{\mathcal{G}}_{k+1}$ où r'_k correspond au rang de la SVD (tronquée ou non).

Amélioration de l'efficacité des calculs Dans le cas général, la taille des matrices mises en jeu peut rendre le calcul des pseudo-inverses extrêmement couteux. Il est possible de réduire de manière importante les coûts de calcul en imposant à U_k et V_k de posséder respectivement des colonnes et lignes orthogonales. Dans ce cas, l'équation (III.36) se réduit à :

$$W_k^{(i_k,i_{k+1})} = U_k^{T} A_k^{(i_k,i_{k+1})} V_k^{T} \tag{III.37}$$

Sans détailler, on montre [110] qu'en exploitant la structure particulière des matrices U_k et V_k, il est possible de conserver la propriété d'orthogonalité en appliquant des décompositions QR au cours des itérations.

Les calculs impliqués dans l'équation (III.37) restent prohibitifs dû à la dimension des matrices. Une solution pour calculer une estimation des $W_k^{(i_k,i_{k+1})}$ est proposée par [110]. Comme les matrices U_k et V_k sont orthogonales, on peut montrer qu'il est possible de trouver une bonne approximation $\widetilde{W}_k^{(i_k,i_{k+1})}$ de $W_k^{(i_k,i_{k+1})}$ à partir d'un sous-ensemble de lignes $\mathcal{I}_k$ de U_k et de colonnes $\mathcal{J}_k$ de V_k telle que :

$$\widetilde{W}_k^{(i_k,i_{k+1})} = \widehat{U}_k^{-1} \widehat{A}_k^{(i_k,i_{k+1})} \widehat{V}_k^{-1}$$

où

$$\begin{aligned} \widehat{U}_k &= U_k(\mathcal{I}_k, :) \\ \widehat{V}_k &= V_k(:, \mathcal{J}_k) \\ \widehat{A}_k^{(i_k,i_{k+1})} &= A_k^{(i_k,i_{k+1})}(\mathcal{I}_k, \mathcal{J}_k) \end{aligned}$$

À la manière des balayages de la TT-cross, [110] propose une procédure pour déterminer les ensembles de lignes et de colonnes en s'appuyant sur le principe d'une maximisation du volume. Vis-à-vis des applications, l'inconvénient principal de cet algorithme est qu'il faut disposer initialement d'une approximation au format TT.

III.5.f Autres algorithmes associés au format TT

Pour pouvoir réaliser numériquement des calculs algébriques entre trains de tenseurs, des implémentations efficaces de plusieurs opérations ont été développées [92]. C'est le cas des opérations élémentaires comme l'addition et la multiplication mais aussi d'opérations et d'opérateurs plus complexes comme la convolution, le produit scalaire, le produit d'Hadamard, les contractions et les normes. Les tenseurs résultants de ces opérations possèdent typiquement des rangs plus grands que les tenseurs initiaux. Pour empêcher une augmentation trop importante des rangs, des algorithmes de recompression [92, Algorithme 2] ou [91, Algorithme 1] ont été proposés pour permettre de réduire le rang en contrôlant la précision. On trouve, d'autre part, dans [92] une procédure pour construire un tenseur TT à partir d'une tenseur au format canonique. Enfin, une méthode d'intégration de fonctions multivariées [93], qui s'inspire de la méthode de quadrature de Gauss, a également été proposée pour des fonctions écrites au format TT.

III.5.g Proper Generalized Decomposition

La Proper Generalized Decomposition a été introduite et développée par [1, 2, 27, 87, 88]. Elle peut être interprétée comme la décomposition radiale de la méthode LaTIn [73]. Cette méthode constitue une approche originale de résolution d'EDP paramétriques en mécanique fondée sur une représentation tensorielle de toutes les quantités du problème ainsi que toutes les solutions possibles dans un espace multidimensionnel. La notion de tenseur fait ici référence à des éléments de produit tensoriel d'espaces vectoriels et est donc plus générale que la notion de tableau multidimensionnel. En particulier, un tenseur possède dans une base donnée, une représentation par un tableau multidimensionnel.

La méthode PGD a montré son efficacité en termes de précision et de temps de calcul dans le cadre de lois matériaux linéaires ou de lois de comportement élastoviscoplastique à énergie libre quadratique [73, 97] décrites par des EDP.

On considère une EDP paramétrique dont la solution est notée f. Le domaine de définition $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ sur lequel est définie la solution peut s'écrire comme un produit cartésien des domaines $\mathcal{D}_k$ associés aux coordonnées spatiales et temporelles et aux paramètres de l'EDP. Le principe de la PGD est de rechercher f sous la forme d'une décomposition tensorielle canonique telle que :

$$f = \sum_{k=1}^{K} \bigotimes_{i=1}^{d} f_{i,l} \tag{III.38}$$

où K désigne le nombre de termes de la décomposition et pour tout $(i,k) \in [\![1,d]\!] \times [\![1,K]\!]$ les *modes* $f_{i,k} : x_i \mapsto f_{i,k}(x_i)$ sont des fonctions d'une seule variable. Les x_i peuvent aussi représenter l'association de plusieurs coordonnées. Par exemple, les coordonnées d'espace

peuvent être regroupées. Notons que la solution d'une EDP peut toujours s'écrire sous la forme (Éq. (III.38)) même si cela peut impliquer un nombre infini de termes.

En termes d'image, f s'écrit sous la forme :

$$f(x_1,\dots,x_d)=\sum_{k=1}^{K} f_{1,k}(x_1)f_{2,k}(x_2)\dots f_{d,k}(x_d) \quad \forall\,(x_1,\dots,x_d)\in\mathcal{D}_1\times\dots\times\mathcal{D}_d$$

L'originalité de la méthode est de considérer les paramètres, les coordonnées spatiales et temporelles de la même manière dans l'écriture séparée de la solution. Dans la thèse [44], une approche alternative est proposée où la solution f est recherchée sous un format de Tucker (Section III.4.c) ou un format hiérarchique de Tucker (Section III.4.d).

Solveur PGD générique La PGD est un solveur qui consiste à injecter la décomposition particulière de f (Éq. (III.38)) dans la formulation faible de l'EDP afin de construire de manière gloutonne les modes $\bigotimes_{i=1}^{d} f_{i,k}$ en résolvant itérativement des problèmes de points fixes.

On présente le formalisme générique de la méthode PGD. Soit une EDP linéaire dont la formulation variationnelle est donnée par $\mathcal{L}(f)=\mathcal{G}$ et où f représente la solution. L'approximation de rang L de cette solution est notée :

$$\widehat{f}^{L}=\sum_{l=1}^{L}\bigotimes_{i=1}^{d} f_{i,l}$$

On note Res^L le résidu obtenu en injectant l'approximation de rang L dans le système d'équations :

$$Res^{L}=\mathcal{L}(\widehat{f}^{L})-\mathcal{G}$$

Le principe de la PGD est de construire de manière gloutonne $\widehat{f}$ en l'enrichissant à chaque itération k d'un terme de correction supplémentaire $\bigotimes_{i=1}^{d} f_{i,k}$. Le terme de correction est solution d'un problème de minimisation du résidu qui est résolu en pratique par une méthode du point fixe. L'algorithme 9 donne les idées essentielles mises en jeux.

Notons que l'algorithme converge théoriquement vers la solution exacte. La vitesse de convergence est variable selon les implémentations particulières de la PGD et il existe différentes stratégies pour améliorer la vitesse de convergence.

Algorithme 9 : PGD

Données : Un système d'équations $\mathcal{L}(f) = \mathcal{G}$ écrit sous forme séparée et une tolérance de convergence ϵ_{tol}.

Résultat : Une solution approchée $\widehat{f}$ donnée sous la forme d'une décomposition canonique.

Initialisation

Soit $\widehat{f}^0$ cinématiquement admissible et $\delta > \epsilon_{\text{tol}}$.

Soit $L := 1$.

Tant que $\delta > \epsilon_{tol}$ **faire**

On cherche les fonctions $f_{i,L}$ tels que le résidu Res^L de l'approximation :

$$\widehat{f}^L = \widehat{f}^{L-1} + \bigotimes_{i=1}^{d} f_{i,L}$$

soit orthogonal au terme de correction $\bigotimes_{i=1}^{d} f_{i,L}$, i.e :

$$\left\langle \mathcal{L}\left(\widehat{f}^L\right), \bigotimes_{i=1}^{d} f_{i,L} \right\rangle = \left\langle \mathcal{G}, \bigotimes_{i=1}^{d} f_{i,L} \right\rangle \tag{III.39}$$

Calcul des modes Initialiser les fonctions $f_{i,L}$ pour tout $i \in [\![1, d]\!]$ de manière aléatoire. Résoudre le problème (Éq. (III.39)) en fixant alternativement tous les modes sauf un.

Incrémentation : $L := L + 1$.

Mise à jour du critère : $\delta = \left\| \bigotimes_{i=1}^{d} f_{i,L} \right\|_{L^2}$ où $\|.\|_{L^2}$ est une norme bien choisie.

Finalisation

La norme de la dernière correction étant inférieure à la tolérance prédéfinie, on considère que l'algorithme a convergé et $\widehat{f} = \widehat{f}^L$ est définie comme la solution PGD.

Calculs numériques En pratique, le calcul des modes nécessite d'introduire une discrétisation des espaces. On cherche les fonctions $f_{i,k}$ comme les combinaisons linéaires

$$f_{i,k} = \sum_{j=1}^{n_i} A_i(j,k)\varphi_i^{(j)} \tag{III.40}$$

où les $\left\{\varphi_k^{(i)} : x_k \mapsto \varphi_k^{(i)}(x_k)\right\}_{i=1}^{n_k}$ sont des fonctions de base qui engendrent un espace vectoriel noté E_k et définies sur le domaine $\mathcal{D}_k$. Pour la variable d'espace, ces fonctions correspondent typiquement aux fonctions chapeaux utilisées dans la méthode des éléments finis. La mise en œuvre numérique de la méthode PGD consiste donc à calculer explicitement les coefficients $A_i(j,k)$ des modes PGD.

Exploitation en ligne En injectant la forme des modes (Éq. (III.40)) dans la forme générale de la solution (Éq. (III.38)), on peut écrire f comme

$$f = \sum_{i_1=1}^{n_1} \dots \sum_{i_d=1}^{n_d} \boldsymbol{\mathcal{A}}(i_1,\dots,i_d)\varphi_1^{(i_1)} \otimes \dots \otimes \varphi_d^{(i_d)}$$

où $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ correspond à un tenseur multidimensionnel donné au format canonique (Section III.4.b).

L'idée de la PGD revient finalement à chercher la solution f comme un élément du produit tensoriel des espaces E_k ($f \in E_1 \otimes \dots \otimes E_d$) en imposant une décomposition canonique des coefficients $\boldsymbol{\mathcal{A}}$.

Au terme de l'algorithme PGD, on dispose en pratique du tableau multidimensionnel $\boldsymbol{\mathcal{A}}$ au format canonique permettant via la connaissance l'ensemble des fonctions de base $\varphi_i^{(j)}(x_i)$ d'approcher extrêmement rapidement pour n'importe quelles valeurs de paramètres les solutions du système d'EDP. Cette représentation des solutions sous forme de données stockées en mémoire est parfois [28] désignée par le terme d'abaque virtuel (*Virtual Charts*). L'originalité de l'approche PGD pour la communauté mécanique vient du fait que la phase en ligne ne fait plus intervenir de résolution d'équations, contrairement aux méthodes classiques de réduction de modèle. En ce sens, cette méthode peut être interprétée comme une méthode de surfaces de réponses.

Chapitre IV

Cadre générique d'approximation par décomposition en train de tenseurs

Parmi les multiples formats de décompositions tensorielles existants, le choix a été d'étudier, dans le cadre de cette thèse le format en train de tenseurs (tensor train format) introduit par [93]. Ce choix est guidé par des résultats théoriques et pratiques mettant en évidence son efficacité en termes de compression de données et de qualité de l'approximation pour représenter des tenseurs de grande dimension.

L'objectif de la première section de ce chapitre est d'introduire un formalisme d'approximation de fonctions multivariées basé sur le format en train de tenseurs. Dans la seconde section, on présente une formalisation algorithmique générique développée au cours de la thèse, basé sur une suite d'approximation matricielle de faible rang, permettant d'approcher des tenseurs (au sens de tableau multidimensionnel) à partir de décompositions en train de tenseurs.

Table des matières

IV.1 Approximation de fonctions multivariées 80
IV.1.a Représentation tensorielle 80
IV.1.b Représentation discrète 80
IV.1.c Représentation en train de tenseurs 81
IV.1.d Interpolation multilinéaire par morceaux 82
IV.2 Algorithme générique de décomposition TT par approximations matricielles successives 84
IV.2.a Description de l'algorithme 84
IV.2.b Erreur d'approximation 85
IV.2.c Algorithme de refactorisation 89
IV.3 Particularisation de l'algorithme de décomposition TT par approximations matricielles successives 91

IV.1 Approximation de fonctions multivariées

On considère une fonction f multivariée à valeurs vectorielles définie sur le domaine $\mathcal{D}$ correspondant à un produit cartésien d'intervalle $\mathcal{D}_k = \left[x_k^{(min)}, x_k^{(max)}\right] \subset \mathbb{R}$

$$\begin{array}{rcccl} f & : & \mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_d & \rightarrow & \mathbb{R}^m \\ & & (x_1, \ldots, x_d) & \mapsto & \begin{pmatrix} f_1(x_1, \ldots, x_d) \\ \vdots \\ f_m(x_1, \ldots, x_d) \end{pmatrix} \end{array} \tag{IV.1}$$

dont on dispose d'une procédure d'évaluation plus ou moins couteuse. En définissant les vecteurs de base canonique $\{e_j\}_{j=1}^m$ de $\mathbb{R}^m$, on réécrit f sous la forme :

$$f = \sum_{j=1}^{m} f_j e_j$$

IV.1.a Représentation tensorielle

À la manière de la PGD, on introduit pour tout $k \in [\![1, d]\!]$ les bases $\left\{\varphi_k^{(i)} : x_k \mapsto \varphi_k^{(i)}(x_k)\right\}_{i=1}^{n_k}$ définies sur les domaines $\mathcal{D}_k$ qui engendrent respectivement les espaces vectoriels E_k. Une représentation tensorielle de la fonction f consiste à la définir comme un élément de $[E_1 \otimes \cdots \otimes E_d]^m$ tel que :

$$f = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \sum_{j=1}^{m} \boldsymbol{\mathcal{A}}(i_1, \ldots, i_d, j) \left(\varphi_1^{(i_1)} \otimes \cdots \otimes \varphi_d^{(i_d)}\right) e_j \tag{IV.2}$$

où $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d \times m}$ est un tenseur multidimensionnel. Notons que chaque f_j appartient au produit tensoriel d'espace $E_1 \otimes \cdots \otimes E_d$. La connaissance de la fonction f se résume à la connaissance du tableau multidimensionnel $\boldsymbol{\mathcal{A}}$.

IV.1.b Représentation discrète

La manipulation de fonction continue est faite numériquement via l'utilisation de représentations discrètes. Dans le cadre de cette thèse, on fait l'hypothèse que les détails de calcul de l'évaluation de la fonction que l'on cherche à approcher ne sont pas connus. On souhaite donc construire une approximation par une méthode non intrusive devant être basée uniquement sur l'évaluation de la fonction f en un certain nombre de points.

Pour représenter numériquement la fonction f, on introduit une discrétisation $D = D_1 \times \cdots \times D_d$ du domaine de définition avec pour tout $k \in [\![1, d]\!]$, D_k une discrétisation de

l'intervalle $\mathcal{D}_k = \left[x_k^{(min)}, x_k^{(max}\right]$ telle que :

$$D_k = \left\{x^{(i_k)} \in \mathbb{R} \mid \forall\, i_k \in I_k\right\} \quad \text{où} \quad I_k = [\![1, n_k]\!] \tag{IV.3}$$

avec

$$x_k^{(min)} = x_k^{(1)} < x_k^{(2)} < \cdots < x_k^{(n_k)} = x_k^{(max)} \tag{IV.4}$$

Sans information a priori sur la forme de la fonction f, il est naturel de choisir pour chaque espace E_k des fonctions de bases à support compact correspondant à des partitions de l'unité. On impose d'autre part, que les fonctions de bases soient, soit nulles soit égales à 1 aux points de discrétisation. Ces fonctions correspondent typiquement aux fonctions chapeaux utilisées dans la méthode des éléments finis.

Avec ce type de restriction, on peut montrer que le tenseur $\boldsymbol{\mathcal{A}}$ introduit dans la représentation tensorielle (Éq. (IV.2)) de la fonction f est tel que :

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d, j) = f_j\left(x_1^{(i_1)}, \ldots, x_d^{(i_d)}\right) \quad \forall\, (i_1, \ldots, i_d, j) \in I_1 \times \cdots \times I_d \times [\![1, m]\!] \tag{IV.5}$$

Finalement, via cette représentation, les techniques d'approximation tensorielle de faible rang peuvent être transposées pour approcher des fonctions continues à partir de leur connaissance uniquement sur un domaine discrétisé.

IV.1.c Représentation en train de tenseurs

En raison du fléau de la dimension, le stockage explicite du tenseur $\boldsymbol{\mathcal{A}}$ de la représentation (Éq. (IV.2)) est typiquement inenvisageable. Pour surmonter ce problème, on propose dans cette thèse de ne pas stocker le tenseur $\boldsymbol{\mathcal{A}}$, mais uniquement des approximations sous forme de décomposition TT telles que :

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d, j) = G_1(i_1) \ldots G_d(i_d) G_{d+1}(j)$$

On montre que dans ce cas, la représentation tensorielle (Éq. (IV.2)) de f se réécrit :

$$f_j = \left(f^{(1)} \otimes f^{(2)} \otimes \cdots \otimes f^{(d)}\right) g_j \quad \forall\, j \in [\![1, m]\!] \tag{IV.6}$$

où les *facteurs* $f^{(k)} : x_k \mapsto f^{(k)}(x_k) \in \mathbb{R}^{r_{k-1} \times r_k}$ sont des fonctions d'une variable à valeurs matricielles et où $g_j = G_{d+1}(j)$. Les opérations impliquées lors de l'évaluation des images sont à présent des multiplications matricielles. La correspondance entre les $f^{(k)}$ et les

facteurs G_k du train de tenseurs (Éq. (III.26)) est donnée par :

$$f^{(k)} = \sum_{i=1}^{n_k} G_k(i)\varphi_k^{(i)}$$

La formulation (Éq. (IV.6)) peut être interprétée comme la représentation continue d'un train de tenseurs.

Les décompositions en train de tenseurs des f_j se réécrivent :

$$f_j(x_1, \ldots, x_d) = f^{(1)}(x_1) \ldots f^{(d)}(x_d) g_j \tag{IV.7}$$

où les opérations impliquées entre les $f^{(k)}(x_k)$ sont des multiplications matricielles.

L'évaluation du train de tenseurs f_j pour un point $(x_1, \ldots, x_d)$ fixé implique $d-1$ multiplications matrice-vecteur et 1 produit scalaire entre deux vecteurs. Si les tailles des facteurs du train sont relativement faibles, l'évaluation de la fonction f peut être réalisée en temps réel. Ce type de format est donc particulièrement adapté pour être intégré dans le cadre de modèle de substitution.

IV.1.d Interpolation multilinéaire par morceaux

Par souci de simplicité, on se restreint dans cette section à des fonctions f à valeur scalaire. Lorsqu'aucune information a priori n'est connue sur une fonction f, l'interpolation multilinéaire par morceaux est la méthode la plus naturelle pour l'approcher lorsque l'on dispose uniquement de ses valeurs sur un domaine discrétisé $D = D_1 \times \cdots \times D_d$ (Éq. (IV.3)). Cette interpolation se définit comme la généralisation en dimension arbitraire des interpolations standard bi- et tri-linéaires par morceaux .

L'interpolation multilinéaire par morceaux $\widetilde{f}$ de la fonction f sur le domaine $\mathcal{D}$ est telle que :

$$\widetilde{f}(x_1, \ldots, x_d) = \sum_{\eta_1, \ldots, \eta_d = \pm 1} \left[f\left(\bar{x}_1^{-\eta_1}, \ldots, \bar{x}_d^{-\eta_d}\right) \prod_{k=1}^{d} w_{\eta_k}(x_k) \right] \tag{IV.8}$$

où pour tout $k \in [\![1, d]\!]$, $\bar{x}_k^{-1}, \bar{x}_k^{+1} \in D_k$ correspondent aux points respectivement inférieur et supérieur les plus proches de x_k. Pour $\eta_k = \pm 1$, w_{η_k} sont des fonctions affines d'une seule variable x_k :

$$\begin{array}{rccc} w_{\eta_k} & : & [\bar{x}_k^{-1}, \bar{x}_k^{+1}] & \to & \mathbb{R} \\ & & x_k & \mapsto & \dfrac{\eta_k \left(\bar{x}_k^{\eta_k} - x_k\right)}{\left(\bar{x}_k^{+1} - \bar{x}_k^{-1}\right)} \end{array} \tag{IV.9}$$

La dépendance implicite de $\bar{x}_k^{-1}$, $\bar{x}_k^{+1}$, w_- et w_+ en fonction de x_k n'est pas mentionnée

pour alléger les notations.

La formule de l'équation (IV.8) présente une structure tensorielle traduisant le fait que l'interpolation multilinéaire correspond au produit tensoriel d'interpolations linéaires indépendantes associées à chaque dimension.

Dans le cas où les valeurs de f sur une grille de discrétisation sont stockées dans un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ représenté par une décomposition en train de tenseurs (Éq. (III.26)), il est possible d'exploiter cette structure pour dériver une formule simplifiée pour l'interpolation multilinéaire par morceaux.

Proposition 2. *Soit f une fonction définie par l'équation (IV.1) sur $\mathcal{D}$ dont les valeurs sur le domaine discrétisé $D = D_1 \times \cdots \times D_d$ définissent un tenseur $\boldsymbol{\mathcal{A}}$ par l'équation (IV.5).*

L'interpolation multilinéaire par morceaux $\widetilde{f}$ de f définie sur le domaine continu $\mathcal{D}$ est égale à la décomposition en train de tenseurs continue :

$$\widetilde{f}(x_1, \ldots, x_d) = \widetilde{f}^{(1)}(x_1) \ldots \widetilde{f}^{(d)}(x_d) \tag{IV.10}$$

où pour tout $k \in [\![1, d]\!]$, les fonctions à valeurs matricielles $\widetilde{f}^{(k)}$ sont les interpolations linéaires par morceaux des fonctions $f^{(k)}$, i.e :

$$\widetilde{f}^{(k)}(x_k) = w_+(x_k) G_k\left(i_k^{-1}\right) + w_-(x_k) G_k\left(i_k^{+1}\right)$$

avec w_+, w_-, $\bar{x}_k^{-1}$ et $\bar{x}_k^{+1}$ définis comme dans l'équation (IV.9) et i_k^{-1}, i_k^{+1} tels que :

$$\bar{x}_k^{-1} = x_k^{(i_k^{-1})} \in D_k \quad \text{et} \quad \bar{x}_k^{+1} = x_k^{(i_k^{+1})} \in D_k$$

La démonstration de la proposition 2 est donnée en annexe B

En notant $r = \max(r_1, \ldots, r_{d-1})$, on a les complexités de calcul suivantes :

— Interpolation linéaire d'une matrice $\tilde{f}^{(k)}(x_k)$: $\mathcal{O}(r^2)$;

— Suite des produits matriciels de l'équation (IV.10) : $\mathcal{O}\left(dr^2\right)$;

D'après la proposition 2, la complexité pour évaluer $\widetilde{f}$ en un point quelconque de $\mathcal{D}$ est donc $\mathcal{O}\left(2dr^2\right)$, ce qui est très inférieur à la complexité d'une implémentation naïve de l'interpolation multilinéaire par morceaux : $\mathcal{O}\left(dr^2 2^d\right)$ (évaluation du train de tenseurs en 2^d valeurs). On note d'autre part qu'entre l'évaluation de la fonction $\tilde{f}$ sur le domaine continu et discrétisé, il n'y a qu'un facteur 2 sur la complexité. La formule (Éq. (IV.10)) représente donc une méthode d'interpolation efficace.

Remarque 12. *Le coût de calcul pour trouver les points $\bar{x}_k^{-1}$ et $\bar{x}_k^{+1}$, correspondant à une recherche de points dans un ensemble, n'est pas nécessairement totalement négligeable. En particulier, lorsque D_k est une grille non régulière (c'est à dire, avec un pas non*

constant) la complexité de calcul est $\mathcal{O}(\log n_k)$ (recherche dans une liste ordonnée de n_k éléments). Cependant, pour une grille régulière, une implémentation adaptée (c'est-à-dire un accès aléatoire) offre un coût indépendant de n_k.

IV.2 Algorithme générique de décomposition TT par approximations matricielles successives

D'après la section précédente, il est possible de représenter une fonction continue par un train de tenseurs représentant la valeur de la fonction sur un domaine discrétisé. Une telle représentation permet un stockage compact et des temps d'évaluation particulièrement faibles. Cette section est dédiée à la construction de telles représentations. Lorsque la fonction de référence est couteuse à évaluer, il n'est pas envisageable de calculer sa valeur sur l'ensemble du domaine discrétisé. L'objectif est donc d'en construire une approximation basée sur la connaissance de ses valeurs en un nombre réduit de points du domaine discrétisé.

On propose dans cette section un cadre algorithmique générique développé au cours de cette thèse pour construire des approximations de tenseurs multidimensionnels au format TT. Nous verrons dans la suite que cette formulation généralise la TT-SVD et la TT-cross et permet par ailleurs une extension à un algorithme efficace dans le cadre des applications (Chapitre VII).

IV.2.a Description de l'algorithme

L'algorithme présenté dans cette section repose essentiellement sur deux opérations : le réarrangement tensoriel et l'approximation matricielle de rang faible. L'idée essentielle consiste à construire successivement des approximations de rang faible sur différents réarrangements de tenseurs.

Soit un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ typiquement issu de la discrétisation d'une fonction et donnée par l'équation (IV.5). L'algorithme 10 génère un ensemble de matrices $\{H_1, \ldots, H_d\}$ définissant le train de tenseurs $\boldsymbol{\mathcal{T}}$ (Éq. (IV.11)) qui approche $\boldsymbol{\mathcal{A}}$ avec

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{H}}_1^\star \bullet \boldsymbol{\mathcal{H}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_{d-1} \bullet \boldsymbol{\mathcal{H}}_d^\star \in \mathbb{R}^{n_1 \times \cdots \times n_d} \tag{IV.11}$$

tel que pour tout $k \in [\![1, d]\!]$:

$$\boldsymbol{\mathcal{H}}_k = \underset{s_{k-1} \times n_k \times s_k}{\text{reshape}} (H_k) \in \mathbb{R}^{s_{k-1} \times n_k \times s_k} \tag{IV.12}$$

où $H_k \in \mathbb{R}^{(s_{k-1} n_k) \times s_k}$ est une matrice.

La figure IV.1 illustre le lien entre les matrices H_k et les tenseurs $\boldsymbol{\mathcal{H}}$.

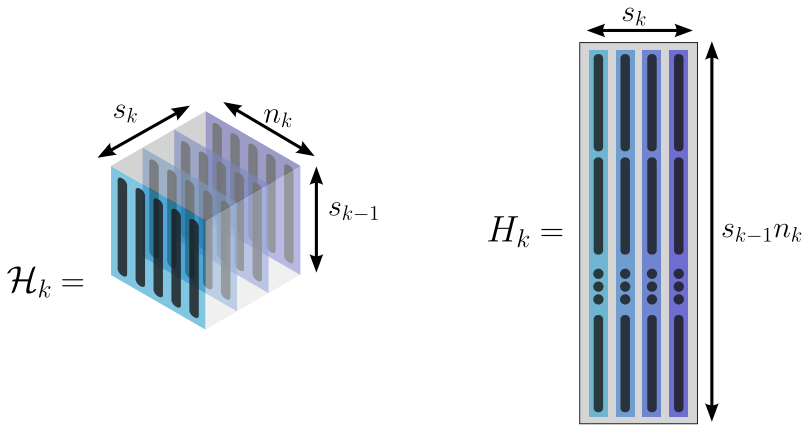


FIGURE IV.1 – Le tenseur $\boldsymbol{\mathcal{H}}_k$ et la matrice H_k.

Notons que la formulation générale de l'algorithme 10 laisse la liberté de considérer des décompositions matricielles (Éq. (IV.14)) différentes à chaque itération.

Remarque 13. *Pour $k = d-1$, l'opération de réarrangement (Éq. (IV.15)) retourne une matrice $A_d \in \mathbb{R}^{(s_{d-1}n_d)\times 1}$, par conséquent on choisit la convention $s_d = 1$ de telle sorte que $A_d \in \mathbb{R}^{(s_{d-1}n_d)\times s_d}$.*

Définition 5. *Avec les notations de l'algorithme 10, on définit pour tout $k \in [\![1, d]\!]$, le tenseur $\boldsymbol{\mathcal{A}}^{(k)} \in \mathbb{R}^{s_{k-1}\times n_k\times\cdots\times n_d}$ en inversant la relation :*

$$\left\langle \boldsymbol{\mathcal{A}}^{(k)} \right\rangle_2 = A_k \tag{IV.17}$$

Remarque 14. *D'après la définition 5 des tenseurs $\boldsymbol{\mathcal{A}}^{(k)}$ et l'étape de réarrangement (Éq. (IV.15)), on a pour tout $k \in [\![1, d-1]\!]$:*

$$\left\langle \boldsymbol{\mathcal{A}}^{(k+1)} \right\rangle_1 = \widehat{A}_k \in \mathbb{R}^{s_k\times(n_{k+1}\ldots n_d)} \tag{IV.18}$$

IV.2.b Erreur d'approximation

Pour $k \in [\![1, d]\!]$, on définit les matrices d'erreur $E_k \in \mathbb{R}^{(s_{k-1}n_k)\times(n_{k+1}\ldots n_d)}$ par :

$$E_k = A_k - T_k \tag{IV.19}$$

et les tenseurs correspondants :

$$\boldsymbol{\mathcal{E}}_k \in \mathbb{R}^{s_{k-1}\times n_k\times\cdots\times n_d}$$

avec

$$\boldsymbol{\mathcal{E}}_k = \underset{s_{k-1}\times n_k\times\cdots\times n_d}{\text{reshape}}\ (E_k) \tag{IV.20}$$

Algorithme 10 : Décomposition TT par approximations matricielles successives

Données : Un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

Résultat : L'ensemble de matrices $\{H_1, \ldots, H_d\}$ permettant de définir le train de tenseurs.

Initialisation

Définir la matrice $A_1 \in \mathbb{R}^{n_1 \times (n_2 \ldots n_d)}$ issue d'un réarrangement de $\boldsymbol{\mathcal{A}}$:

$$A_1 = \underset{n_1 \times (n_2 \ldots n_d)}{\text{reshape}} \left(\boldsymbol{\mathcal{A}}\right) \tag{IV.13}$$

Définir $s_0 = 1$.

Pour $k \in [\![1, d-1]\!]$ **faire**

Approximation de rang faible

Calculer une approximation T_k de rang r_k de A_k donnée par la décomposition :

$$T_k = H_k \widehat{A}_k \simeq A_k \tag{IV.14}$$

avec

$$H_k \in \mathbb{R}^{(s_{k-1} n_k) \times s_k} \quad \text{et} \quad \widehat{A}_k \in \mathbb{R}^{s_k \times (n_{k+1} \ldots n_d)}$$

tel que $s_k \geq r_k$.

Réarrangement tensoriel

Définir la matrice $A_{k+1} \in \mathbb{R}^{(s_k n_{k+1}) \times (n_{k+2} \ldots n_d)}$ à partir d'un réarrangement de $\widehat{A}_k$:

$$A_{k+1} = \underset{(s_k n_{k+1}) \times (n_{k+2} \ldots n_d)}{\text{reshape}} \left(\widehat{A}_k\right) \tag{IV.15}$$

Finalisation

Définir la matrice $H_d \in \mathbb{R}^{(s_{d-1} n_d) \times s_d}$ telle que :

$$H_d = A_d \tag{IV.16}$$

avec $s_d = 1$.

Remarque 15. *Pour $k = d$, on a $\boldsymbol{\mathcal{E}}_d = 0 \in \mathbb{R}^{s_{d-1} n_d \times 1}$ d'après l'égalité (IV.16).*

On établit la propriété suivante :

Proposition 3. *L'erreur d'approximation entre le tenseur de référence $\boldsymbol{\mathcal{A}}$ et son approximation en train de tenseurs $\boldsymbol{\mathcal{T}}$ obtenue par l'algorithme 10 est donnée par :*

$$\begin{aligned} \boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}} = \quad & \boldsymbol{\mathcal{H}}_1^{\star} \bullet \boldsymbol{\mathcal{H}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_{d-2} \bullet \boldsymbol{\mathcal{E}}_{d-1} \\ & + \boldsymbol{\mathcal{H}}_1^{\star} \bullet \boldsymbol{\mathcal{H}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_{d-3} \bullet \boldsymbol{\mathcal{E}}_{d-2} \\ & + \ldots \\ & + \boldsymbol{\mathcal{H}}_1^{\star} \bullet \boldsymbol{\mathcal{E}}_2 \\ & + \boldsymbol{\mathcal{E}}_1^{\star} \end{aligned} \tag{IV.21}$$

Ce résultat prouve qu'il est possible de représenter n'importe quel tenseur par une décomposition en train de tenseurs. En effet, si à chaque itération on construit une approximation telle que $T_k = A_k$ alors $\boldsymbol{\mathcal{E}}_k$ est nul et on obtient $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{T}}$.

Démonstration de la proposition 3.

La démonstration repose sur le lemme 2 donné en annexe. Pour $k \in [\![1, d-1]\!]$, on définit

$$\boldsymbol{\mathcal{H}}_{(k)} = \boldsymbol{\mathcal{H}}_1 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_k \tag{IV.22}$$

et par convention $\boldsymbol{\mathcal{H}}_{(0)} = \mathbb{I}$

Montrons que pour tout $k \in [\![1, d-1]\!]$, la proposition $\mathcal{P}_k$ suivante est vraie :

$$\boldsymbol{\mathcal{A}}^{(1)} = \boldsymbol{\mathcal{H}}_{(k)} \bullet \boldsymbol{\mathcal{A}}^{(k+1)} + \sum_{k'=0}^{k-1} \boldsymbol{\mathcal{H}}_{(k')} \bullet \boldsymbol{\mathcal{E}}_{k'+1} \tag{IV.23}$$

Initialisation : Pour $k = 1$, le lemme 2 donne :

$$\boldsymbol{\mathcal{A}}^{(1)} = \boldsymbol{\mathcal{H}}_1 \bullet \boldsymbol{\mathcal{A}}^{(2)} + \boldsymbol{\mathcal{E}}_1 \tag{IV.24}$$

$$= \boldsymbol{\mathcal{H}}_{(1)} \bullet \boldsymbol{\mathcal{A}}^{(2)} + \boldsymbol{\mathcal{H}}_{(0)} \bullet \boldsymbol{\mathcal{E}}_1 \qquad \text{d'après (Éq. (IV.22))} \tag{IV.25}$$

Par conséquent $\mathcal{P}_1$ est vraie.

Récurrence : On suppose que $\mathcal{P}_k$ est vraie pour $k \in [\![1, d-2]\!]$ fixé. Montrons que

$\mathcal{P}_{k+1}$ est vraie. On a :

$$\begin{aligned}
\boldsymbol{\mathcal{A}}^{(1)} &= \boldsymbol{\mathcal{H}}_{(k)} \bullet \boldsymbol{\mathcal{A}}^{(k+1)} + \sum_{k'=0}^{k-1} \boldsymbol{\mathcal{H}}_{(k')} \bullet \boldsymbol{\mathcal{E}}_{k'+1} && \text{car } \mathcal{P}_k \text{ vraie} \\
&= \boldsymbol{\mathcal{H}}_{(k)} \bullet \left[\boldsymbol{\mathcal{H}}_{k+1} \bullet \boldsymbol{\mathcal{A}}^{(k+2)} + \boldsymbol{\mathcal{E}}_{k+1}\right] + \sum_{k'=0}^{k-1} \boldsymbol{\mathcal{H}}_{(k')} \bullet \boldsymbol{\mathcal{E}}_{k'+1} && \text{d'après le lemme 2} \\
&= \boldsymbol{\mathcal{H}}_{(k)} \bullet \boldsymbol{\mathcal{H}}_{k+1} \bullet \boldsymbol{\mathcal{A}}^{(k+2)} + \boldsymbol{\mathcal{H}}_{(k)} \bullet \boldsymbol{\mathcal{E}}_{k+1} + \sum_{k'=0}^{k-1} \boldsymbol{\mathcal{H}}_{(k')} \bullet \boldsymbol{\mathcal{E}}_{k'+1} \\
&= \boldsymbol{\mathcal{H}}_{(k+1)} \bullet \boldsymbol{\mathcal{A}}^{(k+2)} + \sum_{k'=0}^{k} \boldsymbol{\mathcal{H}}_{(k')} \bullet \boldsymbol{\mathcal{E}}_{k'+1}
\end{aligned}$$

Donc $\mathcal{P}_{k+1}$ est vraie.

La propriété $\mathcal{P}_k$ est vraie pour tout $k \in [1 : d-1]$ et en particulier, on a pour $k = d-1$:

$$\boldsymbol{\mathcal{A}}^{(1)} = \boldsymbol{\mathcal{H}}_{(d-1)} \bullet \boldsymbol{\mathcal{A}}^{(d)} + \sum_{k'=0}^{d-2} \boldsymbol{\mathcal{H}}_{(k')} \bullet \boldsymbol{\mathcal{E}}_{k'+1} \tag{IV.26}$$

Par ailleurs, on a :

$$\boldsymbol{\mathcal{A}}^{(d)} = \boldsymbol{\mathcal{H}}_d^{\star} \tag{IV.27}$$

Les équations (IV.26) et (IV.27) donnent donc :

$$\begin{aligned}
\boldsymbol{\mathcal{A}}^{(1)} = \quad & \boldsymbol{\mathcal{H}}_{(d-1)} \bullet \boldsymbol{\mathcal{H}}_d^{\star} \\
& + \boldsymbol{\mathcal{H}}_{(d-2)} \bullet \boldsymbol{\mathcal{E}}_{d-1} \\
& + \dots \\
& + \boldsymbol{\mathcal{H}}_{(0)} \bullet \boldsymbol{\mathcal{E}}_1
\end{aligned} \tag{IV.28}$$

D'après la définition de $\boldsymbol{\mathcal{T}}$ (Éq. (IV.11)) :

$$\boldsymbol{\mathcal{T}} = \left(\boldsymbol{\mathcal{H}}_{(d-1)} \bullet \boldsymbol{\mathcal{H}}_d^{\star}\right)^{\star} \tag{IV.29}$$

On obtient finalement :

$$\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}} = \sum_{k=1}^{d-1} \left[\boldsymbol{\mathcal{H}}_{(k-1)} \bullet \boldsymbol{\mathcal{E}}_k\right]^{\star} \tag{IV.30}$$

□

On établit un corollaire (Proposition 4) de la proposition 3 qui donne une borne

supérieure sur la norme de l'erreur d'approximation.

Proposition 4. *Avec les hypothèses de la proposition 3 :*

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sum_{k=1}^{d-1} \prod_{k'=1}^{k-1} \|H_{k'}\|_2 \, \|E_k\| \tag{IV.31}$$

Démonstration. L'inégalité triangulaire de la norme de Frobenius appliquée à l'équation (IV.30) donne d'après l'indépendance de la norme par réarrangement :

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sum_{k=1}^{d-1} \left\| \boldsymbol{\mathcal{H}}_{(k-1)} \bullet \boldsymbol{\mathcal{E}}_k \right\| \tag{IV.32}$$

En appliquant le lemme 1 à $\left\| \boldsymbol{\mathcal{H}}_{(k-1)} \bullet \boldsymbol{\mathcal{E}}_k \right\|$ de manière récursive, on obtient :

$$\left\| \boldsymbol{\mathcal{H}}_{(k-1)} \bullet \boldsymbol{\mathcal{E}}_k \right\| \leq \|\langle \boldsymbol{\mathcal{H}}_1 \rangle_2\|_2 \, \|\langle \boldsymbol{\mathcal{H}}_2 \rangle_2\|_2 \ldots \|\langle \boldsymbol{\mathcal{H}}_{k-1} \rangle_2\|_2 \, \|E_k\|$$

Par définition de $\boldsymbol{\mathcal{H}}_k$ (Éq. (IV.12)), on a :

$$\left\| \boldsymbol{\mathcal{H}}_{(k-1)} \bullet \boldsymbol{\mathcal{E}}_k \right\| \leq \|H_1\|_2 \, \|H_2\|_2 \ldots \|H_{k-1}\|_2 \, \|E_k\|$$

Finalement, l'équation (IV.32) devient :

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sum_{k=1}^{d-1} \|H_1\|_2 \ldots \|H_{k-1}\|_2 \, \|E_k\|$$

□

Lors de la construction des approximations matricielles, à chaque itération, on peut fixer une tolérance d'erreur ϵ_k telle que $\|E_k\| \leq \epsilon_k$. La proposition 4 suggère que l'erreur d'approximation de la décomposition TT peut être d'une certaine manière contrôlée par les tolérances ϵ_k. Comme les matrices H_k dépendent également de l'approximation matricielle et donc de la tolérance ϵ_k, la proposition ne suffit pas à affirmer une convergence sur l'erreur d'approximation TT. On observe toutefois en pratique une convergence de l'erreur vis-à-vis des tolérances ϵ_k (Chapitre VII). Cette borne d'erreur est affinée pour une spécification particulière de l'algorithme 10 en proposition 11.

IV.2.c Algorithme de refactorisation

Selon les décompositions matricielles (Éq. (IV.14)) sélectionnées dans l'algorithme 10, la décomposition TT obtenue, notée $\boldsymbol{\mathcal{T}}$, n'est pas nécessairement minimale, c'est-à-dire telle que les tailles de stockage ne correspondent pas aux rangs de compression. Cela se produit en particulier lorsque les décompositions matricielles sont telles que $r_k < s_k$.

Tout tenseur admet théoriquement une décomposition TT minimale. Cela signifie qu'il existe une décomposition alternative de $\boldsymbol{\mathcal{T}}$ dont les facteurs possèdent moins d'éléments et donc plus compactes. On propose une procédure de refactorisation (Section IV.2.c) qui permet de calculer à partir de $\boldsymbol{\mathcal{T}}$ une décomposition *refactorisée* plus compacte qui n'est pas nécessairement une décomposition minimale.

La décomposition refactorisée dépend uniquement des sorties H_k de l'algorithme 10 et par conséquent aucun calcul supplémentaire d'éléments du tenseur de référence $\boldsymbol{\mathcal{A}}$ n'est nécessaire. Cette décomposition est d'autre part équivalente à la décomposition initiale au sens où leurs évaluations coïncident exactement (au moins en arithmétique exacte).

L'approximation de rang faible (Éq. (IV.14)) fait apparaitre une matrice $H_k \in \mathbb{R}^{(s_{k-1}n_k)\times s_k}$ de rang $r_k \leq s_k$. Il existe, par conséquent, deux matrices $K_k \in \mathbb{R}^{(s_{k-1}n_k)\times r_k}$ et $L_k \in \mathbb{R}^{r_k \times s_k}$ telles que :

$$H_k = K_k L_k \tag{IV.33}$$

Les sorties H_k de l'algorithme 10 ne sont pas nécessairement données explicitement, mais peuvent aussi être retournées sous forme de décompositions matricielles de la forme (Éq. (IV.33)). Cette décomposition particulière peut apparaitre au cours de la procédure de calcul de l'approximation de rang faible (Éq. (IV.14)). Lorsque c'est le cas, il est préférable de conserver cette décomposition pour appliquer dans un second temps la procédure de refactorisation présentée dans la suite. Dans le cas contraire, une fois l'algorithme de construction terminé, il est également possible de calculer les décompositions matricielles sous la forme (Éq. (IV.33)).

Pour $k \in [\![1, d-1]\!]$, on peut définir à partir des matrices K_k et L_k les tenseurs $\boldsymbol{\mathcal{K}}_k \in \mathbb{R}^{s_{k-1}\times n_k \times r_k}$ et $\boldsymbol{\mathcal{L}}_k \in \mathbb{R}^{r_k \times s_k}$ tels que :

$$K_k = \langle \boldsymbol{\mathcal{K}}_k \rangle_2 \quad \text{et} \quad L_k = \boldsymbol{\mathcal{L}}_k$$

On a donc :

$$\boldsymbol{\mathcal{H}}_k = \boldsymbol{\mathcal{K}}_k \bullet \boldsymbol{\mathcal{L}}_k \tag{IV.34}$$

En injectant l'équation (IV.34) dans l'équation (IV.11), on obtient :

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{K}}_1^\star \bullet \boldsymbol{\mathcal{L}}_1 \bullet \boldsymbol{\mathcal{K}}_2 \bullet \boldsymbol{\mathcal{L}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{K}}_{d-1} \bullet \boldsymbol{\mathcal{L}}_{d-1} \bullet \boldsymbol{\mathcal{H}}_d^\star \tag{IV.35}$$

La refactorisation consiste à définir les tenseurs $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ tels que :

$$\boldsymbol{\mathcal{G}}_1 = \boldsymbol{\mathcal{K}}_1 \tag{IV.36}$$

$$\boldsymbol{\mathcal{G}}_k = \boldsymbol{\mathcal{L}}_{k-1} \bullet \boldsymbol{\mathcal{K}}_k \text{ pour } k \in [\![2, d-1]\!] \tag{IV.37}$$

$$\boldsymbol{\mathcal{G}}_d = \boldsymbol{\mathcal{L}}_{d-1} \bullet \boldsymbol{\mathcal{H}}_d \tag{IV.38}$$

D'après les équations (IV.35), (IV.36), (IV.37) et (IV.38), on a finalement :

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}}_1^\star \bullet \boldsymbol{\mathcal{G}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{G}}_{d-1} \bullet \boldsymbol{\mathcal{G}}_d^\star \tag{IV.39}$$

La procédure de refactorisation donne donc bien une représentation TT exacte de $\boldsymbol{\mathcal{T}}$.

