Parallel methods in time and in space

Thi Bich Thuy Tran

UNIVERSITE PARIS-SUD

ÉCOLE DOCTORALE de Mathématiques la région Paris-Sud

Laboratoire de Mathématiques d'Orsay

DISCIPLINE Mathématiques appliquées

THÈSE DE DOCTORAT

soutenue le 24/09/2013

par

TRAN THI BICH THUY

Méthodes de résolution parallèle en temps et en espace

Rapporteurs

Florence HUBERT
Martin GANDER

Directeurs de thèse

Laurence HALPERN
Bertrand MAURY

Co-Encadrant de thèse

Juliet RYAN

Président du jury

Frédéric LAGOUTIÈRE

Acknowledgements

"*An error does not become truth by reason of multiplied propagation,*
nor does truth become error because nobody sees it."
Mahatma Gandhi

Je tiens dans un premier temps à remercier Madame Laurence Halpern, Professeur de l'université Paris 13 - ma directrice de thèse, pour m'avoir confiée ce travail de recherches ainsi que pour son aide, ses idées et ses précieux conseils au cours des années.

Je remercie également Madame Juliette Ryan, ONERA - ma co-encadrante, pour la gentillesse et la patience qu'elle a manifestée à mon égard, pour m'avoir guidée, encouragée, conseillée et accueillie au sein de l'équipe CHP de l'ONERA durant cette thèse.

Merci à Monsieur Bertrand Maury, Professeur de l'université Paris 11 - mon co-directeur de thèse, pour sa sympathie et ses conseils.

Je tiens à remercier Monsieur Martin Gander, Professeur de l'université Genève, et Madame Florence Hubert, Professeur de l'université Aix-Marseille, d'avoir accepté d'être les rapporteurs de ce travail. Ils ont également contribué par leur remarques et suggestions à améliorer la qualité de ce mémoire et je leur suis très reconnaissante.

Je remercie Monsieur Frédéric Lagoutière, Professeur de l'université Paris 11, d'avoir accepté d'être Président du jury.

Je remercie tous ceux sans qui cette thèse ne serait pas ce qu'elle est: aussi bien par les discussions que j'ai eu la chance d'avoir avec eux, leurs suggestions ou contributions. Je pense ici en particulier à Monsieur Xavier Juvigny (pour toutes ses connaissances et son humour) et Mlle Oana Ciobanu (pour avoir été à mes côtes pendant tout ce temps) et ainsi qu'à l'équipe Calcul Haute Performance de l'ONERA et l'équipe Modélisation et Calcul scientifique du LAGA.

Je remercie aussi le laboratoire LAGA de l'université Paris 13 et le laboratoire de Mathématiques de l'université Paris 11 de m'avoir m'accueillie pendant ma thèse.

Avec mon état de santé pas comme les autres, j'ai eu des moments très difficiles et je ne sais pas comment exprimer toute ma gratitude pour mes encadrantes Madame Laurence Halpern et Madame Juliette Ryan, encore une fois, qui étaient là pour moi pendant ces dures épreuves loin

de ma famille, je suis tellement chanceuse de les avoir eu comme encadrantes.

Ces remerciements ne seraient pas complets sans une pensée à mon ami Benoît qui a m'a beaucoup aidé et encouragé pour que je puisse terminer ce travail.

Mes dernières pensées iront vers ma famille, et surtout mes parents, qui m'auront permis de poursuivre mes études jusqu'à aujourd'hui.

Contents

Abstract

Domain decomposition methods in space applied to Partial Differential Equations (PDEs) expanded considerably thanks to their effectiveness (memory costs, calculation costs, better conditioned local problems) and this related to the development of massively parallel machines. Domain decomposition in space-time brings an extra dimension to this optimization.
In this work, we study two different direct time-parallel methods for the resolution of Partial Differential Equations.
The first part of this work is devoted to the Tensor-product space-time method introduced by R. E. Lynch, J. R. Rice, and D. H. Thomas in 1963. We analyze it in depth for Euler and Crank-Nicolson schemes in time applied to the heat equation. The method needs all time steps to be different, while accuracy is optimal when they are all equal (in the Euler case). Furthermore, when they are close to each other, the condition number of the linear problems involved becomes very big. We thus give for each scheme an algorithm to compute optimal time steps, and present numerical evidences of the quality of the method.
The second part of this work deals with the numerical implementation of the Block method of Amodio and Brugnano presented in 1997 to solve the heat equation with Euler and Crank-Nicolson time schemes and the elasticity equation with Euler and Gear time schemes. Our implementation shows how the method is accurate and scalable.