IV.3 Particularisation de l'algorithme de décomposition TT par approximations matricielles successives

Les efforts de calcul impliqués dans l'algorithme 10 correspondent principalement aux constructions des approximations (Éq. (IV.14)) des matrices A_k. Elles impliquent des opérations de décomposition matricielle, mais également des évaluations du tenseur de référence $\boldsymbol{\mathcal{A}}$. La construction d'une approximation TT s'appuie en effet fondamentalement sur la connaissance d'un certain nombre d'éléments du tenseur de référence.

Dans l'algorithme 10 la définition récursive des matrices A_k ne fait pas apparaitre clairement leur lien avec le tenseur de référence $\boldsymbol{\mathcal{A}}$. Pourtant, en pratique les accès au tenseur de référence peuvent être déterminants en termes de temps de calcul. L'objectif de l'algorithme 11 est de spécialiser l'algorithme 10 pour faire apparaitre clairement ce lien de sorte que les éléments du tenseur $\boldsymbol{\mathcal{A}}$ qui doivent être calculés au cours des itérations soient explicitement désignés.

À partir d'un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, l'algorithme 11 génère un ensemble de matrices $\{H_1, \ldots, H_d\}$ définissant le train de tenseurs approché $\boldsymbol{\mathcal{T}}$ défini par l'équation (IV.11). Notons que les résultats de la section IV.2.b relatifs aux erreurs restent valables et la procédure de refactorisation est toujours applicable.

Il est essentiel de comprendre que l'approximation TT fournie par les algorithmes 10 et 11 dépend du type d'approximation matricielle, mais aussi de la forme de la décomposition utilisée, c'est-à-dire du découpage entre les matrices H_k et $\widehat{A}_k$ qui détermine :

— la partie H_k qui sera retournée en sortie de l'algorithme ;

— la partie $\widehat{A}_k$ qui sera à approcher à l'itération suivante.

La décomposition a donc un effet sur les données qui seront calculées et traitées au cours de l'algorithme ainsi que sur le résultat final de l'approximation. La différence entre les algorithmes 10 et 11 concerne précisément la forme de cette décomposition. En

Algorithme 11 : Décomposition TT par approximations matricielles successives spécifiques

Données : Un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ dont tous les éléments sont accessibles.
Résultat : Un ensemble de matrices $\{H_1, \ldots, H_d\}$.
Initialisation

Définir la matrice $A_1 \in \mathbb{R}^{n_1 \times (n_2 \ldots n_d)}$ comme le premier déploiement du tenseur $\boldsymbol{\mathcal{A}}$:

$$A_1 = \langle \boldsymbol{\mathcal{A}} \rangle_1 \tag{IV.40}$$

Définir $s_0 = 1$.

Pour $k \in [\![1, d-1]\!]$ **faire**

Approximation de faible rang

Construire une approximation T_k de rang r_k de A_k donnée sous la forme :

$$T_k = H_k P_k^T A_k \simeq A_k \tag{IV.41}$$

où $H_k \in \mathbb{R}^{(s_{k-1} n_k) \times s_k}$ est de rang r_k, $P_k \in \mathbb{R}^{(s_{k-1} n_k) \times s_k}$ possède des colonnes orthogonales et $s_k \geq r_k$.

Réarrangement tensoriel

Définir la matrice $A_{k+1} \in \mathbb{R}^{(s_k n_{k+1}) \times (n_{k+2} \ldots n_d)}$ par :

$$A_{k+1} = \underset{(s_k n_{k+1}) \times (n_{k+2} \ldots n_d)}{\text{reshape}} \left(P_k^T A_k \right) \tag{IV.42}$$

Finalisation

Définir la matrice $H_d \in \mathbb{R}^{(s_{d-1} n_d) \times s_d}$ telle que :

$$H_d = A_d \tag{IV.43}$$

avec $s_d = 1$.

particulier, la décomposition utilisée dans l'algorithme 11 contraint la forme de la matrice $\widehat{A}_k$ transportée à l'itération suivante. La matrice $\widehat{A}_k = P_k^T A_k$ correspond au produit de la transposée d'une matrice de colonnes orthogonales et de la matrice de l'itération courante A_k.

Les choix de l'approximation matricielle et du découpage influencent d'une part le nombre d'accès au tenseur de référence et la qualité de l'approximation et d'autre part l'espace mémoire sollicité et les temps de calcul engendrés. Ils déterminent donc totalement la possibilité d'appliquer en pratique l'algorithme. Il s'agit donc de faire des choix judicieux parmi les nombreuses alternatives possibles. L'objet du chapitre suivant est justement de discuter des différentes options.

Décomposition qui s'appuie sur une sélection de lignes Comme pour la TT-cross (Section III.5.d), on considère à présent un cas particulier où P_k représente une matrice de sélection associée aux s_k indices de lignes $\mathcal{I}_k$. À l'itération $k \in [\![1, d]\!]$ la matrice $P_k^T A_k$ correspond à un sous-ensemble de lignes de la matrice A_k :

$$P_k^T A_k = A_k(\mathcal{I}_k, :) \tag{IV.44}$$

L'étape de réarrangement (Éq. (IV.42)) s'écrit donc :

$$A_{k+1} = \underset{(s_k n_{k+1}) \times (n_{k+2} \dots n_d)}{\text{reshape}} \left(A_k(\mathcal{I}_k, :) \right) \tag{IV.45}$$

On observe que dans ce cas, les matrices A_k sont définies de manière récursive par sélection de lignes et réarrangement. Cela suggère que les matrices A_k correspondent à des réarrangements de sélection de sous-tenseurs du tenseur de référence $\boldsymbol{\mathcal{A}}$. La proposition 5 explicite ce lien.

Proposition 5. *Avec les notations de l'algorithme 11, pour tout $k \in [\![1, d]\!]$:*

$$A_k = \left\langle \boldsymbol{\mathcal{A}}(\mathcal{I}_{k-1}, I_k, \dots, I_d) \right\rangle_2 \tag{IV.46}$$

avec $I_i = [\![1, n_i]\!]$ qui indique que les éléments associés à tous les indices de la dimension i sont sélectionnés. On définit par convention $\mathcal{I}_0 = \{1\}$.

Et où les sélections de multi-indices $\mathcal{I}_k$ sont définies de manière récursive par

$$\mathcal{I}_k \subset \mathcal{I}_{k-1} \times I_k \subset I_1 \times \cdots \times I_k$$

en adoptant l'identification de la remarque 11.

Remarque 16. *Afin que la proposition 5 soit valide pour $k = 1$, on adopte la convention où $\boldsymbol{\mathcal{A}}(\{1\}, I_1, \dots, I_d)$ correspond au tenseur $\boldsymbol{\mathcal{A}}$ auquel une dimension de taille 1 a été*

ajoutée.

Remarque 17. *Si le tenseur $\boldsymbol{\mathcal{A}}$ que l'on cherche à approcher est issu de la discrétisation d'une fonction multivariée à valeurs vectorielles $f = (f_1, \ldots, f_m)$ tel que :*

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) = f_{i_d}\left(x_1^{(i_1)}, \ldots, x_{d-1}^{(i_{d-1})}\right)$$

alors la construction des matrices A_k revient à évaluer f pour une sélection réalisée sur ses $k-1$ premières variables. Ainsi, plus le processus progresse et plus on restreint le domaine d'évaluation de f.

Démonstration de la proposition 5.

On démontre par récurrence la proposition notée $\mathcal{P}_k$ pour $k \in [\![1, d]\!]$ fixé.

Initialisation : Par définition (Éq. (IV.40)) :

$$A_1 = \langle \boldsymbol{\mathcal{A}} \rangle_1 = \langle \boldsymbol{\mathcal{A}}(I_1, \ldots, I_d) \rangle_1 = \langle \boldsymbol{\mathcal{A}}(\{1\}, I_1, \ldots, I_d) \rangle_2$$

Donc $\mathcal{P}_1$ est vraie avec $\mathcal{I}_0 = \{1\}$

Récurrence : Pour $k \in [\![1, d-1]\!]$ fixé, on suppose $\mathcal{P}_k$ vraie. D'après l'hypothèse de récurrence :

$$A_k(\mathcal{I}_k, :) = \left\langle \boldsymbol{\mathcal{A}}(\mathcal{I}_{k-1}, I_k, \ldots, I_d) \right\rangle_2 (\mathcal{I}_k, :)$$

La sélection de s_k d'indices lignes $\mathcal{I}_k$ du second déploiement de $\boldsymbol{\mathcal{A}}(\mathcal{I}_{k-1}, I_k, \ldots, I_d)$ constitué de $s_{k-1}n_k$ lignes peut être interprétée comme une sélection de multi-indices parmi $\mathcal{I}_{k-1} \times I_k \subset I_1 \times \cdots \times I_k$. Plus précisément :

$$\mathcal{I}_{k-1} \times I_k = \left\{ \left(i_1^{(\alpha)}, \ldots, i_{k-1}^{(\alpha)}, i_k \right) \mid \forall\, (\alpha, i_k) \in [\![1, s_{k-1}]\!] \times [\![1, n_k]\!] \right\}$$

et la sélection d'indices de lignes $\mathcal{I}_k \subset [\![1, s_{k-1}n_k]\!]$ associée à la matrice P_k est identifiée à la sélection de multi-indices :

$$\mathcal{I}_k = \left\{ \left(i_1^{(\beta)}, \ldots, i_{k-1}^{(\beta)}, i_k^{(\beta)} \right) \mid \forall\, \beta \in [\![1, s_k]\!] \right\} \subset \mathcal{I}_{k-1} \times I_k$$

On a donc

$$A_k(\mathcal{I}_k, :) = \left\langle \boldsymbol{\mathcal{A}}(\mathcal{I}_k, I_{k+1}, \ldots, I_d) \right\rangle_2$$

Finalement, d'après l'équation (IV.45) :

$$\begin{aligned} A_{k+1} &= \underset{(s_k n_{k+1}) \times (n_{k+2} \dots n_d)}{\text{reshape}} \left(\left\langle \boldsymbol{\mathcal{A}}(\mathcal{I}_k, I_{k+1}, \dots, I_d) \right\rangle_2 \right) \\ &= \left\langle \boldsymbol{\mathcal{A}}(\mathcal{I}_k, I_{k+1}, \dots, I_d) \right\rangle_2 \end{aligned}$$

et donc $\mathcal{P}_{k+1}$ est vraie. □

Notons que les ensembles d'indices, définis de manière récursive, sont emboîtés à gauche (*left-nested*), c'est-à-dire que pour tout $k \in [\![1, d]\!]$:

$$(i_1, \dots, i_k, i_{k+1}) \in \mathcal{I}_{k+1} \Rightarrow (i_1, \dots, i_k) \in \mathcal{I}_k$$

On propose dans la proposition 6 une formule équivalente à celle de la proposition 5 qui met en évidence le lien entre les matrices A_k et le déploiement matriciel du tenseur $\boldsymbol{\mathcal{A}}$.

Proposition 6. *Avec les notations de l'algorithme 11, on a pour tout $k \in [\![1, d]\!]$:*

$$A_k = \langle \boldsymbol{\mathcal{A}} \rangle_k (\widetilde{\mathcal{I}}_k, :) \tag{IV.47}$$

avec

$$\widetilde{\mathcal{I}}_k = \mathcal{I}_{k-1} \times I_k$$

où à nouveau la sélection d'indices $\widetilde{\mathcal{I}}_k \subset [\![1, n_1]\!] \times \cdots \times [\![1, n_k]\!]$ et identifiée à une sélection d'indices parmi $[\![1, n_1 \dots n_k]\!]$.

La figure IV.2 illustre la définition de la matrice A_k via la sélection de lignes du $k^{\text{ème}}$ déploiement de $\boldsymbol{\mathcal{A}}$.

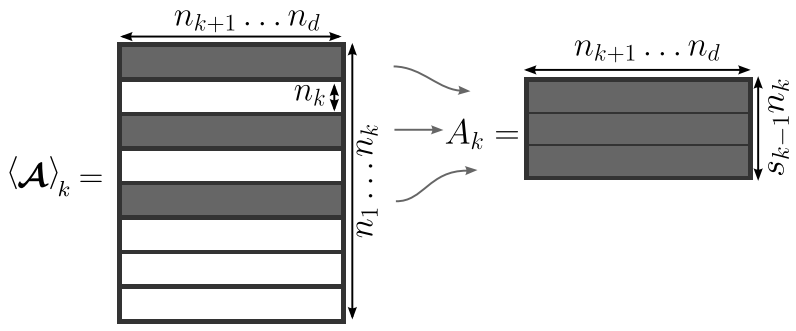


FIGURE IV.2 – Définition de A_k à partir de $\boldsymbol{\mathcal{A}}$ d'après la proposition 6. Dans l'illustration, le nombre de lignes sélectionnées à l'itération précédente $k-1$ est $s_{k-1} = 3$.

En choisissant P_k comme une matrice de sélection de lignes, l'approximation matricielle (Éq. (IV.41)) de l'étape k consiste finalement à approcher une sélection de lignes du $k^{\text{ème}}$

déploiement matriciel du tenseur $\boldsymbol{\mathcal{A}}$. L'approximation de faible rang s'écrit donc :

$$T_k = H_k \left[\langle \boldsymbol{\mathcal{A}} \rangle_k \left(\overline{\mathcal{I}}_k, : \right) \right]$$

L'intérêt de cette approche est d'exprimer explicitement l'ensemble des éléments du tenseur $\boldsymbol{\mathcal{A}}$ qui doivent être pris en compte à chaque itération.

Chapitre V

Spécialisation de l'algorithme générique

Les algorithmes 10 et 11 définissent des procédures générales, mais masquent les éléments fondamentaux (approximation matricielle de faible rang et décomposition) pourtant déterminants vis-à-vis de leur l'opérabilité pratique. L'objet de ce chapitre est de présenter différentes alternatives et les spécialisations correspondantes du cadre algorithmique commun.

Table des matières

V.1 Approximation matricielle 98
V.1.a Formulation particulière 98
V.2 Spécialisations préliminaires 99
V.2.a TT-SVD 99
V.2.b TT-POD 100
V.2.c TT-PSD 101
V.3 TT-gappy 102
V.3.a Décomposition matricielle 103
V.3.b Décomposition tensorielle 105
V.3.c Détails algorithmiques 107
V.4 Application pratique 109
V.5 Interprétation en termes de projecteur 111
V.5.a Matrice de projection 111
V.5.b Formulation en termes de projecteurs 112
V.5.c Principe général d'approximation 113
V.6 Intérêt de la procédure de refactorisation 114

V.1 Approximation matricielle

Pour une matrice $A \in \mathbb{R}^{n\times m}$, le problème d'approximation de rang faible se formule de manière informelle sous la forme : trouver une matrice $T \in \mathbb{R}^{n\times m}$ de rang r "faible" telle que :

$$A = T + E$$

tel que la norme $\|E\|$ de l'erreur d'approximation $E \in \mathbb{R}^{n\times m}$ soit "petite".

Plus spécifiquement, il peut être donné sous la forme d'un problème de minimisation : trouver la matrice $T \in \mathbb{R}^{n\times m}$ solution de :

$$\|A - T\| = \min_{X\in\mathbb{T}} \|A - X\| \tag{V.1}$$

où $\mathbb{T}$ est un ensemble de matrices de rang r choisi a priori.

Déterminer numériquement le rang d'une matrice est déjà un problème difficile, même si des algorithmes d'estimation existent tels que la SVD (Section III.3.b) ou la décomposition QR avec révélation de rang [46] (*rank revealing QR decomposition*). Déterminer un rang de troncature judicieux permettant d'obtenir une approximation matricielle suffisamment précise est un problème radicalement plus difficile et dépend étroitement de l'usage fait de l'approximation.

Il n'est pas nécessairement possible, voire souhaitable, de trouver la matrice qui satisfasse l'optimalité du problème (Éq. (V.1)), entre autres pour les raisons suivantes :

— La recherche de la matrice optimale est trop couteuse ;
— Le stockage de la matrice optimale est trop couteux.

D'autre part, lorsque le rang de troncature n'est pas connu, il est possible de résoudre ce qu'on définit comme le problème dual : trouver la matrice $T \in \mathbb{T}$ de petit rang r telle que :

$$\|A - T\| \leq \epsilon \tag{V.2}$$

où $\mathbb{T}$ est un ensemble de matrices choisi a priori et ϵ est une tolérance d'approximation fixée au préalable.

V.1.a Formulation particulière

On rappelle la forme particulière de la décomposition matricielle adoptée dans l'algorithme 11. Pour une matrice $A \in \mathbb{R}^{n\times m}$, on cherche une approximation T de rang r sous la

forme :

$$T = HP^T A \tag{V.3}$$

où $H \in \mathbb{R}^{n\times s}$ est de rang r avec $s \leq r$ et $P \in \mathbb{R}^{n\times s}$ possède des colonnes orthogonales.

Nous nous intéressons au cas particulier où $P \in \mathbb{R}^{n\times s}$ est une matrice de sélection de lignes associée à un ensemble $\mathcal{I}$ de s indices de lignes avec $s \leq n$. La décomposition met en évidence que l'approximation T est basée uniquement sur la connaissance partielle de A aux indices $\mathcal{I}$ (la matrice $A(\mathcal{I}, :)$).

Lorsque les P_k représentent des matrices de sélection, les matrices A_k sont constituées d'éléments du tenseur de référence $\boldsymbol{\mathcal{A}}$. Ces matrices peuvent être particulièrement grandes de sorte que ni leur calcul ni leur stockage complet soient envisageables. La construction d'une approximation n'implique cependant pas nécessairement l'accès à tous les éléments des matrices A_k.

Pour comprendre, on propose une interprétation alternative de l'approximation matricielle. La décomposition (Éq. (V.3)) exprime que tout $j \in [\![1, m]\!]$, on est capable de construire une approximation $T(:, j)$ du vecteur colonne $A(:, j)$ sous la forme :

$$T(:, j) = HP^T A(:, j)$$

En d'autres termes, on est capable d'approcher le vecteur colonne $A(:, j)$ à partir de la connaissance partielle de ce vecteur aux indices $\mathcal{I}$: $A(\mathcal{I}, j)$.

Cette formulation permet de comprendre que si l'on dispose effectivement de H (stocké en mémoire) et que l'on est capable d'obtenir à la volée des lignes particulières de la matrice A, alors on est capable d'approcher n'importe quelle colonne de A. Finalement, une approximation matricielle peut être formulée de manière abstraire uniquement à partir du stockage de la matrice H et de la liste d'indices $\mathcal{I}$.

V.2 Spécialisations préliminaires

On détaille dans cette section concrètement trois choix d'approximations/décompositions matricielles aboutissant à des spécialisations différentes de l'algorithme 11.

V.2.a TT-SVD

La technique canonique consiste à considérer la décomposition SVD. Pour une matrice $A \in \mathbb{R}^{n\times m}$, l'approximation T_{svd} obtenue par SVD tronquée de rang r (Section III.3.b)

s'écrit :

$$A = \underbrace{V_{\text{svd}}\Sigma_{\text{svd}}W_{\text{svd}}^T}_{=T_{\text{svd}}} + E_{\text{svd}} \quad \text{avec} \quad \|E_{\text{svd}}\| \leq \epsilon_{\text{svd}} \|A\| \tag{V.4}$$

où ϵ_{svd} est la tolérance de troncature.

L'algorithme 11 se spécialise en la TT-SVD (Algorithm 8) avec les substitutions matricielles suivantes :

$$T = T_{\text{svd}} \quad \text{et} \quad H = P = V_{\text{svd}} \tag{V.5}$$

Contrairement aux autres approximations/décompositions présentées dans la suite, la matrice P_k n'est ici pas une matrice de sélection de lignes. Par conséquent, à chaque itération l'ensemble des éléments du tenseur $\boldsymbol{\mathcal{A}}$ sont impliqués dans la construction de l'approximation de A_k. Les coûts de calcul et stockage correspondant rendent ainsi la TT-SVD non applicable en grande dimension.

Une solution partagée par les approximations de rang faible qui suivent consiste à utiliser uniquement une exploration parcimonieuse du tenseur de référence pour conserver des coûts raisonnables.

V.2.b TT-POD

Pour aboutir à une méthode plus parcimonieuse que la TT-SVD, une idée intuitive consiste à se passer de la matrice complète A pour construire la base POD et à modifier la forme de la décomposition en choisissant P comme une matrice de sélection de lignes. Une base réduite POD approchée de rang r peut être obtenue via la méthode de Snapshot POD (Section III.1.a). La matrice de base réduite V est construite en appliquant la SVD tronquée sur une sous-matrice $\widetilde{A} = A_k(:, \mathcal{J}_{\text{pod}})$ constituée d'une sélection de colonnes $\mathcal{J}_{\text{pod}}$ de la matrice A. La précision de la base POD résultante repose alors sur la qualité l'échantillonnage de colonnes.

L'utilisation de la Snapshot POD permet de spécialiser l'algorithme 11 en l'algorithme *TT-POD* avec les substitutions suivantes :

$$T = T_{\text{pod}}, \quad H = V_{\text{pod}}V_{\text{pod}}^T, \quad \text{et} \quad P = \mathbb{I}$$

où V_{pod} correspond à la matrice des vecteurs singuliers à gauche de la SVD tronquée au rang r de $\widetilde{A}$. Notons que la différence entre la TT-SVD et TT-POD ne provient pas du type d'approximation matricielle utilisé, mais de la procédure de construction et la forme de la décomposition.

Cette approche utilise une matrice de sélection égale à l'identité. La définition récursive

des matrices A_k indique que pour tout $k \in [\![1, d-1]\!]$, A_k correspond au $k^{\text{ème}}$ déploiement du tenseur $\boldsymbol{\mathcal{A}}$. L'utilisation de la Snapshot POD pour construire la matrice de base réduite V_k n'est donc pas suffisante pour résoudre le fléau de la dimension. En effet, le nombre de lignes de la matrice A_k croît exponentiellement avec k et devient rapidement trop grand au cours des itérations pour calculer ne serait-ce qu'une seule de ses colonnes. Le calcul de la Snapshot POD n'est donc pas envisageable et par conséquent la TT-POD reste inapplicable pour des tenseurs en grande dimension. Les spécifications TT-SVD et TT-POD illustrent le fait que lorsqu'aucune sélection de lignes n'est effectuée, les coûts de calcul et de mémoire engendrés sont trop importants.

V.2.c TT-PSD

Une approche permettant une exploration parcimonieuse du tenseur $\boldsymbol{\mathcal{A}}$ et envisageable d'un point de vue pratique consiste à utiliser la décomposition pseudo-skeleton en temps qu'approximation de faible rang.

V.2.c.1 Décomposition matricielle

On rappelle la forme d'une approximation au format pseudo-skeleton pour une matrice quelconque $A \in \mathbb{R}^{n \times m}$:

$$A = \underbrace{AQ_{\text{psd}} \left[P_{\text{psd}}^T A Q_{\text{psd}}\right]^{-1} P_{\text{psd}}^T A}_{= T_{\text{psd}}} + E_{\text{psd}} \tag{V.6}$$

où $P_{\text{psd}} \in \mathbb{R}^{n \times r}$ et $Q_{\text{psd}} \in \mathbb{R}^{m \times r}$ sont respectivement des matrices de sélection de lignes et colonnes associées aux indices $\mathcal{I}_{\text{psd}}$ et $\mathcal{J}_{\text{psd}}$. La définition est valide uniquement lorsque la matrice $P_{\text{psd}}^T A Q_{\text{psd}}$ est inversible et donc en particulier le nombre de lignes et de colonnes doit être identique.

On propose un résultat qui n'a pas été retrouvé dans la littérature et pourtant intéressant permettant de définir la décomposition PSD comme la solution d'un problème d'interpolation.

Proposition 7. *Soit une matrice $A \in \mathbb{R}^{n \times m}$, un ensemble de r lignes $\mathcal{I}$ et un ensemble de r colonnes $\mathcal{J}$. Si $\widetilde{A}$ est une matrice de rang r interpolant A en $\mathcal{I}$ et $\mathcal{J}$, c'est-à-dire telle que :*

$$\widetilde{A}(\mathcal{I}, :) = A(\mathcal{I}, :)$$
$$\widetilde{A}(:, \mathcal{J}) = A(:, \mathcal{J})$$

et si $A(\mathcal{I}, \mathcal{J})$ est inversible alors $\widetilde{A}$ correspond à la décomposition PSD (Éq. (V.6)).

La démonstration de la proposition 7 est donnée en annexe B.

On observe par ailleurs que l'approximation PSD correspond exactement à l'approximation interpolante matricielle (Définition 3), lorsque la matrice de base correspond à une sélection de colonnes linéairement indépendantes de A, i.e : $V = A(:, \mathcal{J})$.

V.2.c.2 Décomposition tensorielle

On obtient l'algorithme *TT-PSD* à partir de l'algorithme 11 avec les substitutions suivantes :

$$T = T_{\text{psd}}, \quad H = A(:, \mathcal{J}_{\text{psd}}) \left[A(\mathcal{I}_{\text{psd}}, \mathcal{J}_{\text{psd}})\right]^{-1} \quad \text{et} \quad P = P_{\text{psd}} \tag{V.7}$$

où P_{psd} est la matrice de sélection associée aux s indices de lignes $\mathcal{I}_{\text{psd}}$ de A. La TT-PSD est en réalité équivalente à l'algorithme TT-cross (Section III.5.d).

Le format TT-PSD peut être interprété comme une généralisation du format PSD. Il faut cependant faire attention à ne pas généraliser l'ensemble des résultats relatifs à l'approximation PSD. En particulier, contrairement à l'approximation PSD, une approximation TT-PSD ne possède pas de propriété d'interpolation. Cela est dû aux réarrangements tensoriels effectués aux différentes itérations. À ce jour, il n'existe pas de résultats spécifiques à la TT-PSD caractérisant l'erreur d'approximation.

Les remarques de la section IV.3 relatives à la forme des matrices A_k lorsque P_k est une matrice de sélection sont valides ici. Ainsi, chaque itération équivaut au calcul d'une approximation PSD d'une matrice de taille $(s_{k-1}n_k) \times (n_{k+1} \dots n_d)$ constituée d'éléments du tenseur $\boldsymbol{\mathcal{A}}$. Lorsqu'à l'itération k le nombre s_k de lignes sélectionnées est tel que $s_k \ll s_{k-1}n_k$ la croissance du nombre de lignes des matrices A_k est limitée. Cela permet de construire effectivement l'approximation tensorielle via une exploration parcimonieuse du tenseur $\boldsymbol{\mathcal{A}}$.

V.3 TT-gappy

Une stratégie développée dans cette thèse consiste à utiliser la gappy POD comme approximation matricielle de faible rang. L'approche vise à combiner les avantages de la Snapshot POD et de la décomposition PSD sans chercher à avoir une approximation interpolante à l'image de l'hyper-réduction [103] et de la GNAT [20] pour les tenseurs d'ordre 2. Plus précisément, les deux points clés sont :

— L'utilisation d'une estimation de la base POD possible à calculer ;
— L'utilisation d'une matrice de sélection qui limite la croissance du nombre de lignes des matrices à approcher.

La combinaison de ces deux ingrédients permet de construire des approximations TT précises à partir d'une exploration parcimonieuse du tenseur de référence.

V.3.a Décomposition matricielle

Une approximation gappy (définition 4) T_{gap} d'une matrice $A \in \mathbb{R}^{n\times m}$ est associée à une matrice de base et un ensemble de lignes :

$$A = \underbrace{\left[V_{\text{gap}}^T P_{\text{gap}} P_{\text{gap}}^T V_{\text{gap}}\right]^{-1} V_{\text{gap}}^T P_{\text{gap}} P_{\text{gap}}^T A}_{= T_{\text{gap}}} + E_{\text{gap}} \tag{V.8}$$

où $V_{\text{gap}} \in \mathbb{R}^{n\times r}$ est une matrice de base de rang r, $P_{\text{gap}} \in \mathbb{R}^{n\times s}$ est une matrice de sélection associée aux s indices de lignes $\mathcal{I}_{\text{gap}} \subset [1:n]$ et $E_{\text{gap}}\mathbb{R}^{n\times m}$ est la matrice d'erreur.

La matrice $P_{\text{gap}}^T V_{\text{gap}}$ doit être de rang plein pour que la formulation ait un sens. Comme V_{gap} est de rang r, il existe nécessairement un ensemble de s lignes tel que cette propriété soit assurée tant que $s \geq r$.

Contrairement à la PSD, l'approximation gappy permet de considérer un nombre de lignes d'apprentissage s supérieur au rang r de l'approximation. Nous verrons que cette particularité est essentielle pour la suite des développements.

On rappelle une propriété relative à la définition de la pseudo-inverse de Moore-Penrose [46, Section 5.5.2] :

Proposition 8. *Soit une matrice $M \in \mathbb{R}^{s\times r}$ de rang plein telle que $s \geq r$. Sa pseudo-inverse de Moore-Penrose est égal à :*

$$M^\dagger = \left[M^T M\right]^{-1} M^T \tag{V.9}$$

De plus, $M^\dagger$ est l'inverse à gauche de M, i.e :

$$M^\dagger M = \mathbb{I}_r$$

D'après la proposition 8, l'approximation gappy s'écrit de manière condensée :

$$T_{\text{gap}} = V_{\text{gap}} \left[P_{\text{gap}}^T V_{\text{gap}}\right]^\dagger P_{\text{gap}}^T A \tag{V.10}$$

L'approximation gappy constitue une généralisation de l'approximation interpolante matricielle (Définition 3) pour laquelle la propriété d'interpolation n'est plus vérifiée. En effet, lorsque le nombre de lignes s sélectionnées est égal au rang r de la matrice de base V_{gap}, la matrice $P_{\text{gap}}^T V_{\text{gap}}$ devient inversible et l'équation (V.10) devient :

$$T_{\text{gap}} = V_{\text{gap}} \left[P_{\text{gap}}^T V_{\text{gap}}\right]^{-1} P_{\text{gap}}^T A$$

ce qui correspond strictement à l'approximation interpolante matricielle (Éq. (III.11)).

D'après la remarque 9, l'approximation PSD est un cas particulier de l'approximation interpolante matricielle (Définition 3) lorsque la base est constituée de colonnes de A linéairement indépendantes. Par extension des observations du paragraphe précédent, l'approximation PSD constitue un cas particulier de l'approximation gappy.

On considère dans la suite de cette section une matrice de base $V \in \mathbb{R}^{n\times r}$ dont les colonnes sont orthogonales.

Définition 6. *On définit :*

La matrice de projection (orthogonale) de référence $\mathbb{P}_{\text{ref}} \in \mathbb{R}^{n\times n}$ *:*

$$\mathbb{P}_{\text{ref}} = VV^T$$

La matrice de projection gappy $\mathbb{P}_{\text{gap}} \in \mathbb{R}^{n\times n}$ *associée à la matrice de sélection de lignes* $P \in \mathbb{R}^{n\times s}$ *:*

$$\mathbb{P}_{\text{gap}} = V\left[P^TV\right]^\dagger P^T$$

Et la matrice d'écart dû à l'approximation gappy :

$$\mathbb{E}_{\text{gap}} = \mathbb{P}_{\text{gap}} - \mathbb{P}_{\text{ref}}$$

La proposition 9 donne deux résultats relatifs aux erreurs d'approximation gappy.

Proposition 9. *On considère une projection gappy* $\mathbb{P}_{\text{gap}} \in \mathbb{R}^{n\times n}$ *associée à la matrice de base* $V \in \mathbb{R}^{n\times r}$ *et une matrice de sélection de lignes* $P \in \mathbb{R}^{n\times n}$*. Pour une matrice* $A \in \mathbb{R}^{n\times m}$*, l'erreur d'approximation gappy est définie par :*

$$E_{\text{gap}} = A - \mathbb{P}_{\text{gap}}A$$

La norme de l'erreur d'approximation gappy vérifie les relations suivantes :

$$\|E_{\text{gap}}\|^2 = \|(\mathbb{I} - \mathbb{P}_{\text{ref}})\,A\|^2 + \|\mathbb{E}_{\text{gap}}A\|^2 \tag{V.11}$$

$$\|E_{\text{gap}}\| \leq \frac{1}{\sigma_{\min}(P^TV)}\,\|A - \mathbb{P}_{\text{ref}}A\| \tag{V.12}$$

où $\mathbb{E}_{\text{gap}}$ *est la matrice d'écart dû à la projection gappy.*

La preuve de la proposition 9 est donnée en annexe B.

La proposition 10 donne une relation entre les matrices d'écarts associés à deux sélections de lignes dont l'une est incluse dans l'autre.

Proposition 10. *Soit les projections gappy $\mathbb{P}_1$ et $\mathbb{P}_2$ associées à la même base orthogonale V et respectivement aux indices de lignes $\mathcal{I}_1$ et $\mathcal{I}_2$ telle que :*

$$\mathcal{I}_2 \subset \mathcal{I}_1$$

Les matrices d'écarts $\mathbb{E}_1$ et $\mathbb{E}_2$ associés à $\mathbb{P}_1$ et $\mathbb{P}_2$ vérifient :

$$\|\mathbb{E}_1\| \leq \|\mathbb{E}_2\|$$
$$\|\mathbb{E}_1\|_2 \leq \|\mathbb{E}_2\|_2$$

Le résultat est un corollaire direct des propositions 20 et 21 données en annexe.

L'équation (V.11) montre qu'une fois la base réduite choisie, l'erreur d'approximation gappy est uniquement contrôlée par la matrice d'écart $\mathbb{E}_{\text{gap}}$ qui dépend notamment de la sélection de lignes. D'autre part, la proposition 10 montre qu'ajouter des lignes permet de réduire la norme de la matrice d'écart. Ces deux résultats suggèrent que la sélection d'un nombre plus important de lignes améliore la précision de l'approximation. En particulier, les approximations gappy qui permettent d'intégrer un nombre plus important de lignes que les approximations interpolantes sont plus précises.

V.3.b Décomposition tensorielle

L'algorithme 10 est spécialisé en l'algorithme nommé *TT-gappy* avec les substitutions matricielles suivantes :

$$T = T_{\text{gap}}, \quad H = V_{\text{gap}} \left[P_{\text{gap}}^T V_{\text{gap}}\right]^\dagger \quad \text{et} \quad P = P_{\text{gap}} \tag{V.13}$$

Borne d'erreur Pour quantifier de manière théorique l'accumulation des erreurs introduites par les approximations gappy successives, on prouve la proposition 11 qui donne une borne supérieure sur la norme de l'erreur d'approximation.

Proposition 11. *Soit un $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ et son approximation $\boldsymbol{\mathcal{T}}$ construite via la TT-gappy. En supposant que pour tout $k \in [\![1, d-1]\!]$:*

$$\exists \nu_k \geq 0, \quad \left\| \left(\mathbb{I} - V_k V_k^T\right) A_k \right\| \leq \nu_k \|A_k\| \tag{V.14}$$

on a :

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sum_{k=1}^{d-1} \frac{\nu_k}{\sigma_{\min}\left(P_k^T V_k\right)} \prod_{k'=1}^{k-1} \frac{\min\left(\sigma_{\max}(P_{k'}^T V_{k'}) + \nu_{k'}, 1\right)}{\sigma_{\min}\left(P_{k'}^T V_{k'}\right)} \|\boldsymbol{\mathcal{A}}\| \tag{V.15}$$

En particulier, la proposition est toujours vraie pour $\nu_k = 1$.

Démonstration. On a :

$$\begin{aligned}
\|H_k\|_2 &= \left\| V_k \left[P_k^T V_k \right]^\dagger \right\|_2 \\
&= \left\| \left[P_k^T V_k \right]^\dagger \right\|_2 && \text{comme } V_k \text{ possède des colonnes orthogonales} \\
&= \frac{1}{\sigma_{\min}\left(P_k^T V_k\right)} && \text{comme } P_k^T V_k \text{ est de rang plein} \qquad \text{(V.16)}
\end{aligned}$$

D'autre part pour tout $k \in [\![1, d-1]\!]$:

$$\begin{aligned}
\|E_k\| &\leq \frac{1}{\sigma_{\min}(P_k^T V_k)} \left\| A_k - V_k V_k^T A_k \right\| && \text{d'après la proposition 9} \\
&\leq \frac{1}{\sigma_{\min}(P_k^T V_k)} \left\| (\mathbb{I} - V_k V_k^T) A_k \right\| \\
&\leq \frac{\nu_k}{\sigma_{\min}\left(P_k^T V_k\right)} \|A_k\| && \text{d'après l'hypothèse (V.14)} \qquad \text{(V.17)}
\end{aligned}$$

D'autre part, d'après l'équation (IV.42) et l'invariance de la norme de Frobenius par réarrangement :

$$\begin{aligned}
\|A_k\| &= \|P_{k-1} A_{k-1}\| \\
&\leq \sigma_{\max}\left(P_{k-1}^T V_{k-1}\right) \|A_{k-1}\| + \left\| (\mathbb{I} - V_{k-1} V_{k-1}^T) A_{k-1} \right\| && \text{d'après la proposition 18} \\
&\leq \left[\sigma_{\max}\left(P_{k-1}^T V_{k-1}\right) + \nu_{k-1} \right] \|A_{k-1}\| && \text{d'après l'hypothèse (V.14)} \qquad \text{(V.18)}
\end{aligned}$$

(V.19)

D'autre part, on a clairement :

$$\|P_{k-1} A_{k-1}\| \leq \|A_{k-1}\| \qquad \text{(V.20)}$$

Donc d'après les équations (V.18) et (V.20) :

$$\begin{aligned}
\|A_k\| &\leq \|A_{k-1}\| \min\left(\sigma_{\max}(P_{k-1}^T V_{k-1}) + \nu_{k-1}, 1 \right) \\
&\leq \|A_1\| \prod_{k'=1}^{k-1} \min\left(\sigma_{\max}(P_{k'}^T V_{k'}) + \nu_{k'}, 1 \right) && \text{par récurrence} \\
&\leq \|\boldsymbol{\mathcal{A}}\| \prod_{k'=1}^{k-1} \min\left(\sigma_{\max}(P_{k'}^T V_{k'}) + \nu_{k'}, 1 \right) && \text{d'après l'équation (IV.40)} \qquad \text{(V.21)}
\end{aligned}$$

Finalement, en injectant les équations (V.16), (V.17) et (V.21) dans l'équation de la

proposition 4 on obtient :

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sum_{k=1}^{d-1} \frac{\nu_k}{\sigma_{\min}\left(P_k^T V_k\right)} \prod_{k'=1}^{k-1} \frac{\min\left(\sigma_{\max}(P_{k'}^T V_{k'}) + \nu_{k'}, 1\right)}{\sigma_{\min}\left(P_{k'}^T V_{k'}\right)} \|\boldsymbol{\mathcal{A}}\| \tag{V.22}$$

□

La proposition 11 suggère que la précision de l'approximation TT peut être en partie contrôlée par la précision des approximations matricielles aux différentes itérations. L'hypothèse (Éq. (V.14)) reste cependant difficile à vérifier lorsque la base V_k provient d'un échantillonnage de colonnes de la matrice A_k. La convergence doit par conséquent toujours être justifiée empiriquement. En remplaçant l'hypothèse, on peut modifier la proposition 11 pour faire apparaitre la tolérance de troncature SVD ϵ_{tol}.

Proposition 12. *Avec les notations de la proposition 11, si on remplace l'hypothèse (Éq. (V.14)) par le fait que la matrice V_k correspond à la base POD tronquée de rang r_k de la matrice complète A_k alors :*

$$E_{\text{svd},k} = A_k - V_k V_k^T A_k \quad \textit{avec} \quad \|E_{\text{svd},k}\| = \epsilon_k \|A_k\| \tag{V.23}$$

et la borne devient :

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\| \leq \sum_{k=1}^{d-1} \frac{\epsilon_k}{\sigma_{\min}\left(P_k^T V_k\right)} \prod_{k'=1}^{k-1} \frac{\min\left(\sigma_{\max}(P_{k'}^T V_{k'}) + \epsilon_{k'}, 1\right)}{\sigma_{\min}\left(P_{k'}^T V_{k'}\right)} \|\boldsymbol{\mathcal{A}}\| \tag{V.24}$$

En faisant l'hypothèse que l'échantillonnage de colonnes est suffisamment exhaustif, on peut finalement calculer une estimation de la borne sur l'erreur d'approximation. On observe dans les applications (Sections VII.3.b.2, VII.4.b.2 et V.4) que la valeur de la borne calculée numériquement est à un facteur 10 près de l'ordre de la valeur de la tolérance de troncature et reste également proche de la valeur de l'erreur d'approximation estimée. Cette borne d'erreur, particulièrement intéressante, découle de l'intégration de la gappy POD au sein de la décomposition en train de tenseurs. En particulier, il n'existe pas de borne équivalente pour les autres types d'approximations en train de tenseurs telles que la TT-cross.

V.3.c Détails algorithmiques

L'approximation gappy POD correspond à une approximation gappy pour laquelle la matrice de base est une base réduite issue de l'application de la POD. En termes d'implémentation, c'est cette approximation qui est privilégiée. On discute dans cette section de plusieurs détails algorithmiques relatifs à la construction pratique des approximations/décompositions matricielles.

Construction de la matrice de base réduite À chaque itération, on utilise la méthode de Snapshot POD pour construire la matrice de base réduite. La méthode implique deux étapes détaillées dans la suite :

— La sélection des colonnes ;
— La détermination de la base POD tronquée.

Sélection des colonnes L'objectif de la sélection de colonnes est d'obtenir une sous-matrice $A(:, \mathcal{J})$ dont les colonnes engendrent un espace d'approximation que l'on espère le plus proche possible de l'espace engendré par les colonnes de A. L'erreur d'échantillonnage mesure la différence entre l'espace d'approximation et l'espace de référence. L'objectif de la sélection de colonnes est donc de minimiser cette erreur. Dans le cas général, il n'existe pas d'approche de sélection de colonnes qui permettent de garantir la construction d'une approximation de rang faible suffisamment précise pour représenter une matrice A quelconque. Par conséquent, on doit en pratique compter sur l'hypothèse que la sélection de colonnes est suffisamment riche. Une discussion plus détaillée sur la manière de sélectionner les colonnes dans le cadre des applications (Chapitre VII) est réalisée en section VI.3.a. À ce stade, un algorithme glouton n'a pas pu être mis en œuvre car les applications traitées ne possèdent pas d'estimateurs d'erreur de faible complexité.

Construction de la base La difficulté de la construction de la base vient du fait que le rang de la matrice complète A_k n'est pas connu à l'avance. Dans un premier temps la SVD exacte de la matrice $\widetilde{A} = A(:, \mathcal{J}) \in \mathbb{R}^{n \times m}$ est calculée afin d'obtenir la matrice des vecteurs singuliers à gauche $\widetilde{V}_{\text{svd}} \in \mathbb{R}^{n \times R}$.

Pour déterminer le rang de troncature r, on choisit préalablement une tolérance d'erreur ϵ_{tol} et on cherche à résoudre le problème dual (Éq. (V.2)). Le problème s'écrit : trouver le plus petit $r \leq R$ tel que :

$$\frac{\left\| \widetilde{A} - V_{\text{pod}} V_{\text{pod}}^T \widetilde{A} \right\|}{\left\| \widetilde{A} \right\|} \leq \epsilon_{\text{tol}} \tag{V.25}$$

avec

$$V_{\text{pod}} = \widetilde{V}_{\text{svd}}(:, [\![1, r]\!])$$

C'est-à-dire la matrice constituée des r premiers vecteurs de $\widetilde{V}_{\text{svd}}$.

D'après le théorème d'Eckart-Young-Mirsky, la contrainte de minimisation (Éq. (V.25))

se réécrit :

$$\sqrt{\frac{\sum_{r+1}^{R} \sigma_k^2}{\sum_{k=1}^{R} \sigma_k^2}} \leq \epsilon_{\text{tol}}$$

où les σ_k sont les valeurs singulières de la matrice $\widetilde{A}$. Comme les σ_k sont disponibles au terme du calcul de la SVD, le calcul du rang de troncature peut être réalisée très rapidement.

Construction de la matrice de sélection de lignes L'objectif est de définir l'ensemble de lignes donnant la meilleure approximation tout en limitant le nombre d'évaluations d'éléments du tenseur aux itérations suivantes. La section III.3.c présente un état de l'art de méthodes permettant d'effectuer des sélections de lignes. Dans les applications, l'algorithme DEIM a dans un premier temps été utilisé puis a été remplacé par l'algorithme Q-DEIM. Même si la complexité algorithmique de la Q-DEIM est plus importante que celle de la DEIM, ce n'est pas un problème en pratique. En effet, comme la Q-DEIM est basée sur un algorithme de référence, son utilisation se résume à appeler des fonctions de librairies standards dont les implémentations sont nettement plus optimisées qu'une implémentation "maison" de la DEIM. Dans les applications, des résultats en termes de précision et rapidité de calcul ont montré de manière empirique la supériorité de l'approche Q-DEIM par rapport à la DEIM. Notons enfin que l'algorithme Maxvol est une alternative envisageable. Une comparaison systématique entre ces trois méthodes n'a cependant pas été réalisée.

V.4 Application pratique

On propose dans cette section, une application pratique de l'algorithme 12 pour approcher une fonction multivariée à valeurs vectorielles basée sur une généralisation de la fonction de Rosenbrock en dimension arbitraire.