Les méthodes de décomposition de domaine en espace ont prouvé leur utilité dans le cadre des architectures parallèles. Pour les problèmes d'évolution en temps, il est nécessaire d'introduire une dimension supplémentaire de parallélisme dans la direction du temps. Ceci peut alors être couplé avec des méthodes de type *optimized Schwarz waveform relaxation*. Nous nous intéressons dans cette thèse aux méthodes directes de décomposition en temps. Nous en étudions particulièrement deux.
Dans une première partie nous étudions la méthode de produit tensoriel, introduite par R. E. Lynch, J. R. Rice, et D. H. Thomas in 1963. Nous proposons une méthode d'optimisation des pas de temps, basée sur une étude d'erreur en variable de Fourier en temps. Nous menons cette étude sur les schémas d'Euler et de Newmark pour la discrétisation en temps de l'équation de la chaleur. Nous présentons ensuite des tests numériques établissant la validité de cette approche.
Dans la seconde partie, nous étudions les méthodes dites de Bloc, introduites par Amodio et Brugnano en 1997. Nous comparons diverses implémentations de la méthode, basées sur différentes approximations de l'exponentielle de matrice. Nous traitons l'équation de la chaleur et l'équation des ondes, et montrons par une étude numérique bidimensionnelle la puissance de la méthode.

Keywords: Domain decomposition, Time-parallel method, Tensor-product space-time method, Blocks method, Optimal parallelism.

Résumé

0.1 Contexte

Les méthodes de décomposition de domaine en espace appliquées aux équations aux dérivées partielles (EDP) ont considérablement élargi leur domaines d'application grâce à leur efficacité (coûts mémoire, coûts calcul, problèmes locaux mieux conditionnés, ...) et ceci lié au développement des machines massivement parallèles. La décomposition de domaine en espace et en temps apporte une dimension supplémentaire à cette optimisation. D'autre part indépendamment de l'espace, de nombreuses réflexions sur la parallélisation en temps ont eu lieu depuis les trente dernières années avec différentes techniques telles que une méthode multigrille en temps [28], une technique de multi-pas de temps modifiée [57], la méthode Pararéel [33], parallélisation par blocs de temps-espace [1, 2] ou méthode de Produit Tensoriel espace-temps [42]. Parmi toutes ces méthodes, ce travail de thèse s'est surtout intéressé aux méthodes directes, et en particulier, la méthode Tenseur Produit espace-temps qui reste inexploité jusqu'à présent et la méthode de blocs temps-espace en raison de sa similitude avec la méthode Pararéel.

0.2 Historique

Les équations aux dérivées partielles (EDP) sont utilisées de plus en plus pour modéliser des phénomènes physiques complexes. L'invention de processeurs multi-core en informatique a permis de grands progrès dans les simulations industrielles. Pour utiliser à grande échelle ces ordinateurs parallèles (cluster) de manière efficace, de nouveaux algorithmes parallèles devaient être développés.
Les méthodes de décomposition de domaine sont naturellement adaptées pour fonctionner en parallèle et en particulier pour les problèmes complexes de modélisation.
Le concept de ces méthodes peut être vu par leurs noms: nous structurons les problèmes en sous-domaines; pour chaque sous-domaine, toutes les données et les calculs sont gérés par un processeur de la machine parallèle, et il y a des communications entre processeurs grâce à un système spécial (par exemple Message Passing Interface (MPI)). La résolution du problème est alors obtenue par itération entre les sous-domaines et les sous-domaines peuvent même être des domaines espace-temps.

L'ancêtre des méthodes de décomposition de domaine (DDM) dans l'espace pourrait être la méthode de Schwarz publiée en 1870 [53] où il propose une façon de résoudre un problème géométrique relativement complexe (l'union d'un disque et d'un rectangle) donné à l'aide des

sous-domaines avec des conditions de transmission entre les sous-domaines. Un siècle plus tard, à la fin des années 80, cette méthode a été ré-exploré et depuis lors elle a été très développée, stimulée par l'invention des machines parallèles. Les DDM sont utilisés dans de nombreux domaines tels que: automobile, météorologie, astrophysique, médecine, aéronautique, etc .., ou pour des problèmes plus complexes avec des propriétés physiques différentes sur différents domaines.
Miller en 1965 a proposé des analogues numériques à la procédure alternative de Schwarz [45], basés sur l'utilisation de séries de Fourier discrétisées pour calculer des domaines simples tels que des rectangles et des disques. Mais chaque itération de cette méthode est coûteuse. Puis, en 1988, P. L. Lions propose une extension brillante de la méthode de Schwarz [34, 35, 36], très adapté au calcul parallèle sur les clusters et également prouve la convergence de la méthode de Schwarz pour un certain nombre de sous-domaines. Ces articles ont donné un nouvel intérêt pour la méthode de Schwarz et, par conséquent, de nombreux développements ont suivi cette voie [10, 54, 49, 56]. En 2000, M. J. Gander, L. Halpern, F. Nataf ont présenté une méthode de Schwarz optimisée [22] une nouvelle classe de méthode de Schwarz avec de meilleures propriétés de convergence et de nombreuses applications pour les différentes EDP: l'équation de Helmholtz [21, 20], l'équation de Maxwell [11], l'équation d'onde [19], l'équation d'advection-diffusion [26], l'équation de Navier Stokes [9], etc...

Une seconde famille de méthodes de décomposition de domaine en espace repose sur une décomposition de Schur, autrefois appelée méthode de sous-structuration. Elle a été introduite par Przemieniecki en 1963 [48] dans le contexte des calculs aéronautiques. Un problème d'éléments finis symétrique est divisé en sous-domaines non recouvrants, et les inconnues à l'intérieur des sous-domaines sont éliminées, réduisant le problème global en un problème sur les interfaces. La matrice correspondante appelée complément de Schur appliquées aux inconnues associées aux interfaces du sous-domaine est résolu par la méthode du gradient conjugué [54, 49, 10]. Il y a deux avantages à cette méthode. Tout d'abord, l'élimination des inconnues intérieures aux sous-domaines, correspond à des problèmes de Dirichlet locaux , qui peut être fait en parallèle. Deuxièmement, le passage au complément de Schur réduit le nombre d'inconnues et tend donc à diminuer le nombre d'itérations. Trois décennies après, la méthode FETI (Finite Element Tearing and Interconnecting), une méthode de Schur exceptionnelle, a été proposé par F.X. Roux et C. Farhat [15, 17]. FETI contient deux ingrédients supplémentaires: préconditionneur grille grossière naturel en utilisant des sous-domaines flottants et conditionnement des problèmes locaux amélioré. Cette méthode a immédiatement attiré beaucoup d'attention des chercheurs et a été étendu et utilisé pour le calcul massivement parallèle pour résoudre les équations aux dérivées partielles en parallèle [16, 31, 14].

Les méthodes de décomposition de domaine en espace se sont tellement étendues que le développement devient stationnaire par rapport à l'amélioration de la capacité des machines

parallèles. Afin d'exploiter au maximum les machines massivement parallèles, nous devons envisager une autre dimension qui est la parallélisation en temps des EDP temps-espace