Soit la fonction :

$$F(x_1, x_2, x_3, x_4, x_5) = \sum_{i=2}^{5} 100 \left[(x_i - x_{i-1}^2)^2 \right] + (1 - x_i)^2$$

Après avoir discrétisé l'intervalle $[-2, 2]$ associé à la coordonnée x_5 en 40 points

$\left\{x_5^{(j)}\right\}_{j=1}^{40}$, on peut définir la fonction f qu'on cherche à approcher :

$$\begin{array}{rccc} f & : & [-3,3]^4 & \to & \mathbb{R}^{40} \\ & & (x_1,x_2,x_3,x_4) & \mapsto & \begin{pmatrix} f_1(x_1,x_2,x_3,x_4) \\ \vdots \\ f_{40}(x_1,x_2,x_3,x_4) \end{pmatrix} \end{array} \tag{V.26}$$

où pour tout $j \in [\![1,40]\!]$:

$$f_j(x_1,x_2,x_3,x_4) = F(x_1,x_2,x_3,x_4,x_5^{(j)})$$

En discrétisant les intervalles associés à chaque variable x_i ($i \in [\![1,4]\!]$) avec des grilles régulières de 50 points $\left\{x_i^{(j)}\right\}_{j=1}^{50}$, on peut définir à la manière de l'équation (IV.5) un tenseur $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{50\times50\times50\times50\times40}$ tel que :

$$\boldsymbol{\mathcal{A}}(i_1,\ldots,i_5) = f_{i_5}(x_1^{(i_1)},x_2^{(i_2)},x_3^{(i_3)},x_4^{(i_4)})$$

On applique l'algorithme TT-gappy sur le tenseur $\boldsymbol{\mathcal{A}}$ en choisissant pour chaque itération une tolérance de troncature $\epsilon_{\text{tol}} = 10^{-3}$ et 1000 colonnes d'échantillonnages pour le calcul de la SVD. La construction implique l'évaluation de F 152 000 fois. Les tailles de stockage du train de tenseurs approché après refactorisation sont $(3,2,4,5)$. En supposant que les échantillonnages de colonnes sont suffisamment exhaustifs, la valeur de la borne sur l'erreur d'approximation donnée en proposition 12 est $0,013$. La borne d'erreur sur l'erreur totale est donc particulièrement fine par rapport à la tolérance de troncature ϵ_{tol}.

On peut finalement définir une fonction $\widetilde{f}$ qui approche f sur l'ensemble du domaine continu $[-3,3]^4$ grâce l'interpolation multilinéaire par morceaux (Section IV.1.d). Notons qu'il est également possible d'interpréter f et $\widetilde{f}$ comme des fonctions scalaires multivariées dont les variables sont $(x_1,\ldots,x_5)$.

On définit l'erreur d'approximation relative $e(x_1,x_2,x_3,x_4)$ par :

$$e(x_1,x_2,x_3,x_4) = \frac{\left\| f(x_1,x_2,x_3,x_4) - \widetilde{f}^i(x_1,x_2,x_3,x_4) \right\|_2}{\|f(x_1,x_2,x_3,x_4)\|_2}$$

En calculant la valeur de $e(x_1,x_2,x_3,x_4)$ pour un ensemble de points 10 000 tirés aléatoirement sur le domaine paramétrique discrétisé, on obtient une erreur d'approximation moyenne de l'ordre de $2,5.10^{-17}$. Le tirage aléatoire est effectué en choisissant une loi uniforme sur chaque intervalle du domaine paramétrique discrétisé.

La figure V.1 présente les sorties associées à 4 vecteurs (x_1,x_2,x_3,x_4) du domaine

paramétrique continu, pour le modèle de référence (la fonction f) et pour le modèle de substitution (la fonction $\widetilde{f}$).

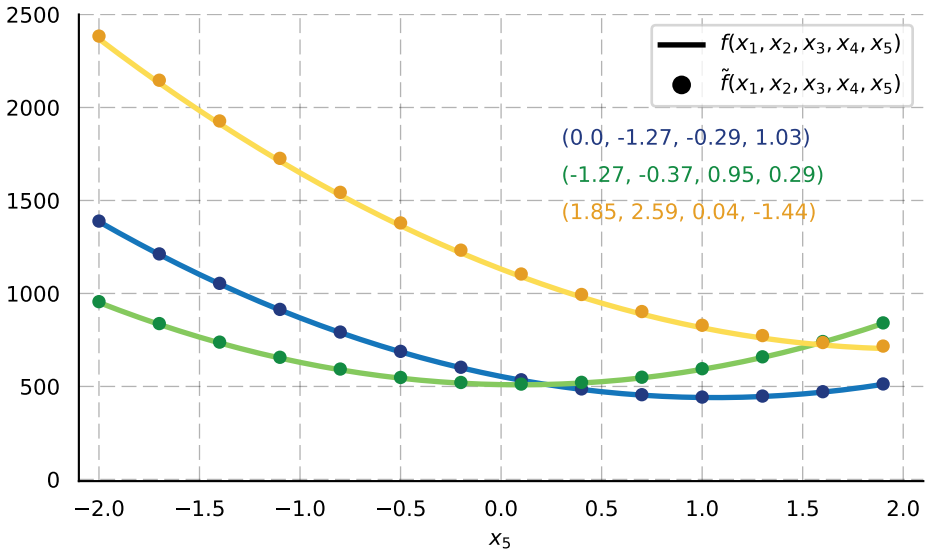


FIGURE V.1 – Comparaison de sorties entre le modèle de référence Rosenbrock et le modèle de substitution.

V.5 Interprétation en termes de projecteur

V.5.a Matrice de projection

Un projecteur (matriciel) $\mathbb{P}$ de $\mathbb{R}^n$ est caractérisé par son espace de projection $\mathrm{Im}(\mathbb{P})$ et son noyau $\mathcal{N}(\mathbb{P})$ tel que :

$$\mathbb{R}^n = \mathrm{Im}(\mathbb{P}) \oplus \mathcal{N}(\mathbb{P})$$

On dit que $\mathbb{P}$ est la projection sur $\mathrm{Im}(\mathbb{P})$ parallèlement $\mathcal{N}(\mathbb{P})$.

On prouve un résultat général sur les matrices de projection :

Proposition 13. *Soient deux matrices $U \in \mathbb{R}^{n\times r}$ et $C \in \mathbb{R}^{n\times s}$ de rang plein telles que $r \leq s \leq n$. On a l'équivalence entre les deux propositions suivantes :*

- $C^T U \in \mathbb{R}^{s\times r}$ *est de rang* r (V.27)
- $\mathbb{R}^n = \mathrm{Im}(U) \oplus \mathcal{N}(U^T C C^T)$ *(décomposition de* $\mathbb{R}^n$ *en somme directe)* (V.28)

Et la projection associée à cette décomposition en somme directe est donnée par :

$$\mathbb{P} = U[C^T U]^\dagger C^T \tag{V.29}$$

La démonstration de la proposition 13 est donnée en annexe B.

Remarque 18. *Dans la proposition 13, les matrices U et C peuvent être de tailles différentes ($r \neq s$). Cependant lorsque les propositions sont vérifiées et que $r = s$, on a :*

$$\mathcal{N}(\mathbb{P}) = \mathcal{N}(U^T C C^T) = \mathcal{N}(C^T)$$

et la proposition 13 correspond à un résultat d'algèbre plus classique.

La démonstration de la remarque 18 est donnée en annexe B.

Un projecteur orthogonal $\mathbb{P}$ correspond au cas ou $U = C$ ou $C = \mathbb{I}$. Dans le cas contraire, on parle de projections obliques.

V.5.b Formulation en termes de projecteurs

Il est possible de formuler l'ensemble des approximations matricielles considérées dans cette partie de la manière suivante :

$$T = \mathbb{P}A$$

où les matrices $\mathbb{P}$ relatives à chaque approximation s'écrivent :

$$\begin{aligned}
\mathbb{P}_{\text{svd}} &= V_{\text{svd}} V_{\text{svd}}^T \\
\mathbb{P}_{\text{pod}} &= V_{\text{pod}} V_{\text{pod}}^T \\
\mathbb{P}_{\text{psd}} &= AQ_{\text{psd}} \left[P_{\text{psd}}^T AQ_{\text{psd}} \right]^{-1} P_{\text{psd}}^T \\
\mathbb{P}_{\text{gap}} &= V_{\text{gap}} \left[P_{\text{gap}}^T V_{\text{gap}} \right]^{\dagger} P_{\text{gap}}^T
\end{aligned}$$

Il est clair que toutes ces approximations correspondent donc à des projetés de la matrice A.

La formulation de l'équation (V.29) permet d'une part d'englober toutes les approximations matricielles, mais aussi de distinguer les différentes décompositions utilisées pour spécialiser l'algorithme 11.

En définissant les matrices H et P telle que :

$$\begin{aligned}
H &= U \left[C^T U \right]^{\dagger} \\
P &= C
\end{aligned} \tag{V.30}$$

la correspondance entre les matrices associées aux différentes spécifications et les matrices U et C sont données dans le tableau V.1.

Algorithme	U	C
TT-SVD	V_{svd}	V_{svd}
TT-POD	V_{pod}	$\mathbb{I}_n$
TT-PSD	AQ_{psd}	P_{psd}
TT-gappy	V_{gap}	P_{gap}

TABLE V.1 – Tableau d'équivalence entre approximation matricielle et projection.

Par construction, le produit $C^T U$ est toujours de rang r et donc la projection est bien associée à la décomposition en somme directe de l'équation (V.28).

V.5.c Principe général d'approximation

Pour l'ensemble des spécifications présentées, la construction des approximations matricielle (Éq. (V.3)) peut finalement être résumée en deux étapes :

— Construire d'une matrice de base réduite $V \in \mathbb{R}^{n\times r}$;

— Définir une matrice de sélection $P \in \mathbb{R}^{n\times s}$ associée à s indices de lignes, appelées aussi indices d'apprentissage ou point d'apprentissage ;

Matrice de base réduite Informellement, la construction de la matrice de base réduite $V \in \mathbb{R}^{n\times r}$ consiste à avoir :

$$\text{Im}(V) \simeq \text{Im}(A)$$

de manière à minimiser l'erreur d'approximation.

On note que cette matrice est nécessairement une combinaison linéaire de colonnes de A :

$$V = AW \quad \text{où} \quad W \in \mathbb{R}^{m\times r} \tag{V.31}$$

Dans le cas de la PSD, la matrice de base réduite est constituée uniquement d'une sélection de colonnes (donc W est une matrice de sélection de colonnes). Pour les approximations POD et gappy, la matrice de base réduite est constituée d'une combinaison linéaire d'une sélection de colonnes. Pour la SVD, il n'y a pas de sélection de colonnes.

Sélection de lignes Selon les approximations utilisées, le nombre s d'indices de lignes sélectionnées varie. Dans le cas de la TT-SVD et de la TT-POD, toutes les lignes sont requises ce qui empêche l'application de ces algorithmes en grande dimension. Dans le cas de la PSD uniquement un nombre de lignes égales au rang de la base de projection est

requis. Enfin pour la TT-gappy, le nombre de lignes est arbitraire tant qu'il est supérieur au rang de la matrice de base réduite.

V.6 Intérêt de la procédure de refactorisation

La formulation en termes de projection permet de donner la forme générale des matrices à utiliser dans la procédure de refactorisation pour obtenir une représentation TT minimale.

On rappelle que l'application de l'algorithme de refactorisation consiste à décomposer $H \in \mathbb{R}^{n\times s}$ sous la forme d'un produit matriciel $H = KL$ en imposant $K \in \mathbb{R}^{n\times r}$ et $L \in \mathbb{R}^{r\times s}$. D'après l'équation (V.30), la matrice H peut être développée sous la forme :

$$H = U\left[U^T CC^T U\right]^{-1} U^T C$$

Or les matrices impliquées dans cette décomposition sont de tailles :

$$\begin{aligned} U &\in \mathbb{R}^{n\times r} \\ \left[U^T CC^T U\right]^{-1} &\in \mathbb{R}^{r\times r} \\ U^T C &\in \mathbb{R}^{r\times s} \end{aligned}$$

Par conséquent, il existe deux manières naturelles de définir les matrices K et L donnant un format TT minimal équivalent :

$$\begin{aligned} K = U \quad \text{et} \quad L = \left[U^T CC^T U\right]^{-1} U^T C \\ \text{ou} \quad K = U\left[U^T CC^T U\right]^{-1} \quad \text{et} \quad L = U^T C \end{aligned}$$

L'intérêt de la procédure de refactorisation varie selon l'approximation de faible rang sélectionnée. Dans le cas de la SVD et la PSD, le tenseur $\boldsymbol{\mathcal{H}}$ (Éq. (IV.12)) est déjà donné dans format compact. La procédure de refactorisation ne fournira donc pas une décomposition TT plus compacte. Même si la TT-POD n'a pas d'intérêt pratique, on observe que la procédure de refactorisation fournit une compression importante permettant théoriquement de stocker la décomposition TT si les rangs sont suffisamment faibles.

La refactorisation prend tout son intérêt dans le cadre de la TT-gappy. En effet, la taille des tenseurs $\boldsymbol{\mathcal{H}}$ construits en suivant l'approche gappy POD dépendent du nombre de lignes échantillonnées qui est supérieur ou égal au rang de l'approximation. La procédure de refactorisation permet dans ce cas de redéfinir les facteurs du train afin d'obtenir une décomposition TT minimale.

Chapitre VI

Décomposition en trains de tenseurs hétérogènes

Ce chapitre présente une extension de l'algorithme TT-gappy adaptée pour construire des modèles de substitution de modèles physiques dont les sorties sont hétérogènes et à valeurs vectorielles. Ce type de modèle physique est omniprésent en science des matériaux : en effet les lois de comportement mettent en relation des tenseurs de contrainte, des tenseurs de déformation, des variables internes et des forces thermodynamiques généralisées, qui ont leur unité propre et des ordres de grandeur qui ne sont pas comparables. La procédure résultante, nommée décomposition en trains de tenseurs hétérogènes (Heterogeneous TT decomposition), permet de construire simultanément plusieurs approximations TT à partir d'un ensemble commun de résultats de calcul en se basant sur des échantillonnages parcimonieux des domaines paramétriques.

Table des matières

VI.1 Cadre général des modèles à approcher . . . 117
VI.1.a Modèle de référence . . . 117
VI.1.b Représentations tensorielles d'un modèle de référence . . . 119
VI.2 Algorithme de décomposition en trains de tenseurs hétérogènes 121
VI.3 Détails algorithmiques . . . 122
VI.3.a Sélection de colonnes . . . 122
VI.3.b Sélection de lignes . . . 126
VI.3.c Calcul de la SVD . . . 127
VI.4 Détails relatifs aux applications . . . 128
VI.4.a Définition des sorties du modèle . . . 129
VI.4.b Définition des entrées du modèle . . . 131
VI.4.c Accumulation des erreurs . . . 133
VI.5 Applications théoriques . . . 133
VI.5.a Agrégation . . . 134
VI.5.b Raffinement du rang de troncature . . . 136
VI.6 Améliorations possibles . . . 137
VI.6.a SVD incrémentale . . . 137

VI.6.b Autres perspectives d'amélioration 138

VI.1 Cadre général des modèles à approcher

VI.1.a Modèle de référence

Systèmes d'équations différentielles Soit un modèle physique décrit par le système d'équations différentielles algébriques (EDA) paramétriques suivant :

$$\boldsymbol{R}_{\boldsymbol{\mu}}\left(\boldsymbol{Y}_{\boldsymbol{\mu}}^{1},\ldots,\boldsymbol{Y}_{\boldsymbol{\mu}}^{N},\dot{\boldsymbol{Y}}_{\boldsymbol{\mu}}^{1},\ldots,\dot{\boldsymbol{Y}}_{\boldsymbol{\mu}}^{N},t\right)=0 \tag{VI.1}$$

où

- Le vecteur $\boldsymbol{\mu}=(\mu_1,\ldots,\mu_{d-1})\in\mathcal{D}$ avec $d>0$ correspond aux paramètres ;
- Le domaine paramétrique $\mathcal{D}=\mathcal{D}_1\times\cdots\times\mathcal{D}_{d-1}$ est défini comme un produit cartésien d'intervalles de $\mathbb{R}$;
- La variable $t\in[0,T]\subset\mathbb{R}$ avec $T>0$ correspond au temps ;
- $\boldsymbol{R}_{\boldsymbol{\mu}}$ est un opérateur paramétrique ;
- Les $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ sont les variables du système d'équations dont les solutions dépendent de $\boldsymbol{\mu}$.

De telles EDA sont typiquement introduites en mécanique [76, 13] pour modéliser des lois de comportement des matériaux. Les variables $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ correspondent dans ces applications typiquement au tenseur des contraintes, au tenseur des déformations ou encore à des variables internes qui peuvent avoir ou non un sens physique.

Remarque 19. *Il faut faire attention à la différence de concept mathématique entre le terme tenseur utilisé dans ce manuscrit et en science des matériaux. En mécanique et en science matériaux, le terme de tenseur fait référence à des applications linéaires définies sur des espaces vectoriels [13]. Dans cette thèse, le terme de tenseur fait référence à une représentation de ces objets donnée sous la forme d'un tableau multidimensionnel (Éq.* (III.21)*).*

En supposant que les hypothèses du théorème des fonctions implicites soient vérifiées, la résolution numérique du modèle (Éq. (VI.1)) consiste à trouver pour un $\boldsymbol{\mu}\in\mathcal{D}$ fixé la solution constituée de l'ensemble des variables $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ sous la forme

$$\begin{array}{rcccl}\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi} & : & [0,T] & \to & \mathbb{R}^{m^{\chi}} \\ & & t & \mapsto & \boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}(t)=\begin{pmatrix} Y_{\boldsymbol{\mu},1}^{\chi}(t) \\ \vdots \\ Y_{\boldsymbol{\mu},m^{\chi}}^{\chi}(t)\end{pmatrix}\end{array} \tag{VI.2}$$

Cette formulation englobe, à réorganisation des éléments près, les cas où $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}(t)$ correspond à un scalaire, un vecteur, une matrice ou un tenseur (au sens tableau multidimension-

nel). On peut envisager le cas particulier où la fonction $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ est indépendante du temps. Dans ce cas, $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ s'interprète directement comme un vecteur de $\mathbb{R}^{m^{\chi}}$.

À partir de ces grandeurs, il est possible de définir des *quantités d'intérêt* formulées également sous la forme (Éq. (VI.2)). En pratique, une quantité d'intérêt correspond à une grandeur dont la connaissance est nécessaire pour conduire une étude particulière du modèle physique. Par exemple, dans le cadre de la calibration de lois matériaux, il est nécessaire de connaitre l'évolution temporelle de variables mécaniques telles que le tenseur des contraintes ou le tenseur des déformations, mais il peut être intéressant d'observer également des grandeurs dérivées des variables mécaniques telles que des instants caractéristiques, des vitesses de déformation ou encore des seuils de contrainte. Si par exemple, une zone temporelle d'une certaine variable joue un rôle important vis-à-vis de la calibration, il peut être intéressant de définir une quantité d'intérêt où la zone temporelle est dilatée afin améliorer la précision de son approximation.

Interdépendance des sorties Les quantités d'intérêt (Éq. (VI.2)) étant définies à partir de la résolution d'un modèle physique monolithique (tel qu'un système d'EDA), il existe une forme de dépendance entre elles, on dit alors qu'elles sont *interdépendantes*. D'une part, la résolution du modèle physique pour un vecteur de paramètres $\boldsymbol{\mu}$ consiste à trouver l'ensemble des $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ pour tout $\chi \in [\![1, N]\!]$. D'autre part, le temps est une coordonnée causale, c'est-à-dire que le calcul de $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}(t)$ au temps t implique que $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}(t')$ ait été calculé pour tout $t' \in [0, t]$ (au moins sur l'intervalle de temps discrétisé). L'interdépendance des quantités d'intérêt s'exprime donc par le fait que la connaissance (via une résolution du modèle physique) de $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}(t) \in \mathbb{R}^{m^{\chi}}$ pour $(t, \boldsymbol{\mu}, \chi)$ fixé implique nécessairement que l'on dispose de $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi'}(t') \in \mathbb{R}^{m^{\chi'}}$ pour tout $\chi' \in [\![1, N]\!]$ et $t' \in [0, t]$ sans effectuer de calcul supplémentaire.

Hétérogénéité des sorties Les quantités d'intérêt $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ peuvent être de natures différentes et ne sont donc généralement pas comparables, on dit qu'elles sont *hétérogènes*.

Elles peuvent :

— avoir différentes unités physiques ;

— avoir différents ordres de grandeur en termes d'amplitude ;

— présenter des variations dissemblables en fonction des paramètres et du temps.

Par exemple, la comparaison entre le tenseur des contraintes et le tenseur des déformations n'a pas de sens dans le cas général.

Définition générale d'un modèle de référence La résolution numérique du modèle (Éq. (VI.1)) pour un choix de paramètres $\boldsymbol{\mu} \in \mathcal{D}$ fournit un ensemble de quantités d'intérêt $\boldsymbol{Y}_{\boldsymbol{\mu}}^{\chi}$ (Éq. (VI.2)). Lors d'une étude paramétrique, on cherche à comprendre la relation entre les paramètres et les quantités d'intérêt. Le modèle qu'on cherche à étudier peut

finalement se formuler comme un ensemble de N fonctions multivariées à valeurs vectorielles $\left\{\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^N\right\}$ définies sur l'espace paramétrique $\mathcal{D}$ et l'intervalle de temps $[0, T]$ telles que pour tout $\chi \in [\![1, N]\!]$

$$\begin{array}{rcccl} \boldsymbol{Y}^\chi & : & \mathcal{D} \times [0,T] & \to & \mathbb{R}^{m^\chi} \\ & & (\boldsymbol{\mu}, t) & \mapsto & \boldsymbol{Y}^\chi(\boldsymbol{\mu}, t) = \begin{pmatrix} Y_1^\chi(\boldsymbol{\mu}, t) \\ \vdots \\ Y_{m^\chi}^\chi(\boldsymbol{\mu}, t) \end{pmatrix} \end{array} \tag{VI.3}$$

Dans la suite, le concept de *modèle de référence* fait référence à un ensemble de fonctions $\left\{\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^N\right\}$ données sous la forme (Éq. (VI.3)) et issues de la résolution numérique d'un modèle physique. D'autre part, les paramètres $\boldsymbol{\mu}$ et les quantités d'intérêt $\boldsymbol{Y}_{\boldsymbol{\mu}}^\chi$ sont considérés respectivement comme les entrées et les sorties du modèle de référence.

VI.1.b Représentations tensorielles d'un modèle de référence

Pour définir les représentations tensorielles associées aux fonctions $\boldsymbol{Y}^\chi$ (Éq. (VI.3)), des discrétisations du domaine paramétrique et de l'intervalle de temps doivent être introduites.

L'échantillonnage temporel, noté $\mathcal{T}^\Delta$, de l'intervalle de temps $[0, T]$ est généralement fourni par le schéma d'intégration numérique utilisé pour résoudre le modèle physique :

$$\mathcal{T}^\Delta = \left\{t^{(i_t)} \in [0,T] \mid \forall\, i_t \in I_t\right\} \quad \text{avec} \quad t^{(1)} < \cdots < t^{(n_t)} \tag{VI.4}$$

où $I_t = [\![1, n_t]\!]$ et n_t est le nombre de pas de discrétisation.

La discrétisation du domaine paramétrique $\mathcal{D}$ est similaire à celle introduite à l'équation (IV.3) et est donnée par $D = D_1 \times \cdots \times D_{d-1}$.

Pour tout $\chi \in [\![1, N]\!]$, on définit l'indice de composante $i_{comp} \in I_{comp}^\chi = [\![1, m^\chi]\!]$. Pour rendre la notation plus claire, l'exposant χ est omis sur l'indice i_{comp} même si les intervalles de variation sont différents pour chaque χ.

Pour chaque $\chi \in [\![1, N]\!]$, on associe à la fonction $\boldsymbol{Y}^\chi$ un tenseur $\boldsymbol{\mathcal{A}}^\chi \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times n_t \times m^\chi}$ tel que :

$$\begin{gathered} \boldsymbol{\mathcal{A}}^\chi\left(i_1, \ldots, i_{d-1}, i_t, i_{comp}\right) = Y_{i_{comp}}^\chi\left(\mu_1^{(i_1)}, \ldots, \mu_{d-1}^{(i_{d-1})}, t^{(i_t)}\right) \\ \forall\, \left(i_1, \ldots, i_{d-1}, i_t, i_{comp}\right) \in I_1 \times \cdots \times I_{d-1} \times I_t \times I_{comp}^\chi \end{gathered}$$

Les tenseurs $\boldsymbol{\mathcal{A}}^\chi$ associés au modèle de référence sont appelés *tenseurs de référence* ou *tenseurs physiques* (au sens où ils sont issus de modèles physiques).

Sans restreindre la généralité de l'approche, les deux derniers indices sont associés en un seul multi-indice $i_d = (i_t, i_{comp}) \in I_t \times I_{comp}^\chi = I_d^\chi$ de telle sorte que $\boldsymbol{\mathcal{A}}^\chi$ définisse un

tenseur d'ordre d :

$$\boldsymbol{\mathcal{A}}^{\chi} \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times n_d^{\chi}} \quad \text{avec } n_d^{\chi} = n_t m^{\chi} \tag{VI.5}$$

avec

$$\begin{aligned} \boldsymbol{\mathcal{A}}^{\chi}\left(i_1, \ldots, i_{d-1}, (i_t, i_{comp})\right) &= Y_{i_{comp}}^{\chi}\left(\mu_1^{(i_1)}, \ldots, \mu_{d-1}^{(i_{d-1})}, t^{(i_t)}\right) \\ \forall\ (i_1, \ldots, i_{d-1}, i_t, i_{comp}) &\in I_1 \times \cdots \times I_{d-1} \times I_t \times I_{comp}^{\chi} \end{aligned} \tag{VI.6}$$

Les tenseurs de référence $\boldsymbol{\mathcal{A}}^{\chi}$ sont ainsi définis implicitement, c'est-à-dire que les éléments ne sont pas stockés en mémoire, mais que leur accès est possible typiquement via la résolution numérique du modèle physique pour les valeurs de paramètres appropriées.

Remarque 20. *L'interdépendance des quantités d'intérêt, mentionnée dans la section VI.1.a, s'exprime pour les tenseurs physiques de la manière suivante. Une seule résolution du modèle physique permet d'obtenir pour chaque tenseur $\boldsymbol{\mathcal{A}}^{\chi}$ ($\chi \in [\![1, N]\!]$) la fibre*

$$\boldsymbol{\mathcal{A}}^{\chi}\left(i_1, \ldots, i_{d-1}, I_d^{\chi}\right) \in \mathbb{R}^{n_d^{\chi}} \tag{VI.7}$$

associée aux $(d-1)$ premiers indices $(i_1, \ldots, i_{d-1}) \in I_1 \times \cdots \times I_{d-1}$ communs.

Donc avoir accès à un élément d'un tenseur $\boldsymbol{\mathcal{A}}^{\chi}$ implique nécessairement que l'ensemble des fibres (Éq. (VI.7)) pour tous les tenseurs $\boldsymbol{\mathcal{A}}^{\chi}$ sont également connues. Puisque la construction d'approximations tensorielles repose sur des sélections d'indices, cette considération est prise en compte dans la suite pour élaborer un algorithme efficace.

Rappel du contexte Selon le modèle physique considéré, la résolution peut être chère, c'est d'ailleurs le principal frein à la calibration. En termes de tenseurs physiques, cela s'exprime par le fait que les accès aux éléments sont couteux empêchant typiquement une exploration efficace des tenseurs.

Ce constat motive la construction préalable de ces tenseurs (phase hors ligne) pour assurer un accès temps réel aux éléments lors de leur exploitation (phase en ligne). Une solution naïve consiste à calculer et stocker en mémoire l'ensemble des éléments pour n'avoir ensuite qu'à réaliser des accès mémoires lors de la phase en ligne. Cette solution n'est évidemment envisageable ni en termes de temps de calcul ni en termes d'espace mémoire. On illustre ce problème sur un exemple simple. Considérons un modèle physique dont on cherche à étudier l'influence de 9 paramètres définis sur des intervalles discrétisées en 10 points. On suppose que le modèle est relativement simple de telle sorte que sa résolution nécessite seulement 1s de calcul et le stockage des solutions correspond à 10ko. Si l'on cherchait à calculer et stocker l'ensemble des solutions, cela nécessiterait 31 années de calcul

et 9To de mémoire. Cet exemple trivial permet de comprendre la difficulté que représente l'exploitation de modèles paramétriques en termes de temps de calcul et espace mémoire.

Pour surmonter ce fléau de la dimension, l'idée développée dans cette thèse est de construire des modèles de substitution fondés sur un ensemble de décompositions en train de tenseurs qui approchent les tenseurs physiques. D'une part, la construction de tenseurs approchés nécessite de résoudre les modèles physiques pour un nombre limité de valeurs de paramètres. Cela permet de réduire à des valeurs envisageables les temps de calcul hors lignes. D'autre part, l'utilisation de décompositions tensorielles permet de réduire significativement l'empreinte mémoire des tenseurs résolvant ainsi le problème de stockage.

VI.2 Algorithme de décomposition en trains de tenseurs hétérogènes

Cette section présente un algorithme, développé au cours de cette thèse, de construction de décompositions TT approchant des tenseurs de référence.

L'algorithme permet de tenir compte de trois caractéristiques spécifiques aux applications qui nous intéressent, à savoir :

- les coûts de calcul pour accéder aux éléments des tenseurs de référence sont importants ;
- les tenseurs de référence sont interdépendants au sens de la remarque 20 ;
- les tenseurs de référence sont hétérogènes au sens où ils représentent typiquement des grandeurs physiques ayant des unités de mesure qui leur sont propres.

En tenant compte de l'interdépendance des quantités d'intérêt, on propose l'algorithme 12 qui autorise l'intégration de l'ensemble des données couteuses produites par la résolution des modèles physiques afin d'améliorer la précision des approximations et limiter le nombre de résolutions nécessaires. L'idée de l'algorithme 12 est d'appliquer la TT-gappy (Section V.3) simultanément sur tous les tenseurs physiques en se basant sur des résolutions partagées du modèle physique. Les représentations en train de tenseurs résultantes, bien qu'issues de solutions des mêmes équations, ont des rangs de compression qui leur sont propres et a fortiori des tailles de stockages différentes.

Contrairement à la méthode PGD (Section III.5.g), l'approche est ici non intrusive au sens où elle n'implique pas de reformulation des équations physiques durant la phase hors ligne. La condition suffisante pour appliquer l'algorithme est d'être capable d'accéder à n'importe quels éléments des tenseurs physiques. Les détails d'implémentation de la résolution numérique des équations physiques ne sont pas pertinents dans ce cadre.

L'algorithme 12 fournit pour chaque tenseur physique $\boldsymbol{\mathcal{A}}^\chi$ un ensemble de matrices $\{H_1^\chi, \ldots, H_d^\chi\}$ permettant de définir une approximation en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$ d'après l'équation (IV.11). La procédure de refactorisation (Section IV.2.c) peut être transposée

directement pour chaque décomposition et permet d'obtenir des représentations en train de tenseurs dont les tailles de stockage sont différentes.

VI.3 Détails algorithmiques

L'algorithme 12 ne fait pas apparaitre explicitement les opérations impliquées lors de la construction des approximations matricielles. L'objet de cette section est de donner des précisions sur ces opérations.

On s'intéresse ici à l'étape essentielle, réalisée à chaque itération k, qui consiste à approcher pour tout $\chi \in [\![1, N]\!]$ les matrices A_k^χ par une approximation gappy T_k^χ donnée par :

$$T_k^\chi = H_k^\chi A_k^\chi(\mathcal{I}_k, :) \tag{VI.16}$$

VI.3.a Sélection de colonnes

À chaque itération $k \in [\![1, d-1]\!]$, la construction de l'approximation gappy passe par une sélection initiale pour chaque matrice $A_k^\chi \in \mathbb{R}^{(s_{k-1}n_k)\times\left(n_{k+1}\ldots n_{d-1}n_d^\chi\right)}$ d'un ensemble de $\widetilde{n}_k^\chi$ colonnes associées aux indices :

$$\widetilde{J}_k^\chi = \left\{\widetilde{J}_k^{\chi,(1)}, \ldots, \widetilde{J}_k^{\chi,\left(\widetilde{n}_k^\chi\right)}\right\} \subset [\![1, n_{k+1}\ldots n_{d-1}n_d^\chi]\!]$$

La figure VI.1 illustre la sélection de colonnes correspondantes.

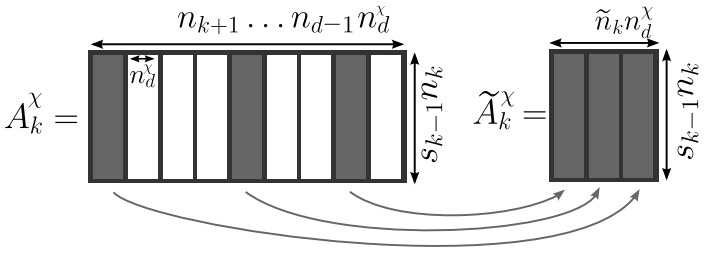


FIGURE VI.1 – Défintion de la sous-matrice $\widetilde{A}_k^\chi$ utilisée pour construire la base réduite POD. Dans l'illustration, la taille de l'échantillonnage est de $\widetilde{n}_k = 3$

Les sous-matrices $\widetilde{A}_k^\chi$ (Éq. (VI.9)) doivent être ensuite explicitement calculées via la résolution du modèle physique. Pour exploiter au maximum les résultats des simulations, les sélections de colonnes doivent être cohérentes vis-à-vis de l'interdépendance des tenseurs physiques $\boldsymbol{\mathcal{A}}^\chi$. On dit que les sélections de colonnes sont *cohérentes* lorsque les éléments des $\widetilde{A}_k^\chi$ correspondent aux mêmes solutions du modèle physique.

D'après la proposition 5, pour tout $(k, \chi) \in [\![1, d]\!] \times [\![1, N]\!]$ la matrice A_k^χ est définie

Algorithme 12 : Décomposition en trains de tenseurs hétérogènes

Données : Des tenseurs de référence interdépendants $\boldsymbol{\mathcal{A}}^\chi \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times n_d^\chi}$ définis implicitement et des tolérances de troncatures ϵ_k^χ pour tout $(\chi, k) \in [\![1, N]\!] \times [\![1, d-1]\!]$.

Résultat : N ensembles des matrices $\{H_1^\chi, \ldots, H_d^\chi\}$ pour $\chi \in [\![1, N]\!]$.

Initialisation

Pour tout $\chi \in [\![1, N]\!]$, définir la matrice $A_1^\chi \in \mathbb{R}^{n_1 \times \left(n_2 \ldots n_{d-1} n_d^\chi\right)}$ comme le premier déploiement du tenseur $\boldsymbol{\mathcal{A}}^\chi$:

$$A_1^\chi = \langle \boldsymbol{\mathcal{A}}^\chi \rangle_1 \tag{VI.8}$$

Définir $s_0 = 1$.

Pour $k \in [\![1, d-1]\!]$ **faire**

Échantillonnage des colonnes

Définir des échantillonnages de colonnes cohérents $\widetilde{J}_k^\chi$ (Section VI.3.a) et calculer les sous-matrices :

$$\widetilde{A}_k^\chi = A_k^\chi \left(:, \widetilde{J}_k^\chi\right) \quad \text{pour tout } \chi \in [\![1, N]\!] \tag{VI.9}$$

Construction des bases

Calculer la SVD tronquée (Éq. (V.4)) de chaque $\widetilde{A}_k^\chi$ associée à la tolérance ϵ_k^χ afin d'obtenir les matrices de rang r_k^χ :

$$V_k^\chi \in \mathbb{R}^{(s_{k-1} n_k) \times r_k^\chi} \quad \text{pour tout } \chi \in [\![1, N]\!] \tag{VI.10}$$

Sélection des lignes

Sélectionner pour chaque $\chi \in [\![1, N]\!]$, un ensemble de r_k^χ lignes pertinentes $\mathcal{I}_k^\chi$ (Section VI.3.b). Définir l'union des lignes :

$$\mathcal{I}_k = \bigcup_{\chi=1}^{N} \mathcal{I}_k^\chi \tag{VI.11}$$

et la matrice de sélection correspondante :

$$P_k = \in \mathbb{R}^{(s_{k-1} n_k) \times s_k} \quad \text{où} \quad s_k = \mathrm{Card}(\mathcal{I}_k) \tag{VI.12}$$

Définitions des sorties

Calculer les matrices $H_k^\chi \in \mathbb{R}^{(s_{k-1} n_k) \times s_k}$ telles que :

$$H_k^\chi = V_k^\chi \left[(P_k)^T V_k^\chi \right]^\dagger \tag{VI.13}$$

Réarrangement tensoriel

Définir pour tout $\chi \in [\![1, N]\!]$, la matrice $A_{k+1}^\chi \in \mathbb{R}^{(s_k n_{k+1}) \times \left(n_{k+2} \ldots n_{d-1} n_d^\chi\right)}$:

$$A_{k+1}^\chi = \underset{(s_k n_{k+1}) \times \left(n_{k+2} \ldots n_{d-1} n_d^\chi\right)}{\mathrm{reshape}} \left(P_k^T A_k^\chi\right) \tag{VI.14}$$

Finalisation

Pour chaque $\chi \in [\![1, N]\!]$, calculer la matrice $H_d^\chi \in \mathbb{R}^{\left(s_{d-1} n_d^\chi\right) \times s_d}$ telle que :

$$H_d^\chi = A_d^\chi \tag{VI.15}$$

avec $s_d = 1$.

par :

$$A_k^\chi = \left\langle \boldsymbol{\mathcal{A}}^\chi(\mathcal{I}_{k-1}, I_k, \ldots, I_{d-1}, I_d^\chi) \right\rangle_2$$

Une sélection de colonnes de A_k^χ correspond donc à une sélection de multi-indices parmi $I_{k+1} \times \cdots \times I_{d-1} \times I_d^\chi$ du tenseur $\boldsymbol{\mathcal{A}}^\chi(\mathcal{I}_{k-1}, I_k, \ldots, I_{d-1}, I_d^\chi) \in \mathbb{R}^{s_{k-1} \times n_k \times \cdots \times n_d \times n_d^\chi}$. D'après la remarque 20, la résolution du modèle physique pour des valeurs de paramètres associées aux indices communs

$$(\mathcal{I}_{k-1}^{(\alpha)}, i_k, \ldots, i_{d-1}) \in \mathcal{I}_{k-1} \times I_k \times \cdots \times I_{d-1}$$

permet d'obtenir l'ensemble des éléments des fibres :

$$\boldsymbol{\mathcal{A}}^\chi(\mathcal{I}_{k-1}^{(\alpha)}, i_k, \ldots, i_{d-1}, I_d^\chi) \in \mathbb{R}^{n_d^\chi}$$

Pour sélectionner des colonnes cohérentes, il suffit donc de sélectionner $\widetilde{n}_k$ multi-indices $\mathcal{J}_k \subset I_{k+1} \times \cdots \times I_{d-1}$. Les sous-matrices $\widetilde{A}_k^\chi$ sont alors définies par :

$$\widetilde{A}_k^\chi = \left\langle \boldsymbol{\mathcal{A}}^\chi(\mathcal{I}_{k-1}, I_k, \mathcal{J}_k, I_d^\chi) \right\rangle_2 \tag{VI.17}$$

Et les sélections de colonnes cohérentes sont données par $\widetilde{J}_k^\chi = \mathcal{J}_k \times I_d^\chi$ et telles que $\mathrm{Card}\left(\widetilde{J}_k^\chi\right) = \widetilde{n}_k^\chi = \widetilde{n}_k n_d^\chi$.

Finalement, d'après la proposition 6, la matrice $\widetilde{A}_k^\chi$ qu'on cherche à approcher par une décomposition de faible rang correspond à une sous-matrice du $k^{\text{ème}}$ déploiement de $\boldsymbol{\mathcal{A}}^\chi$:

$$\widetilde{A}_k^\chi = \langle \boldsymbol{\mathcal{A}}^\chi \rangle_k (\widetilde{\mathcal{I}}_k, \widetilde{\mathcal{J}}_k^\chi)$$

Pour tout $k \in [\![1, d-1]\!]$, on note :

$$D_{\leq k} = D_1 \times \cdots \times D_k$$
$$D_{\geq k} = D_k \times \cdots \times D_{d-1}$$

Pour $k \in [\![1, d-1]\!]$, on a alors

$$D_{\leq k-1}^{\mathcal{I}_{k-1}} \subset D_{\leq k-1} \quad \text{et} \quad D_{\geq k+1}^{\mathcal{J}_k} \subset D_{\geq k+1}$$

les points du domaine paramétrique respectivement associés aux multi-indices

$$\mathcal{I}_{k-1} \subset I_1 \times \cdots \times I_{k-1} \quad \text{et} \quad \mathcal{J}_k \subset I_{k+1} \times \cdots \times I_{d-1}$$

La sélection de colonnes correspond à la définition de $D_{\geq k+1}^{\mathcal{J}_k}$ par échantillonnage de $\widetilde{n}_k$ points de l'espace paramétrique $D_{\geq k+1}$. Le calcul des matrices $\widetilde{A}_k^{\chi}$ nécessite donc la résolution du modèle physique pour toutes les combinaisons de paramètres du produit cartésien $D_{\leq k-1}^{\mathcal{I}_{k-1}} \times D_k \times D_{\geq k+1}^{\mathcal{J}_k}$ contenant $s_{k-1} n_k \widetilde{n}_k$ points. Les points de $D_{\leq k-1}^{\mathcal{I}_{k-1}} \times D_k \times D_{\geq k+1}^{\mathcal{J}_k}$ peuvent être interprétés comme un faisceau de fibre dans la direction k du domaine paramétrique discrétisé $D = D_1 \times \dots D_{d-1}$.

Remarque 21. *Le nombre total de résolutions total du modèle physique au cours de l'algorithme est de*

$$\sum_{k=1}^{d} s_{k-1} n_k \widetilde{n}_k$$

En comparaison, les approximations tensorielles obtenues permettent d'obtenir instantanément une estimation de $n_1 \dots n_{d-1}$ solutions pour des valeurs de paramètres différentes du domaine discrétisé.

D'autre part, l'utilisation de l'interpolation multilinéaire par morceaux (Propostion 2) permet d'avoir une estimation sur le domaine paramétrique continu $\mathcal{D}_1 \times \mathcal{D}_{d-1}$. Cela illustre clairement l'intérêt de ce type d'approche pour représenter des solutions de modèle physique.

Échantillonnage de l'espace paramétrique L'objectif de l'échantillonnage (Éq. (VI.9)) est de sélectionner suffisamment de colonnes pour que l'espace engendré par les colonnes $\widetilde{\mathcal{J}}_k^{\chi}$ approche au mieux l'espace engendré par l'ensemble des colonnes de A_k^{χ}.

La sélection de colonnes étant équivalente à une sélection de points de $D_{k+1} \times \dots \times D_{d-1}$, elle peut être guidée par la connaissance empirique de la sensibilité des paramètres sur les solutions du modèle. Sans connaissance préalable, l'approche usuelle consiste à utiliser des méthodes de plans d'expérience (Section II.2) pour échantillonner les domaines paramétriques successifs.

Dans les applications (Chapitre VII), on considère des plans d'expérience basés sur des suites de Halton [57]. Les suites de Halton définies en dimension arbitraire sont caractérisées par leur faible disprépance. Elles permettent de générer des échantillonnages pseudo-aléatoires, répartis de manière homogène dans le domaine.

On fait l'hypothèse que les colonnes correspondantes sont suffisamment nombreuses et suffisamment bien distribuées pour avoir une erreur d'échantillonnage faible. La qualité de l'approximation doit ainsi être réalisée a posteriori une fois que la représentation tensorielle approchée a été construite. On la quantifie empiriquement sur un nombre réduit d'éléments via des mesures d'erreur entre les tenseurs physiques et leur approximation.

VI.3.b Sélection de lignes

À l'itération $k \in [\![1, d-1]\!]$, des ensembles de lignes $\mathcal{I}_k^\chi$ spécifiques à chaque sortie χ sont d'abord déterminés indépendamment. Dans les applications, on utilise typiquement l'algorithme 5 Q-DEIM qui prend en entrée la matrice V_k^χ et sélectionne un nombre de lignes égal au rang de V_k^χ.

À partir de cette sélection de lignes, il serait possible de définir pour chaque A_k^χ les approximations gappy T_k^χ suivantes :

$$T_k^\chi = V_k^\chi \left[V_k^\chi(\mathcal{I}_k^\chi, :)\right]^\dagger A_k^\chi(\mathcal{I}_k^\chi, :) \tag{VI.18}$$

Il est cependant possible d'exploiter l'interdépendance des tenseurs physiques pour définir des approximations plus précises.

D'après la proposition 5, les sélections de lignes $\mathcal{I}_k^\chi$ déterminent les matrices A_{k+1}^χ qu'il s'agira d'approcher à l'itération suivante :

$$A_{k+1}^\chi = \left\langle \boldsymbol{\mathcal{A}}(\mathcal{I}_k^\chi, I_{k+1}, \ldots, I_{d-1}, I_d^\chi) \right\rangle_2$$

L'approximation de ces matrices passe par le calcul d'un certain nombre de leurs colonnes (Section VI.3.a) associées à des résultats de simulations partagés.