La décomposition de domaine en temps a pris son essor longtemps après la décomposition de domaine dans l'espace, mais a été rapidement développée par les chercheurs. Principalement, il y a deux types de méthode parallèle en temps: itératives et directes. La première fois qu'une méthode itérative parallèle a été présenté est dans l'article "Parallel Methods for Integrating Ordinary Differential Equations" présenté par J.Nievergelt en 1964 [47] qui est finalement devenu la méthode de tirs multiples (Multiple Shooting) pour des Problèmes aux Valeurs Limites [30]. Puis, en 1967 , W. Miranker et W. Liniger présentent "Parallel Methods for the Numerical Integration of Ordinary Differential Equations" [46] comme une méthode parallèle multi-pas en temps qui est devenu, en 1990 par D. Womble un algorithme multi-pas de temps pour les machines parallèles. Mais le potentiel de décomposition de domaine de temps est devenu considérable quand J.L. Lions, Y. Maday, G. Turinici ont présenté le "Parareal algorithm in Time Discretization of PDEs" dans [33] comme une méthode numérique pour résoudre les problèmes d'équations d'évolution en parallèle. Cette méthode propose de diviser le problème d'évolution global en temps en une série des problèmes d'évolution indépendants sur des intervalles de temps plus petits, la méthode se rapprochant de la solution à la fin de l'intervalle global en temps avant d'avoir des approximations précises sur les sous intervalles précédents. L'algorithme itératif est basé sur une approche prédicteur-correcteur qui converge généralement assez rapidement, et conduit , lorsque de très nombreux processeurs sont disponibles, à un coût en temps réel des procédures de la solution. Dans le flux de développement de la machine parallèle , la méthode Pararéel a reçu beaucoup d'attention dès qu'elle est apparue. Des expériences approfondies peuvent être trouvées pour le fluide et les problèmes de structure dans [13], pour les équations de Navier- Stokes dans [18], et pour la simulation de réservoir dans [23]. Plusieurs variantes de la méthode ont été proposées dans [6, 13] et une analyse plus profonde et la stabilité de la méthode ont été faites [43, 44, 55, 7].

Une méthode parallèle directe en temps fut d'abord présentée en 1963 grâce a une analyse du produit tensoriel d'EDP dans [38]. L'idée était que certains problèmes multidimensionnels peuvent être résolus par la résolution de quelques problèmes unidimensionnels. Puis en 1964, R.Lynch, J.Rice et D.Thomas continuent à exploiter cette méthode pour la résolution directe des équations aux dérivées partielles [37]. L'approche est naturelle et classique. Si un problème est séparable, alors la solution peut être exprimée en termes de produits tensoriels de solutions de problèmes de dimensions inférieures. Cela implique que la matrice impliquée dans l'EDP correspondante peut être exprimée en termes de produit tensoriel de matrices de rangs inférieurs donc beaucoup plus simple à résoudre. Cela conduit à une méthode simple et directe pour l'analyse de schémas implicites par direction alternée [39]. Le solveur produit tensoriel rapide de Lynch, Rice et Thomas a une applicabilité limitée , mais reste néanmoins encore très attractif le cas échéant. Cependant, cette méthode n'a pas été beaucoup étudié jusqu'en 2008 lorsque Y. Maday et E.M. Ronquist ont appliqué les solveurs de produits tensoriels rapides aux problèmes à trois dimensions dans l'espace [40]. Dans la même année, une autre application de solveurs de produit

tensoriel au EDP en fonction du temps a été publié dite méthode du produit tensoriel espace-temps qui résout une EDP en temps d'une façon parallèle à toutes les étapes de temps grâce à un système d'équations algébriques avec une contrainte sur des pas de temps [42]. Cependant, peu d'analyses était fait. Peu de temps après, la discrétisation spectrale dans le temps et dans l'espace [41] a été publiée à l'aide du produit tensoriel pour une approximation d'ordre plus élevé dans le temps et l'espace pour la résolution numérique des équations aux dérivées partielles dépendant du temps qui a reçu beaucoup d'intérêt.
Différente de la méthode produit tensoriel espace-temps qui divise un problème global en une série de problèmes d'évolution indépendants qui peuvent être résolus directement et ceci de façon complètement parallèle, la méthode de Bloc [1] - dérivée de Méthodes aux Valeurs Limites par Bloc (BVMs)- a la même vision de sous-structuration dans le domaine temporel comme la méthode Pararéel, mais la solution du problème de l'évolution dans chaque intervalle de temps (avec des pas de temps raffinés) peut être réglé directement à partir du système algébrique sans faire aucune itération comme dans la méthode Pararéel. Seule l'application de l'équation de la chaleur avec schéma de la seconde l'ordre Gear dans le temps a été présenté.

0.3 Objectif de ce travail

Dans cette thèse, deux méthodes parallèles en temps seront étudiés avec des solveurs directs: la méthode de produit tensoriel espace-temps de Y. Maday et E. Rønquist [42] et la méthode Bloc de Amodio P. et Brugnano L. [3] pour résoudre différentes équations aux dérivées partielles parallèles en temps et aussi réaliser des expériences numériques avec Matlab et Fortran(MPI).
Dans le contexte que la méthode de Produit Tensoriel espace-temps est encore inexploitée, une étude de la stabilité et de l'estimation de l'erreur de la méthode numérique avec différentes méthodes d'approximation de temps implicites, comme Euler implicite, méthode de $Newmark - \beta$ [29] pour l'équation de la chaleur et de la méthode d'Euler et méthode de Newmark pour l'équation d'élasticité [51] seront effectuées.
La seconde méthode parallèle en temps est appliqué à l'équation de la chaleur avec le schéma en temps Euler et Crank-Nicolson [27] et l'équation d'élasticité [51] avec les schémas Euler et Newmark pour l'approximation de dérivée en temps. L'objectif de cette thèse est d'étudier théoriquement et numériquement ces méthodes de parallélisme en temps pour évaluer et trouver les conditions optimales de ces méthodes pour enfin avoir une application industrielle en aéro-élasticité et l'acoustique de vibration (travail dans lequel le département DTIM de l'ONERA Palaiseau est impliqué).

0.4 Plan de ce travail

Partout dans ce travail, la méthode des différences finies est utilisée pour discrétiser la solution des équations aux dérivées partielles, à la fois dans l'espace et le temps, et les applications

présentées concernent les géométries simples à mailles cartésiennes.
- La première partie introduit la méthode de produit tensoriel espace-temps pour l'équation de la chaleur. Ensuite, nous utilisons l'approximation de premier ordre en temps, méthode d'Euler implicite, avec la méthode de produit tensoriel espace-temps pour résoudre l'équation de la chaleur. Une particularité de la méthode Produit tensoriel est que tous les pas de temps doivent être différents, c'est pourquoi, dans le chapitre suivant, une étude de l'erreur pour la méthode séquentielle à pas de temps distincts est effectuée. Ensuite, nous montrons que la présence d'une condition sur la suite de pas de temps rend la matrice des vecteurs propres S (en raison de la diagonalisation de la matrice en temps) du système linéaire, une matrice presque singulière, ce qui provoque l'erreur de la méthode numérique. Donc, avec l'estimation de l'erreur séquentielle et en tenant compte des erreurs dues à la matrice S, nous trouvons une condition sur la suite des pas de temps pour s'assurer que l'erreur finale de la méthode numérique est optimale. A la fin de ce chapitre, les applications numériques sont présentés pour valider l'algorithme d'optimisation proposé ainsi que l'efficacité de la méthode de produit tensoriel espace-temps.

Le second chapitre de cette partie est dédié au schéma de Newmark, qui est normalement utilisé pour l'équation d'élasticité, et appliqué à l'équation de la chaleur pour obtenir une approximation de deuxième ordre de la dérivée en temps. Ce chapitre suivra la même structure que le chapitre précédent sur l'etude de la stabilité et la précision de la méthode Crank Nicolson (un cas particulier de Newmark) pour l'équation de la chaleur. Puis l'erreur due à la diagonalisation de la méthode numérique temps-parallèle sera étudiée et les applications numériques dans la fin confirmeront les paramètres optimaux.