La colonne d'indice α de la matrice A_{k+1}^χ correspond aux éléments de

$$\boldsymbol{\mathcal{A}}(\mathcal{I}_k^\chi, I_{k+1}, i_{k+1}^{(\alpha)}, \ldots, i_{d-1}^{(\alpha)}, i_d^{(\alpha)}) \in \mathbb{R}^{r_k^\chi \times n_{k+1}}$$

D'après l'interdépendance des tenseurs physiques (Remarque 20), le calcul de la colonne α pour toutes les matrices A_{k+1}^χ permet d'obtenir l'ensemble des éléments :

$$\boldsymbol{\mathcal{A}}(\mathcal{I}_k, I_{k+1}, i_{k+1}^{(\alpha)}, \ldots, i_{d-1}^{(\alpha)}, i_d^{(\alpha)}) \in \mathbb{R}^{s_k \times n_{k+1}}$$

où $\mathcal{I}_k$ correspond à l'union des indices de lignes $\mathcal{I}_k^\chi$:

$$\mathcal{I}_k = \bigcup_{\chi=1}^{N} \mathcal{I}_k^\chi$$

Finalement, les calculs de l'itération suivante fournissent un certain nombre d'éléments qu'il serait regrettable de ne pas utiliser sachant que les résolutions du modèle physique sont chères. On souhaite en effet intégrer dans la construction des décompositions TT un maximum de résultats disponibles pour permettre d'approcher les tenseurs physiques le plus précisément possible.

L'utilisation de l'approximation gappy permet d'approcher les matrices A_k^χ en se basant sur un nombre de lignes supérieur au rang de l'approximation et permet ainsi de construire

les approximations utilisées dans l'algorithme 12 :

$$T_k^\chi = H_k^\chi A_k^\chi(\mathcal{I}_k, :) \tag{VI.19}$$

La différence avec l'approximation gappy usuelle de l'équation (VI.18) correspond au fait que le nombre de lignes prises en compte est ici plus important permettant ainsi, en vertu de la proposition 10, de diminuer l'erreur d'approximation.

VI.3.c Calcul de la SVD

Dans les applications qui nous intéressent, les temps d'accès aux éléments des tenseurs physiques ne sont pas négligeables du fait qu'ils impliquent typiquement la résolution de systèmes d'équations différentielles. Lors de l'application de l'algorithme, 12 c'est essentiellement le nombre d'accès aux tenseurs physiques qui est déterminant en termes de temps de calcul.

Les matrices effectivement construites au cours de la procédure sont donc relativement petites dans le sens où les temps de calcul relatifs à l'application d'algorithmes d'algèbre linéaire comme la SVD et la Q-DEIM sont négligeables par rapport à la construction des matrices. En particulier, cela implique qu'il est possible de calculer la SVD complète (Éq. (VI.10)) des matrices $\widetilde{A}_k^\chi$. La détermination du rang de troncature reste toutefois un problème difficile et on utilise classiquement la méthode détaillée en section V.3.c pour le définir.

Troncature SVD Il est important de comprendre le rôle de la troncature SVD. L'objectif de la troncature des bases POD n'est pas intrinsèquement de réduire de rang de l'approximation puisque cette opération détériore a priori la qualité de l'approximation. L'objectif fondamental de la troncature est de limiter le nombre de résolutions du modèle physique à effectuer lors des itérations suivantes.

Les ensembles d'indices de lignes $\mathcal{I}_k^\chi$ sont déterminés en appliquant la Q-DEIM sur les matrices V_k^χ de rang r_k^χ. Ils sont donc chacun constitués de r_k^χ éléments. Par ailleurs d'après l'équation (VI.11), $\mathcal{I}_k$ correspond à l'union de ces sélections de lignes. Or, à l'itération $k+1$, le nombre de résolutions nécessaires du modèle physique est proportionnel au nombre de lignes des matrices A_{k+1}^χ qui lui même dépend du nombre de lignes $\mathcal{I}_k$ sélectionnées à l'itération courante k (Éq. (VI.17)).

Les valeurs des rangs de troncature r_k^χ influencent donc le nombre de résolutions physique à l'itération suivante. Plus le rang est grand et plus le nombre de calculs sera important. Les valeurs des tolérances ϵ_k^χ qui déterminent les rangs de troncature doivent donc assurer un compromis entre des approximations précises et des temps de calcul raisonnables. Le choix de leurs valeurs est en pratique basé sur l'expérience de l'utilisateur vis-à-vis des

rapports observés entre temps de calcul hors ligne et qualité des approximations.

Un autre élément est à prendre en compte pour le choix des tolérances ϵ_k^χ lorsque les tenseurs physiques sont entachés d'incertitudes. Aux différentes itérations, l'application de la SVD fournit dans ces cas des modes de bruit associés aux plus petites valeurs singulières. Il est fondamental de choisir un rang qui ne tient pas compte de ces modes. Ainsi, les valeurs de tolérance ϵ_k^χ doivent être cohérentes avec la précision du modèle physique sous peine de construire lors de la phase en ligne des approximations sur des bases non pertinentes.

Raffinement du rang de troncature Lors de l'application de la SVD, le choix en pratique d'une « bonne » valeur de troncature est un problème difficile bien connu. Le choix proposé dans la section V.3.c peut sembler arbitraire et peut d'autre part mener à ne pas tenir compte de modes qui auraient été pertinents. Une étude spécifique sur les méthodes de troncature qui tiennent typiquement compte de l'évolution des valeurs singulières n'a pas été menée dans cette thèse, mais serait intéressante.

On propose cependant une méthode qui améliore en pratique la qualité des approximations sans engendrer de résolution supplémentaire du modèle physique. La méthode consiste à redéfinir les matrices V_k^χ juste avant la définition des sorties H_k^χ dans l'algorithme 12.

Pour tout $\chi \in [\![1, N]\!]$, on calcule les matrices exactes des vecteurs singuliers à gauche $\widetilde{V}_k^\chi$ de $\widetilde{A}_k^\chi$ puis on cherche la valeur du rang $\bar{r}_k^\chi$ tel que l'approximation gappy :

$$\widetilde{T}_k^\chi = \widetilde{V}_k^\chi(:, [\![1 : \bar{r}_k^\chi]\!]) \left[(P_k)^T \, \widetilde{V}_k^\chi(:, [\![1 : \bar{r}_k^\chi]\!]) \right]^\dagger P_k^T \, \widetilde{A}_k^\chi$$

minimise la norme :

$$\left\| \widetilde{A}_k^\chi - \widetilde{T}_k^\chi \right\|$$

Le rang obtenu $r_k^{\chi,\text{best}}$ permet de définir la matrice de base réduite :

$$V_k^\chi := \widetilde{V}_k^\chi(:, [\![1, r_k^{\chi,\text{best}}]\!])$$

utilisée dans l'équation (VI.13).

L'application de cette méthode est illustrée dans la section VI.5.b.

VI.4 Détails relatifs aux applications

L'objectif de la méthodologie qu'on propose est d'approcher les applications $\boldsymbol{Y}^\chi$: $(\boldsymbol{\mu}, t) \mapsto \boldsymbol{Y}^\chi(\boldsymbol{\mu}, t)$ (Éq. (VI.3)) qui constituent le modèle de référence et définies à partir d'un modèle physique. On donne dans cette section des précisions sur la définition des entrées et sorties associées aux applications $\boldsymbol{Y}^\chi$.

VI.4.a Définition des sorties du modèle

Les sorties sont définies à partir de l'agrégation de m^χ quantités d'intérêt $Y^\chi_{\boldsymbol{\mu},j} : t \mapsto Y^\chi_{\boldsymbol{\mu},j}(t)$. Selon le modèle physique qu'on cherche à étudier, il s'agit de définir de manière la plus pertinente possible les quantités d'intérêt ainsi que les agrégations à effectuer.

VI.4.a.1 Domaine de définition des sorties dépendant du paramétrage

Pour certains modèles physiques, il est possible que l'intervalle de définition des sorties dépende des valeurs des entrées. C'est notamment le cas lors qu'on étudie des modèles d'endommagement dont où l'on cherche à mesurer l'influence des coefficients sur le temps de rupture du matériau.

Le fait que l'intervalle de définition temporelle dépende des valeurs des coefficients pose problème pour appliquer la méthodologie proposée. En effet, dans la définition des tenseurs physiques $\boldsymbol{\mathcal{A}}^\chi$ (Éq. (VI.5)), la taille de la dernière dimension (associée aux composantes et indices de temps) ne peut pas dépendre de la valeur des $d-1$ premiers indices.

On propose une méthode pour appliquer malgré tout l'algorithme 12. Soit $\boldsymbol{Y}_{\boldsymbol{\mu}}(t)$ une sortie du modèle associé au vecteur de paramètres $\boldsymbol{\mu}$ et définie sur l'intervalle de temps $[0, T_{\boldsymbol{\mu}}]$. Dans un premier temps, on définit la nouvelle sortie $\overline{Y}_{\boldsymbol{\mu}}(\bar{t})$ pour $\bar{t} \in [0,1]$ en renormalisant l'intervalle tel que $\bar{t} = \frac{t}{T_{\boldsymbol{\mu}}}$. Puis dans un second temps, on échantillonne les valeurs de $\overline{Y}_{\boldsymbol{\mu}}(\bar{t})$ sur une grille régulière de $[0,1]$. La figure VI.2 illustre les étapes de renormalisation des sorties.

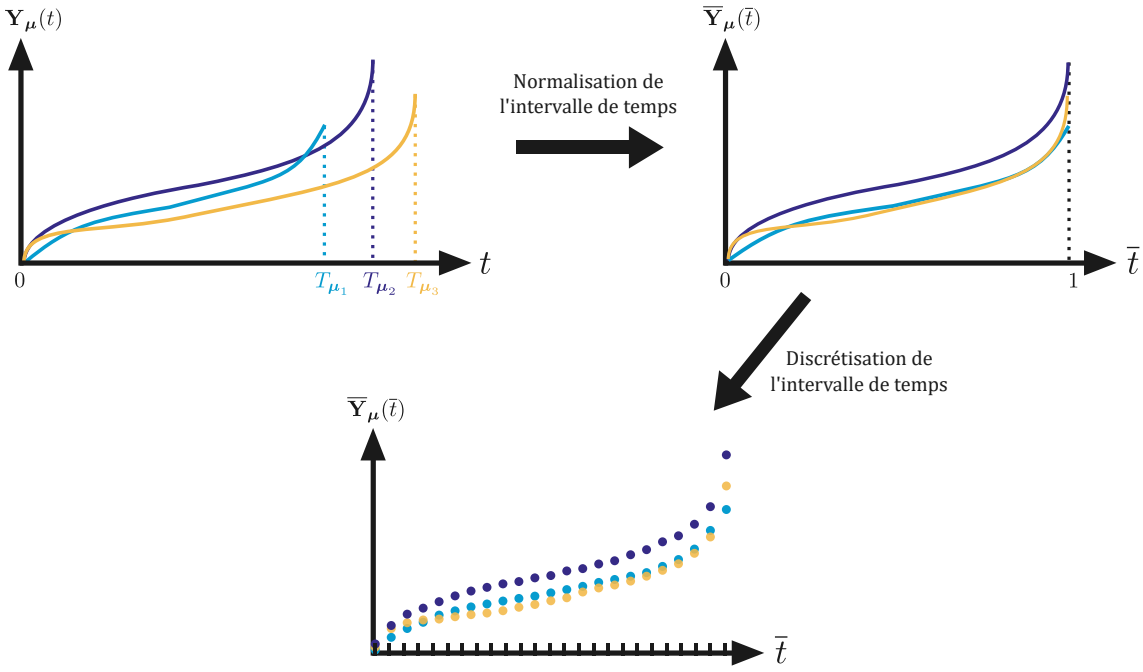


FIGURE VI.2 – Illustration de la normalisation et discrétisation temporelle.

Pour faire en sorte que le modèle de substitution soit capable de prédire la quantité

$\boldsymbol{Y}_{\boldsymbol{\mu}}(t)$ définie sur l'intervalle $[0, T_{\boldsymbol{\mu}}]$, on définit comme sorties du modèle de référence la quantité $\overline{Y}_{\boldsymbol{\mu}}$ discrétisé ainsi que la quantité $T_{\boldsymbol{\mu}}$.

VI.4.a.2 Influence de l'agrégation des quantités d'intérêt

Au cours de l'algorithme 12, on construit pour chaque χ des bases POD V_k^χ représentant en quelque sorte l'évolution de la sortie de $\boldsymbol{Y}^\chi$ en fonction des paramètres. Les quantités d'intérêt agrégées dans la sortie de $\boldsymbol{Y}^\chi$ ont donc une influence sur la forme des bases POD. D'autre part, les éléments des tenseurs physiques sont approchés lors de la phase en ligne par les décompositions TT qui reposent sur ces bases POD. Par conséquent, la manière dont sont agrégées les quantités d'intérêt joue un rôle majeur sur la précision des approximations. Sachant que les trains de tenseurs sont basés essentiellement sur une suite d'approximations matricielles, on illustre l'influence des agrégations dans le cas de matrices.

Soit deux matrices $A^1 \in \mathbb{R}^{n\times m_1}$ et $A^2 \in \mathbb{R}^{n\times m_2}$ que l'on cherche à approcher par des approximations de faible rang. Soit $A = [A^1; A^2]$ la concaténation des matrices et V^1, V^2, V les matrices des bases POD respectivement associées à A^1, A^2, A. On cherche à montrer dans les deux paragraphes suivants les conséquences de l'agrégation ou non des matrices A^1 et A^2 en termes de précision des approximations et de coûts de calcul engendrés.

Qualité des approximations On veut comparer la différence entre des approximations de A^1 et A^2 basées d'une part sur V^1 et V^2 et d'autre part sur V.

Si les normes des colonnes de A^1 sont grandes devant celles des colonnes de A^2 alors la base POD V représentera davantage la matrice A^1. Pour approcher A^2 dans la base V il sera alors nécessaire de choisir un rang de troncature important.

D'autre part si $\dim\left(\mathrm{Im}(A^1) \bigcap \mathrm{Im}(A^2)\right)$ est petit, c'est-à-dire si les colonnes de A^1 et A^2 engendrent des espaces très différents alors pour un rang de troncature fixée les approximations sur les bases V^1 et V^2 seront nécessairement meilleures que les approximations sur la base V.

Plus généralement, l'utilisation de deux bases POD distinctes V^1 et V^2 est toujours meilleure en termes de précision.

Coût de calcul Le nombre de résolutions du modèle aux itérations suivantes dépend du nombre de lignes sélectionnées sur chaque matrice de base POD associée aux différentes sorties χ (Section VI.3.b).

Si $\mathrm{Im}(A^1) \simeq \mathrm{Im}(A^2)$ de telle sorte que les approximations sur les bases V^1 et V^2 ou sur la base jointe V soient comparables pour un rang de troncature r fixé. Comme les sélections des lignes $\mathcal{I}^1$ pour V_1 et $\mathcal{I}^2$ pour V_2 (via la Q-DEIM) sont réalisées indépendamment, les ensembles $\mathcal{I}^1$ et $\mathcal{I}^2$ sont couramment distincts. Par conséquent le nombre d'éléments

dans l'union $\mathcal{I}^1 \bigcup \mathcal{I}^2$ est en général plus grand que le nombre de lignes sélectionnées uniquement sur la base jointe V. Dans ce cas, les calculs engendrés aux itérations suivantes en considérant des bases distinctes seront plus importants alors que les approximations ne seront pas plus précises.

Bilan Les deux paragraphes précédents montrent à quel point les choix d'agrégation ont une importance majeure vis-à-vis de la précision des approximations et des coûts de calcul hors lignes. Le cas d'application présenté dans la section VI.5.a illustre l'influence des agrégations sur la précision des modèles de substitution résultants. L'idée fondamentale consiste donc à agréger ensemble les quantités d'intérêt dont l'amplitude et la forme des variations sont les plus similaires. Ces choix doivent être réalisés en connaissance de cause par les experts. On recommande de séparer les grandeurs physiques en fonction de leur unité de mesure.

VI.4.b Définition des entrées du modèle

La définition des applications à approcher passe par la définition des sorties(Section VI.4.a), mais également par la définition des entrées. Nous allons voir qu'une fois encore la connaissance des modèles physiques par les experts joue un rôle capital vis-à-vis de la qualité des représentations construites par l'algorithme.

Reparamétrage Jusqu'à maintenant, on a considéré que les paramètres utilisés dans la définition des applications $\boldsymbol{Y}^\chi$ s'identifiaient aux coefficients apparaissant dans les équations des modèles physiques. Il est possible en réalité possible de définir le modèle de référence à approcher en utilisant de nouveaux paramètres résultant d'un reparamétrage des coefficients physiques.

Un reparamétrage est typiquement à effectuer lorsque, dans les équations, des relations explicites apparaissent entre les paramètres physiques. Par exemple si les paramètres (x, y) apparaissent uniquement sous la forme $(x - y)$, cela signifie que seule la différence entre les deux coordonnées a un effet sur les solutions. Dans ce cas, il est plus pertinent de considérer la variable $z = x - y$ pour définir les fonctions à approcher. C'est en pratique aux experts qui connaissent le mieux les modèles physiques de déterminer les reparamétrages les plus pertinents. Leurs choix peut néanmoins être guidé par l'utilisation de modèles de substitution grossiers pouvant par exemple reposer sur des décompositions en trains de tenseurs.

Le reparamétrage de paramètres pris individuellement peut également avoir un intérêt. Pour appliquer l'algorithme 12, les intervalles de chaque paramètre doivent être discrétisés. Par conséquent, un changement de variable n'impliquant qu'un seul paramètre est strictement équivalent à une discrétisation différente de l'intervalle de variation. Il n'y a pas

de généralité sur la meilleure manière de discrétiser un intervalle de variation. Il peut par exemple être intéressant de discrétiser plus finement certaines zones si l'on sait que les variations y sont plus importantes. En pratique, c'est à l'expert de juger, en fonction de ce qu'il cherche à représenter, quelle est la meilleure manière de discrétiser les intervalles.

Organisation des dimensions L'algorithme 12 procède par itération sur les dimensions du tenseur par conséquent l'ordre des dimensions a potentiellement une influence sur la décomposition TT vis-à-vis de la qualité de l'approximation, mais aussi des tailles de stockages. Plus généralement, d'après [61, Remarque 3.2], la représentation TT d'un tenseur quelconque n'est pas invariante par permutation des dimensions. Cela implique en particulier qu'il existe des ordres de dimensions pour lesquels les rangs de compression sont plus faibles et donc les représentations sont plus compactes. Une analyse sur l'influence de l'organisation des dimensions mériterait d'être effectuée. On propose ici quelques pistes de réflexion relatives à cette problématique.

Les dimensions des tenseurs physiques sont associées à différents types de variables :

— la variable de temps ;
— les composantes des sorties ;
— les paramètres du modèle physique.

Elles sont en particulier associées soit à des entrées (paramètres) soit à des sorties (temps et composantes) du modèle physique.

Un premier constat est qu'il est préférable de placer les dimensions associées aux sorties en dernière position. En effet, le principe de l'algorithme est de procéder à des sélections d'indices successivement sur les dimensions. Or les dimensions de sortie sont interdépendantes (Remarque 20), c'est-à-dire que les éléments associés à tous les indices des dimensions de sortie ne peuvent pas être obtenus indépendamment. Cette interdépendance est en partie due à la causalité des évolutions temporelles. Effectuer une sélection sur ces dimensions n'a donc pas d'intérêt et ne peut que dégrader la solution à temps de calcul égal. Pour cette même raison, il est usuel d'associer les dimensions de sortie ensemble pour obtenir des tenseurs réarrangés dont uniquement la dernière dimension est associée aux sorties (comme ce qui est fait à la fin de la section VI.1.b sur les tenseurs $\boldsymbol{\mathcal{A}}^{\chi}$).

On s'intéresse à présent aux dimensions associées aux entrées. La sensibilité d'une entrée est quantifiée par l'ampleur des variations produites sur les sorties lorsqu'on modifie sa valeur. La sensibilité des entrées du modèle de référence n'est pas identique. On préconise d'organiser les dimensions d'entrée par ordre décroissant de sensibilité. Cela est suggéré par le fait qu'à la première itération la première dimension est explorée de manière exhaustive. Il semble donc naturel de placer en première position la dimension la plus sensible. Une étude plus poussée serait nécessaire pour confirmer ou infirmer cette supposition.

VI.4.c Accumulation des erreurs

On liste ici l'ensemble des erreurs introduites aux différentes étapes de la méthodologie proposée. Pour modéliser un phénomène physique, on met en place ce qu'on appelle un modèle physique typiquement formulé par un ensemble d'équations différentielles. Entre la réalité et le modèle physique, il existe nécessairement ce qu'on appelle une *erreur de modélisation*.

On appelle *modèle numérique* la procédure de résolution permettant d'approcher numérique les solutions du modèle physique. En mécanique, on utilise typiquement des méthodes éléments finies et des schémas numériques d'intégration temporelle. Des *erreurs de discrétisation* temporelle et spatiale sont alors introduites lors de l'intégration des équations sur les domaines discrétisés.

Dans l'approche proposée, c'est concrètement le modèle numérique qui est considéré comme « la vérité » au sens où l'on chercher à approcher les solutions qu'il génère sans se préoccuper des erreurs introduites en amont. *L'erreur d'approximation* introduite lors de la construction du modèle de substitution correspond ici à la différence entre les tenseurs physiques et leurs représentations TT approchées. Cette erreur correspond à l'accumulation des erreurs des approximations de rang faible. On peut sous certaines hypothèses donner une borne supérieure pour cette erreur (Proposition 11).

Le modèle de substitution basé sur des décompositions TT approche les solutions du modèle numérique essentiellement pour des valeurs de paramètres du domaine discrétisé. L'utilisation d'une méthode d'interpolation, telle que l'interpolation multilinéaire par morceaux (Éq. (IV.8)) qu'on propose, est indispensable pour définir un modèle de substitution sur l'ensemble du domaine paramétrique. L'*erreur d'interpolation paramétrique*, introduite lors de l'évaluation du modèle de substitution pour des valeurs de paramètres hors du domaine discrétisé, peut être réduite en raffinant la grille de discrétisation.

Notons que l'interpolation multilinéaire par morceaux ne s'applique pas à toutes les dimensions des tenseurs TT. En effet, certaines d'entre elles sont associées à des variables intrinsèquement discrètes. C'est le cas de la dimension correspondant à l'indice i_{comp} qui énumère les composantes des quantités d'intérêt.

VI.5 Applications théoriques

Cette section illustre l'application de l'algorithme 12 pour approcher deux fonctions f et g multivariées à valeurs vectorielles hétérogènes. En particulier, on montre que l'importance de l'agrégation des sorties et on illustre l'amélioration de la précision permise par la méthode de raffinement du rang de troncature proposée dans la section VI.3.c.

La fonction f correspond à la fonction de Rosenbrock (Éq. (V.26)). Pour la fonction g que procède de manière similaire en se basant sur une généralisation en dimension arbitraire

de la fonction de Ackley.

Soit la fonction :

$$G(x_1, x_2, x_3, x_4, x_5) = -20 \exp\left(0,2\sqrt{1/5 \sum_{i=1}^{5} x_i}\right) - \exp\left(1/5 \sum_{i=1}^{5} \cos(2\pi x_i)\right) + 20 + \exp(1)$$

Avec les mêmes intervalles de définition et discrétisations que dans la section V.4, on définit g telle que :

$$\begin{array}{rccc} g & : & [-3,3]^4 & \to & \mathbb{R}^{40} \\ & & (x_1, x_2, x_3, x_4) & \mapsto & \begin{pmatrix} g_1(x_1, x_2, x_3, x_4) \\ \vdots \\ g_{40}(x_1, x_2, x_3, x_4) \end{pmatrix} \end{array}$$

où pour tout $j \in [\![1, 40]\!]$:

$$g_j(x_1, x_2, x_3, x_4) = G(x_1, x_2, x_3, x_4, x_5^{(j)})$$

On suppose que les fonctions f et g ne sont pas explicitement connues mais que l'on dispose qu'une procédure de calcul (le modèle de référence) qui prend en entrée des valeurs pour les coordonnées (x_1, x_2, x_3, x_4) et retourne simultanément les deux vecteurs $f(x_1, x_2, x_3, x_4) \in \mathbb{R}^{40}$ et $g(x_1, x_2, x_3, x_4) \in \mathbb{R}^{40}$. Le domaine paramétrique correspond ici au domaine $[-3, 3]^4$ sur lequel sont définis les paramètres $(x_1, \ldots, x_4)$.

Toujours avec les mêmes discrétisations que dans la section V.4, on définit les tenseurs $\boldsymbol{\mathcal{A}}^1, \boldsymbol{\mathcal{A}}^2 \in \mathbb{R}^{50\times50\times50\times50\times40}$ tels que :

$$\begin{aligned} \boldsymbol{\mathcal{A}}^1(i_1, \ldots, i_5) &= f_{i_5}(x_1^{(i_1)}, x_2^{(i_2)}, x_3^{(i_3)}, x_4^{(i_4)}) \\ \boldsymbol{\mathcal{A}}^2(i_1, \ldots, i_5) &= g_{i_5}(x_1^{(i_1)}, x_2^{(i_2)}, x_3^{(i_3)}, x_4^{(i_4)}) \end{aligned}$$

VI.5.a Agrégation

Afin d'évaluer l'influence de l'agrégation des quantités d'intérêt, on définit également le tenseur $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{50\times50\times50\times50\times80}$ qui correspond à d'agrégation des fonctions f et g :

$$\boldsymbol{\mathcal{B}}(i_1, \ldots, (i_5, i_{comp})) = \begin{cases} f_{i_5}(x_1^{(i_1)}, x_2^{(i_2)}, x_3^{(i_3)}, x_4^{(i_4)}) & \text{si} \quad i_{comp} = 1 \\ g_{i_5}(x_1^{(i_1)}, x_2^{(i_2)}, x_3^{(i_3)}, x_4^{(i_4)}) & \text{si} \quad i_{comp} = 2 \end{cases}$$

On applique l'algorithme 12 d'une part sur les tenseurs $\boldsymbol{\mathcal{A}}^\chi$ pour construire un modèle de substitution dit *découplé* et d'autre part sur le tenseur $\boldsymbol{\mathcal{B}}$ pour construire un modèle

de substitution dit *couplé*. Pour chaque itération, on choisit une tolérance de troncature $\epsilon_{\text{tol}} = 10^{-3}$ et $\widetilde{n}_k = 50$ points d'échantillonnage du domaine paramétrique.

Les constructions des metamodèles couplé et découplé impliquent respectivement l'évaluation du modèle de référence 35400 et 22750 fois. Les tailles de stockage après refactorisation des représentations approchées sont :

— Pour $\boldsymbol{\mathcal{A}}^1$: $(3, 6, 8, 4)$;
— Pour $\boldsymbol{\mathcal{A}}^2$: $(5, 6, 6, 6)$;
— Pour $\boldsymbol{\mathcal{B}}$: $(3, 5, 5, 4)$;

Pour les modèles de substitution couplé et découplé, on associe respectivement les fonctions $\widetilde{f}^1$, $\widetilde{g}^1$ et $\widetilde{f}^2$, $\widetilde{g}^2$ qui approchent les fonctions f et g sur l'ensemble du domaine continue $[-3, 3]^4$ grâce l'interpolation multilinéaire par morceaux (Section IV.1.d). Il est également possible de les interpréter comme des fonctions scalaires multivariées définies pour tout $(x_1, \ldots, x_5)$.

Pour chaque modèle de substitution les erreurs d'approximation relatives e_f^i et e_g^i sont définies par :

$$e_f^i(x_1, x_2, x_3, x_4) = \frac{\left\| f(x_1, x_2, x_3, x_4) - \widetilde{f}^i(x_1, x_2, x_3, x_4) \right\|_2}{\| f(x_1, x_2, x_3, x_4) \|_2} \tag{VI.20}$$

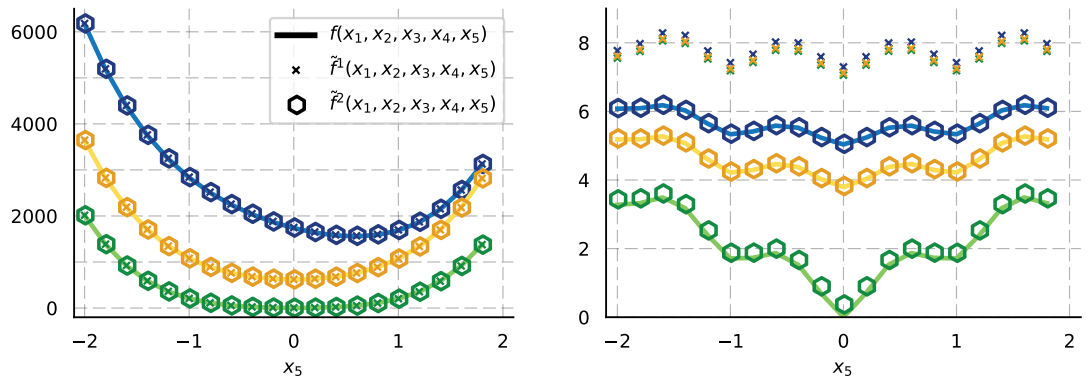
$$e_g^i(x_1, x_2, x_3, x_4) = \frac{\| g(x_1, x_2, x_3, x_4) - \widetilde{g}^i(x_1, x_2, x_3, x_4) \|_2}{\| g(x_1, x_2, x_3, x_4) \|_2} \tag{VI.21}$$

Par un tirage aléatoire de 10 000 points sur le domaine paramétrique discrétisé (voir section V.4), on estime que :

$$e_f^1 = 4,8.10^{-8} \quad \text{et} \quad e_f^2 = 1,8.10^{-17}$$
$$e_g^1 = 1,3.10^{-3} \quad \text{et} \quad e_g^2 = 1,2.10^{-6}$$

On remarque que les faibles coûts de calcul supplémentaires pour construire un modèle découplé par rapport au modèle couplé apportent des gains de précision particulièrement importants.

On peut facilement comprendre ce phénomène en visualisant la forme des fonctions f et g. La figure VI.3 présente les sorties associées à 3 vecteurs (x_1, x_2, x_3, x_4) du domaine paramétrique continu, pour le modèle de référence (f et g) et les modèles de substitution ($\widetilde{f}^1$, $\widetilde{f}^2$ et $\widetilde{g}^1$, $\widetilde{g}^2$). On remarque que les ordres de grandeur des fonctions f et g sont très différents, par conséquent la construction des modes POD lors de l'algorithme aura tendance à représenter la fonction f plutôt que la fonction g. C'est pour cette raison que la fonction g est très mal approchée par le modèle de substitution couplé.



(a) Première quantité d'intérêt.

(b) Seconde quantité d'intérêt.

FIGURE VI.3 – Comparaison de sorties entre les métamodèles couplé et découplé.

VI.5.b Raffinement du rang de troncature

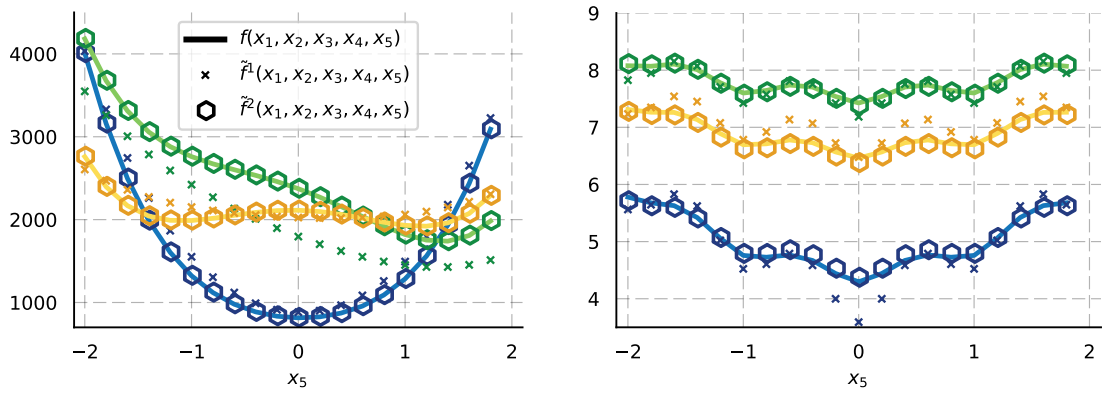
On applique l'algorithme 12 sur les tenseurs A^χ en choisissant dans un premier temps la méthode de troncature proposée dans la section V.3.c (metamodèle simple) puis dans un second temps avec la méthode de raffinement du rang proposée dans la section VI.3.c (modèle raffiné). Pour chaque itération, on choisit une tolérance de troncature $\epsilon_{\text{tol}} = 0.015$ et $\widetilde{n}_k = 50$ points d'échantillonnage du domaine paramétrique. On choisit une valeur de troncature assez grande pour mieux illustrer le propos. Les constructions des modèles simple et raffiné impliquent le même nombre d'évaluations du modèle de référence : 30300.

Pour les modèles de substitution simple et raffiné, on associe respectivement les fonctions $\widetilde{f}^1$, $\widetilde{g}^1$ et $\widetilde{f}^2$, $\widetilde{g}^2$ qui approchent les fonctions f et g sur l'ensemble du domaine continue $[-3, 3]^4$ grâce l'interpolation multilinéaire par morceaux (Section IV.1.d).

Pour chaque modèle de substitution les erreurs d'approximation relatives e_f^i et e_g^i sont définies par les équations (VI.20) et (VI.21). Par un tirage aléatoire de 10 000 points sur le domaine paramétrique discrétisé (Section V.4), on estime :

$$e_f^1 = 1,9.10^{-4} \quad \text{et} \quad e_f^2 = 4.10^{-17}$$
$$e_g^1 = 1,8.10^{-4} \quad \text{et} \quad e_g^2 = 5,9^{-5}$$

On remarque que pour des coûts de calcul égaux, le modèle raffiné est significativement plus précis que le modèle simple. La figure VI.4 présente les sorties associées à 3 vecteurs (x_1, x_2, x_3, x_4) du domaine paramétrique continu, pour le modèle de référence (f et g) et les modèles de substitution ($\widetilde{f}^1$, $\widetilde{f}^2$ et $\widetilde{g}^1$, $\widetilde{g}^2$).



(a) Première quantité d'intérêt.

(b) Seconde quantité d'intérêt.

FIGURE VI.4 – Comparaison de sorties entre les métamodèles simple et raffiné.

VI.6 Améliorations possibles

Cette section évoque quelques améliorations possibles de l'algorithme 12.

VI.6.a SVD incrémentale

Dans les applications, les matrices $\widetilde{A}_k^\chi$ peuvent avoir des tailles particulièrement importantes principalement dues au grand nombre de colonnes. Les espaces mémoire impliqués (de l'ordre du gigaoctet) peuvent alors poser des problèmes pour appliquer numériquement les algorithmes.

Des techniques ont été proposées [17, 102] pour approcher des décompositions en valeurs singulières de matrices $A \in \mathbb{R}^{n\times m}$ par des calculs incrémentaux. L'idée de ces méthodes est de mettre à jour les approximations des termes de la SVD en incorporant au fur et à mesure les colonnes de A (soit une à une, soit par bloc).

Comme mentionné dans la section VI.3.c les matrices considérées dans nos applications possèdent un grand nombre de colonnes et sont donc couteuses à stocker en mémoire. Cependant puisqu'elles possèdent un nombre de lignes limité, le calcul de la SVD exacte est envisageable.

On prouve deux résultats relatifs au calcul de la SVD exacte.

Proposition 14. *Soient $X_1 \in \mathbb{R}^{n\times m_1}$ et $X_2 \in \mathbb{R}^{n\times m_2}$ dont les décompositions en valeurs singulières sont données par :*

$$X_1 = V_1 \Sigma_1 W_1^T$$
$$X_2 = V_2 \Sigma_2 W_2^T$$

Alors les matrices des vecteurs singuliers à gauche correspondant aux concaténations de matrices

$$[X_1; X_2] \ , \ [X_1; V_2\Sigma_2] \ \textit{et} \ [V_1\Sigma_1; V_2\Sigma_2]$$

sont identiques.

Remarque 22. *La démonstration de la proposition 14 est évidente d'après la proposition 17 et en remarquant que*

$$[X_1; X_2]\,[X_1; X_2]^T = [X_1; V_2\Sigma_2]\,[X_1; V_2\Sigma_2]^T = [V_1\Sigma_1; V_2\Sigma_2]\,[V_1\Sigma_1; V_2\Sigma_2]^T$$

D'après l'équation (VI.17), les matrices $\widetilde{A}_k^{\chi}$ sont constituées de bloc de colonnes $\widetilde{A}_k^{\chi,(\alpha)}$ calculable indépendamment :

$$\widetilde{A}_k^{\chi,(\alpha)} = \left\langle \boldsymbol{\mathcal{A}}^{\chi}(\mathcal{I}_{k-1}, I_k, i_{k+1}^{(\alpha)}, \ldots, i_{d-1}^{(\alpha)}, I_d^{\chi}) \right\rangle_2 \in \mathbb{R}^{(s_{k-1}n_k)\times n_d^{\chi}}$$

pour tout $(i_{k+1}^{(\alpha)}, \ldots, i_{d-1}^{(\alpha)}) \in \mathcal{J}_k$ avec $\alpha \in [\![1, \widetilde{n}_k]\!]$ et tels que :

$$\widetilde{A}_k^{\chi} = \left[\widetilde{A}_k^{\chi,(1)}; \ldots; \widetilde{A}_k^{\chi,(\widetilde{n}_k)}\right] \tag{VI.22}$$

Au cours de l'algorithme 12, on cherche à calculer les matrices des vecteurs singuliers à gauche V_k^{χ} des matrices $\widetilde{A}_k^{\chi}$. En vertu de la proposition 14 et de la décomposition (Éq. (VI.22)), il est possible de les calculer :

- soit de manière incrémentale en mettant à jour les matrices à V_k^{χ} et Σ_k^{χ} au fur et à mesure du calcul des $\widetilde{A}_k^{\chi,(\alpha)}$;
- soit de manière parallèle en calculant les matrices des vecteurs singuliers et les matrices des valeurs singulières pour chaque $\widetilde{A}_k^{\chi,(\alpha)}$ puis en recombinant ;
- soit par une approche mixte à la fois incrémentale et parallèle.

Ces différentes approches permettant de limiter les espaces mémoire impliqués au cours de l'algorithme, car ne font pas intervenir le stockage complet des matrices $\widetilde{A}_k^{\chi}$.

Il est important de noter que cette approche est valable seulement si on considère des calculs exacts. Par conséquent, il est nécessaire d'effectuer la troncature une fois seulement la matrice V_k^{χ} finale obtenue. Ce type d'approche peut par ailleurs faire apparaitre des instabilités de calculs numériques qui n'ont pas été étudiées en détail dans cette thèse.

VI.6.b Autres perspectives d'amélioration

Les perspectives d'amélioration de la méthode sont nombreuses, on donne ici deux pistes supplémentaires.

Dans l'algorithme actuel, les sélections d'indices de lignes $\mathcal{I}_k$ sont définies comme l'union des points d'interpolation ayant été sélectionnés indépendamment sur chaque base. Cette approche est clairement sous optimale en général. L'exemple donné dans la section VI.4.a illustre un cas où cette méthode sélectionne nombre de points trop importants qui n'apportent pas de gain de précision. Une piste d'amélioration de la méthode consiste à sélectionner les lignes en considérant simultanément l'ensemble des matrices de base réduites. On peut par exemple s'inspirer de méthodes basées sur la résolution de problème de minimisation telles que la BPIM (Section III.2.a.4). La difficulté principale qui émerge lorsqu'on considère les matrices de base réduites simultanément est qu'il est nécessaire d'introduire une notion de distance commune, or ce problème est difficile puisque les sorties sont par définition hétérogènes.

Une seconde amélioration concerne cette fois-ci, la sélection des colonnes utilisées dans la Snapshot POD. On a évoqué la difficulté de déterminer a priori et de manière pertinente suffisamment de colonnes pour espérer engendrer l'espace complet. La validation croisée (*Cross-validation*) est un type d'heuristique permettant de vérifier si la sélection de colonnes est fiable. La méthode *leave-one-out* fait partie des approches de validation croisée. Elle consiste à successivement sélectionner un vecteur de l'espace d'échantillonnage et de vérifier qu'il peut être représenté par l'ensemble des vecteurs restants. Pour un échantillonnage de l'espace suffisamment exhaustif, cette condition est nécessairement respectée. Cette méthode qui vérifie que cette condition est respectée permet donc de donner un indicateur sur la fiabilité de l'échantillonnage. Pour les équations ayant un estimateur d'erreur à faible complexité, la sélection de colonnes peut être faite par algorithme glouton incrémental, à la façon de ce qui a été fait dans la méthode APHR [99].

Chapitre VII

Application de la méthodologie

Ce chapitre présente l'application de la méthodologie pour construire des modèles de substitution associés à des lois de comportement matériaux à complexité croissante. Les applications représentent typiquement des cas d'études représentatifs des besoins industriels.

Table des matières

VII.1 Généralités . . . 142
VII.2 Loi élasto-viscoplastique non linéaire . . . 142
VII.2.a Modèle physique . . . 143
VII.2.b Résultats . . . 146
VII.3 Premier cas industriel : Polystar . . . 153
VII.3.a Modèle physique . . . 153
VII.3.b Résultats . . . 156
VII.3.c Identification . . . 159
VII.4 Second cas industriel : RaftX . . . 160
VII.4.a Définition du modèle physique de référence . . . 160
VII.4.b Résultats . . . 162
VII.4.c Identification . . . 164
VII.5 Couplage avec l'hyper-réduction . . . 164
VII.5.a Cas d'étude . . . 165
VII.5.b Description de la méthodologie de couplage . . . 166
VII.5.c Construction du modèle hyper-réduit . . . 167
VII.5.d Sélection des points d'apprentissage pour l'hyper-réduction . . . 167

VII.1 Généralités

Dans l'ensemble de ce chapitre, on s'intéresse à des modèles de lois de comportement relatives à des matériaux métalliques. Les variables dépendantes du temps impliquées dans les équations sont :

- Le tenseur des déformations : $\underset{\sim}{\boldsymbol{\varepsilon}} = \underset{\sim}{\boldsymbol{\varepsilon}}_e + \underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$ [sans dimension] (somme d'une contribution élastique et viscoplastique) ;
- Le tenseur des contraintes : $\underset{\sim}{\boldsymbol{\sigma}}$ [MPa] ;
- Une variable interne d'écrouissage : $\underset{\sim}{\boldsymbol{X}}$ [MPa] ;
- La déformation viscoplastique cumulée : p [sans dimension].

Les grandeurs $\underset{\sim}{\boldsymbol{\varepsilon}}$, $\underset{\sim}{\boldsymbol{\varepsilon}}_e$, $\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$, $\underset{\sim}{\boldsymbol{\sigma}}$ et $\underset{\sim}{\boldsymbol{X}}$ sont des tenseurs (au sens mécanique) du second ordre.

On se place dans le cadre de la théorie des petites déformations. En pratique, on considère que cette hypothèse est respectée lorsque $\max(\underset{\sim}{\boldsymbol{\varepsilon}}) = 8\%$.

La précision des modèles de substitution est estimée a posteriori en mesurant une distance (différente selon les applications) entre les sorties du modèle de substitution et les sorties du modèle physique de référence. L'estimation est réalisée sur la base de sorties associées à un nombre élevé de vecteurs de paramètres échantillonnés de manière aléatoire sur le domaine discrétisé. Chaque vecteur est obtenu en tirant la valeur des paramètres selon une loi uniforme sur les intervalles discrétisés respectifs. En choisissant des points du domaine discrétisé, on ne fait pas intervenir d'erreur d'interpolation sur le domaine paramétrique. Lorsque les sorties dépendent du temps, seule une erreur d'interpolation temporelle est introduite lors de la prédiction par le modèle de substitution.

Les erreurs d'approximation sont définies à l'aide des normes suivantes :

$$\|x\|^2_{[0,T]} = \int_0^T x^2 dt \quad \text{et} \quad \left\|\underset{\sim}{\boldsymbol{x}}\right\|^2_{[0,T]} = \int_0^T \underset{\sim}{\boldsymbol{x}} : \underset{\sim}{\boldsymbol{x}}\, dt$$

où x et $\underset{\sim}{\boldsymbol{x}}$ correspondent à des fonctions dépendantes du temps à valeurs respectivement scalaires et tensorielles.

Dans l'ensemble des applications, la même valeur de troncature ϵ_{tol} est utilisée à chaque itération de l'algorithme 12 et la procédure de refactorisation est systématiquement appliquée.

VII.2 Loi élasto-viscoplastique non linéaire

On considère une loi de comportement élasto-viscoplastique [13, 76] hautement non linéaire sollicitée lors d'essais de fatigue. Cette application permet de démontrer la faisabilité de l'approche proposée pour traiter des solutions de modèles matériaux formulés par des équations différentielles algébriques.

VII.2.a Modèle physique

Le modèle implique huit coefficients matériaux : E, ν, n, K, R_0, Q, b et C. Les coefficients de Young et Poisson sont respectivement fixés à $E = 200\,000$ MPa et $\nu = 0.3$. Le tableau VII.1 présente les intervalles de variation des autres coefficients matériaux, considérés comme les entrées du modèle de référence.

	n	K	R_0	Q	b	C
Unité		MPa.s^{-n}	MPa	MPa		MPa
Borne inférieure	2	100	1	1	0.02	150
Borne supérieure	12	10 000	200	2 000	2 000	150 000

TABLE VII.1 – Intervalles de variation des paramètres du modèle élasto-viscoplastique.

VII.2.a.1 Système d'équations

On présente ici les équations du modèle en mettant en évidence en vert les paramètres variables du modèle. Le comportement élastique est régi par :

$$\underset{\sim}{\boldsymbol{\sigma}} = \frac{E}{1+\nu}\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e + \frac{\nu}{1-2\nu} Tr\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e\right)\underset{\sim}{\mathbb{I}}\right) \tag{VII.1}$$

Le comportement viscoplastique est décrit par la loi de Norton (Éq. (VII.2)) formulée avec un critère de von Mises (Éq. (VII.5)). La fonction de charge et la normale à la surface de plasticité sont données par les équations (VII.3) et (VII.4). (VII.6) rappelle la définition de la partie déviatorique du tenseur des contraintes impliqué dans l'équation (VII.5).

$$\frac{d}{dt}\underset{\sim}{\boldsymbol{\varepsilon}}_{vp} = \underset{\sim}{\boldsymbol{N}}\left(\frac{f}{K}\right)_+^n \tag{VII.2}$$

$$f = J\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right) - R \tag{VII.3}$$

$$\underset{\sim}{\boldsymbol{N}} = \frac{3}{2}\frac{\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}}{J\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right)} \tag{VII.4}$$

$$J\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right) = \sqrt{\frac{3}{2}\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right) : \left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right)} \tag{VII.5}$$

$$\underset{\sim}{\boldsymbol{\sigma}}^D = \underset{\sim}{\boldsymbol{\sigma}} - \frac{1}{3} Tr\left(\underset{\sim}{\boldsymbol{\sigma}}\right)\underset{\sim}{\mathbb{I}} \tag{VII.6}$$

où $(.)_+$ est la partie positive.

L'opérateur **:** dénote le produit contracté défini tel que :

$$\underset{\sim}{\boldsymbol{Z}}_1 : \underset{\sim}{\boldsymbol{Z}}_2 = \sum_{i=1}^{3}\sum_{j=1}^{3} Z_1^{ij} Z_2^{ij} \quad \text{pour} \quad \underset{\sim}{\boldsymbol{Z}}_1, \underset{\sim}{\boldsymbol{Z}}_2 \in \mathbb{R}^{3\times 3}$$

L'écrouissage non linéaire isotropique est modélisé par l'équation (VII.7) où (VII.8) donne le taux de déformation viscoplastique.

$$R = R_0 + Q\left(1 - e^{-bp}\right) \tag{VII.7}$$

$$\frac{dp}{dt} = \sqrt{\frac{2}{3}\frac{d}{dt}\underset{\sim}{\boldsymbol{\varepsilon}}_{vp} : \frac{d}{dt}\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}} \tag{VII.8}$$

Finalement, l'écrouissage linéaire cinématique est donné par :

$$\underset{\sim}{\boldsymbol{X}} = \frac{2}{3} C \underset{\sim}{\boldsymbol{\varepsilon}}_{vp} \tag{VII.9}$$

Ce modèle est hautement non linéaire. Premièrement, la loi d'écrouissage isotrope introduit une non-linéarité exponentielle. Mais la non-linéarité la plus significative résulte de la fonction partie positive qui intervient dans la loi de Norton (Éq. (VII.2)). Capturer l'effet de seuil résultant est particulièrement difficile pour un modèle de substitution.

VII.2.a.2 Sollicitations et conditions initiales

On considère le cas d'un essai de cyclage uniaxial en traction/compression avec déformation imposée. Le chargement est appliqué en imposant $\varepsilon^{11}(t)$ donné en figure VII.1 pour $t \in [0, T]$ et avec $\sigma^{12}(t) = \sigma^{13}(t) = \sigma^{23}(t) = \sigma^{22}(t) = \sigma^{33}(t) = 0$. Le chargement $\varepsilon^{11}(t)$ consiste en motif triangulaire de période 400s avec une amplitude pic-à-pic de 2% centrée en 0.

Les conditions initiales pour les variables internes sont :

$$p(t=0) = 0 \quad \text{et} \quad \varepsilon_{\text{vp}}(t=0) = \underset{\sim}{\mathbf{0}}$$

VII.2.a.3 Abstraction tensorielle

On cherche à construire un modèle de substitution représentant la relation (VII.10) entre des coefficients matériaux (entrées du modèle) et des variables mécaniques dépendantes du temps (sorties de modèles) :

$$(n, K, R_0, Q, b, C) \mapsto \left(\underset{\sim}{\boldsymbol{\varepsilon}}(t), \underset{\sim}{\boldsymbol{\varepsilon}}_{vp}(t), \underset{\sim}{\boldsymbol{\sigma}}(t), p(t)\right) \tag{VII.10}$$

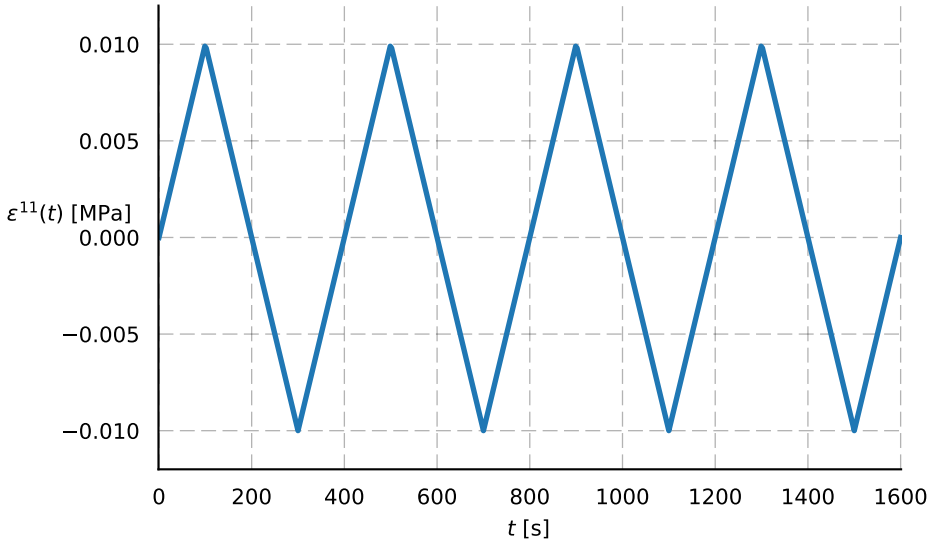


FIGURE VII.1 – Déformation imposée dans le temps de $\varepsilon^{11}(t)$.

Avec les notations de la section VI.1.a, on définit :

$$(\mu_1, \mu_2, \ldots, \mu_6) = (n, K, R_0, Q, b, C)$$

et en suivant la notation de Voigt, les sorties sont regroupées de la manière suivante :

$$\begin{aligned}
\boldsymbol{Y}^1 &= \left(\varepsilon^{11}(t), \varepsilon^{22}(t), \varepsilon^{33}(t), \varepsilon^{12}(t), \varepsilon^{13}(t), \varepsilon^{23}(t)\right)^T \\
\boldsymbol{Y}^2 &= \left(\varepsilon^{11}_{\mathrm{vp}}(t), \varepsilon^{22}_{\mathrm{vp}}(t), \varepsilon^{33}_{\mathrm{vp}}(t), \varepsilon^{12}_{\mathrm{vp}}(t), \varepsilon^{13}_{\mathrm{vp}}(t), \varepsilon^{23}_{\mathrm{vp}}(t)\right)^T \\
\boldsymbol{Y}^3 &= \left(\sigma^{11}(t), \sigma^{22}(t), \sigma^{33}(t), \sigma^{12}(t), \sigma^{13}(t), \sigma^{23}(t)\right)^T \\
\boldsymbol{Y}^4 &= p(t)
\end{aligned}$$

Pour chaque paramètre, l'intervalle de définition est discrétisé comme dans (Éq. (IV.3)) par une grille régulière de 30 points :

$$n_1 = n_2 = n_3 = n_4 = n_5 = n_6 = 30$$

L'intervalle de temps utilisé pour la résolution numérique est discrétisé comme dans l'équation (VI.4) par une grille régulière de $n_t = 537$ points.

La discrétisation permet de définir 4 tenseurs de référence :

$$\boldsymbol{\mathcal{A}}^\chi \in \mathbb{R}^{n_1 \times \cdots \times n_6 \times n_7^\chi} \quad \text{pour} \quad \chi \in [\![1, 4]\!] \tag{VII.11}$$

avec les indices de temps et de composante associées dans le dernier multi-indice (Éq. (VI.5)).

$$n_7^1 = n_7^2 = n_7^3 = 6n_t \quad \text{et} \quad n_7^4 = n_t$$

VII.2.b Résultats

L'algorithme 12 est appliqué pour construire des approximations en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$ des tenseurs de référence $\boldsymbol{\mathcal{A}}^\chi$ en choisissant à chaque itération la tolérance de troncature $\epsilon_{\text{tol}} = 10^{-3}$. Dans cette application, le calcul des rangs des matrices V_k^χ n'est pas effectué via la méthode de raffinement du rang présentée dans la section VI.3.c mais via la méthode présentée en section V.3.c.

Les tailles de l'échantillonnage de snapshots (définie à la section VI.3.a) relatives à chaque itération sont :

$$\widetilde{n}_1 = \widetilde{n}_2 = \widetilde{n}_3 = \widetilde{n}_4 = \widetilde{n}_5 = 100 \quad \text{et} \quad \widetilde{n}_6 = 30$$

VII.2.b.1 Indicateurs de performance

Avec un code de calcul adapté, la résolution du modèle physique pour un unique jeu de paramètres prend en moyenne $1,5$s. La phase hors ligne de construction des décompositions en train de tenseurs requiert la résolution du système d'équations 514 050 fois pour autant de vecteurs de paramètres différents. Sur une machine dotée de 16 cœurs, le temps de construction est de 15 heures. 98% de l'effort de calcul est dédié à la résolution des équations physiques via le solveur et les 2% restants aux opérations de décompositions tensorielles.

Le modèle de substitution est évalué en 6ms ce qui correspond à un facteur d'accélération de 250 pour la phase en ligne.

Pour tout $\chi \in [\![1, 4]\!]$, on note $(r_1^\chi, \ldots, r_6^\chi)$ les tailles de stockages des représentations en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$. Le tableau VII.2 donne leurs valeurs.

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
r_k^1	7	9	10	24	27	30
r_k^2	13	23	29	123	143	134
r_k^3	11	17	20	67	90	100
r_k^4	9	12	14	24	20	21

TABLE VII.2 – Tailles de stockages des décompositions TT associés au modèle élasto-viscoplastique.

Le stockage des approximations TT nécessite donc de garder en mémoire 2 709 405 valeurs flottantes en double précision. En comparaison, le stockage d'une seule solution du

système d'EDA (constituée de toutes les sorties d'intérêt dépendantes du temps) nécessite 10 203 valeurs. Le stockage du modèle de substitution équivaut donc au stockage d'environ 265 solutions alors qu'il permet de prédire des solutions associées à 30^6 points du domaine paramétrique.

VII.2.b.2 Erreurs d'approximation

Les exposants MP et TT font référence aux sorties respectives du modèle physique et du modèle de substitution. On définit les erreurs relatives associées aux différentes variables mécaniques :

- Tenseur des déformations totales : $e_{\varepsilon} = \dfrac{\left\| \underset{\sim}{\boldsymbol{\varepsilon}}^{\mathrm{MP}} - \underset{\sim}{\boldsymbol{\varepsilon}}^{\mathrm{TT}} \right\|_{[0,T]}}{\left\| \underset{\sim}{\boldsymbol{\varepsilon}}^{\mathrm{MP}} \right\|_{[0,T]}}$;
- Tenseur des déformations viscoplastiques : $e_{\varepsilon_{\mathrm{vp}}} = \dfrac{\left\| \underset{\sim}{\boldsymbol{\varepsilon}}_{\mathrm{vp}}^{\mathrm{MP}} - \underset{\sim}{\boldsymbol{\varepsilon}}_{\mathrm{vp}}^{\mathrm{TT}} \right\|_{[0,T]}}{\left\| \underset{\sim}{\boldsymbol{\varepsilon}}^{\mathrm{MP}} \right\|_{[0,T]}}$;
- Tenseur des contraintes : $e_{\sigma} = \dfrac{\left\| \underset{\sim}{\boldsymbol{\sigma}}^{\mathrm{MP}} - \underset{\sim}{\boldsymbol{\sigma}}^{\mathrm{TT}} \right\|_{[0,T]}}{\left\| \underset{\sim}{\boldsymbol{\sigma}}^{\mathrm{MP}} \right\|_{[0,T]}}$;
- Déformation viscoplastique cumulée : $e_{\mathrm{p}} = \dfrac{\left\| p^{\mathrm{MP}} - p^{\mathrm{TT}} \right\|_{[0,T]}}{\left\| \underset{\sim}{\boldsymbol{\varepsilon}}^{\mathrm{MP}} \right\|_{[0,T]}}$.

Selon la valeur des paramètres, le comportement viscoplastique peut être ou non négligeable. Dans le cas où le comportement est uniquement élastique, $\|p\|$ et $\left\| \underset{\sim}{\boldsymbol{\varepsilon}}_{vp} \right\|$ sont théoriquement nuls. Par conséquent, mesurer l'erreur d'approximation relativement à $\left\| p^{\mathrm{MP}} \right\|$ ou $\left\| \underset{\sim}{\boldsymbol{\varepsilon}}_{vp}^{\mathrm{MP}} \right\|$ n'a pas de sens. Lors d'une étude mécanique, c'est le rapport entre le comportement élastique et viscoplastique qui a du sens. Par conséquent, dans l'application proposée, on mesure l'erreur sur $\underset{\sim}{\boldsymbol{\varepsilon}}$, $\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$ et p relativement à la norme de $\underset{\sim}{\boldsymbol{\varepsilon}}$.

Les histogrammes tracés sur les figures VII.2, VII.3, VII.4 et VII.5 présentent pour chaque variable mécanique la distribution empirique des erreurs relatives pour un ensemble des sorties associées à 20 000 vecteurs de paramètres tirés aléatoirement.

On observe que les modèles de substitution basés sur les trains de tenseurs présentent un niveau d'erreur suffisamment faible pour réaliser des études paramétriques telles que la calibration. En effet, dans ce cadre on tolère typiquement des erreurs de l'ordre de 2%.

VII.2.b.3 Convergence de l'erreur vis-à-vis de la tolérance de troncature

Cette partie illustre la convergence de l'erreur d'approximation en fonction des valeurs de tolérance de troncature utilisées au cours de l'algorithme 12. Cette convergence est suggérée par la proposition 12.

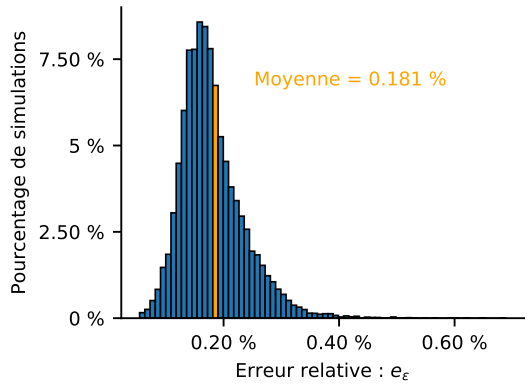


FIGURE VII.2 – Distribution empirique pour e_ε. La largeur des barres est de 0.009%.

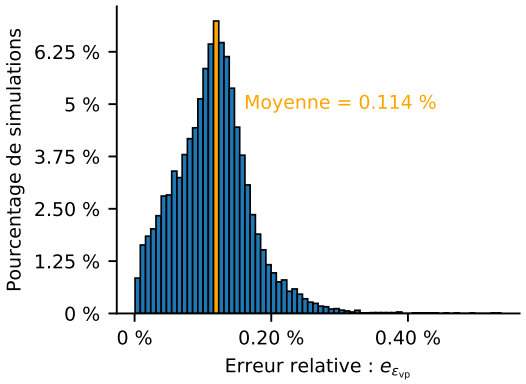


FIGURE VII.3 – Distribution empirique pour $e_{\varepsilon_{\mathrm{vp}}}$. La largeur des barres est de 0.008%.

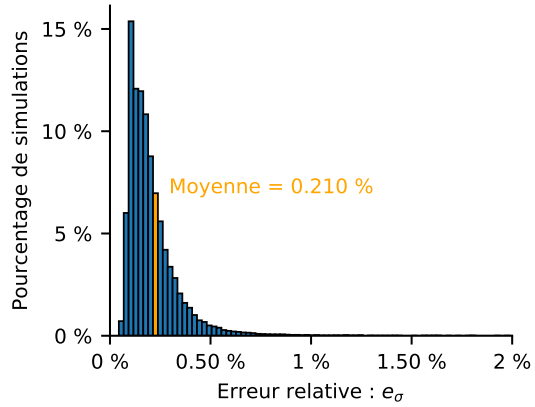


FIGURE VII.4 – Distribution empirique pour e_σ. La largeur des barres est de 0.024%.

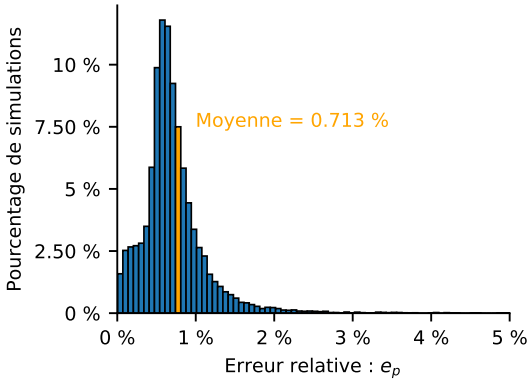


FIGURE VII.5 – Distribution empirique pour e_{p}. La largeur des barres est de 0.066%.

On commence par construire un modèle de substitution initial pour une valeur de tolérance de troncature $\epsilon_{\mathrm{tol}} = 10^{-3}$. L'algorithme 12 est ensuite appliqué sur le modèle de substitution initial en choisissant les valeurs de tolérance de troncature croissantes suivantes :

$$\epsilon_{\mathrm{tol}} \in \left\{1.10^{-3};\ 2.10^{-3};\ 4,6.10^{-3};1.10^{-2};\ 2.10^{-2};\ 4,6.10^{-2};\ 1.10^{-1}\right\}$$

Cela permet d'obtenir 7 nouveaux modèles de substitution. Les calculs hors lignes sont particulièrement rapides (de 1 min à 10 min) puisqu'ils sont basés sur un modèle de substitution. D'autre part, étant donné la manière particulière dont sont définis les points à évaluer lors l'application de l'algorithme 12, il est possible d'exploiter le format TT du modèle de substitution initial pour accélérer davantage les calculs (Section VIII.3.b.1).

Les figures VII.6, VII.7, VII.8 et VII.9 présentent l'évolution des distributions des d'erreurs relatives (pour les différentes variables mécaniques) par rapport à la tolérance de

troncature. L'estimation des distributions d'erreur est réalisée à nouveau par un échantillonnage aléatoire de 20 000 valeurs de paramètres.

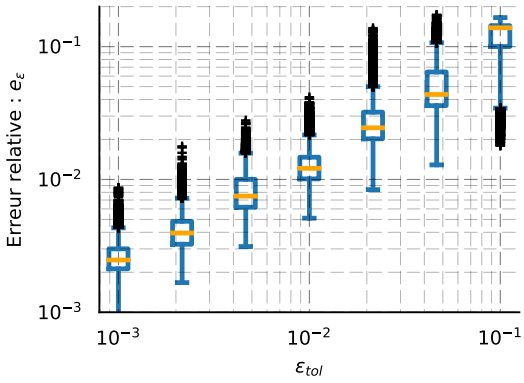


FIGURE VII.6 – Distribution empirique de e_ε en fonction de ϵ_{tol}.

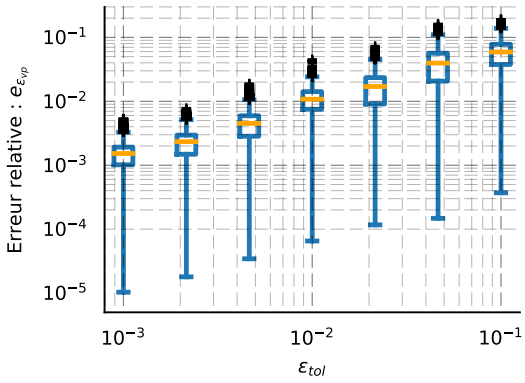


FIGURE VII.7 – Distribution empirique de $e_{\varepsilon_{\text{vp}}}$ en fonction de ϵ_{tol}.

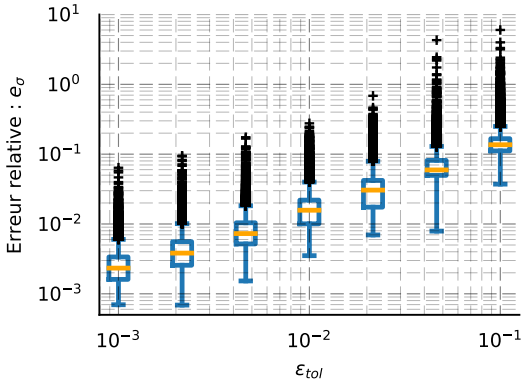


FIGURE VII.8 – Distribution empirique de e_σ en fonction de ϵ_{tol}.

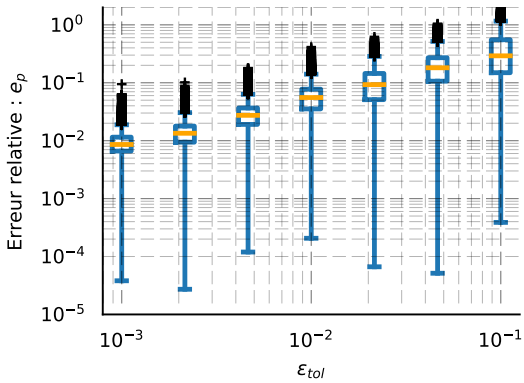


FIGURE VII.9 – Distribution empirique de e_{p} en fonction de ϵ_{tol}.

La figure VII.10 détaille les notations graphiques. Les côtés gauche et droit correspondent au premier et troisième quartiles (respectivement Q_1 et Q_3). La ligne à l'intérieur de la boite représente la médiane. La longueur des moustaches correspond à 1.5 fois l'intervalle interquartile (IQR). Les croix représentent les points dont les valeurs sont au-delà des moustaches.

Les résultats indiquent de manière empirique que pour toutes les variables mécaniques, l'erreur relative décroît en fonction de ϵ_{tol}, ce qui est cohérent avec le comportement attendu de l'algorithme.

Les figures VII.11 et VII.12 montrent la dépendance du nombre d'éléments stockés et du nombre d'appels au modèle de référence par rapport à ϵ_{tol}.

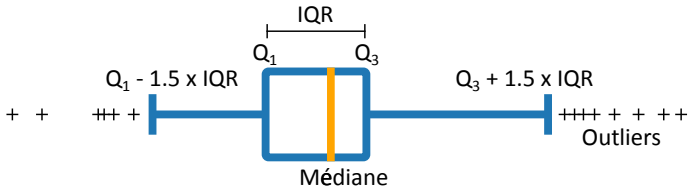


FIGURE VII.10 – Notations des utilisées pour les diagrammes en boîte.

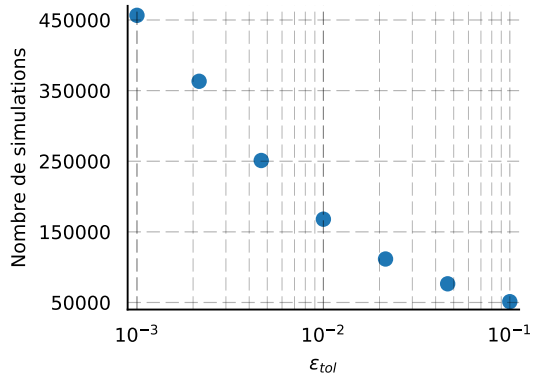


FIGURE VII.11 – Dépendance du nombre d'appels aux modèles de référence par rapport à ϵ_{tol}.

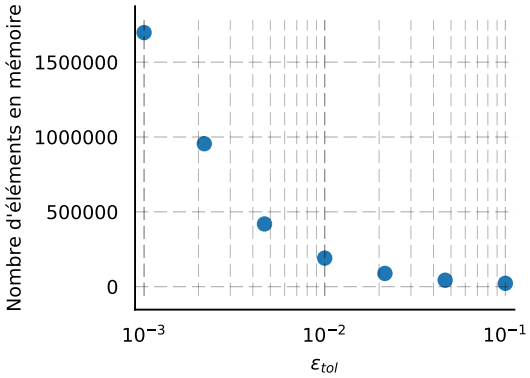


FIGURE VII.12 – Dépendance du nombre d'éléments stockés pour les modèles de substitution par rapport à ϵ_{tol}.

VII.2.b.4 Estimateur de cohérence en ligne

Les sorties générées par le modèle de substitution approchent celles du modèle physique, cependant elles peuvent être inconsistantes vis-à-vis des équations physiques. Cela signifie qu'elles ne vérifient pas nécessairement les équations qui constituent le modèle physique.

On propose un *estimateur de cohérence* permettant de mesurer à quel point les équations physiques sont vérifiées par les sorties du modèle de substitution. L'estimateur correspond au résidu engendré lorsqu'on injecte les sorties dans les équations physiques. Moins le modèle de substitution est précis et plus la probabilité que les équations soient vérifiées est faible. Il est donc raisonnable de s'attendre à voir une corrélation entre la précision du modèle de substitution et l'estimateur de cohérence.

En utilisant l'équation (VII.1), on définit :

$$\underset{\sim}{\boldsymbol{\sigma}}^{eq,TT} = \frac{E}{1+\nu}\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e^{\mathrm{TT}} + \frac{\nu}{1-2\nu} Tr\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e^{\mathrm{TT}}\right)\underset{\sim}{\mathbb{I}}\right)$$

ainsi que l'estimateur de cohérence :

$$\eta_\sigma = \frac{\left\|\underset{\sim}{\boldsymbol{\sigma}}^{\mathrm{TT}} - \underset{\sim}{\boldsymbol{\sigma}}^{eq,TT}\right\|_{[0,T]}}{\left\|\underset{\sim}{\boldsymbol{\sigma}}^{\mathrm{TT}}\right\|_{[0,T]}} \tag{VII.12}$$

On définit l'effectivité de l'estimateur de cohérence par

$$\eta_\sigma / e_\sigma$$

Pour chacune des 20 000 simulations tirées aléatoirement on trace dans la figure VII.13 un point ayant pour abscisse la valeur de l'erreur relative associée à $\underset{\sim}{\boldsymbol{\sigma}}$ et pour ordonnée la valeur de l'effectivité de l'estimateur de cohérence. La couleur des points correspond à la valeur finale de la déformation viscoplastique cumulée et caractérise donc la non-linéarité de la réponse du modèle.

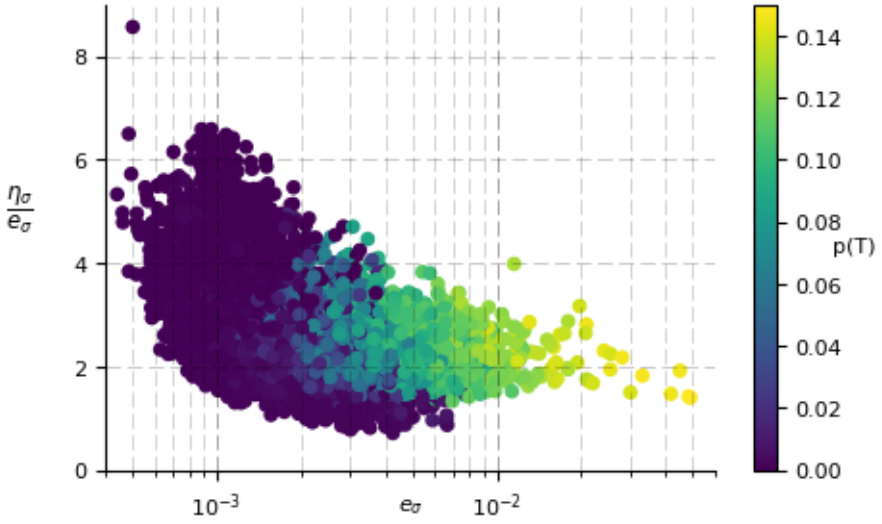


FIGURE VII.13 – Effectivité de l'estimateur de cohérence η_σ en fonction de l'erreur relative e_σ.

Le nuage de points montre que l'erreur augmente avec la valeur de la déformation viscoplastique cumulée, c'est-à-dire lorsque le matériau présente un comportement viscoplastique plus intense. D'autre part, on observe une corrélation entre l'estimateur de cohérence et l'erreur relative. L'effectivité tend à être plus grande que 1 ce qui indique que l'estimateur de cohérence se comporte comme une borne supérieure de l'erreur relative. Hormis pour quelques rares exceptions, l'estimateur de cohérence surestime l'erreur relative d'un facteur 7 au maximum.

Finalement, l'effectivité de l'estimateur de cohérence converge empiriquement vers 1 (ce qui signifie que l'estimateur est plus fin) lorsque l'amplitude de l'erreur relative augmente. Cet estimateur de cohérence est très rapide à calculer et repose uniquement sur les sorties du modèle de substitution. Les résultats suggèrent que cet estimateur peut être utilisé comme un indicateur d'erreur en ligne au point courant lors de l'exploration du domaine paramétrique en temps réel.

Plus généralement, il est possible de définir un estimateur de cohérence pour chaque

équation impliquée dans le modèle physique pour laquelle le modèle de substitution approche toutes les variables mécaniques qui y interviennent.

VII.2.b.5 Exemples de courbes

Les figures VII.14, VII.15 et VII.16 présentent pour différentes variables mécaniques, la correspondance des sorties entre le modèle physique et le modèle de substitution. On remarque d'autre part la variété des comportements approchés par le modèle de substitution.

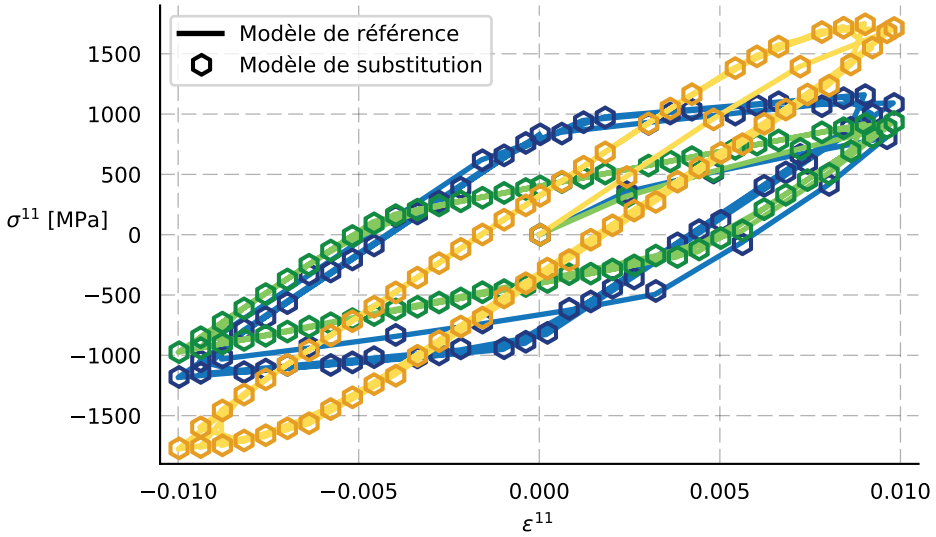


FIGURE VII.14 – Courbes $\epsilon^{11}(\sigma^{11})$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents.

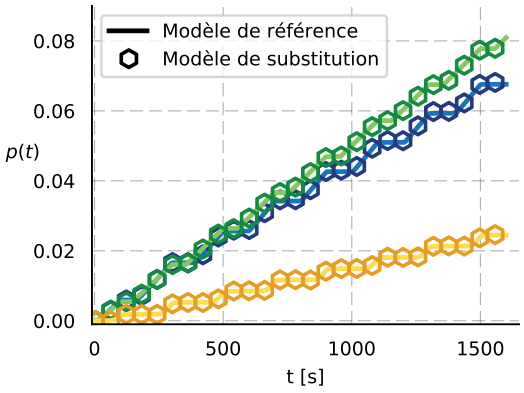


FIGURE VII.15 – $p(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents.

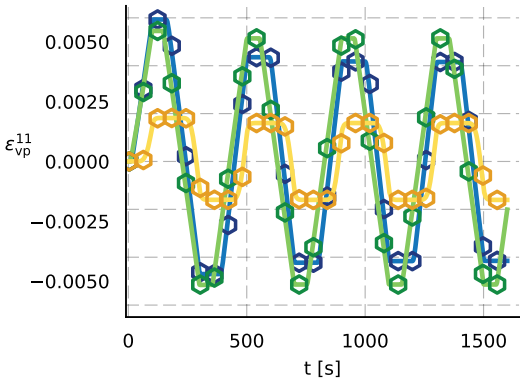


FIGURE VII.16 – $\epsilon^{11}_{vp}(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents.

VII.3 Premier cas industriel : Polystar

Cette section s'intéresse à un modèle de lois matériaux de plus grande complexité utilisé industriellement. Le modèle Polystar développé par [30, 43, 42] est destiné à décrire en particulier le comportement et l'endommagement anisotherme des monocristaux de superalliage base nickel. Il a été utilisé dans [45] pour modéliser le comportement du matériau CMSX-4.

VII.3.a Modèle physique

Le modèle est formulé comme un système d'équations différentielles non linéaires couplées présentant de fortes non-linéarités et un effet de seuil. On se place dans le cas où la température est constante.

Une trentaine de coefficients matériaux sont impliqués dans le modèle. Dans cette étude, on considère comme paramètres d'entrée les neuf coefficients matériaux suivants : n, K, d, c, β^*, Q_{s_0}, r^0, b et Q.

Le tableau VII.3 présente les intervalles de variation considérés pour les coefficients matériaux. Pour des raisons de confidentialité, les valeurs des bornes ne sont pas données.

	n	K	d	c	β^*	Q_{s_0}	r^0	b	Q
Unité		MPa.s^{-n}		MPa	s	MPa	MPa		MPa
Borne inférieure	$n_{\min}$	$K_{\min}$	$d_{\min}$	$c_{\min}$	$\beta^*_{\min}$	$Q_{s_0,\min}$	$r^0_{\min}$	$b_{\min}$	$Q_{\min}$
Borne supérieure	$n_{\max}$	$K_{\max}$	$d_{\max}$	$c_{\max}$	$\beta^*_{\max}$	$Q_{s_0,\max}$	$r^0_{\max}$	$b_{\max}$	$Q_{\max}$

TABLE VII.3 – Intervalles de variation des paramètres associés au modèle Polystar.

VII.3.a.1 Équations physiques

Le modèle est construit à partir du couplage de plusieurs « briques » de modèle élémentaires. On donne ici les équations pour rendre compte de la complexité du modèle, mais on renvoie le lecteur à [45] pour une description plus précise du modèle. Les paramètres d'entrée du modèle apparaissent en vert dans les équations.

Viscoplasticité cristalline

$$\begin{aligned}
\dot{\gamma}^s &= (1 + d^s_{dislo}) \left(\frac{\langle f^s \rangle_+}{K(1 - d^s_c)} \right)^n \text{signe}(\tau^s - x^s) \\
f^s(\tau^s, x^s, r^s) &= |\tau^s - x^s| - r^s \\
\tau^s &= \underset{\sim}{\boldsymbol{\sigma}} : \underset{\sim}{\boldsymbol{m}}^s \\
\underset{\sim}{\boldsymbol{m}}^s &= \frac{1}{2} (\underline{n}^s \otimes \underline{l}^s + \underline{l}^s \otimes \underline{n}^s) \\
\dot{d}^s_{dislo} &= C_{dislo} \dot{\gamma}^s \\
\underset{\sim}{\dot{\boldsymbol{\varepsilon}}}^p &= \sum_s \dot{\gamma}^s \underset{\sim}{\boldsymbol{m}}^s \text{signe}(\tau^s) \\
\dot{\nu} &= \sqrt{\frac{2}{3} \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}^p : \underset{\sim}{\dot{\boldsymbol{\varepsilon}}}^p} \\
\dot{\nu}^s &= \dot{\gamma}^s \text{signe}(\tau^s - x^s)
\end{aligned}$$

Microstructure

$$\begin{aligned}
\dot{f}_l &= \left[1 - \delta_l \exp\left(-\frac{\nu}{e_{cfl}} \right) \right] \frac{f_{\text{eq}} - f_l}{\alpha_l} \\
\dot{f}_s &= \begin{cases} -\dfrac{f_s}{\alpha_s} & \text{si} \quad f_{\text{eq}} - f_l \leq 0 \\ 0 & \text{sinon} \end{cases} \\
w_{001} &= \frac{\alpha_0}{\delta} \left(f_l^{m_l} - d_{tp} f_s^{m_s} \right)
\end{aligned}$$

Écrouissage

$$\begin{aligned}
\dot{\alpha}^s &= \dot{\nu}^s \left[\text{signe}(\tau^s - x^s) - d\alpha^s \right] \\
r^s &= r^0 + (Q + Q^*) \sum_j h_{s_j} \rho^j + \sqrt{\frac{2}{3}} \frac{G_B}{w_{001}} \\
\dot{\alpha}^* &= -\frac{\alpha^*}{\beta^*} \\
\dot{\rho}^s &= (1 - b\rho^s) \dot{\gamma}^s \\
Q^* &= Q_{s_0} a^* \\
x^s &= c\alpha^s
\end{aligned}$$

Dommage Lorsque le critère

$$\nu > \nu_d$$

est vérifié, les équations de dommage s'activent :

$$\begin{aligned}
\dot{\alpha}^d &= \dot{\nu}^d \left[\text{signe}(\tau^s - w^s) - d_{\text{dom}}\alpha^d\right] - M(\alpha^d)^{m_\alpha} \\
w^s &= C_{\text{dom}}\alpha^d \\
\dot{d}_c^s &= \left(\frac{|w^s|}{K_x(1-d_c^s)}\right)^{m_x} \\
K_x &= K_{x_0} + g_{ta}\max|f_{\text{eq}} - f_l|^{g_{tb}}
\end{aligned}$$

VII.3.a.2 Sollicitations et conditions initiales

On considère numériquement un essai de fluage isotherme avec contraintes imposées jusqu'à rupture. La contrainte imposée $\sigma^{33}(t)$ suit une montée en charge linéaire de 0 MPa jusqu'à une valeur maximale fixée σ^{33}_{MAX} puis est maintenue jusqu'à rupture. Les autres contraintes sont imposées nulles au cours du temps $\sigma^{11}(t) = \sigma^{12}(t) = \sigma^{13}(t) = \sigma^{23}(t) = \sigma^{22}(t) = 0$. Notons qu'expérimentalement les essais sont plutôt réalisés à effort imposé.

VII.3.a.3 Abstraction tensorielle

On cherche à construire un modèle de substitution représentant la relation (VII.13) entre des coefficients matériaux et la déformation viscoplastique cumulée p :

$$(n, K, d, c, \beta^*, Q_{s_0}, r^0, b, Q) \mapsto p(t) \tag{VII.13}$$

L'objectif de cette application est d'étudier l'influence des coefficients sur le temps de rupture du matériau. Comme pour chaque vecteur de paramètres $(n, K, d, c, \beta^*, Q_{s_0}, r^0, b, Q)$, la solution $p(t)$ est définie sur un intervalle $[0, T_{(n,K,d,c,\beta^*,Q_{s_0},r^0,b,Q)}]$ différent, on applique la méthodologie de normalisation du temps présentée dans la section VI.4.a.1. Le temps $T_{(n,K,d,c,\beta^*,Q_{s_0},r^0,b,Q)}$ représente le temps de rupture du matériau. En pratique, le temps de rupture correspond à la valeur du dernier pas de temps qui a pu être calculé par le code numérique. On note $\bar{p}(\bar{t})$ la nouvelle variable correspondant à la déformation viscoplastique cumulée définie sur l'intervalle de temps normalisé $[0, 1]$. Finalement, le modèle de référence que l'on cherche à approcher correspond à la relation :

$$(n, K, d, c, \beta^*, Q_{s_0}, r^0, b, Q) \mapsto (T, \bar{p}(\bar{t})) \tag{VII.14}$$

Avec les notations de la section VI.1.a, on définit :

$$(\mu_1, \ldots, \mu_9) = (n, K, d, c, \beta^*, Q_{s_0}, r^0, b, Q)$$

et

$$\boldsymbol{Y}^1 = T$$
$$\boldsymbol{Y}^2 = \bar{p}(\bar{t})$$

L'intervalle de temps normalisé est discrétisé uniformément par une grille de $n_t = 200$ points.

Pour chaque paramètre, l'intervalle de définition est discrétisé comme dans l'équation (IV.3) par une grille régulière de 20 points :

$$n_1 = n_2 = n_3 = n_4 = n_5 = n_6 = n_7 = n_8 = n_9 = 20$$

Les discrétisations permettent de définir 2 tenseurs de référence :

$$\boldsymbol{\mathcal{A}}^\chi \in \mathbb{R}^{n_1 \times \cdots \times n_9 \times n_{10}^\chi} \quad \text{pour} \quad \chi \in [\![1, 2]\!] \tag{VII.15}$$

avec

$$n_{10}^1 = 1 \quad \text{et} \quad n_{10}^2 = n_t$$

VII.3.b Résultats

L'algorithme 12 est appliqué pour construire des approximations en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$ des tenseurs de référence $\boldsymbol{\mathcal{A}}^\chi$ en choisissant à chaque itération la tolérance de troncature $\epsilon_{\text{tol}} = 10^{-3}$.

Les tailles de l'échantillonnage de snapshots (définie à la section VI.3.a) relatives à chaque itération sont :

$$\widetilde{n}_1 = \widetilde{n}_2 = \widetilde{n}_3 = \widetilde{n}_4 = \widetilde{n}_5 = \widetilde{n}_6 = 20$$

On compare les résultats en utilisant la méthode de troncature du rang classique présentée en section V.3.c (modèle de substitution simple) et la méthode de raffinement du rang présentée en section VI.3.c (modèle de substitution raffiné).

VII.3.b.1 Indicateurs de performance

Avec un code de calcul adapté, la résolution du modèle physique pour un unique vecteur de paramètres prend en moyenne $2,7$s.

Les constructions des deux modèles de substitution requièrent la résolution du système d'équations environ 220 000 fois pour autant de vecteurs de paramètres différents. La

différence du nombre de simulations nécessaires est due à la sélection aléatoire de colonnes qui peut engendrer numériquement une différence sur les rangs d'approximation. Le fait que le nombre de simulations soit proche suggère que le nombre de colonnes sélectionnées est pertinent. Sur une machine dotée de 20 coeurs, le temps de construction correspond à environ $8,5$ heures.

Le modèle de substitution est évalué en 1ms ce qui correspond à un facteur d'accélération de 2 700 pour la phase en ligne.

Pour tout $\chi \in [\![1, 2]\!]$, on note $(r_1^\chi, \ldots, r_9^\chi)$ les tailles de stockages des représentations en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$. Les tableaux VII.4 et VII.5 donnent les valeurs des tailles de stockage pour les trains de tenseurs correspondant respectivement au modèle simple et au modèle raffiné.

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
r_k^1	11	17	18	17	17	17	17	16	1
r_k^2	19	50	64	72	91	88	84	84	11

TABLE VII.4 – Tailles de stockages des décompositions TT pour le modèle simple.

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
r_k^1	18	19	12	14	12	11	13	19	1
r_k^2	19	19	13	14	13	12	12	18	10

TABLE VII.5 – Tailles de stockages des décompositions TT pour le modèle raffiné.

Le stockage des approximations TT du modèle simple nécessite de garder en mémoire 815 681 valeurs flottantes en double précision. Le stockage des approximations TT du modèle raffiné nécessite de garder en mémoire 65 041 valeurs flottantes en double précision ce qui correspond à un gain en taille mémoire d'un facteur 12. En comparaison, le stockage d'une unique solution associée à un vecteur de paramètres nécessite 201 valeurs.

À titre indicatif, le tableau VII.6 donne les valeurs des tailles de stockage pour les trains de tenseurs correspondant au modèle raffiné avant refactorisation.

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
$r_k^1 = r_k^2$	19	60	76	86	105	103	99	94	13

TABLE VII.6 – Tailles de stockages des décompositions TT pour le modèle raffiné avant refactorisation.

On note que les deux représentations en train de tenseurs $\boldsymbol{\mathcal{A}}^1$ et $\boldsymbol{\mathcal{A}}^2$ ont les mêmes tailles de stockages avant refactorisation. Le stockage des approximations TT du modèle raffiné avant refactorisation nécessite de garder en mémoire 2 113 026 valeurs flottantes

en double précision. La refactorisation permet donc de gagner un facteur 32 sur la taille mémoire. D'autre part, elle permet également de réduire le nombre d'opérations impliquées lors de l'évaluation du modèle de substitution et donc les temps de calcul en ligne.

VII.3.b.2 Erreurs d'approximation

Les exposants MP,1 et 2 font référence aux sorties respectives du modèle physique, du modèle de substitution simple et du modèle de substitution raffiné.

Pour $i \in [\![1, 2]\!]$, on définit l'erreur relative associée au temps final par :

$$e_T^i = \frac{|T^{\mathrm{MP}} - T^i|}{|T^{\mathrm{MP}}|}$$

Il n'existe pas de mesure d'erreur naturelle associée à la déformation viscoplastique cumulée, car les courbes p^{MP} et $p^i(t)$ ne sont pas nécessairement définies sur le même intervalle de temps. Pour $i \in [\![1, 2]\!]$, on propose de mesurer l'erreur entre la courbe $p^{\mathrm{MP}}(t)$ et une courbe renormalisée $\widehat{p}^i(t)$ qui correspond à $p^i(t)$ défini sur l'intervalle $[0, T^{\mathrm{MP}}]$. Cela permet de dissocier les erreurs d'approximation faites sur le temps final et la forme de la courbe de déformation. On définit l'erreur relative associée à la déformation viscoplastique cumulée par :

$$e_{\mathrm{p}}^i = \frac{\left\| p^{\mathrm{MP}} - \widehat{p}^i \right\|_{[0,T^{\mathrm{MP}}]}}{\left\| \underset{\sim}{\varepsilon}^{\mathrm{MP}} \right\|_{[0,T^{\mathrm{MP}}]}}$$

avec $\underset{\sim}{\varepsilon}^{\mathrm{MP}}$ la déformation totale obtenue par le modèle physique.

Les histogrammes tracés sur les figures VII.17 et VII.18 présentent les distributions empiriques des erreurs relatives pour un ensemble de sorties associées à 20 000 paramétrages tirés aléatoirement pour le modèle simple et le modèle raffiné.