- La deuxième partie de cette thèse est l'application de la méthode Bloc. Tout d'abord, nous présentons la méthode Bloc pour les EDO, puis son application à l'équation de la chaleur à l'aide de la méthode d'Euler et la méthode de Crank-Nicolson pour discrétiser la dérivée en temps. Ensuite, la méthode sera utilisée pour résoudre une equation d'ordre supérieur - l'équation d'élasticité à l'aide de schémas Euler et Newmark. Cette méthode directe de Bloc est basée sur une approximation de conditions initiales de type exponentielle d'une matrice. Trois techniques différentes seront comparées: un calcul complet en utilisant la méthode de Padé [4], une approximation de Arnoldi -Krylov consistant à projeter la matrice sur un espace de Krylov réduit et un développement limité de Taylor de l'exponentielle. Des expériences numériques sont présentées montrant l'efficacité de la méthode.

0.5 Problématique

0.5.1 Méthode de produit tensoriel espace-temps

La méthode de produit tensoriel a été introduite dans [42] comme une méthode algébrique pour gérer le parallélisme en temps. Supposons une équation aux dérivées partielles discrétisée

dans le temps et l'espace, avec des dimensions $M \times N$. Supposons que le vecteur discret de toutes les inconnues dans le temps et dans l'espace est la solution d'un grand système

$$(B \otimes I_x + I_t \otimes A)U = F. \tag{0.5.1}$$

Le vecteur U est ordonné ainsi $U = (U^1, \cdots, U^M)$, $U^i \in \mathbb{R}^N$.
Si la matrice B est diagonalisable, avec $B = SDS^{-1}$, l'équation précédente peut être décomposée en

$$\begin{aligned} &(1) \quad (S \otimes I_x)G = F, \\ &(2) \quad (\lambda_m + A)V^m = G^m, \quad 1 \leq m \leq M, \\ &(3) \quad U = (S \otimes I_x)V. \end{aligned}$$

M équations dans l'espace peuvent ainsi être résolues indépendamment sur les processeurs. L'idée semble très attrayante, mais l'application nécessite certaine prudence:

1. Tout d'abord, l'obtention de la forme (0.5.1) dans chaque schéma n'est pas une tâche facile (voir l'application du schéma Newmark ci-dessous).
2. Puis la matrice B n'est pas diagonalisable, à moins que les pas de temps soient tous différents.
3. Dans ce cas, la précision du système est généralement affectée par rapport à la configuration pas de temps égaux.
4. Il est donc préférable de garder les pas de temps les plus proches possible .
5. Ensuite, le conditionnement de la matrice S augmente de façon exponentielle avec M, d'où une détérioration des résultats des étapes (1) et (3).

Le but de ce travail est de fournir une méthode rigoureuse pour déterminer le nombre d'étapes M et le pas de temps $\Delta t_i = t_{i+1} - t_i$ comme une suite géométrique $\Delta t_i = \rho^{i-1} \Delta t_i$. Nous avons mis en place un processus d'optimisation de l'équation scalaire $\partial_t u + au = 0$, avec $a > 0$. Il consiste à réaliser une série de Taylor pour $\rho = 1 + \varepsilon$ de l'erreur et du conditionnement de S, et effectuer une répartition équilibrée de ces erreurs.

Application de la méthode avec schéma d'Euler

Nous présentons d'abord la méthode pour le schéma d'Euler implicite, puisque la forme (0.5.1) est alors facile à obtenir. L'introduction d'un maillage constitué de nœuds $0 = t_0 < t_1 < t_2 < \cdots < t_M = T$, $\Delta t_m = t_m - t_{m-1}$ pour lequel le schéma d'Euler est

$$\frac{U^m - U^{m-1}}{\Delta t_m} + AU^m = F^m, \quad m = 1, \ldots, M, \tag{0.5.2}$$

où le vecteur $U^m \in \mathbb{R}^N$ désigne l'approximation numérique de la solution au temps t_m. Le vecteur $F^m \in \mathbb{R}^N$ représente les données indiquées aux points de grille internes au temps t_m.