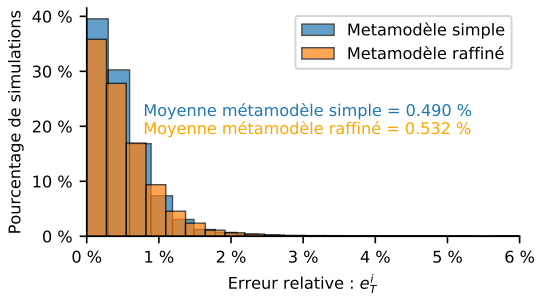


FIGURE VII.17 – Distribution empirique pour e_T^i. La largeur des barres de l'histogramme est de $0,0025\%$ pour le modèle simple et de $0,0027\%$ pour le modèle raffiné.

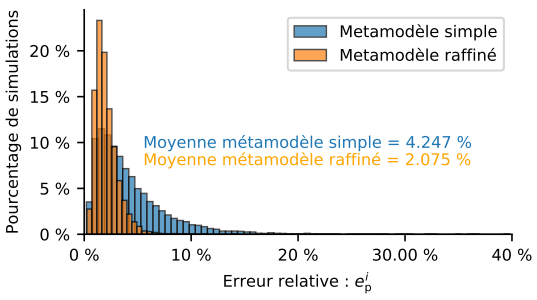


FIGURE VII.18 – Distribution empirique pour e_{p}^i. La largeur des barres de l'histogramme est de $0,008\%$ pour le modèle simple et de $0,002\%$ pour le modèle raffiné.

Les deux modèles présentent des précisions comparables pour la prédiction du temps final. Pour ce qui est de la prédiction de la déformation viscoplastique cumulée, le modèle raffiné est nettement plus précis.

On mesure pour le modèle de substitution raffiné, l'erreur relative d'approximation entre $\boldsymbol{\mathcal{T}}^\chi$ et $\boldsymbol{\mathcal{A}}^\chi$ à partir de l'échantillonnage de 20 000 points :

$$\frac{\left\|\boldsymbol{\mathcal{A}}^1 - \boldsymbol{\mathcal{T}}^1\right\|}{\left\|\boldsymbol{\mathcal{A}}^1\right\|} \simeq 0,012 \qquad \frac{\left\|\boldsymbol{\mathcal{A}}^2 - \boldsymbol{\mathcal{T}}^2\right\|}{\left\|\boldsymbol{\mathcal{A}}^2\right\|} \simeq 0,087$$

En faisant l'hypothèse que les échantillonnages de colonnes sont suffisamment exhaustifs, on peut calculer les bornes d'erreur théoriques données à la proposition 12 :

$$\frac{\left\|\boldsymbol{\mathcal{A}}^1 - \boldsymbol{\mathcal{T}}^1\right\|}{\left\|\boldsymbol{\mathcal{A}}^1\right\|} \leq 0,024 \qquad \frac{\left\|\boldsymbol{\mathcal{A}}^2 - \boldsymbol{\mathcal{T}}^2\right\|}{\left\|\boldsymbol{\mathcal{A}}^2\right\|} \leq 0,013$$

On remarque que les bornes calculées numériquement sont proches des estimations des erreurs relatives d'approximation. Toutefois, la relation est fausse pour le second tenseur. Cela est dû au fait que l'hypothèse d'échantillonnage complet de colonnes n'est pas vérifiée.

VII.3.c Identification

La figure VII.19 présente les courbes de déformation viscoplastique cumulée obtenue par les acquisitions expérimentales ainsi qu'avec le modèle physique et le modèle de substitution raffiné.

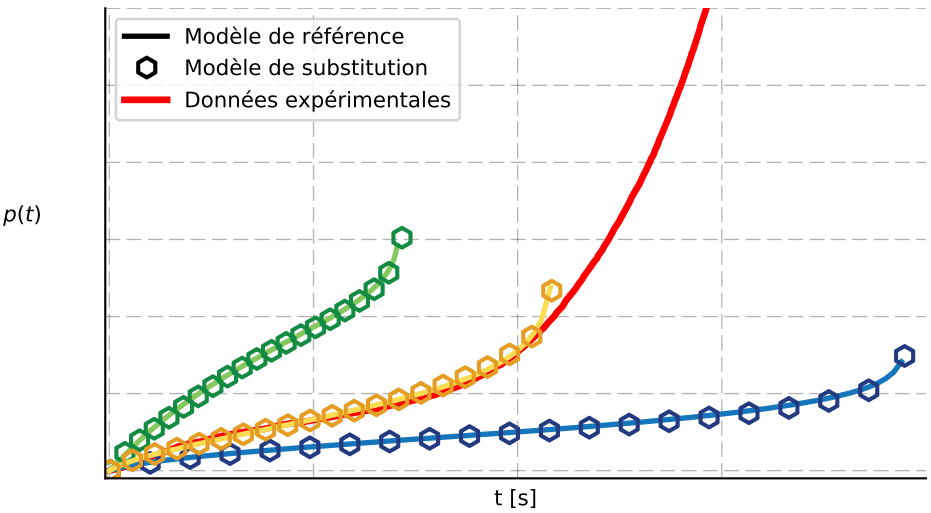


FIGURE VII.19 – Courbes $p(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents ainsi que les données expérimentales.

Avec un outil de visualisation interactif (Section VIII.4), il est possible d'exploiter le modèle de substitution de manière très efficace et intuitive. Cela permet de trouver particulièrement rapidement (5 min) une courbe qui se rapproche de la courbe expérimentale.

Après exploration du domaine paramétrique, il semble que le modèle physique tel qu'il est défini n'est pas capable de reproduire le comportement physique observé expérimentalement et en particulier le fluage tertiaire (la dernière partie de la courbe).

Trois raisons principales peuvent expliquer ce constat :

— Les intervalles paramétriques ne sont pas définis de manière suffisamment étendue ;
— Les autres paramètres qui interviennent dans le modèle doivent être modifiés ;
— Le modèle Polystar n'est pas adapté pour reproduire ce type de comportement.

Ceci donne un objectif d'amélioration du modèle physique aux experts en mécanique des matériaux. Il suffit de quelques minutes d'exploration du modèle de substitution pour en discuter avec eux et les en convaincre. La méthode proposée ouvre donc la voie à de nouvelles méthodes de travail collaboratif.

Il serait intéressant d'étudier les deux premières hypothèses, cependant :

— L'objectif de cette thèse n'est pas de réaliser effectivement le travail d'identification des paramètres, mais de mettre à disposition des outils pour simplifier cette tâche ;
— En parallèle de cette thèse, des travaux [34] se sont basés sur le modèle Polystar pour en proposer une variante thermodynamiquement valide et plus simple permettant des calculs éléments finis avec une durée de résolution acceptable en industrie : le modèle RaftX.

VII.4 Second cas industriel : RaftX

Le modèle RaftX [34] est une évolution du modèle Polystar permettant de reproduire les effets du vieillissement microstructurel grâce à une modélisation du couplage entre la mise en radeaux et le comportement viscoplastique. Cette modélisation est essentielle pour reproduire les comportements observés expérimentalement sur le CMSX-4. On utilise dans le cas présent une version qui tient compte des effets d'endommagement. Le modèle particulièrement complexe ne sera pas présenté en détail.

VII.4.a Définition du modèle physique de référence

Dans cette étude, on considère comme entrées du modèle sept coefficients matériaux jouant un rôle dans les équations de comportement et d'endommagement : N, N_K, σ_{vinf}, GK_{oro}, K_{raft}, Eps_{pth} et u_{raft}.

Le tableau VII.7 présente les intervalles de variation respectifs pour les coefficients matériaux. Pour des raisons de confidentialité, les valeurs des bornes ne sont pas données.

	N	N_K	σ_{vinf}	GK_{oro}	K_{raft}	Eps_{pth}	u_{raft}
Unité			MPa	MPa	s^{-1}		MPa^{-1}
Borne inférieure	$N_{\min}$	$N_{K,\min}$	$\sigma_{vinf,\min}$	$GK_{oro,\min}$	$K_{raft,\min}$	$Eps_{pth,\min}$	$u_{raft,\min}$
Borne supérieure	$N_{\max}$	$N_{K,\max}$	$\sigma_{vinf,\max}$	$GK_{oro,\max}$	$K_{raft,\max}$	$Eps_{pth,\max}$	$u_{raft,\max}$

TABLE VII.7 – Intervalles de variation des paramètres associés au modèle RaftX.

Comme dans le cas du modèle Polystar, on considère un essai de fluage isotherme avec contraintes imposées jusqu'à rupture. La contrainte imposée $\sigma^{11}(t)$ suit une montée en charge linéaire de 0 MPa jusqu'à une valeur maximale fixée σ^{33}_{MAX} puis est maintenue jusqu'à rupture. Les autres contraintes sont imposées nulles au cours du temps $\sigma^{12}(t) = \sigma^{13}(t) = \sigma^{23}(t) = \sigma^{22}(t) = \sigma^{33}(t) = 0$. Notons qu'expérimentalement les essais sont plutôt réalisés à effort imposé.

On cherche à construire un modèle de substitution pour la déformation viscoplastique cumulée p. Comme p est définie sur un intervalle de temps dépendant de la valeur des paramètres, on effectue, comme pour le modèle Polystar, une normalisation de l'échelle de temps.

Le modèle de référence que l'on cherche à approcher correspond donc à la relation :

$$(N, N_K, \sigma_{vinf}, GK_{oro}, K_{raft}, Eps_{pth}, u_{raft}) \mapsto (T, \bar{p}(\bar{t})) \tag{VII.16}$$

Avec les notations de la section VI.1.a, on définit :

$$(\mu_1, \dots, \mu_7) = (N, N_K, \sigma_{vinf}, GK_{oro}, K_{raft}, Eps_{pth}, u_{raft})$$

et

$$\boldsymbol{Y}^1 = T$$
$$\boldsymbol{Y}^2 = \bar{p}(\bar{t})$$

L'intervalle de temps est discrétisé uniformément par une grille de $n_t = 200$ points.

Pour le paramètre K_{raft}, on choisit une discrétisation logarithmique de 20 points. Pour les autres paramètres, on choisit une grille régulière de 20 points.

$$n_1 = n_2 = n_3 = n_4 = n_5 = n_6 = n_7 = 20$$

Les discrétisations permettent de définir 2 tenseurs de référence :

$$\boldsymbol{\mathcal{A}}^\chi \in \mathbb{R}^{n_1 \times \cdots \times n_7 \times n_8^\chi} \quad \text{pour} \quad \chi \in [\![1, 2]\!] \tag{VII.17}$$

avec

$$n_8^1 = 1 \quad \text{et} \quad n_8^2 = n_t$$

VII.4.b Résultats

L'algorithme 12 est appliqué pour construire des approximations en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$ des tenseurs de référence $\boldsymbol{\mathcal{A}}^\chi$ en choisissant à chaque itération la tolérance de troncature $\epsilon_{\text{tol}} = 10^{-3}$.

Les tailles de l'échantillonnage de snapshots (définie à la section VI.3.a) relatives à chaque itération sont :

$$\widetilde{n}_1 = \widetilde{n}_2 = \widetilde{n}_3 = \widetilde{n}_4 = \widetilde{n}_5 = \widetilde{n}_6 = 20$$

VII.4.b.1 Indicateurs de performance

Avec un code de calcul adapté, la résolution du modèle physique pour un unique vecteur de paramètres est beaucoup plus couteuse et variable que les modèles précédents. Sur un échantillon de 20 000, on estime un temps moyen de $9,1$s.

La phase hors ligne de construction des décompositions en train de tenseurs requiert la résolution du modèle 251 220 fois pour autant de vecteurs de paramètres différents. Sur une machine dotée de 20 coeurs, le temps de construction est de $34,4$ heures.

Le modèle de substitution est évalué en 1ms ce qui correspond à un facteur d'accélération d'environ 9000 pour la phase en ligne.

Pour tout $\chi \in [\![1, 2]\!]$, on note $(r_1^\chi, \ldots, r_7^\chi)$ les tailles de stockages des représentations en train de tenseurs $\boldsymbol{\mathcal{T}}^\chi$. Le tableau VII.8 donne les valeurs des tailles de stockage.

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
r_k^1	20	22	13	17	16	9	1
r_k^2	20	21	11	13	13	9	8

TABLE VII.8 – Tailles de stockages des décompositions TT associées au modèle RaftX.

Le stockage des approximations TT nécessite donc de garder en mémoire 52 881 valeurs flottantes en double précision. En comparaison, le stockage d'une unique solution associée à un vecteur de paramètres, nécessite 201 valeurs.

VII.4.b.2 Erreurs d'approximation

Les exposants MP et TT font référence aux sorties respectives du modèle physique et du modèle de substitution.

On définit l'erreur relative associée au temps final par :

$$e_T = \frac{|T^{\mathrm{MP}} - T^{\mathrm{TT}}|}{|T^{\mathrm{MP}}|}$$

Pour l'erreur relative associée à la déformation viscoplastique cumulée, on utilise la même mesure que pour le modèle Polystar :

$$e_{\mathrm{p}} = \frac{\left\| p^{\mathrm{MP}} - \widehat{p}^{\mathrm{TT}} \right\|_{[0,T^{\mathrm{MP}}]}}{\left\| \underset{\sim}{\varepsilon}^{\mathrm{MP}} \right\|_{[0,T^{\mathrm{MP}}]}}$$

avec $\underset{\sim}{\varepsilon}^{\mathrm{MP}}$ la déformation totale obtenue par le modèle physique.

Les histogrammes tracés sur les figures VII.20 et VII.21 présentent les distributions empiriques des erreurs relatives pour un ensemble de sorties associées à 20 000 paramétrages tirés aléatoirement.

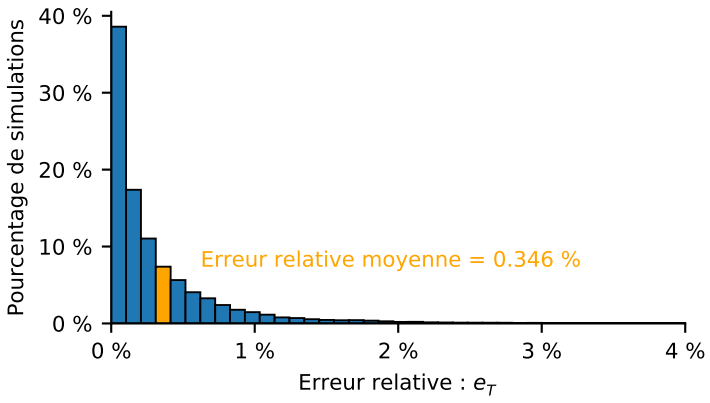


FIGURE VII.20 – Distribution empirique pour e_T^i. La largeur des barres de l'histogramme est de $0,001\%$.

À nouveau, on observe des valeurs d'erreur suffisamment faibles pour effectuer des études de calibration.

On mesure l'erreur relative d'approximation entre $\boldsymbol{\mathcal{T}}^\chi$ et $\boldsymbol{\mathcal{A}}^\chi$ à partir de l'échantillonnage de 20 000 points :

$$\frac{\left\|\boldsymbol{\mathcal{A}}^1 - \boldsymbol{\mathcal{T}}^1\right\|}{\left\|\boldsymbol{\mathcal{A}}^1\right\|} \simeq 0,0066 \quad \frac{\left\|\boldsymbol{\mathcal{A}}^2 - \boldsymbol{\mathcal{T}}^2\right\|}{\left\|\boldsymbol{\mathcal{A}}^2\right\|} \simeq 0,049$$

En faisant l'hypothèse que les échantillonnages de colonnes sont suffisamment exhaustifs,

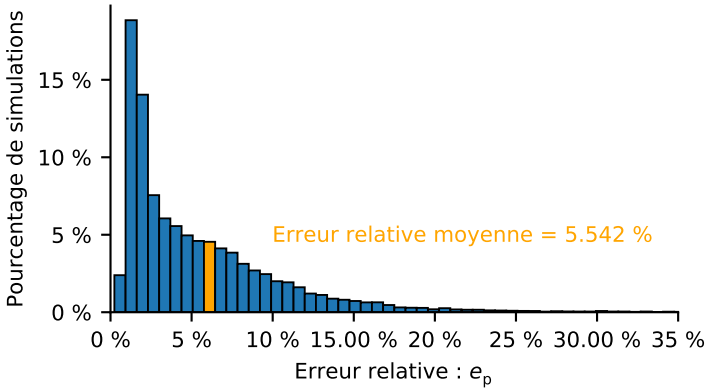


FIGURE VII.21 – Distribution empirique pour e_{p}^{i}. La largeur des barres de l'histogramme est de $0,007\%$.

on peut calculer les bornes d'erreur théoriques données à la proposition 12 :

$$\frac{\left\|\boldsymbol{\mathcal{A}}^1 - \boldsymbol{\mathcal{T}}^1\right\|}{\left\|\boldsymbol{\mathcal{A}}^1\right\|} \leq 0,021 \quad \frac{\left\|\boldsymbol{\mathcal{A}}^2 - \boldsymbol{\mathcal{T}}^2\right\|}{\left\|\boldsymbol{\mathcal{A}}^2\right\|} \leq 0,018$$

On remarque que les bornes calculées numériquement sont proches des estimations des erreurs relatives d'approximation. Toutefois, la relation est fausse pour le second tenseur. Cela est dû au fait que l'hypothèse d'échantillonnage complet de colonnes n'est pas vérifiée.

VII.4.c Identification

La figure VII.22 présente les courbes de déformation viscoplastique cumulée obtenue par les acquisitions expérimentales ainsi qu'avec le modèle physique et le modèle de substitution.

Avec un outil de visualisation interactif (Section VIII.4), il est possible de trouver relativement rapidement (10min) une courbe numérique qui se rapproche de la courbe expérimentale. Ce type d'identification manuelle ne sera absolument pas possible en se basant sur le modèle physique sachant les temps de calcul trop importants pour obtenir des solutions.

VII.5 Couplage avec l'hyper-réduction

L'identification d'une loi matériau sur un modèle 0D, c'est-à-dire uniquement défini sur un seul point de Gauss, n'est pas toujours robuste. En particulier, il est courant de trouver plusieurs vecteurs de paramètres admissibles tels que les courbes d'évolution temporelle numériques associées soient toutes superposées à la courbe expérimentale.

D'autre part, ces modèles 0D ne permettent pas de faire apparaitre les comportements liés au couplage entre les lois matériaux et la structure géométrique. Lorsque ce couplage a

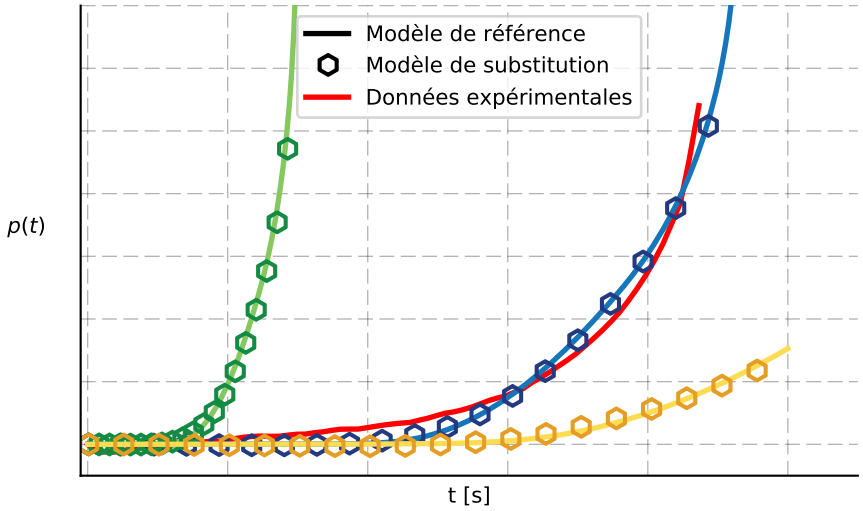


FIGURE VII.22 – Courbes $p(t)$ pour le modèle de référence et le modèle de substitution associées à 3 vecteurs de paramètres différents ainsi que les données expérimentales.

des effets non négligeables, il peut être judicieux de conduire l'identification en comparant des données relatives à des calculs de structure. Comme les temps de calcul impliqués sont beaucoup plus couteux, ce type d'étude est rarement réalisée.

Pour facilité la conduite de ce type d'approche, on propose dans cette section une méthodologie pour construire un modèle de substitution d'un problème de mécanique de structure. La méthodologie repose sur un couplage entre l'algorithme 12 et la méthode d'hyper-réduction [106] évoquée dans la section III.2.b.2. On illustre ici la méthodologie sur un calcul d'éprouvette.

VII.5.a Cas d'étude

On cherche à étudier l'influence de coefficients matériaux sur l'évolution des déformations dans un essai de fluage sur une éprouvette. L'éprouvette considérée (représentée en figure VII.23) est constituée de 21500 éléments. L'essai consiste à imposer une force constante aux extrémités de l'éprouvette de sorte que les contraintes engendrées au niveau de la zone d'intérêt (en vert sur la figure VII.23) soient de l'ordre de celles imposées dans le modèle 0D.

On choisit le modèle de comportement matériau Polystar tel qu'il a été présenté à la section VII.3. On étudie donc les paramètres n, K, d, c, β^*, Q_{s_0}, r^0, b et Q définis sur les intervalles donnés par le tableau VII.3.

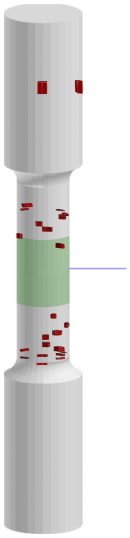


FIGURE VII.23 – Éprouvette d'étude. Les zones rouges correspondent au domaine réduit utilisé pour l'hyper-réduction

VII.5.b Description de la méthodologie de couplage

L'objectif est de construire un modèle de substitution de la relation :

$$(n, K, d, c, \beta^*, Q_{s_0}, r^0, b, Q) \mapsto (T, \bar{p}(\boldsymbol{x}, \bar{t}))$$

où la déformation viscoplastique cumulée est définie sur tout le domaine de l'éprouvette.

La définition des tenseurs physiques $\boldsymbol{\mathcal{A}}^1 \in \mathbb{R}^{n_1 \times n_9 \times 1 \times n_{\text{dom}}}$ et $\boldsymbol{\mathcal{A}}^2 \in \mathbb{R}^{n_1 \times n_9 \times n_t \times n_{\text{dom}}}$ font donc intervenir une dimension supplémentaire, associée au domaine spatial, par rapport aux tenseurs du modèle de la section VII.3. La valeur de n_{dom} correspond au nombre d'éléments du domaine spatial. En théorie, il est possible d'appliquer l'algorithme 12 sur les tenseurs $\boldsymbol{\mathcal{A}}^\chi$ pour lequel les deux dernières dimensions ont été associées. Toutefois, les temps de calcul pour accéder aux éléments de $\boldsymbol{\mathcal{A}}^\chi$ sont extrêmement importants si on cherche à résoudre le modèle physique par une méthode classique de résolution (type éléments finis). Cela ne permet pas en pratique d'envisager l'application de l'algorithme tel quel.

La solution qu'on propose est d'utiliser la méthode d'hyper-réduction, évoquée dans la section III.2.b.2, pour résoudre le modèle physique. La phase de construction hors ligne du modèle de substitution se base donc sur des résolutions en lignes du modèle hyper-réduit. Une étape préalable est donc de construire le modèle hyper-réduit.

Une seconde alternative est de construire un modèle de substitution non pas des solutions produites par le modèle hyper-réduit, mais directement sur les coefficients des modes des solutions hyper-réduites.

VII.5.c Construction du modèle hyper-réduit

La formulation du modèle hyper-réduit est essentiellement basée sur un ensemble de snapshots qui correspondent à des solutions du modèle physique et sur la définition d'un domaine réduit. Dans ce cas, le domaine réduit est composé de 114 éléments. Puisque la résolution du modèle physique complet est couteuse, toute la difficulté consiste à choisir un nombre de snapshots réduit permettant de représenter un maximum de comportements. Pour sélectionner des points d'apprentissage pertinents associés aux snapshots, on propose une méthodologie basée sur la construction préalable d'un train de tenseurs.

Le principe de la méthode est de construire dans un premier temps avec l'algorithme 12, un modèle de substitution du matériau étudié uniquement sur un point de Gauss (comme ce qui est fait dans la section VII.3). On appelle ce modèle de substitution, le modèle de substitution 0D. Lors de la phase d'apprentissage, un ensemble de points du domaine paramétrique est sélectionné de manière à minimiser l'erreur d'approximation. En d'autres termes, les points sont choisis de telle sorte que les solutions associées présentent des comportements les plus variés possible. Comme les contraintes appliquées dans le cas de l'éprouvette sont identiques à celle appliquée dans le modèle 0D, on peut considérer que ces sélections peuvent être exploitées pour définir des snapshots pertinents pour l'hyper-réduction.

VII.5.d Sélection des points d'apprentissage pour l'hyper-réduction

Au cours de l'algorithme 12, pour tout les indices sélectionnés $\mathcal{I}_k$ (Éq. (VI.11)) sont inclus dans l'ensemble $I_1 \times \cdots \times I_k$ d'après la proposition 5 et sont emboités à gauche, c'est-à-dire que :

$$\mathcal{I}_k \subset \mathcal{I}_{k-1} \times I_k$$

Le graphe VII.24 donne une représentation en arbre d'un exemple de sélection de lignes emboitées pour un tenseur d'ordre 10 (donc il y a 9 sélections successives) dont la taille de chaque dimension est de 20. Les ensembles de noeuds entourées en rouge et orange représentent respectivement des multi-indices appartenant à $\mathcal{I}_4$ et $\mathcal{I}_6$.

On considère ici que $D = D_1 \times \cdots \times D_9$ fait référence au domaine paramétrique discrétisé. D'après la section VI.3.a les indices $\mathcal{I}_k$ correspondent à des ensembles de points, notés $D_{\leq k}^{\mathcal{I}_k}$, dans le domaine discrétisé $D_{\leq k}$. Cela implique que l'algorithme fournit un ensemble d'échantillonnage de points $\left\{D_{\leq k}^{\mathcal{I}_k}\right\}_{k=1}^{9}$ emboités :

$$D_{\leq k}^{\mathcal{I}_k} \subset D_{\leq k-1}^{\mathcal{I}_{k-1}} \times D_k \subset D_{\leq k} \quad \forall\, k \in [\![1, 9]\!]$$

avec par convention $D_0 = \emptyset$. Ces échantillonnages ne permettent pas de définir naturellement

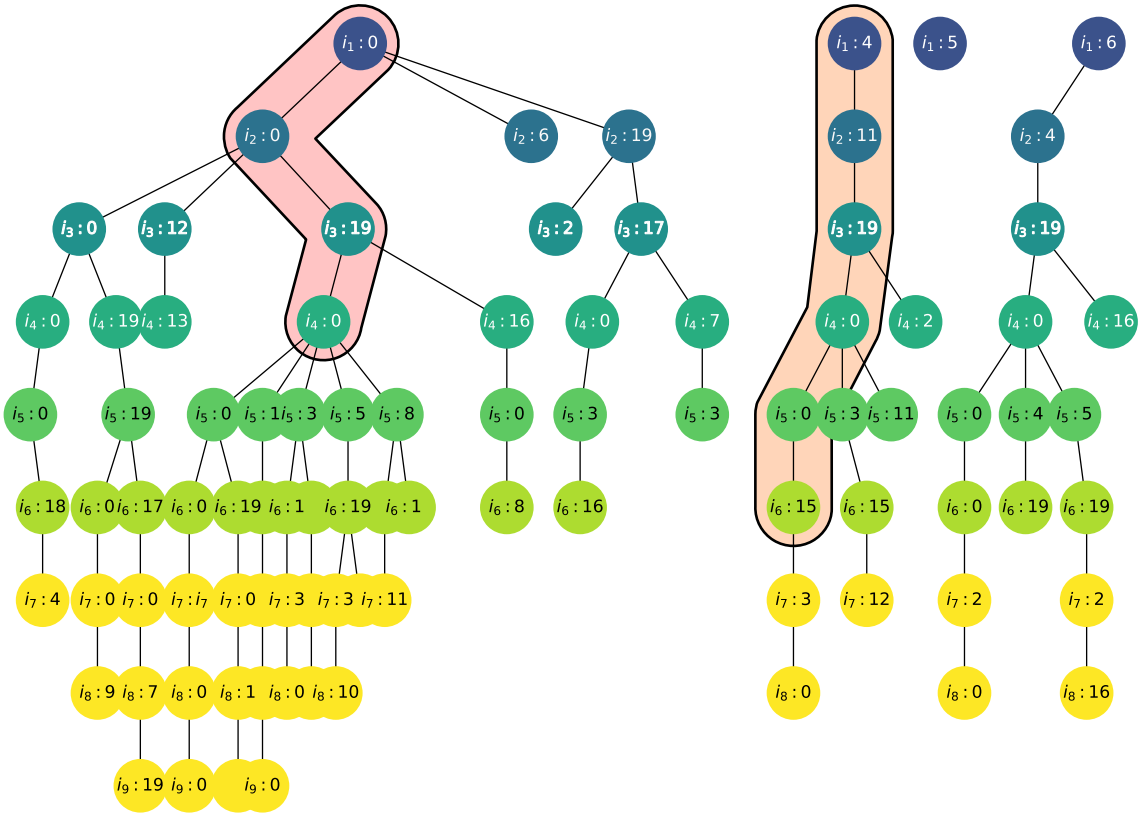


FIGURE VII.24 – Sélection de multi-indices emboités obtenue par l'algorithme 12. Certaines légendes de points sont omises pour alléger le schéma.

des points d'apprentissage définis sur le domaine paramétrique complet D.

On propose d'appliquer l'algorithme 12 sur le modèle de substitution 0D en réorganisant l'ordre des paramètres dans le sens inverse de manière à obtenir une sélection d'indices $\mathcal{I}'_k$ associés aux points d'apprentissage $D^{\mathcal{I}'_k}_{\geq 10-k}$ emboités dans le sens inverse : On a pour tout $k \in [\![1, 9]\!]$:

$$D^{\mathcal{I}'_k}_{\geq 10-k} \subset D_{10-k} \times D^{\mathcal{I}'_{k-1}}_{\geq 11-k} \subset D_{\geq 10-k} \quad \forall\, k \in [\![1, 9]\!]$$

avec par convention $D_{10} = \emptyset$.

Pour tout $k \in [\![1, 8]\!]$, à partir des échantillonnages partiels $D^{\mathcal{I}_k}_{\leq k} \subset D_{\leq k}$ et $D^{\mathcal{I}_{9-k}}_{\geq k+1} \subset D_{\geq k+1}$, on définit les échantillonnages sur le domaine complet suivant :

$$\widehat{D}_k = D^{\mathcal{I}_k}_{\leq k} \times D^{\mathcal{I}'_{9-k}}_{\geq k+1} \subset D$$

Finalement pour chaque $k \in [\![1, 8]\!]$, l'ensemble de points d'échantillonnage $\widehat{D}_k$ correspond à un ensemble de multi-indices $\widehat{\mathcal{I}}_k = \mathcal{I}_k \times \mathcal{I}'_{9-k} \subset I_1 \times \dots I_9$. La figure VII.25 représente les fibres du tenseur $\boldsymbol{\mathcal{A}}^\chi$ réarrangé correspondant aux points d'échantillonnage $\widehat{\mathcal{I}}_k$. Chaque fibre correspond à une sortie du modèle de substitution pour un paramétrage donnée.

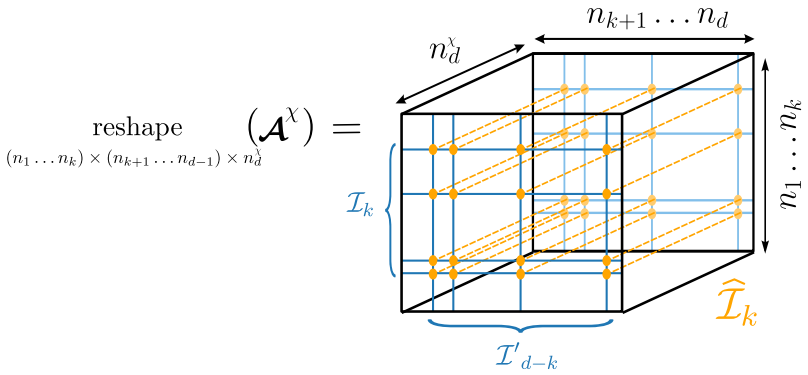


FIGURE VII.25 – Points d'échantillonnage $\widehat{\mathcal{I}}_k$.

Les points d'apprentissage utilisés pour définir les snapshots de l'hyper-reduction sont choisis parmi D_{ech} constitué de l'union des échantillonnages $\widehat{D}_k$ et de $D_{\leq 9}^{\mathcal{I}_9}$ et $D_{\geq 1}^{\mathcal{I}'_9}$:

$$D_{\text{ech}} = \bigcup D_{\leq 9}^{\mathcal{I}_9} \bigcup D_{\geq 1}^{\mathcal{I}'_9} \bigcup_{k=1}^{8} \widehat{D}_k$$

Cet ensemble contient trop de points d'échantillonnages pour définir les snapshots. Il s'agit à présent d'en sélectionner un sous-ensemble pertinent. Pour chaque point de D_{ech}, on calcul grâce au modèle de substitution 0D, le temps final de rupture ainsi que la déformation finale, générant ainsi un nuage de point en 2 dimensions. Comme on cherche à représenter un maximum de comportements différents, on souhaite sélectionner des points d'apprentissage tels que les temps finaux et déformations finales soient les plus variés possibles.

La méthode retenue consiste à définir a priori un nombre de snapshots, dans notre cas 20, puis de classifier le nuage de points en 20 zones. La classification est réalisée via l'algorithme k-means. Les points d'apprentissage sont définis comme les médoïdes des différentes classes. La figure VII.26 illustre la méthode de classification mise en œuvre.

Finalement, on obtient 20 points d'apprentissage pour l'hyper-réduction permettant de représenter des comportements variés. La figure VII.27 présente grâce à un graphique de coordonnées parallèles, l'ensemble des points d'apprentissage sélectionnés et leur déformation finale et temps de rupture associés. On peut observer que les points semblent se repartir de manière relativement uniforme sur l'ensemble des intervalles.

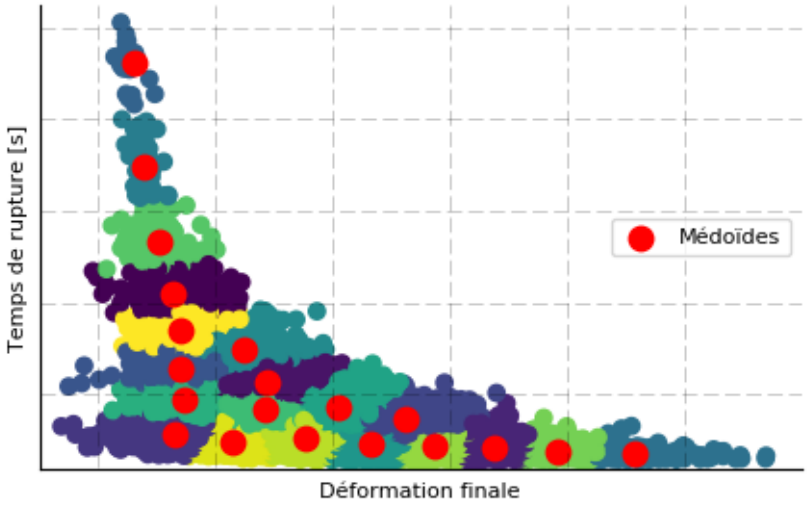


FIGURE VII.26 – Classification par k-means des points d'échantillonnages.

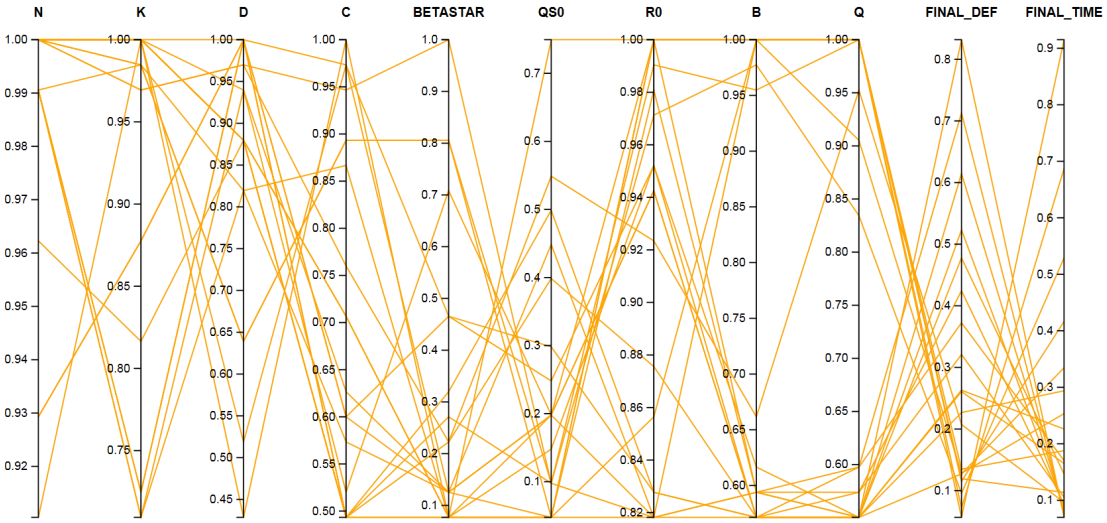


FIGURE VII.27 – Coordonnées parallèles des points d'apprentissage sélectionnés pour l'hyper-réduction.

Chapitre VIII

Implémentation algorithmique et outils numériques

Ce chapitre présente les outils numériques développés en Python au cours de la thèse. Ces outils sont essentiellement intégrés dans deux librairies Python. La première permet d'interfacer un code de calcul de résolution de problème mécanique. La seconde implémente l'algorithme de décomposition en trains de tenseurs hétérogènes. Une approche orientée objet a été suivie de sorte que les modèles physiques et les modèles de substitution exposent une interface commune. La dernière section du chapitre fait un retour sur les procédures de calibration constituant la principale préoccupation industrielle de la thèse. On y présente les nouvelles opportunités de travail permises par les modèles de substitution notamment via l'utilisation d'outils de visualisation interactifs et collaboratifs.

Table des matières

VIII.1 Présentation générale de l'implémentation des modèles 172
VIII.2 Interfaçage du modèle numérique 172
VIII.2.a Mise en place d'un calcul par le solveur numérique 173
VIII.2.b Interface du solveur numérique 173
VIII.2.c Interface d'un modèle de référence 177
VIII.2.d Définition du domaine paramétrique 179
VIII.3 Modèle de substitution basé sur la décomposition TT 180
VIII.3.a Méthode de construction 181
VIII.3.b Méthode d'évaluation 181
VIII.4 Exploitation des modèles 183
VIII.4.a Retour sur le processus de calibration 184
VIII.4.b Outils d'aide à la décision 184

VIII.1 Présentation générale de l'implémentation des modèles

Les modèles physiques ainsi que les modèles de substitution sont représentés par des instances de classes possédant des interfaces similaires. En particulier, l'évaluation des modèles est effectuée de manière identique via l'utilisation d'une méthode admettant en argument les mêmes objets.

La section VIII.2 décrit essentiellement l'implémentation de la classe `Zset_model` interfaçant de manière générique un solveur numérique permettant de résoudre des problèmes mécaniques. Cette classe permet d'instancier n'importe quel modèle physique résolu par le code de calcul. La méthode d'évaluation permet d'obtenir simultanément des solutions associées à un nombre arbitraire de paramètres. La méthode exploite l'indépendance du calcul des solutions paramétriques via une implémentation parallèle adaptée pour être exécutée sur des machines de calculs.

La section VIII.3 est consacrée à la classe `TT_model` qui permet de représenter des fonctions multivariées à valeurs vectorielles (Éq. (IV.1)) basées sur des décompositions en train de tenseurs. Ces fonctions correspondant typiquement à des approximations de solutions de modèles physiques. De manière similaire à la classe `Zset_model`, la méthode d'évaluation permet de calculer simultanément des sorties associées à un nombre arbitraire de paramètres d'entrée. La méthode `fit` correspond à l'implémentation de l'algorithme 12 et permet d'approcher un modèle de référence par des décompositions de train de tenseurs. Les arguments principaux qu'elle admet en entrée sont : une instance d'un modèle de référence à approcher, la discrétisation du domaine paramétrique ainsi que les tolérances de troncature utilisées au cours de l'algorithme.

VIII.2 Interfaçage du modèle numérique

La résolution d'un modèle physique mécanique est effectuée numériquement par un code de calcul. Dans cette thèse, le code commercial `Z-set`[3] est utilisé. Dans cette section, on présente tout d'abord l'utilisation classique du code de calcul `Z-set`. On décrit dans un second temps, l'implémentation de la classe `Zset_module` qui permet d'interfacer `Z-set`. Enfin la dernière partie présente la classe `Zset_model` permettant d'instancier spécifiquement des modèles physiques basés sur `Z-set`.

3. http://www.zset-software.com/

VIII.2.a Mise en place d'un calcul par le solveur numérique

La résolution d'un problème mécanique par le solveur `Z-set` passe par la définition préalable d'un ensemble de fichiers d'entrée décrivant le modèle physique. `Z-set` interprète ces fichiers, résout les équations et génère un ensemble de fichiers de sortie contenant les solutions du problème ainsi que des informations relatives à la procédure de résolution.

Pour la résolution d'un modèle de loi de comportement matériaux, deux fichiers d'entrée sont nécessaires :

— Un fichier `material.mat` qui décrit la loi de comportement et donne les valeurs des coefficients matériaux ;

— Un fichier `computation.inp` qui contient les informations de conditions limites appliquées, du solveur utilisé et des variables mécaniques à écrire dans les fichiers de sortie.

Le code est lancé en exécutant la commande console

```
Zrun computation.inp
```

Les fichiers de sortie générés sont :

— Un fichier `results.test` qui contient les valeurs des variables mécaniques de sortie en fonction du temps ;

— Un fichier `results.msg` qui donne des détails sur la résolution numérique et en particulier sa convergence.

VIII.2.b Interface du solveur numérique

La procédure de construction du modèle de substitution qu'on propose est basée sur la résolution du modèle de référence pour un nombre élevé de paramétrages différents.

L'objectif de l'interfaçage du code `Z-set` est de pouvoir lancer au sein d'une routine Python un nombre arbitraire de simulations pour des valeurs de coefficients matériaux différentes et d'extraire les données des fichiers de sortie.

La définition d'un modèle physique paramétrique nécessite deux fichiers modèles `computation.inp.tpl` et `material.inp.tpl`.

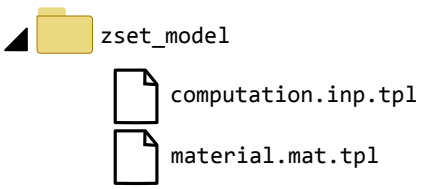


FIGURE VIII.1 – Dossier du modèle paramétrique.

Ces fichiers contiennent les informations de mise en données ainsi que des chaines de caractères destinées à être remplacées pour correspondre à différentes simulations. Les

chaines de caractères à modifier correspondent typiquement aux noms de fichiers de sortie et aux valeurs des coefficients matériaux qui sont considérés comme les entrées du modèle. Les figures VIII.2 et VIII.3 illustrent le contenu des fichiers `computation.inp.tpl` et `material.inp.tpl`. Les chaines de caractères à substituer sont entre accolades.

```
***behavior gen_evp
    **elasticity isotropic
        young 200.e+03
        poisson 0.3
    **potential associated ev
         *flow norton
             n {n}
             K {K}
         *criterion mises
         *isotropic nonlinear
             R0 {R0}
             Q {Q}
             b {b}
         *kinematic linear
             C {C}
***return
```

FIGURE VIII.2 – Exemple de fichier `material.inp.tpl`.

La classe Python `Zset_module` qui interface le code de calcul contient les attributs suivant :

— Le contenu d'un fichier modèle `.inp` ;
— Le contenu d'un fichier modèle `.mat` ;
— La liste des paramètres à substituer ;
— Le lien vers un répertoire temporaire de travail.

ainsi que les méthodes :

— Lecture des fichiers de modèle ;
— Écriture des fichiers d'entrée ;
— Lancement des calculs ;
— Lecture des fichiers de sortie ;
— Suppression des fichiers de calcul.

Instanciation L'instanciation d'un cas de calcul correspond à la ligne :

```
zset_module = Zset_module(zset_model_directory)
```

où `zset_model_directory` contient le lien vers le répertoire des fichiers modèles. Lors de l'instanciation, les attributs de l'instance `zset_module` sont définis à partir de la lecture de fichiers modèles. Le dossier temporaire est fourni par le système d'exploitation.

```
****simulate
    ***solver newton
    ...
    ***test {test_filename}
        **load cycle 4
            *segment 400

time eto11 eto22 eto33 eto12 eto23 eto31
0.  0.0 0.0 0.0 0.0 0.0 0.0
100. 0.01 0.0 0.0 0.0 0.0 0.0
200. 0.0 0.0 0.0 0.0 0.0 0.0
300. -0.01 0.0 0.0 0.0 0.0 0.0
400. 0.0 0.0 0.0 0.0 0.0 0.0
        **model
            *file {mat_filename}
        ...
        **output
            *dtime 3.
time sig11 eto11 evcum

****return
```

FIGURE VIII.3 – Exemple de fichier `computation.inp.tpl`.

Évaluation L'évaluation d'un cas de calcul correspond à la ligne :

```
output = zset_module.evaluate(input)
```

`input` correspond à une liste `input = [input1,input2,...,inputM]` de M vecteurs paramètres `intputj = (intputj_1,...,intputj_d)` pour lesquels on souhaite obtenir la solution numérique. Chaque `intputj_i` contient la valeur du i$^{\text{ème}}$ paramètre de la j$^{\text{ème}}$ simulation.

`output` correspond à un dictionnaire (au sens du langage Python) contenant M éléments. Pour chaque élément, la clé correspond au numéro j de la simulation et la valeur de l'élément contient pour la j$^{\text{ème}}$ simulation :

— Les valeurs du vecteur de paramètres d'entrée ;

— Le statut d'infrastructure ;

— Le statut numérique ;

— L'ensemble des valeurs des sorties écrites dans le fichier de sortie `.test`.

Le calcul d'une solution implique des interactions avec le système de fichier et le code de calcul `Z-set`. En particulier, les fichiers d'entrée doivent être écrits sur le disque, les fichiers de sortie doivent être lus et le code de calcul doit être exécuté. Le statut d'infrastructure contient des informations sur le bon déroulement de ces tâches, notamment :

— Si l'écriture des fichiers d'entrée a fonctionné ;

— Si le code de calcul a été lancé ;
— Si le code de calcul s'est terminé correctement ;
— Si les fichiers de sortie ont été lus correctement.

Lorsque le système opératif est défaillant, ces informations sont importantes pour savoir s'il est nécessaire de relancer les calculs.

Le statut numérique contient des booléens indiquant si la résolution numérique du code de calcul a convergé.

VIII.2.b.1 Méthode d'évaluation parallèle

On décrit dans cette section l'ensemble des opérations de parallélisation et de gestion des données impliquées lors de évaluation d'une instance de la classe `Zset_module` :

```
output = zset_module.evaluate(input)
```

où la liste `input` contient M vecteurs de paramètres.

Potentiellement, le nombre de simulations à effectuer peut être très important ce qui nécessite de gérer une grande quantité de données. Puisque les simulations sont indépendantes les unes des autres, il est naturel de paralléliser les calculs. On propose une stratégie d'implémentation robuste qui permet de gérer le lancement parallèle d'un grand nombre de simulations en limitant au maximum l'empreinte mémoire.

Le principe de la stratégie consiste à faire tourner en parallèle n_{proc} processus indépendants capables d'exécuter des simulations parmi les M simulations à réaliser. En pratique, on instancie n_{proc} threads d'une classe qui hérite de la classe `Thread` de la librairie Python `threading`.

Chaque thread effectue dans un premier temps les tâches suivantes :
— Extraction de $m \ll M$ éléments de la liste `input` ;
— Écriture de tous les fichiers d'entrée correspondants dans le dossier temporaire (Fig. VIII.4).
— Lancement séquentiel des m simulations ;
— Lecture de tous les fichiers de sortie correspondants (Fig. VIII.4).

Pour chaque simulation dont les statuts d'infrastructure sont positifs, les valeurs des sorties sont ajoutées au dictionnaire `output` commun à tous les threads. Pour les simulations qui ont échoué, le vecteur de paramètres correspondant est rajouté dans la liste `input` pour que la simulation soit une nouvelle fois exécutée. Ce processus permet de gérer les erreurs qui peuvent survenir indépendamment du code Python. Si une simulation échoue plusieurs fois, on considère que cela vient de la mise en données et non de l'infrastructure.

Une des difficultés pratiques rencontrées pour l'implémentation de ce mécanisme de gestion des échecs de calcul vient du fait que `Z-set` ne possède pas de sortie d'erreur standard. Les statuts sont obtenus via l'analyse des fichiers de sortie.

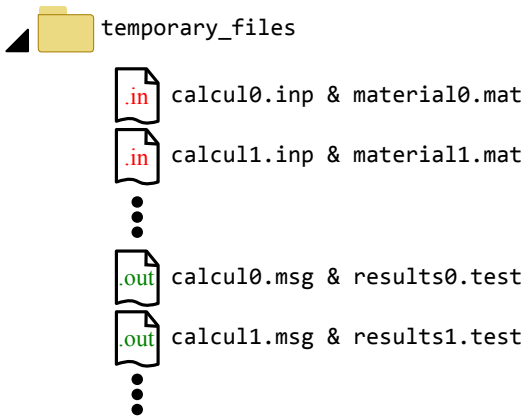


FIGURE VIII.4 – Fichiers temporaires produits au cours des simulations.

En pratique, pour des raisons de couplage entre `Z-set` et le système d'exploitation, certains fichiers restent ouverts au terme de l'exécution de `Z-set` et ne peuvent pas être supprimés. Pour ne pas avoir à attendre que les fichiers soient fermés, la suppression des fichiers n'est pas effectuée par les processus de lancement de calcul, mais par un processus indépendant qui est exécuté en parallèle.

Le fait de lancer les simulations par groupe de $m \ll M$ permet de ne pas surcharger la machine de calcul et également de limiter l'empreinte mémoire puisque les fichiers d'entrée et de sortie sont crées et supprimés au fur et à mesure de l'avancement des calculs.

Un mécanisme de reprise de calcul optionnel a d'autre part été implémenté. Lorsque l'option est spécifiée, l'ensemble des résultats sont sauvegardés dans des fichiers temporaires, à chaque fois qu'un nombre déterminé de simulations est effectué. Ce mécanisme permet d'éviter la perte des données lorsque les programmes terminent de manière anormale ou plus classiquement lorsque les temps alloués sur les machines de calculs sont écoulés.

L'application de la méthodologie demande d'exécuter le code de calcul un grand nombre de fois pour des valeurs de coefficients matériaux très variées. Elle a ainsi permis de déceler de nombreux problèmes dans le fonctionnement même du code de calcul. Par exemple, cela a mis en évidence que le code `Z-set` avait des comportements non souhaités pour certaines valeurs de coefficients (terminaison anormale, arrêt des calculs...). Des problèmes d'implémentation de certaines lois matériaux ont d'autre part été décelés. La méthodologie qu'on propose permet d'une certaine manière de tester la robustesse des codes de calculs.

VIII.2.c Interface d'un modèle de référence

La classe `Zset_module` gère l'ensemble des opérations permettant de lancer de nombreuses simulations et d'en extraire les sorties associées. La construction d'un modèle de substitution par la méthodologie proposée demande cependant de structurer la relation entre les entrées et les sorties du modèle. Un modèle de référence qu'on cherche à approcher

est défini comme une instance `zset_model` de la classe `Zset_model`.

```
zset_model = Zset_model(zset_module)
```

La classe `Zset_model` permet de représenter un modèle comme un ensemble de N fonctions $\boldsymbol{Y}^\chi$ (Éq. (VI.3)) mettant en relation d paramètres d'entrée avec des sorties vectorielles de taille m^χ.

La coordonnée temporelle étant discrétisée en n_t points, pour une valeur de paramètre, l'évaluation d'une instance

```
output = zset_model(input)
```

retourne une liste `output` de N tableaux `output_chi` de taille $m^\chi \times n_t$.

Les attributs de l'instance `zset_module` sont les suivants :

— une instance `zset_module` de la classe `Zset_module` ;

— le nombre `d` de paramètres d'entrée ;

— les tailles `(m1,...,mN)` des tableaux de sortie ;

Définition des entrées L'évaluation d'une instance de `Zset_module` prend en entrée des listes de vecteurs correspondant au coefficient matériaux du modèle physique étudié. La classe `Zset_model` permet de définir les opérations de reparamétrage de ces coefficients, discutées en section VI.4.b.

Définition des sorties L'évaluation d'une instance de `Zset_module` retourne l'ensemble des variables mécaniques en fonction du temps qui sont spécifiées dans les fichiers modèles. La classe `Zset_model` permet de définir à partir de ses sorties, les quantités d'intérêt et leur agrégation, présentées dans la section VI.4.a.

Évaluation d'une instance On présente ici la méthode d'évaluation en supposant, par souci de simplicité, qu'il n'y a pas de reparamétrage. La méthode d'évaluation de `zset_model` prend en argument une liste `X = [X_1,X_2,...,X_d]` de tableaux `X_i` contenant chacun `p_i` valeurs pour le i$^{\text{ème}}$ paramètre.

L'évaluation de `X` consiste à calculer les sorties associées à tous les vecteurs de paramètres du produit cartésien $X_1 \times \cdots \times X_d$, c'est-à-dire $p_1 \dots p_d$ vecteurs de paramètres. La liste `X` est traduite en une liste `input` de vecteurs de paramètres donnés en argument de la méthode d'évaluation de `zset_module` :

```
output = zset_module.evaluate(input)
```

Les valeurs de sortie du dictionnaire `output` sont ensuite réorganisées selon l'agrégation des quantités d'intérêt dans un ensemble de N tableaux Y^χ de taille $p_1 \times \cdots \times p_d \times m^\chi \times n_t$.

VIII.2.d Définition du domaine paramétrique

À partir d'un modèle de référence représenté par une instance de `Zset_model`, la construction d'un modèle de substitution suppose la définition d'un domaine paramétrique. Le domaine paramétrique $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ défini comme un produit cartésien d'intervalle réel $\mathcal{D}_i$ doit être inclus dans :

— le domaine physique admissible ;

— le domaine numérique admissible ;

— un domaine d'intérêt.

Le domaine physique admissible correspond à l'ensemble des valeurs de paramètres pour lesquelles les solutions associées ont un sens physique. Le domaine numérique admissible correspond à l'ensemble des valeurs de paramètres pour lesquelles le code numérique est capable de calculer une solution (qui n'est pas nécessairement correcte physiquement). Le domaine d'intérêt correspond à l'ensemble des valeurs de paramètres pour lesquelles les solutions associées ont du sens vis-à-vis de l'étude paramétrique.

Sans connaissance a priori sur le modèle, cette définition peut être particulièrement délicate. On propose une méthode de recherche de domaine paramétrique basée uniquement sur la connaissance d'un point de référence déterminé par l'utilisateur.

La méthode consiste à calculer des sorties en faisant varier indépendamment la valeur de chaque paramètre (tous les autres étant fixés) autour du point de référence. Pour chaque simulation, on calcule un critère de recherche déterminé préalablement. Les bornes inférieure et supérieure de chaque intervalle paramétrique sont définies lorsque le rapport entre le critère de recherche courant et le critère de recherche du point de référence atteint une valeur fixée préalablement, par exemple 20%. La figure VIII.5 illustre la recherche du domaine paramétrique.

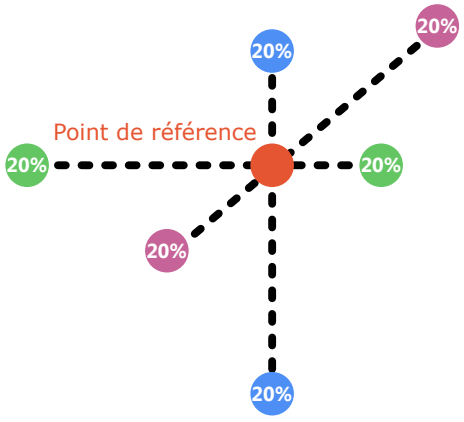


FIGURE VIII.5 – Exploration préliminaire du domaine paramétrique.

Le critère de recherche est défini tel que les solutions sont :

— numériquement admissibles : le code s'exécute normalement et retourne des fichiers

de sortie ;

— physiquement admissibles : par exemple, dans le cadre d'une étude mécanique pour un essai numérique à contraintes imposées, on peut considérer que des déformations au-delà de 8% sortent du cadre des hypothèses de petites déformations et n'ont donc plus de sens physique. De plus pour de grandes déformations, les non-linéarités sont pénalisantes en temps de calcul et il faut si possible éviter les calculs inutiles fortement non linéaires hors du domaine physique d'intérêt ;

— intéressantes du point de vue de l'étude à réaliser : par exemple, on peut s'intéresser uniquement aux solutions dont la déformation viscoplastique cumulée est supérieure à une certaine valeur connue.

En pratique, le parcours des valeurs pour déterminer les bornes des intervalles paramétriques est réalisé par dichotomie. Cette méthode est assez discutable car elle ne tient pas compte de l'influence des couplages entre paramètres, mais permet en pratique de construire une première estimation d'un domaine paramétrique admissible.

VIII.3 Modèle de substitution basé sur la décomposition TT

On cherche à construire un modèle de substitution d'un modèle de référence représenté par une instance `zset_model` de la classe `Zset_model` sur le domaine paramétrique discrétisé $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$. En termes d'implémentation, la discrétisation D est représentée par une liste `Xgrid = [Xgrid_1,...,Xgrid_d]` où chaque `Xgrid_i` contient une liste de `ni` éléments. L'instance `zset_model` et la discrétisation `Xgrid_i` définissent complètement un ensemble de tenseurs interdépendants à partir desquels il est possible d'appliquer l'algorithme 12.

Un modèle de substitution est représenté par une instance de la classe `TT_model` contenant essentiellement les attributs suivants :

— La grille de discrétisation `Xgrid` ;

— Les facteurs des trains de tenseurs.

et les méthodes de :

— Construction ;

— Sauvegarde ;

— Chargement ;

— Évaluation.

L'instanciation de la classe `TT_model` est réalisée par la ligne :

```
tt_model = TT_model.evaluate(input)
```

VIII.3.a Méthode de construction

La méthode de construction `fit` correspond à l'implémentation de l'algorithme 12 et prend en argument :

— une instance `ref_model` correspondant à un modèle de référence ;
— une grille de discrétisation `Xgrid` ;
— une tolérance de troncature `epsilon_tol`.

```
tt_model.fit(ref_model,Xgrid,epsilon_tol)
```

Puisque les classes `TT_model` et `Zset_model` possèdent la même interface pour la méthode d'évaluation, la méthode de construction admet en argument une instance d'une de ces classes. En d'autres termes, pour la méthode de construction le modèle de référence peut aussi bien être basé sur le code de calcul `Z-set` que sur un modèle de substitution TT.

La construction constitue la phase hors ligne et implique des calculs intensifs en particulier dus au nombre élevé d'évaluations du modèle de référence. Cette méthode est par conséquent généralement réalisée sur une machine de calcul dont les capacités de parallélisation et de mémoire sont plus importantes. Une stratégie de reprise de calcul est également implémentée au sein de la méthode de construction. L'application de la méthode de construction est généralement suivie de la méthode de sauvegarde qui permet de stocker dans un fichier `tt_model.tt` les trains de tenseurs construits afin de pouvoir les utiliser lors de la phase en ligne.

```
tt_model.save(tt_model.tt)
```

VIII.3.b Méthode d'évaluation

La méthode d'évaluation est précédée par la méthode de chargement qui permet de charger en mémoire les trains de tenseurs construits préalablement.

```
tt_model.load(tt_model.tt)
```

L'efficacité de l'évaluation du modèle de substitution lors de la phase en ligne constitue le principal intérêt de l'approche qu'on propose. La qualité de l'implémentation de l'évaluation des trains de tenseurs est donc particulièrement importante pour assurer un accès temps réel aux éléments.

L'évaluation d'un élément $\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d)$ correspond à une suite de multiplications matricielles

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = G_1(i_1)G_2(i_2)\ldots G_{d-1}(i_{d-1})G_d(i_d)$$

Pour réaliser les opérations le plus rapidement possible, on découpe le produit en deux vecteurs :

$$\boldsymbol{\mathcal{T}}(i_1, \dots, i_d) = W^{\triangleleft} W^{\triangleright}$$

où

$$W^{\triangleleft} = G_1(i_1) \dots G_{d/2}(i_{d/2})$$
$$W^{\triangleright} = G_{d/2+1}(i_{d/2+1}) \dots G_d(i_d)$$

Dans l'implémentation, le calcul des vecteurs $W^{\triangleleft}$ et $W^{\triangleright}$ est effectué en parallèle, puis on effectue le produit des vecteurs.

L'évaluation d'un élément implique donc $d/2$ multiplications matrice-vecteur (réalisées en parallèle) puis une multiplication vecteur-vecteur. En pratique, les temps de calcul sont faibles et permettent un accès temps réel d'un point de vue perception humaine. On peut remarquer que l'évaluation des éléments du tenseur, n'impliquant que des multiplications matricielles, peut être potentiellement largement accélérée en réalisant les calculs sur GPU (Graphics Processing Unit).

L'implémentation de l'interpolation multilinéaire par morceaux (Section IV.1.d) est également implémentée dans le code. Disposer d'une méthode d'interpolation est essentiel en pratique pour estimer le modèle de substitution sur l'intégralité du domaine paramétrique continu.

VIII.3.b.1 Factorisation de calcul

Si l'on souhaite accéder aux éléments du tenseur correspondant aux indices de l'ensemble $\mathcal{I}_1 \times \cdots \times \mathcal{I}_d$ où pour tout $k \in [\![1, d]\!]$:

$$\mathcal{I}_k = \left\{ \mathcal{I}_k^{(1)}, \dots, \mathcal{I}_k^{(m_k)} \right\}$$

il est clair que pour chaque élément du tenseur, plusieurs multiplications matricielles sont communes.

On présente dans cette section une stratégie de factorisation des calculs dans un cas particulier d'indices auxquels on souhaite accéder. Dans certaines applications (Section VII.2.b.3), on applique l'algorithme 12 sur des modèles basés sur des décompositions tensorielles. D'après la proposition 6, pour calculer une approximation de la matrice A_k via la Snapshot POD, on doit calculer les éléments :

$$\widetilde{A}_k = \{ \boldsymbol{\mathcal{A}}(\mathcal{I}_{k-1}, I_k, \mathcal{J}_k) \}_2$$

où $\mathcal{I}_{k-1}$ et $\mathcal{J}_k$ correspondent respectivement à une sélection de multi-indices parmi $I_1 \times \cdots \times I_{k-1}$ et $I_{k+1} \times \cdots \times I_d$. Avec les notations utilisées précédemment, on a $s_{k-1} = \text{Card}(\mathcal{I}_{k-1})$, $n_k = \text{Card}(I_k)$ et $m_k = \text{Card}(\mathcal{J}_k)$.

Il faut donc effectuer les multiplications matricielles suivantes :

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = W^{\triangleleft}(i_1, \ldots, i_{k-1}) G_k(i_k) W^{\triangleright}(i_{k+1}, \ldots, i_d) \quad \forall\, (i_1, \ldots, i_d) \in \mathcal{I}_{k-1} \times I_k \times \mathcal{J}_k$$

avec

$$W^{\triangleleft}(i_1, \ldots, i_{k-1}) = G_1(i_1) \ldots G_{k-1}(i_{k-1})$$
$$W^{\triangleright}(i_{k+1}, \ldots, i_d) = G_k(i_k) \ldots G_d(i_d)$$

Le calcul de l'ensemble des éléments peut donc être décomposé en plusieurs étapes :

— Calcul des s_{k-1} vecteurs lignes $W^{\triangleleft}$ en effectuant les multiplications matricielles de gauche à droite ;

— Calcul des m_k vecteurs colonnes $W^{\triangleright}$ en effectuant les multiplications matricielles de droite à gauche ;

— Calcul des $s_{k-1} \times n_k$ produits matrice-vecteur :

$$W^{\square}(i_1, \ldots, i_k) = W^{\triangleleft}(i_1, \ldots, i_{k-1}) G_k(i_k)$$

— Calcul des $s_{k-1} n_k \times m_k$ produits vecteur-vecteur :

$$\boldsymbol{\mathcal{T}}(i_1, \ldots, i_d) = W^{\square}(i_1, \ldots, i_k) W^{\triangleright}(i_{k+1}, \ldots, i_d)$$

En termes d'implémentation, les calculs sont effectués en allouant le maximum de processeurs partageant la même mémoire. Les processeurs réalisent ensuite en parallèle la séquence de tâches listées ci-dessus.

VIII.4 Exploitation des modèles

Un des objectifs industriels de cette thèse est de fournir aux experts matériaux des outils numériques pour faciliter la calibration de lois de comportement. On discute dans cette section des opportunités offertes par l'utilisation de modèles de substitutions évaluables en temps réel. L'objectif de l'approche proposée est de valoriser le travail des experts en leur permettant d'exploiter efficacement les modèles matériaux en s'affranchissant des contraintes de coût de calcul important auxquels sont généralement soumis les modèles physiques.

VIII.4.a Retour sur le processus de calibration

L'identification des coefficients matériaux tolère en pratique des marges d'erreur sur la correspondance entre les résultats numériques et expérimentaux. Par conséquent, le fait que les modèles de substitution fournissent des prédictions non exactes n'est intrinsèquement pas problématique dans ces applications. Pour cette raison, l'utilisation d'un modèle de substitution s'accorde particulièrement bien avec la procédure de calibration.

D'autre part, l'ensemble des étapes qui interviennent dans l'approche d'identification des coefficients matériaux et la mise en place de modèle de substitution possèdent des similarités qui rendent ces procédures particulièrement compatibles. Les étapes de calibration se divisent en deux phases distinctes similaires à la décomposition phase hors ligne/phase en ligne utilisée pour les modèles de substitution.

Phase hors ligne La phase hors ligne correspond à la construction du modèle de substitution. Cette phase couteuse d'apprentissage qui implique des calculs intensifs donne le temps aux experts pour préparer et conduire les essais expérimentaux.

Pour effectuer une identification de qualité, le rôle des experts est décisif autant pour la préparation des campagnes expérimentales que pour la préparation du modèle de substitution. Il intervient notamment aux étapes de définition du modèle de référence, c'est-à-dire pour définir :

— les sorties d'intérêt ;
— les entrées pertinentes (reparamétrage) ;
— le domaine paramétrique ;
— la hiérarchisation des dimensions dans les tenseurs ;

Il intervient d'autre part lors de la définition des discrétisations du domaine paramétrique ainsi que dans le choix des tolérances de troncature.

Phase en ligne Une fois les essais expérimentaux réalisés et le modèle de substitution préparé, la phase en ligne consiste à exploiter le modèle pour effectivement identifier les coefficients matériaux. L'efficacité de cette phase résulte de la possibilité d'évaluer en temps réel le modèle sur un large domaine paramétrique.

VIII.4.b Outils d'aide à la décision

Disposer de modèles de substitution évaluables en temps-réel permet de concevoir des interfaces homme-machine de visualisation des données. Ce type d'outils peut particulièrement améliorer l'efficacité du travail d'identification des modèles pour les experts en mécanique des matériaux.

On propose dans cette section deux démonstrateurs interactifs de visualisation temps-réel exploitant les modèles de substitution basés sur la décomposition tensorielle TT.

VIII.4.b.1 Outil de visualisation interactif

L'outil présenté dans la figure VIII.6consiste en une interface graphique permettant de modifier les paramètres matériaux et de visualiser en temps réel l'effet sur les solutions. L'implémentation est basée sur la librairie Python `wxpython`. Dans l'illustration, les courbes sont associées au modèle élasto-viscoplastique de la section VII.2.

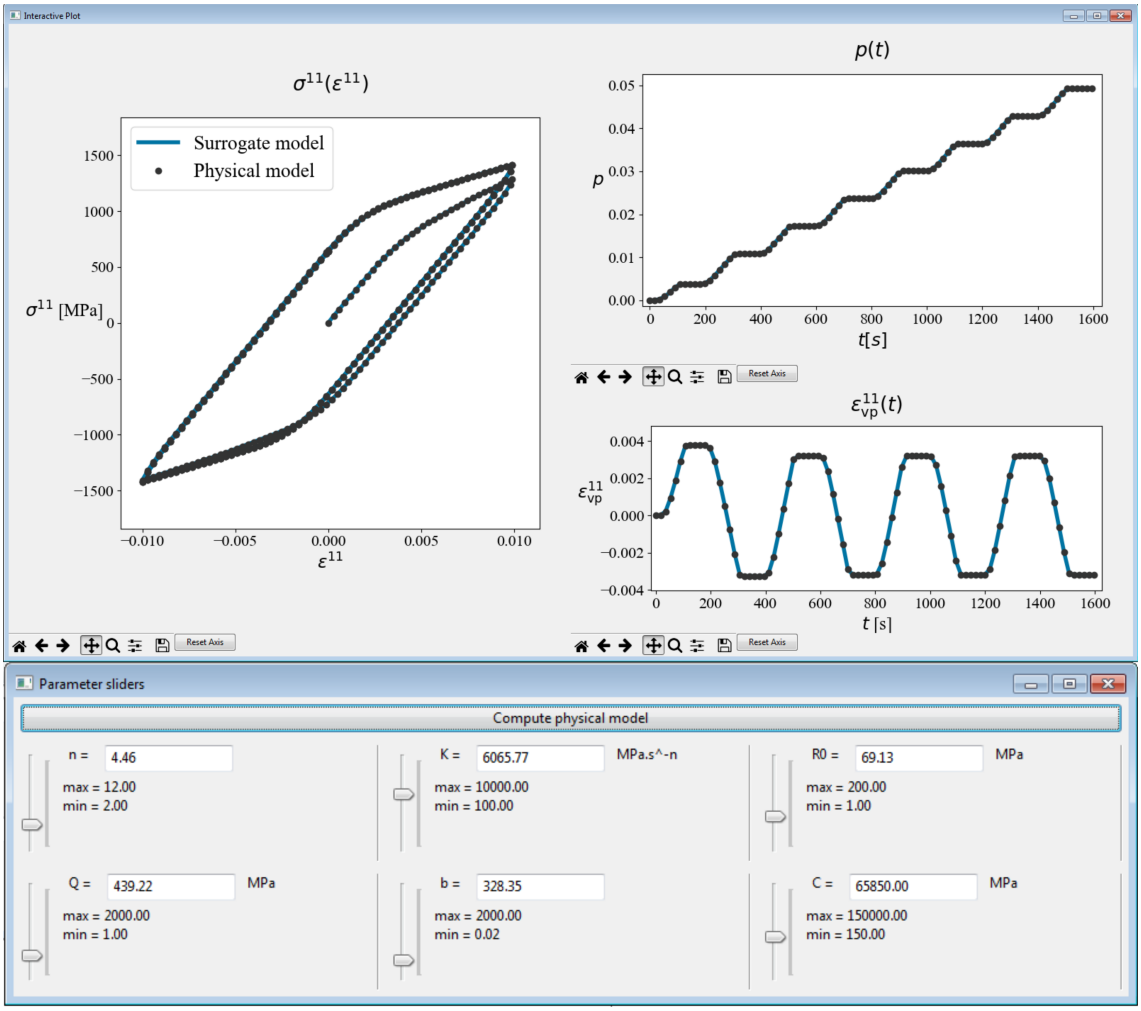


FIGURE VIII.6 – Outil de visualisation pour la loi elasto-viscoplastique.

Ce type d'outil permet de se forger plus rapidement une intuition vis-à-vis de l'influence des paramètres sur les sorties du modèle et aussi de comprendre les couplages complexes difficilement identifiables via la lecture des équations. Cela donne d'autre part, de nouvelles perspectives pour l'analyse de sensibilité ou la quantification d'incertitude.

VIII.4.b.2 Approches collaboratives

Certaines procédures de calibration nécessitent l'intervention de plusieurs experts pour faire un choix de coefficient matériaux pertinent. La possibilité d'évaluation en temps réel donne l'opportunité de promouvoir la conception d'outils de décision collaboratifs où chaque expert à la main sur les paramètres. Le travail collaboratif consiste à réunir des experts et les faire discuter avec pour support la possibilité de connaître en temps réel les résultats. On propose un outil de visualisation interactif et collaboratif en figure VIII.7. L'outil consiste en une interface graphique dont le contrôle est accessible à distance depuis n'importe quel terminal (tablette, mobile, ordinateur personnel).

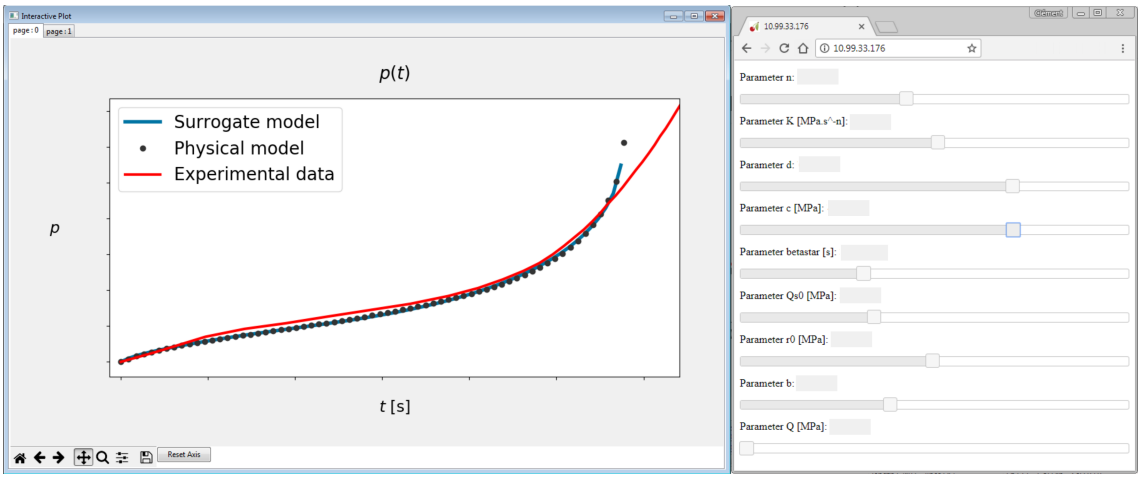


FIGURE VIII.7 – Illustration d'un outil de visualisation collaboratif.

La courbe de la figure VIII.7 correspond au modèle Polystar de la section VII.3.

La partie visualisation de l'outil est basée sur la librairie Python `wxpython` Les contrôles à distance sont codés en JavaScript et le protocole de communication grâce à ws-socket.

Chapitre IX

Conclusions et perspectives

IX.1 Conclusions

La contribution majeure de ce travail de thèse concerne le développement d'une nouvelle méthode non intrusive pour construire des modèles de substitution approchant les solutions hétérogènes de modèles physiques multiparamétriques. Les solutions paramétriques des modèles, exprimées sous la forme de fonctions multivariées à valeurs vectorielles, sont approchées par de multiples représentations tensorielles basées sur la décomposition en train de tenseurs. Le principe de la méthodologie consiste donc essentiellement à approcher des tenseurs multidimensionnels par des décompositions en train de tenseurs.

La première contribution correspond à l'introduction d'un cadre d'algorithmique générique (Chapitre IV) pour construire des décompositions en train de tenseurs approchant des tenseurs multidimensionnels quelconques. La combinaison d'outils mathématiques d'algèbre tensorielle et issus des méthodes de modèles d'ordre réduit a abouti au développement d'une spécialisation de l'algorithme générique – la TT-gappy – (Chapitre V) permettant de construire des approximations en se basant sur une exploration parcimonieuse des tenseurs de référence. Une borne d'erreur particulièrement fine, relative à ces approximations tensorielles, a été démontrée grâce à l'introduction de l'approximation gappy POD. Une extension de l'algorithme TT-gappy – l'algorithme de décomposition en trains de tenseurs hétérogènes – est proposée (Chapitre VI) pour permettre de construire simultanément des d'approximations de fonctions multivariées à valeurs vectorielles hétérogènes. Cet algorithme permet de construire des modèles de substitution de modèles physiques paramétriques, impliquant des quantités hétérogènes, à partir d'un nombre réduit de résolutions des modèles physiques.

La méthodologie est appliquée sur des lois de comportement matériaux (Chapitre VII) formulées par des équations différentielles et présentant des non-linéarités particulièrement importantes telles que des effets de seuils. Les résultats numériques illustrent les performances de l'algorithme pour construire des modèles des substitutions compacts en termes d'espace mémoire et précis étant donné les efforts de calcul modérés dépensés lors de la phase hors ligne.

Les procédures actuelles de calibration de lois matériaux complexes posent de nombreuses problématiques en particulier dues aux calculs couteux qu'elles impliquent, aboutissant parfois à des identifications non satisfaisantes. Ce travail de thèse a permis de proposer une nouvelle méthodologie de travail pour la calibration des lois matériaux via l'utilisation de modèles de substitution évaluables en temps-réel. De tels modèles permettent de concevoir des outils interactifs et collaboratifs de visualisation des résultats (Chapitre VIII). Ces outils permettent de manipuler simplement les modèles et ainsi d'en améliorer de manière significative la compréhension.

IX.2 Perspectives

La méthodologie présentée dans cette thèse est entièrement implémentée et directement applicable pour être testée sur des lois de comportement existantes. La conversion de modèles physiques, potentiellement couteux à évaluer, en jeux de données basés sur le format en train de tenseurs peut rendre leur exploitation beaucoup plus efficace. Il serait intéressant de mesurer la qualité des approximations produites en considérant par exemple des lois matériaux impliquant un plus grand nombre de coefficients matériaux.

Dans la section VI.4.b, l'importance potentielle des reparamétrages sur la qualité des modèles de substitution a été évoquée. Des résultats préliminaires, non présentés dans la thèse, semblent appuyer ces hypothèses. Toutefois, définir des indicateurs des comparaisons entre des modèles de substitution définis sur des domaines paramétriques différents est particulièrement difficile.

Le travail de thèse ouvre des perspectives sur une extension de la méthode dans le cadre de calcul de structures. En particulier, l'application concrète de la méthode de couplage avec l'hyper-réduction présentée en section VII.5 reste à réaliser.

Du point de vue de la méthode de construction, plusieurs perceptives d'améliorations ont été évoquées en section VI.6. Les pistes principales concernent deux aspects de l'algorithme 12. Le développement de méthodes plus robustes pour garantir la qualité des bases POD, par exemple via l'utilisation de plans d'expérience plus adaptés ou par des méthodes de validation croisées. L'amélioration de la méthode de sélection de lignes, par exemple en tenant compte simultanément de toutes les approximations ou en exploitant la borne d'erreur de la proposition 11. Sachant que l'ordre des dimensions peut avoir une grande influence sur la qualité des approximations, une étude des facteurs déterminants serait importante à réaliser.

Pour appliquer la méthodologie proposée, il est indispensable de pouvoir calculer les solutions du modèle physique à approcher sur l'ensemble du domaine paramétrique. Lorsque les modèles physiques sont complexes, les codes de calcul numérique peuvent échouer lors de la résolution des équations pour certains jeux de paramètres. Pour appliquer malgré tout

la méthodologie, l'unique remède proposé consiste à restreindre le domaine paramétrique. Une dernière ouverture concerne donc la possibilité d'ajuster des décompositions en trains de tenseurs sur des données incomplètes. Dans l'algorithme TT-gappy, cela se traduit par la capacité de construire des bases POD à partir de colonnes dont certaines valeurs sont manquantes. La littérature sur les approches d'approximations lacunaires où les approches de complétion de tenseurs peut donner des pistes de solutions.

Annexe A

Résultats d'algèbre linéaire et tensorielle

La proposition 15 présente des résultats classiques d'algèbre linéaire.

Proposition 15. *Soient les matrices $A \in \mathbb{R}^{n\times m}$ et $B \in \mathbb{R}^{m\times p}$ et une matrice $U \in \mathbb{R}^{n\times r}$ dont les colonnes sont orthogonales. Les résultats suivants sont vérifiés :*

$$\|AB\| \leq \|A\|_2 \|B\| \tag{A.1}$$

$$\left\|UU^T A\right\| = \left\|U^T A\right\| \leq \|A\| \tag{A.2}$$

On présente la proposition 16 qui est un résultat d'algèbre linéaire dont la démonstration est non triviale.

Proposition 16. *Soit une matrice $X \in \mathbb{R}^{n\times m}$ de rang r. Les deux propositions suivantes sont équivalentes :*

Décomposition en éléments propres *Une décomposition en éléments propres non nuls de X est donnée par :*

$$XX^T = V\Sigma^2 V^T$$

avec $V \in \mathbb{R}^{n\times r}$ et $\Sigma \in \mathbb{R}^{r\times r}$ une matrice diagonale, en imposant $V^T V = \mathbb{I}_r$ et en ordonnant les éléments diagonaux de Σ par valeurs absolues décroissantes.

Décomposition en valeurs singulières *Une décomposition en valeurs singulières de X tronquée au rang r est donnée par :*

$$X = V\Sigma W^T$$

avec $W = \Sigma^{-1}V^T X$ et en ordonnant les éléments diagonaux de Σ par valeurs absolues décroissantes.

Démonstration de la proposition 16. Le passage de la décomposition en valeurs singulières à la décomposition en éléments propres est trivial. On prouve l'implication dans l'autre sens. On a XX^T symétrique donc d'après le théorème spectral il existe une matrice $\overline{U} \in \mathbb{R}^{n\times n}$

orthogonale et une matrice $\overline{\Sigma} \in \mathbb{R}^{n\times n}$ diagonale tels que :

$$\overline{V}^T X X^T \overline{V} = \overline{\Sigma}^2 \tag{A.3}$$

On note r le nombre d'éléments (valeurs propres) non nuls de la diagonale de $\overline{\Sigma}$. Soit $\Sigma \in \mathbb{R}^{r\times r}$ la sous-matrice de $\overline{\Sigma}$ constituée dans r premières lignes et colonnes. On décompose également $\overline{V}$ en deux sous-matrices $V \in \mathbb{R}^{n\times r}$ et $V' \in \mathbb{R}^{n\times n-r}$ telles que :

$$V = [V; V'] \tag{A.4}$$

Comme par hypothèse $\overline{V}^T\overline{V} = \overline{V}\overline{V}^T = \mathbb{I}_n$ alors on a :

$$\begin{aligned} V^T V &= \mathbb{I}_r \\ V'^T V' &= \mathbb{I}_{n-r} \\ VV^T + V'V'^T &= \mathbb{I}_n \end{aligned} \tag{A.5}$$

D'après l'équation (A.3), la décomposition (Éq. (A.4)) et la définition de Σ, on a :

$$\begin{aligned} V^T X X^T V = \Sigma^2 \Rightarrow & \Sigma^{-1} V^T X X^T V = \Sigma \\ \Rightarrow & W^T X^T V = \Sigma \end{aligned}$$

avec $W^T = \Sigma^{-1} V^T X$

À nouveau, d'après l'équation (A.3), la décomposition (Éq. (A.4)) et la définition de Σ, on a :

$$V'^T X X^T V' = 0$$

donc $\mathrm{rg}(V'^T X X^T V') = 0$ or $\mathrm{rg}(V'^T X X^T V') = \mathrm{rg}(V'^T X)$ et par conséquent :

$$V'^T X = 0 \tag{A.6}$$

Finalement on calcule :

$$\begin{aligned} V\Sigma W^T &= V\Sigma\Sigma^{-1}V^T X \\ &= VV^T X \\ &= (\mathbb{I}_n - V'V'^T)X && \text{d'après (Éq. (A.5))} \\ &= X - V'V'^T X \\ &= X && \text{d'après (Éq. (A.6))} \end{aligned}$$

D'autre part $\Sigma \in \mathbb{R}^{r\times r}$ et on montre facilement que $W^TW = \mathbb{I}_r$ donc cela correspond bien à une décomposition SVD. □

Proposition 17. *Soit $X \in \mathbb{R}^{n\times m_x}$ et $Y \in \mathbb{R}^{n\times m_y}$ tels que $XX^T = YY^T$. Si*

$$X = V\Sigma W_x^T$$

est une décomposition SVD de X alors une décomposition SVD de Y est donnée par

$$Y = V\Sigma W_y^T$$

avec $W_y^T = \Sigma^{-1}U^TY$.

La démonstration de la proposition 17 découle directement de la proposition 16.

Proposition 18. *Soient $A \in \mathbb{R}^{n\times m}$, $P \in \mathbb{R}^{n\times n}$ une matrice de sélection de lignes et $V \in \mathbb{R}^{n\times r}$ une matrice dont les colonnes sont orthogonales. On a :*

$$\left\|P^TA\right\| \leq \sigma_{\max}\left(P^TV\right)\|A\| + \left\|(\mathbb{I} - VV^T)A\right\|$$

Démonstration.

$$\begin{aligned}
\left\|P^TA\right\| &= \left\|PP^TA\right\| && \text{d'après (Éq. (A.2))}\\
&\leq \left\|PP^TVV^TA + PP^T(\mathbb{I} - VV^T)A\right\| &&\\
&\leq \left\|PP^TVV^TA\right\| + \left\|PP^T(\mathbb{I} - VV^T)A\right\| && \text{inégalité triangulaire}\\
&\leq \left\|PP^TVV^T\right\|_2\|A\| + \left\|PP^T\right\|_2\left\|(\mathbb{I} - VV^T)A\right\| && \text{d'après (Éq. (A.1))}\\
&\leq \sigma_{\max}\left(P^TV\right)\|A\| + \left\|(\mathbb{I} - VV^T)A\right\| && \text{car } \left\|PP^T\right\|_2 = 1
\end{aligned}$$

□

Lemme 1. *Soit $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times\cdots\times n_p\times r}$ et $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{r\times m_1\times\cdots\times m_q}$, alors :*

$$\|\boldsymbol{\mathcal{A}} \bullet \boldsymbol{\mathcal{B}}\| \leq \left\|\langle\boldsymbol{\mathcal{A}}\rangle_p\right\|_2\|\boldsymbol{\mathcal{B}}\|$$

Démonstration. D'après la définition du déploiement, on a :

$$\langle\boldsymbol{\mathcal{A}} \bullet \boldsymbol{\mathcal{B}}\rangle_p = \langle\boldsymbol{\mathcal{A}}\rangle_p\langle\boldsymbol{\mathcal{B}}\rangle_1$$

Par conséquent :

$$\begin{aligned}
\|\boldsymbol{\mathcal{A}} \bullet \boldsymbol{\mathcal{B}}\| &= \left\| \langle \boldsymbol{\mathcal{A}} \bullet \boldsymbol{\mathcal{B}} \rangle_p \right\| \\
&= \left\| \langle \boldsymbol{\mathcal{A}} \rangle_p \langle \boldsymbol{\mathcal{B}} \rangle_1 \right\| \\
&\leq \left\| \langle \boldsymbol{\mathcal{A}} \rangle_p \right\|_2 \| \langle \boldsymbol{\mathcal{B}} \rangle_1 \| && \text{d'après (Éq. (A.1))} \\
&\leq \left\| \langle \boldsymbol{\mathcal{A}} \rangle_p \right\|_2 \| \boldsymbol{\mathcal{B}} \|
\end{aligned}$$

□

Annexe B

Démonstrations des propositions

Lemme 2. *Avec les notations de l'algorithme 10, on a pour tout* $k \in [\![1, d-1]\!]$ *:*

$$\boldsymbol{\mathcal{A}}^{(k)} = \boldsymbol{\mathcal{H}}_k \bullet \boldsymbol{\mathcal{A}}^{(k+1)} + \boldsymbol{\mathcal{E}}_k \tag{B.1}$$

avec $\boldsymbol{\mathcal{A}}^{(k)}$ *donné en définition 5.*

Démonstration. En substituant l'approximation matricielle (Éq. (IV.14)) dans la définition de l'erreur (Éq. (IV.19)), on a pour tout $k \in [\![1, d-1]\!]$

$$A_k = H_k \widehat{A}_k + E_k$$

D'après les équations (IV.12),(IV.18),(IV.17) et (IV.20), l'équation se réécrit :

$$\left\langle \boldsymbol{\mathcal{A}}^{(k)} \right\rangle_2 = \langle \boldsymbol{\mathcal{H}}_k \rangle_2 \left\langle \boldsymbol{\mathcal{A}}^{(k+1)} \right\rangle_1 + \langle \boldsymbol{\mathcal{E}}_k \rangle_2$$

Et finalement comme $\boldsymbol{\mathcal{H}}_k$ est d'ordre 3, l'équation se réécrit avec l'opérateur de contraction $\bullet$:

$$\boldsymbol{\mathcal{A}}^{(k)} = \boldsymbol{\mathcal{H}}_k \bullet \boldsymbol{\mathcal{A}}^{(k+1)} + \boldsymbol{\mathcal{E}}_k$$

□

Proposition 19. *Soit une matrice* $A \in \mathbb{R}^{n \times m}$*, un ensemble de* r *lignes* $\mathcal{I}$ *et un ensemble de* r *colonnes* $\mathcal{J}$*. Si* $\widetilde{A}$ *est une matrice de rang* r *interpolant* A *en* $\mathcal{I}$ *et* $\mathcal{J}$*, c'est-à-dire telle que :*

$$\widetilde{A}(\mathcal{I}, :) = A(\mathcal{I}, :)$$
$$\widetilde{A}(:, \mathcal{J}) = A(:, \mathcal{J})$$

et si $A(\mathcal{I}, \mathcal{J})$ *est inversible alors* $\widetilde{A}$ *correspond à la décomposition PSD (Éq. (V.6)).*

Démonstration de la proposition 19. La matrice $\widetilde{A}$ est de rang r donc il existe $B \in \mathbb{R}^{n\times r}$ et $C \in \mathbb{R}^{r\times m}$ telles que :

$$\widetilde{A} = BC$$

Donc d'une part :

$$\widetilde{A}(\mathcal{I}, \mathcal{J}) = B(\mathcal{I}, :)C(:, \mathcal{J})$$

or comme $A(\mathcal{I}, \mathcal{J})$ est inversible, alors $B(\mathcal{I}, :)$ et $C(:, \mathcal{J})$ le sont aussi.

D'autre part, les propriétés d'interpolation se réécrivent :

$$B(\mathcal{I}, :)C = A(\mathcal{I}, :)$$
$$BC(:, \mathcal{J}) = A(:, \mathcal{J})$$

Par conséquent :

$$C = [B(\mathcal{I}, :)]^{-1} A(\mathcal{I}, :)$$
$$B = A(:, \mathcal{J}) [C(:, \mathcal{J})]^{-1}$$

Ce qui implique que :

$$\begin{aligned}\widetilde{A} &= A(:, \mathcal{J}) [C(:, \mathcal{J})]^{-1} [B(\mathcal{I}, :)]^{-1} A(\mathcal{I}, :) \\ &= A(:, \mathcal{J}) [B(\mathcal{I}, :)C(:, \mathcal{J})]^{-1} A(\mathcal{I}, :) \\ &= A(:, \mathcal{J}) \left[\widetilde{A}(\mathcal{I}, \mathcal{J})\right]^{-1} A(\mathcal{I}, :) \\ &= A(:, \mathcal{J}) [A(\mathcal{I}, \mathcal{J})]^{-1} A(\mathcal{I}, :)\end{aligned}$$

□

Proposition 20. *Soit $\mathbb{P}_{\text{gap}} \in \mathbb{R}^{n\times n}$ une matrice de projection gappy associée à la matrice de base $V \in \mathbb{R}^{n\times r}$ dont les colonnes sont orthogonales et à la matrice de sélection $P \in \mathbb{R}^{n\times n}$. La norme de Frobenius et la norme 2 de la matrice d'écart $\mathbb{E} = \mathbb{P}_{\text{gap}} - \mathbb{P}_{\text{ref}}$*

sont données par :

$$\|\mathbb{E}\| = \sqrt{\sum_{k=1}^{n} \frac{1}{\lambda_k(C)} - n}$$
$$\|\mathbb{E}\|_2 = \sqrt{\frac{1}{\lambda_{\min}(C)} - 1}$$

où

$$C = V^T P P^T V$$

et les λ correspondent aux valeurs propres de la matrice C.

Démonstration.

$$\begin{aligned}
\|\mathbb{E}\|_2^2 &= \left\| V \left[C^{-1} V^T P P^T - V^T \right] \right\|_2^2 \\
&= \lambda_{\max} \left(\left[C^{-1} V^T P P^T - V^T \right]^T V^T V \left[C^{-1} V^T P P^T - V^T \right] \right) \\
&= \lambda_{\max} \left(\left[P P^T V C^{-1} - V \right] \left[C^{-1} V^T P P^T - V^T \right] \right) \\
&= \lambda_{\max} \left(\left[C^{-1} V^T P P^T - V^T \right] \left[P P^T V C^{-1} - V \right] \right) \\
&= \lambda_{\max} \left(C^{-1} C C^{-1} - C^{-1} C - C C^{-1} + I \right) \\
&= \lambda_{\max} \left(C^{-1} - I \right) \\
&= \lambda_{\max} \left(C^{-1} \right) - 1 \\
&= \frac{1}{\lambda_{\min}(C)} - 1
\end{aligned}$$

$$\begin{aligned}
\|\mathbb{E}\|^2 &= \left\| V \left[C^{-1} V^T P P^T - V^T \right] \right\|^2 \\
&= \operatorname{tr} \left(\left[C^{-1} V^T P P^T - V^T \right]^T V^T V \left[C^{-1} V^T P P^T - V^T \right] \right) \\
&= \operatorname{tr} \left(\left[P P^T V C^{-1} - V \right] \left[C^{-1} V^T P P^T - V^T \right] \right) \\
&= \operatorname{tr} \left(\left[C^{-1} V^T P P^T - V^T \right] \left[P P^T V C^{-1} - V \right] \right) \\
&= \operatorname{tr} \left(C^{-1} C C^{-1} - C^{-1} C - C C^{-1} + I \right) \\
&= \operatorname{tr} \left(C^{-1} - I \right) \\
&= \sum_{k=1}^{n} \lambda_k \left(C^{-1} - I \right) \\
&= \sum_{k=1}^{n} \frac{1}{\lambda_k(C)} - n
\end{aligned}$$

□

Proposition 21. *Soit une base réduite $V \in \mathbb{R}^{n\times r}$ dont les colonnes sont orthogonales et des sélections d'indices de lignes $\mathcal{I}_1$ et $\mathcal{I}_2$ associées aux matrices de sélection P_1 et P_2 telle que :*

$$\mathcal{I}_2 \subset \mathcal{I}_1 \tag{B.2}$$

On a pour tout $k \in [\![1, n]\!]$

$$\lambda_k(V^T P_1 P_1^T V) \geq \lambda_k(V^T P_2 P_2^T V)$$

où les λ_k désignent les valeurs propres des matrices en argument.

Démonstration. Soit $Y \in \mathbb{R}^{n\times n}$ la matrice de sélection associée aux lignes $\mathcal{I}_1 \setminus \mathcal{I}_2$. On a :

$$P_1 P_1^T = P_2 P_2^T + YY^T$$

Et par conséquent :

$$V^T P_1 P_1^T V = V^T P_2 P_2^T V + V^T Y^T Y V$$

Comme $V^T Y^T Y V$ est symétrique, semi-définie positive, on a :

$$\forall\, x \in \mathbb{R}^n,\ x^T \left(V^T P_1 P_1^T V\right) x \geq x^T \left(V^T P_2 P_2^T V\right) x$$

La fin de la démonstration consiste à montrer que pour toute matrice symétrique A et B le résultat suivant est vérifié :

$$\forall\, x \in \mathbb{R}^n,\ x^T A x \geq x^T B x \implies \forall\, k,\ \lambda_k(A) \geq \lambda_k(B)$$

La relation précédente est une conséquence du théorème min-max de Courant-Fisher qui montre que les valeurs propres d'une matrice symétrique $M \in \mathbb{R}^{n\times n}$ vérifient :

$$\lambda_k(M) = \max_{U \subset \mathbb{R}^n, \dim(U)=k} \left(\min_{x \in U, \|x\|_2 = 1} x^T M x \right) \tag{B.3}$$

Cela implique que pour k fixé, il existe un sous-espace U_k de dimension k tel que

$$\lambda_k(B) = \min_{x \in U_k, \|x\|=1} x^T B x$$

ce qui implique finalement que

$$\lambda_k(A) = \max_{U \subset \mathbb{R}^n, \dim(U)=k} \left(\min_{x \in U, \|x\|_2=1} x^T A x \right) \geq \min_{x \in U_k, \|x\|=1} x^T A x$$

et

$$\min_{x \in U_k, \|x\|=1} x^T A x \geq \min_{x \in U_k, \|x\|=1} x^T B x = \lambda_k(B)$$

□

Proposition 2. *Soit f une fonction définie par l'équation (IV.1) sur $\mathcal{D}$ dont les valeurs sur le domaine discrétisé $D = D_1 \times \cdots \times D_d$ définissent un tenseur $\boldsymbol{\mathcal{A}}$ par l'équation (IV.5).*

L'interpolation multilinéaire par morceaux $\widetilde{f}$ de f définie sur le domaine continu $\mathcal{D}$ est égale à la décomposition en train de tenseurs continue :

$$\widetilde{f}(x_1, \ldots, x_d) = \widetilde{f}^{(1)}(x_1) \ldots \widetilde{f}^{(d)}(x_d) \tag{B.4}$$

où pour tout $k \in [\![1, d]\!]$, les fonctions à valeurs matricielles $\widetilde{f}^{(k)}$ sont les interpolations linéaires par morceaux des fonctions $f^{(k)}$, i.e :

$$\widetilde{f}^{(k)}(x_k) = w_+(x_k) G_k\left(i_k^{-1}\right) + w_-(x_k) G_k\left(i_k^{+1}\right)$$

avec w_+, w_-, $\bar{x}_k^{-1}$ et $\bar{x}_k^{+1}$ définis comme dans (Éq. (IV.9)) et i_k^{-1}, i_k^{+1} tels que :

$$\bar{x}_k^{-1} = x_k^{(i_k^{-1})} \in D_k \quad et \quad \bar{x}_k^{+1} = x_k^{(i_k^{+1})} \in D_k$$

Démonstration de la proposition 2. D'après l'équation (IV.8) et avec les notations corres-

pondantes, on a :

$$\begin{aligned}
\widetilde{f}(x_1,\dots,x_d) &= \sum_{\eta_1,\dots,\eta_d=\pm 1}\left[f\left(\bar{x}_1^{-\eta_1},\dots,\bar{x}_d^{-\eta_d}\right)\prod_{k=1}^{d} w_{\eta_k}(x_k)\right] \\
&= \sum_{\eta_1,\dots,\eta_d=\pm 1}\left[\boldsymbol{\mathcal{A}}\left(i_1^{\eta_1},\dots,i_d^{\eta_d}\right)\prod_{k=1}^{d} w_{\eta_k}(x_k)\right] \qquad \text{d'après (Éq. (IV.5))} \\
&= \sum_{\eta_1,\dots,\eta_d=\pm 1}\left[\prod_{l=1}^{d} G_l\left(i_l^{\eta_l}\right)\prod_{k=1}^{d} w_{\eta_k}(x_k)\right] \qquad \text{d'après (Éq. (III.26))} \\
&= \sum_{\eta_1,\dots,\eta_d=\pm 1}\left[\left(w_{\eta_1}(\mu_1)G_1\left(i_1^{\eta_1}\right)\right)\dots\left(w_{\eta_d}(\mu_d)G_d\left(i_d^{\eta_d}\right)\right)\right] \\
&= \left(\sum_{\eta_1=\pm 1} w_{\eta_1}(\mu_1)G_1\left(i_1^{\eta_1}\right)\right)\dots\left(\sum_{\eta_d=\pm 1} w_{\eta_d}(\mu_d)G_d\left(i_d^{\eta_d}\right)\right) \\
&= \widetilde{f}^{(1)}(x_1)\dots\widetilde{f}^{(d)}(x_d)
\end{aligned}$$

□

Proposition 9. *On considère une projection gappy $\mathbb{P}_{\text{gap}} \in \mathbb{R}^{n\times n}$ associée à la matrice de base $V \in \mathbb{R}^{n\times r}$ et une matrice de sélection de lignes $P \in \mathbb{R}^{n\times n}$. Pour une matrice $A \in \mathbb{R}^{n\times m}$, l'erreur d'approximation gappy est définie par :*

$$E_{\text{gap}} = A - \mathbb{P}_{\text{gap}} A$$

La norme de l'erreur d'approximation gappy vérifie les relations suivantes :

$$\|E_{\text{gap}}\|^2 = \|(\mathbb{I} - \mathbb{P}_{\text{ref}})\,A\|^2 + \|\mathbb{E}_{\text{gap}} A\|^2 \tag{B.5}$$

$$\|E_{\text{gap}}\| \leq \frac{1}{\sigma_{\min}(P^T V)}\,\|A - \mathbb{P}_{\text{ref}} A\| \tag{B.6}$$

où $\mathbb{E}_{\text{gap}}$ est la matrice d'écart dû à la projection gappy.

Démonstration de la proposition 9. On commence par montrer que :

$$\begin{aligned}
\mathbb{P}_{\text{gap}}\mathbb{P}_{\text{ref}} &= V\left[P^T V\right]^{\dagger} P^T V V^T \\
&= V V^T \qquad \text{d'après la proposition 8} \\
&= \mathbb{P}_{\text{ref}} \qquad (\text{B.7}) \\
& \qquad (\text{B.8})
\end{aligned}$$

On a :

$$\begin{aligned} E_{\text{gap}} &= A - \mathbb{P}_{\text{gap}} A \\ &= (A - \mathbb{P}_{\text{ref}} A) + (\mathbb{P}_{\text{ref}} - \mathbb{P}_{\text{gap}})\, A \\ &= (A - \mathbb{P}_{\text{ref}} A) + \mathbb{E}_{\text{gap}} A \\ &= (\mathbb{I} - \mathbb{P}_{\text{ref}})\, A + \mathbb{E}_{\text{gap}} A \end{aligned}$$

Et d'autre part comme $\mathbb{P}_{\text{ref}}$ est orthogonale et d'après l'équation (B.7) on a :

$$\begin{aligned} (\mathbb{I} - \mathbb{P}_{\text{ref}})^T\, \mathbb{E}_{\text{gap}} &= (\mathbb{I} - \mathbb{P}_{\text{ref}})^T\, (\mathbb{P}_{\text{ref}} - \mathbb{P}_{\text{gap}}) \\ &= \mathbb{P}_{\text{ref}} - \mathbb{P}_{\text{gap}} - \mathbb{P}_{\text{ref}}^T \mathbb{P}_{\text{ref}} + \mathbb{P}_{\text{ref}}^T \mathbb{P}_{\text{gap}} \\ &= \mathbb{P}_{\text{ref}} - \mathbb{P}_{\text{gap}} - \mathbb{P}_{\text{ref}} + \mathbb{P}_{\text{gap}} \\ &= 0 \end{aligned}$$

Finalement les matrices $(\mathbb{I} - \mathbb{P}_{\text{ref}})$ et $\mathbb{E}_{\text{gap}}$ ont pour images respectives des espaces mutuellement orthogonaux ce qui prouve l'équation (B.5).

On a aussi :

$$\begin{aligned} E_{\text{gap}} &= [\mathbb{I} - \mathbb{P}_{\text{gap}}]\, A \\ &= [(\mathbb{I} - \mathbb{P}_{\text{ref}}) - (\mathbb{P}_{\text{gap}} - \mathbb{P}_{\text{ref}})]\, A \\ &= [(\mathbb{I} - \mathbb{P}_{\text{ref}}) - (\mathbb{P}_{\text{gap}} - \mathbb{P}_{\text{gap}} \mathbb{P}_{\text{ref}})]\, A \\ &= (\mathbb{I} - \mathbb{P}_{\text{gap}})\, (\mathbb{I} - \mathbb{P}_{\text{ref}})\, A \end{aligned}$$

D'autre part :

$$\begin{aligned} \left\| V \left[P^T V \right]^\dagger \right\|_2 &= \left\| \left[P^T V \right]^\dagger \right\|_2 && \text{d'après (Éq. (A.2))} \\ &= \frac{1}{\sigma_{\min}\left(P^T V \right)} && \text{comme } P^T V \text{ est de rang plein} \end{aligned} \tag{B.9}$$

De plus :

$$\begin{aligned} \|\mathbb{I} - \mathbb{P}_{\text{gap}}\|_2 &= \|\mathbb{P}_{\text{gap}}\|_2 && \text{si } \mathbb{P} \neq 0 \text{ et } \mathbb{P} \neq \mathbb{I} \text{ d'après [116]} \\ &= \left\| V \left[P^T V \right]^\dagger P^T \right\|_2 \\ &\leq \left\| V \left[P^T V \right]^\dagger \right\|_2 && \text{comme } P^T \text{ possède des colonnes orthogonales} \\ &\leq \frac{1}{\sigma_{\min}\left(P^T V \right)} && \text{d'après (Éq. (B.9))} \end{aligned} \tag{B.10}$$

Finalement :

$$\begin{aligned}
\|E_{\text{gap}}\| &= \|(\mathbb{I} - \mathbb{P}_{\text{gap}})\,(\mathbb{I} - \mathbb{P}_{\text{ref}})\,A\| \\
&\leq \|\mathbb{I} - \mathbb{P}_{\text{gap}}\|_2\,\|(\mathbb{I} - \mathbb{P}_{\text{ref}})\,A\| && \text{d'après (Éq. (A.1))} \\
&\leq \frac{1}{\sigma_{\min}\left(P^T V\right)}\,\|(\mathbb{I} - \mathbb{P}_{\text{ref}})\,A\| && \text{d'après (Éq. (B.10))} \qquad \text{(B.11)}
\end{aligned}$$

□

Proposition 13. *Soient deux matrices $U \in \mathbb{R}^{n\times r}$ et $C \in \mathbb{R}^{n\times s}$ de rang plein telles que $r \leq s \leq n$. On a l'équivalence entre les deux propositions suivantes :*

- $C^T U \in \mathbb{R}^{s\times r}$ *est de rang* r (B.12)
- $\mathbb{R}^n = \text{Im}(U) \oplus \mathcal{N}(U^T C C^T)$ *(décomposition de* $\mathbb{R}^n$ *en somme directe)* (B.13)

Et la projection associée à cette décomposition en somme directe est donnée par :

$$\mathbb{P} = U[C^T U]^\dagger C^T \qquad \text{(B.14)}$$

Démonstration de la proposition 13.

Preuve de (Éq. (V.27)) ⇒ (Éq. (V.28))

D'une part :

$$\begin{aligned}
\text{rg}(CC^T U) &\leq \min\Big(\text{rg}(C), \text{rg}(C^T U)\Big) \\
&\leq \min(s, r) \\
&\leq r \qquad \text{(B.15)}
\end{aligned}$$

et d'autre part d'après l'inégalité de Sylvester :

$$\begin{aligned}
&(s - \text{rg}(C)) + (s - \text{rg}(C^T U)) \geq (s - \text{rg}(CC^T U)) \\
\Rightarrow\quad &(s - s) + (s - r) \geq (s - \text{rg}(CC^T U)) \\
\Rightarrow\quad &\text{rg}(CC^T U) \geq r \\
\Rightarrow\quad &\text{rg}(CC^T U) = r \qquad \text{d'après (Éq. (B.15))} \qquad \text{(B.16)}
\end{aligned}$$

Donc :

$$\begin{aligned}\dim\left(\mathcal{N}(U^TCC^T)\right) &= \dim\left(\mathrm{Im}(CC^TU)^\perp\right)\\ &= n - \dim\left(\mathrm{Im}(CC^TU)\right)\\ &= n - \mathrm{rg}(CC^TU)\\ &= n - r \qquad \text{d'après (Éq. (B.16))} \end{aligned} \tag{B.17}$$

De plus :

$$\dim\left(\mathrm{Im}(U)\right) = \mathrm{rg}(U) = r \tag{B.18}$$

D'autre part :

$$\mathrm{rg}(U^TCC^TU) = \mathrm{rg}(C^TU) = r$$

Donc $U^TCC^TU \in \mathbb{R}^{r\times r}$ est inversible.
Soit $x \in \mathrm{Im}(U) \cap \mathcal{N}(U^TCC^T) \subset \mathbb{R}^n$.
$x \in \mathrm{Im}(U)$ donc :

$$\exists y \in \mathbb{R}^r, \quad x = Uy$$

Or $x \in \mathcal{N}(U^TCC^T)$ donc :

$$\begin{aligned} & U^TCC^Tx = 0\\ & \Rightarrow U^TCC^TUy = 0\\ & \Rightarrow y = 0 \qquad\qquad \text{car } U^TCC^TU \text{ inversible}\\ & \Rightarrow x = 0 \end{aligned}$$

Donc $\forall\, x \in \mathrm{Im}(U) \cap \mathcal{N}(U^TCC^T) \quad \Rightarrow x \in 0$.
Pour conclure, on a :

- $\dim\left(\mathcal{N}(U^TCC^T)\right) = n - r$
- $\dim\left(\mathrm{Im}(U)\right) = r$
- $\mathrm{Im}(U) \cap \mathcal{N}(U^TCC^T) = \emptyset$

Donc $\mathbb{R}^n = \mathrm{Im}(U) \oplus \mathcal{N}(U^TCC^T)$

Preuve de (Éq. (V.28)) $\Rightarrow$ (Éq. (V.27))

Soit $y \in \mathbb{R}^r$ tel que $y \neq 0$ alors :

$$\begin{aligned} & Uy \in \mathrm{Im}(U) \\ & \Rightarrow Uy \notin \mathcal{N}(U^T C C^T) \\ & \Rightarrow U^T C C^T U y \neq 0 \end{aligned}$$

Donc $\forall\, y \in \mathbb{R}^r, \left[y = 0 \Leftrightarrow U^T C C^T U y = 0\right]$. Donc $U^T C C^T U \in \mathbb{R}^{r \times r}$ est inversible donc de rang r. Or $\mathrm{rg}(U^T C C^T U) = \mathrm{rg}(C^T U)$ Donc finalement :

$$\mathrm{rg}(C^T U) = r$$

Preuve de la forme de projection (Éq. (V.29))

On a $C^T U \in \mathbb{R}^{s \times r}$ de rang plein donc :

$$\left[C^T U\right]^\dagger = \left[U^T C C^T U\right]^{-1} U^T C \tag{B.19}$$

Soit $\widetilde{\mathbb{P}}$ la projection associée à la décomposition (Éq. (V.28)). Soit $x \in \mathbb{R}^n$:

$$\exists!(u, v) \in \mathrm{Im}(U) \times \mathcal{N}(U^T C C^T), \quad x = u + v$$

Et par définition $\widetilde{\mathbb{P}} x = u$.

Comme $u \in \mathrm{Im}\, U, \quad \exists y \in \mathbb{R}^r, \quad u = Uy$ D'autre part :

$$\begin{aligned} \mathbb{P}x &= U[C^T U]^\dagger C^T x \\ &= U[C^T U]^\dagger C^T (u + v) \\ &= U\left[U^T C C^T U\right]^{-1} U^T C C^T (u + v) && \text{d'après (Éq. (B.19))} \\ &= U\left[U^T C C^T U\right]^{-1} U^T C C^T u && \text{car } v \in \mathcal{N}(U^T C C^T) \\ &= U\left[U^T C C^T U\right]^{-1} U^T C C^T U y \\ &= U y \\ &= u \end{aligned}$$

Finalement $\mathbb{P} = \widetilde{\mathbb{P}}$ □

Remarque 18. *Dans la proposition 13, les matrices U et C peuvent être de tailles*

différentes ($r \neq s$). Cependant lorsque les propositions sont vérifiées et que $r = s$, on a :

$$\mathcal{N}(\mathbb{P}) = \mathcal{N}(U^T C C^T) = \mathcal{N}(C^T)$$

et la proposition 13 correspond à un résultat d'algèbre plus classique.

Démonstration remarque 18. On a $\text{rg}(U^T C) = \text{rg}(C^T U) = r$. Donc $U^T C \in \mathbb{R}^{r \times r}$ inversible.

Soit $x \in \mathbb{R}^n$ tel que $x \neq 0$:

$$\begin{aligned} x \in \mathcal{N}(C^T) &\Leftrightarrow C^T x = 0 \\ &\Leftrightarrow U^T C C^T x = 0 && \text{car } U^T C \text{ inversible} \\ &\Leftrightarrow x \in \mathcal{N}(U^T C C^T) \end{aligned}$$

□

Références

[1] Amine Ammar, Bechir Mokdad, Francisco Chinesta, and Roland Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. Journal of Non-Newtonian Fluid Mechanics, 139(3) :153 – 176, 2006.

[2] Amine Ammar, Bechir Mokdad, Francisco Chinesta, and Roland Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids : Part II : Transient simulation using space-time separated representations. Journal of Non-Newtonian Fluid Mechanics, 144(2) :98–121, 2007.

[3] Hiromasa Arai, Crystal Maung, and Haim Schweitzer. Optimal column subset selection by A-star search. In AAAI, pages 1079–1085, 2015.

[4] Patricia Astrid, Siep Weiland, Karen Willcox, and Ton Backx. Missing point estimation in models described by proper orthogonal decomposition. IEEE Transactions on Automatic Control, 53(10) :2237–2251, 2008.

[5] Jonas Ballani and Lars Grasedyck. Hierarchical tensor approximation of output quantities of parameter-dependent pdes. SIAM/ASA Journal on Uncertainty Quantification, 3(1) :852–872, 2015.

[6] Jonas Ballani, Lars Grasedyck, and Melanie Kluge. Black box approximation of tensors in hierarchical tucker format. Linear Algebra and its Applications, 438(2) :639–657, 2013.

[7] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. An empirical interpolation method : application to efficient reduced-basis discretization of partial differential equations. Comptes-Rendus Mathématiques, 339(9) :667–672, 2004.

[8] Mario Bebendorf. Approximation of boundary element matrices. Numerische Mathematik, 86(4) :565–589, 2000.

[9] Mario Bebendorf. Adaptive cross approximation of multivariate functions. Constructive Approximation, 34(2) :149–179, 2011.

[10] Mario Bebendorf, Yvon Maday, and Benjamin Stamm. Comparison of some reduced representation approximations, pages 67–100. Springer International Publishing, Cham, 2014.

[11] Peter Benner, Volker Mehrmann, and Danny C. Sorensen. Dimension reduction of large-scale systems, volume 45. Springer, 2005.

[12] Michael W. Berry, Shakhina A. Pulatova, and G.W. Stewart. Algorithm 844 : Computing sparse reduced-rank approximations to sparse matrices. ACM Transactions on Mathematical Software (TOMS), 31(2) :252–269, 2005.

[13] Jacques Besson, Georges Cailletaud, Jean-Louis Chaboche, and Samuel Forest. Non-linear mechanics of materials, volume 167. Springer Netherlands, 1 edition, 2010.

[14] Daniele Bigoni, Allan P. Engsig-Karup, and Youssef M. Marzouk. Spectral tensor-train decomposition. SIAM Journal on Scientific Computing, 38(4) :A2405–A2439, 2016.

[15] F. Bonizzoni, F. Nobile, and D. Kressner. Tensor train approximation of moment equations for elliptic equations with lognormal coefficient. Computer Methods in Applied Mechanics and Engineering, 308(Supplement C) :349 – 376, 2016.

[16] Christos Boutsidis and David P. Woodruff. Optimal CUR matrix decompositions. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC '14, pages 353–362, New York, NY, USA, 2014. ACM.

[17] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. Computer Vision—ECCV 2002, pages 707–720, 2002.

[18] Jacob C. Bridgeman and Christopher T. Chubb. Hand-waving and interpretive dance : An introductory course on tensor networks. arXiv preprint arXiv :1603.03039, 2016.

[19] Peter Businger and Gene H. Golub. Linear least squares solutions by Householder transformations. Numerische Mathematik, 7(3) :269–276, 1965.

[20] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations. International Journal for Numerical Methods in Engineering, 86(2) :155–181, 2011.

[21] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction : Effective implementation and application to computational fluid dynamics and turbulent flows. Journal of Computational Physics, 242 :623–647, 2013.

[22] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. Psychometrika, 35(3) :283–319, 1970.

[23] Fabien Casenave, Alexandre Ern, and Tony Lelièvre. Variants of the empirical interpolation method : Symmetric formulation, choice of norms and rectangular extension. Applied Mathematics Letters, 56 :23–28, 2016.

[24] Marco Cavazzuti. Optimization methods : from theory to design scientific and technological aspects in mechanics. Springer-Verlag Berlin Heidelberg, 1 edition, 2013.

[25] Ali Çivril and Malik Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. Theoretical Computer Science, 410(47) :4801–4811, 2009.

[26] Saifon Chaturantabut and Danny C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. SIAM Journal on Scientific Computing, 32(5) :2737–2764, 2010.

[27] Francisco Chinesta, Amine Ammar, François Lemarchand, Pierre Beauchene, and Fabrice Boust. Alleviating mesh constraints : Model reduction, parallel time integration and high resolution homogenization. Computer methods in applied mechanics and engineering, 197(5) :400–413, 2008.

[28] Francisco Chinesta, Adrien Leygue, Felipe Bordeu, Jose Vicente Aguado, Elias Cueto, David Gonzalez, Iciar Alfaro, Amine Ammar, and Antonio Huerta. PGD-based computational vademecum for efficient design, optimization and control. Archives of Computational Methods in Engineering, 20(1) :31–59, 2013.

[29] Andrzej Cichocki. Tensor networks for big data analytics and large-scale optimization problems. arXiv preprint arXiv :1407.3124, 2014.

[30] Jonathan Cormier and Georges Cailletaud. Constitutive modeling of the creep behavior of single crystal superalloys under non-isothermal conditions inducing phase transformations. Materials Science and Engineering : A, 527(23) :6300 – 6312, 2010.

[31] Eduardo Corona, Abtin Rahimian, and Denis Zorin. A tensor-train accelerated solver for integral equations in complex geometries. Journal of Computational Physics, 334 :145 – 169, 2017.

[32] Cécile Daversin and Christophe Prud'homme. Simultaneous empirical interpolation and reduced basis method for non-linear problems. Comptes Rendus Mathematique, 353(12) :1105 – 1109, 2015.

[33] J. Delvillet, L. Ukeiley, L. Cordier, J.P Bonnet, and M. Glauser. Examination of large-scale structures in a turbulent plane mixing layer. Part 1. proper orthogonal decomposition. Journal of Fluid MechaMecnics, 391 :91–122, 1999.

[34] Rodrigue Desmorat, Adriana Mattiello, and Jonathan Cormier. A tensorial thermodynamic framework to account for the gamma-prime rafting in nickel-based single crystal superalloys. International Journal of Plasticity, 95(Supplement C) :43 – 81, 2017.

[35] Alireza Doostan, AbdoulAhad Validi, and Gianluca Iaccarino. Non-intrusive low-rank separated approximation of high-dimensional stochastic models. Computer Methods in Applied Mechanics and Engineering, 263 :42–55, 2013.

[36] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices III : Computing a compressed approximate matrix decomposition. SIAM Journal on Computing, 36(1) :184–206, 2006.

[37] Zlatko Drmač and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. SIAM Journal on Scientific Computing, 38(2) :A631–A648, 2016.

[38] Michael S. Eldred and Daniel M. Dunlavy. Formulations for surrogate-based optimization with data fit, multi-fidelity, and reduced-order models. In Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, number AIAA-2006-7117, Portsmouth, VA, 2006.

[39] Richard Everson and Lawrence Sirovich. Karhunen-Loève procedure for gappy data. Journal of the Optical Society of America A, 12 :1657–1664, August 1995.

[40] Alexander Forrester, Andras Sobester, and Andy Keane. Engineering design via surrogate modelling : a practical guide. John Wiley & Sons, 2008.

[41] Roger Ghanem, David Higdon, and Houman Owhadi. Handbook of uncertainty quantification. Springer, 2017.

[42] Julien Ghighi. Mechanical behavior and creep life of crystal superalloys : crystal anisotropy and microstructure evolutions. Theses, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers, April 2013.

[43] Julien Ghighi, Jonathan Cormier, Elisabeth Ostoja-Kuczynski, José Mendez, George Cailletaud, and Farida Azzouz. A microstructure sensitive approach for the prediction of the creep behaviour and life under complex loading paths. Technische Mechanik, 32(2-5) :205–220, 2012.

[44] Loïc Giraldi. Contributions aux méthodes de calcul basées sur l'approximation de tenseurs et applications en mécanique numérique. PhD thesis, Ecole Centrale de Nantes, 2012.

[45] Rémi Giraud. Influence de l'histoire thermique sur les propriétés mécaniques à haute et très haute température du superalliage monocristallin CMSX-4. PhD thesis, Chasseneuil-du-Poitou, Ecole nationale supérieure de mécanique et d'aérotechnique, 2014.

[46] Gene H. Golub and Charles F. Van Loan. Matrix computations. The Johns Hopkins University Press, 4 edition, 2013.

[47] Sergei Goreinov, Ivan V. Oseledets, Dmitry V. Savostyanov, Eugene E. Tyrtyshnikov, and Nikolai Zamarashkin. How to find a good submatrix. Matrix Methods : Theory, Algorithms and Applications : Dedicated to the Memory of Gene Golub, page 247, 2010.

[48] Sergei Goreinov and Eugene E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. Contemporary Mathematics, 268 :47–51, 2001.

[49] Sergei Goreinov, Eugene E. Tyrtyshnikov, and Nikolai Zamarashkin. A theory of pseudoskeleton approximations. Linear Algebra and its Applications, 261(1) :1 – 21, 1997.

[50] Sergei Goreinov, Nikolai Zamarashkin, and Eugene E. Tyrtyshnikov. Pseudo-skeleton approximations by matrices of maximal volume. Mathematical Notes, 62(4) :515–519, 1997.

[51] Lars Grasedyck. Hierarchical singular value decomposition of tensors. SIAM Journal on Matrix Analysis and Applications, 31(4) :2029–2054, 2010.

[52] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen, 36(1) :53–78, 2013.

[53] Martin A. Grepl, Yvon Maday, Ngoc C. Nguyen, and Anthony T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. ESAIM : M2AN, 41(3) :575–605, 2007.

[54] Ming Gu and Stanley C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. SIAM Journal on Scientific Computing, 17(4) :848–869, 1996.

[55] Wolfgang Hackbusch. Tensor spaces and numerical tensor calculus, volume 42. Springer Science & Business Media, 2012.

[56] Wolfgang Hackbusch and S. Kühn. A new scheme for the tensor representation. Journal of Fourier Analysis and Applications, 15(5) :706–722, Oct 2009.

[57] John. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer. Math., 2(1) :84–90, December 1960.

[58] Richard A. Harshman. Foundations of the PARAFAC procedure : Models and conditions for an "explanatory" multi-modal factor analysis. UCLA Working Papers in Phonetics, 16 :1–84, 1970.

[59] Philip Holmes, John L. Lumley, Gahl Berkooz, and Clarence W. Rowley. Turbulence, coherent structures, dynamical systems and symmetry. Cambridge university press, 2012.

[60] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating

linear scheme for tensor optimization in the tensor train format. SIAM Journal on Scientific Computing, 34(2) :A683–A713, 2012.

[61] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed TT-rank. Numerische Mathematik, 120(4) :701–731, 2012.

[62] Johan Håstad. Tensor rank is NP-complete. Journal of Algorithms, 11(4) :644 – 654, 1990.

[63] Robert Hübener, Volckmar Nebendahl, and Wolfgang Dür. Concatenated tensor network states. New Journal of Physics, 12(2) :025004, 2010.

[64] Thomas Huckle, Konrad Waldherr, and Thomas Schulte-Herbrüggen. Computations in quantum tensor networks. Linear Algebra and its Applications, 438(2) :750–781, 2013.

[65] Ian T. Jolliffe. Principal component analysis and factor analysis. In Principal component analysis, pages 115–128. Springer, 1986.

[66] Kari Karhunen. Über lineare Methoden in der Wahrscheinlichkeitsrechnung, volume 37. Universitat Helsinki, 1947.

[67] Boris N. Khoromskij. Tensors-structured numerical methods in scientific computing : Survey on recent advances. Chemometrics and Intelligent Laboratory Systems, 110(1) :1–19, 2012.

[68] Boris N Khoromskij and I Oseledets. Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic pdes. Computational Methods in Applied Mathematics Comput. Methods Appl. Math., 10(4) :376–394, 2010.

[69] Mei Kobayashi, Georges Dupret, Oliver King, and Hikaru Samukawa. Estimation of singular values of very large matrices using random sampling. Computers & Mathematics with Applications, 42(10-11) :1331–1352, 2001.

[70] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. Siam review, 51(3) :455–500, 2009.

[71] Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. Low-rank tensor completion by riemannian optimization. BIT Numerical Mathematics, 54(2) :447–468, 2014.

[72] Petr Krysl, Sanjay Lall, and J. E. Marsden. Dimensional model reduction in non-linear finite element dynamics of solids and structures. International Journal for Numerical Methods in Engineering, 51(4) :479–504, 2001.

[73] Pierre Ladevèze. Sur une famille d'algorithmes en mécanique des structures. Comptes-rendus des séances de l'Académie des sciences. Série 2, Mécanique-physique, chimie, sciences de l'univers, sciences de la terre, 300(2) :41–44, 1985.

[74] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications, 21(4) :1253–1278, 2000.

[75] Jean-Briac le Graverend, Jonathan Cormier, Franck Gallerneau, P. Villechaise, S. Kruch, and J. Mendez. A microstructure-sensitive constitutive modeling of the inelastic behavior of single crystal nickel-based superalloys at very high temperature. International Journal of Plasticity, 59 :55–83, 2014.

[76] Jean Lemaitre and Jean-Louis Chaboche. Mechanics of solid materials. Cambridge University Press, 1994.

[77] Christiane Lemieux. Monte Carlo and Quasi-Monte Carlo sampling. Springer Series in Statistics, 01 2009.

[78] Lek-Heng Lim. Tensors and hypermatrices. Handbook of Linear Algebra, 2nd Ed., CRC Press, Boca Raton, FL, pages 231–260, 2013.

[79] Michel Loève. Probability theory : foundations, random sequences. van Nostrand Princeton, NJ, 1955.

[80] John Leask Lumley. The structure of inhomogeneous turbulent flows. Atmospheric Turbulence and Radio Wave Propagation (A. M. Yaglom and V.I. Tatarski, eds.), 1967.

[81] Y. Maday and O. Mula. A generalized empirical interpolation method : application of reduced basis techniques to data assimilation, volume 4, pages 221–235. Springer, Milan, 2013.

[82] Yvon Maday, Olga Mula, Anthony T. Patera, and Masayuki Yano. The generalized empirical interpolation method : Stability theory on hilbert spaces with an application to the Stokes equation. Computer Methods in Applied Mechanics and Engineering, 287 :310 – 334, 2015.

[83] Yvon Maday, Olga Mula, and Gabriel Turinici. A priori convergence of the Generalized Empirical Interpolation Method. In 10th international conference on Sampling Theory and Applications (SampTA 2013), pages 168–171, Bremen, Germany, Germany, July 2013.

[84] Yvon Maday, Ngoc Cuong Nguyen, Anthony T. Patera, and George S. H. Pau. A general multipurpose interpolation procedure : the magic points. Communications on Pure and Applied Analysis, 8(1) :383–404, 2009.

[85] Michael W. Mahoney. Randomized algorithms for matrices and data. Foundations and Trends® in Machine Learning, 3(2) :123–224, 2011.

[86] N. C. Nguyen, A. T. Patera, and J. Peraire. A 'best points' interpolation method for efficient approximation of parametrized functions. International Journal for Numerical Methods in Engineering, 73(4) :521–543, 2008.

[87] Anthony Nouy. Recent developments in spectral stochastic methods for the numerical solution of stochastic partial differential equations. Archives of Computational Methods in Engineering, 16(3) :251–285, 2009.

[88] Anthony Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. Computer Methods in Applied Mechanics and Engineering, 199(23–24) :1603 – 1626, 2010.

[89] Clément Olivier, David Ryckelynck, and Julien Cortial. Tensor-train approximation of parametric constitutive equations in elasto-viscoplasticity. Soumis pour publication dans le journal Computer Methods in Applied Mechanics and Engineering, April 2017.

[90] Clément Olivier, David Ryckelynck, Julien Cortial, and Christian Rey. Méthode de décomposition tensorielle pour la calibration de lois de comportement en sciences des matériaux. In Actes du 13ème colloque national en calcul des structures, May 2017.

[91] Ivan V. Oseledets. Compact matrix form of the d-dimensional tensor decomposition. Preprint, 1, 2009.

[92] Ivan V. Oseledets. Tensor-train decomposition. SIAM Journal on Scientific Computing, 33(5) :2295–2317, 2011.

[93] Ivan V. Oseledets and Eugene E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. Linear Algebra and its Applications, 432(1) :70–88, 2010.

[94] Roger Penrose. Applications of negative dimensional tensors. Combinatorial mathematics and its applications, 221244, 1971.

[95] Linda Petzold, Shengtai Li, Yang Cao, and Radu Serban. Sensitivity analysis of differential-algebraic equations and partial differential equations. Computers & Chemical Engineering, 30(10) :1553 – 1559, 2006. Papers form Chemical Process Control VII.

[96] Zu-Qing Qu. Model order reduction techniques with applications in finite element analysis. Springer Science & Business Media, 2013.

[97] Nicolas Relun, David Néron, and Pierre-Alain Boucard. A model reduction technique based on the pgd for elastic-viscoplastic computational analysis. Computational Mechanics, 51(1) :83–92, 2013.

[98] Gianluigi Rozza, D. B. Phuong Huynh, and Anthony T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations : Application to transport and continuum mechanics. Archives of Computational Methods in Engineering, 15(3) :229–275, 2008.

[99] David Ryckelynck. A priori hyperreduction method : an adaptive approach. Journal of Computational Physics, 202(1) :346–366, 2005.

[100] David Ryckelynck. Hyper reduction of mechanical models involving internal variables. International Journal for Numerical Methods in Engineering, 77(1) :75–89, 2009.

[101] David Ryckelynck, Djamel Missoum Benziane, Andrey Musienko, and Georges Cailletaud. Toward "green" mechanical simulations in materials science. European Journal of Computational Mechanics, 19(4) :365–388, 2010.

[102] David Ryckelynck, Francisco Chinesta, Elias Cueto, and Amine Ammar. On the "a priori" model reduction : Overview and recent developments. Archives of Computational Methods in Engineering, 13(1) :91–128, 2006.

[103] David Ryckelynck, Komlanvi Lampoh, and Stéphane Quilicy. Hyper-reduced predictions for lifetime assessment of elasto-plastic structures. Meccanica, 51(2) :309–317, Feb 2016.

[104] David Ryckelynck and Djamel Missoum-Benziane. Multi-level a priori hyper-reduction of mechanical models involving internal variables. Computer Methods in Applied Mechanics and Engineering, 199(17–20) :1134–1142, 2010.

[105] David Ryckelynck, Djamel Missoum-Benziane, Sophie Cartel, and Jacques Besson. A robust adaptive model reduction method for damage simulations. Computational Materials Science, 50(5) :1597–1605, 2011.

[106] David Ryckelynck, Florence Vincent, and Sabine Cantournet. Multidimensional a priori hyper-reduction of mechanical models involving internal variables. Computer Methods in Applied Mechanics and Engineering, 225–228 :28–43, 2012.

[107] Bahram Sarbandi, Sophie Cartel, Jacques Besson, and David Ryckelynck. Truncated integration for simultaneous simulation of sintering using a separated representation. Achives of Computational Methods in Engineering, 17(4) :455–463, 2010.

[108] Syuzanna Sargsyan, Steven L. Brunton, and J. Nathan Kutz. Online interpolation point refinement for reduced order models using a genetic algorithm. arXiv preprint arXiv :1607.07702, 2016.

[109] Dmitry V. Savostyanov. Quasioptimality of maximum-volume cross interpolation of tensors. Linear Algebra and its Applications, 458 :217–244, 2014.

[110] Dmitry V. Savostyanov and Ivan V. Oseledets. Fast adaptive interpolation of multidimensional arrays in tensor train format. In Multidimensional (nD) Systems (nDs), 2011 7th International Workshop on, pages 1–8, Sept 2011.

[111] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. Annals of Physics, 326(1) :96–192, 2011.

[112] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. part 1 : Coherent structures. Quarterly of Applied Mathematics, 45(3) :561–571, 1987.

[113] Danny C. Sorensen and Mark Embree. A DEIM induced CUR factorization. SIAM Journal on Scientific Computing, 38(3) :A1454–A1482, 2016.

[114] Michael Steinlechner. Riemannian optimization for high-dimensional tensor completion. SIAM Journal on Scientific Computing, 38(5) :S461–S484, 2016.

[115] G.W. Stewart. Four algorithms for the the efficient computation of truncated pivoted QR approximations to a sparse matrix. Numerische Mathematik, 83(2) :313–323, 1999.

[116] Daniel B. Szyld. The many proofs of an identity on the norm of oblique projections. Numerical Algorithms, 42(3-4) :309–323, 2006.

[117] Boxin Tang. Orthogonal array-based latin hypercubes. Journal of the American Statistical Association, 88(424) :1392–1397, 1993.

[118] Alex Townsend and Sheehan Olver. The automatic solution of partial differential equations using a global spectral method. Journal of Computational Physics, 299(Supplement C) :106 – 123, 2015.

[119] Ledyard R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, Contributions to mathematical psychology., pages 110–127. Holt, Rinehart and Winston, New York, 1964.

[120] Eugene Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. Computing, 64(4) :367–380, 2000.

[121] Eugene E. Tyrtyshnikov, Sergei Goreinov, and Nikolai Zamarashkin. Pseudo-skeleton approximations. Technical report, Institute of Numerical Mathematics of the Russian Academy of Sci., Leninski Prosp. 32-A Moscow 117334, Russia, 1995.

[122] Steven R. White. Density-matrix algorithms for quantum renormalization groups. Physical Review B, 48(14) :10345, 1993.

[123] Masayuki Yano. A space-time petrov–galerkin certified reduced basis method : Application to the boussinesq equations. SIAM Journal on Scientific Computing, 36(1) :A232–A266, 2014.

[124] Ralf Zimmermann and Karen Willcox. An accelerated greedy missing point estimation procedure. SIAM Journal on Scientific Computing, 38(5) :A2827–A2850, 2016.

Résumé

Cette thèse développe une méthodologie originale et non intrusive de construction de modèles de substitution applicable à des modèles physiques multiparamétriques. La méthodologie proposée permet d'approcher en temps réel, sur l'ensemble du domaine paramétrique, de multiples quantités d'intérêt hétérogènes issues de modèles physiques. Les modèles de substitution sont basés sur des représentations en train de tenseurs obtenues lors d'une phase hors ligne de calculs intensifs. L'idée essentielle de la phase d'apprentissage est de construire simultanément les approximations en se basant sur un nombre limité de résolutions du modèle physique lancées à la volée. L'exploration parcimonieuse du domaine paramétrique couplée au format compact de train de tenseurs permet de surmonter le fléau de la dimension. L'approche est particulièrement adaptée pour traiter des modèles présentant un nombre élevé de paramètres définis sur des domaines étendus. Les résultats numériques sur des lois élasto-viscoplastiques non linéaires montrent que des modèles de substitution compacts en mémoire qui approchent précisément les différentes variables mécaniques dépendantes du temps peuvent être obtenus à des coûts modérés. L'utilisation de tels modèles exploitables en temps réel permet la conception d'outils d'aide à la décision destinés aux experts métiers dans le cadre d'études paramétriques et visent à améliorer la procédure de calibration des lois matériaux.

Mots Clés

Modèle paramétrique, Modèle de substitution, Décomposition en train de tenseurs, Gappy POD, Données hétérogènes, Élasto-viscoplasticité.

Abstract

This thesis presents a novel non-intrusive methodology to construct surrogate models of parametric physical models. The proposed methodology enables to approximate in real-time, over the entire parameter space, multiple heterogeneous quantities of interest derived from physical models. The surrogate models are based on tensor train representations built during an intensive offline computational stage. The fundamental idea of the learning stage is to construct simultaneously all tensor approximations based on a reduced number of solutions of the physical model obtained on the fly. The parsimonious exploration of the parameter space coupled with the compact tensor train representation allows to alleviate the curse of dimensionality. The approach accommodates particularly well to models involving many parameters defined over large domains. The numerical results on nonlinear elasto-viscoplastic laws show that compact surrogate models in terms of memory storage that accurately predict multiple time dependent mechanical variables can be obtained at a low computational cost. The real-time response provided by the surrogate model for any parameter value allows the implementation of decision-making tools that are particularly interesting for experts in the context of parametric studies and aim at improving the procedure of calibration of material laws.

Keywords

Parameter-dependent model, Surrogate modeling, Tensor train decomposition, Gappy POD, Heterogeneous data, Elasto-viscoplasticity.