Puis, nous définissons la matrice B comme

$$B = \begin{pmatrix} \frac{1}{\Delta t_1} & & & & \\ \frac{-1}{\Delta t_2} & \frac{1}{\Delta t_2} & & & \\ & \frac{-1}{\Delta t_3} & \frac{1}{\Delta t_3} & & \\ & & \ddots & \ddots & \\ & & & \frac{-1}{\Delta t_M} & \frac{1}{\Delta t_M} \end{pmatrix}. \tag{0.5.3}$$

Nous définissons aussi I_t à la matrice identité de dimension M (associé avec le domaine temporel) et I_x à la matrice identité de dimension N (associé au domaine spatial). Avec cette notation, nous pouvons écrire l'équation (0.5.2) sous forme produit tensoriel

$$(B \otimes I_x + I_t \otimes A)U = F. \tag{0.5.4}$$

Lemme .1

La matrice B est diagonalisable si et seulement si les étapes de temps Δt_i sont tous différent. Dans ce cas

$$B = S\Lambda S^{-1}, \quad \text{with } \Lambda = diag(1/\Delta t_1, \cdots, 1/\Delta t_M). \tag{0.5.5}$$

Par une transformation de Fourier dans l'espace, nous pouvons voir que l'analyse d'erreur sur une équation $0 - D$, pour $a > 0$ sera pertinente. nous obtenons l'EDO:

$$\frac{d\,u}{d\,t} + au = 0, \tag{0.5.6}$$

approchée par un schéma d'Euler implicite sur l'intervalle $(0, T)$:

$$\frac{u_{n+1} - u_n}{\Delta t_n} + au_{n+1} = 0. \tag{0.5.7}$$

Pour les pas de temps fixés $\Delta t = T/M$, la solution de l'équation (0.5.7) après M itérations est

$$u_M = (1 + a\Delta t)^{-M} u_0.$$

Pour les pas de temps différents Δt_m, la solution de l'équation (0.5.7) après M itérations est

$$v_M = \prod_{m=1}^{M} (1 + a\Delta t_m)^{-1} u_0,$$

et la solution de l'EDO au temps $T = M\Delta t$ est $u(T) = e^{-aT}u_0$.

Pour a et T donnés, définissons le propagateur d'erreur

$$Err(a, T, (\Delta t_1, \cdots, \Delta t_M)) = \prod_{m=1}^{M} (1 + a\Delta t_m)^{-1} - e^{-aT}.$$

Notons que $Err > 0$ pour tous aT et tous M, quand $\Delta t_i = T/M$ pour tous i.

Lemme .2

Pour tout a, T et M, le minimum de Err sur toutes les partitions $\Delta = \{\Delta t_m\}_{1\le m\le M}$ de $(0,T)$ est obtenu lorsque tous les pas de temps sont égaux, i.e $\Delta t_m = \frac{T}{M}$.

Maintenant, considérons les différents pas de temps:

$$\Delta t_m = \rho^{m-1}\Delta t_1,\ i = 1,\dots,M,$$

et écrivons

$$Err(a,T,M,\rho) := Err(a,T,(\Delta t_1,\cdots,\Delta t_M)).$$

Supposons donc que

$$\rho = 1+\varepsilon.$$

Alors, les résultats suivants sont obtenus par l'étude théorique.

Théorème .1

Étant donné a, T et M, $\Delta t = T/M$, pour $\rho = 1+\varepsilon$ quand ε est suffisant petit, nous avons

$$\begin{aligned} Err(a,T,M,1+\varepsilon) &= Err(a,T,M,1)(1+\alpha\varepsilon^2+o(\varepsilon^2)),\\ \alpha(a,T,M) &= \frac{(1+a\Delta t)^{-M}}{(1+a\Delta t)^{-M}-e^{-aT}}\frac{b^2M(M^2-1)}{24} \end{aligned} \tag{0.5.8}$$

où $b = \dfrac{a\Delta t}{1+a\Delta t}$.

Nous définissons la classe des matrices Toeplitz comme:

$$T(X_1,\dots,X_{M-1}) = \begin{pmatrix} 1 & & & & \\ X_1 & \ddots & & 0 & \\ X_2 & \ddots & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \\ X_{M-1} & & X_2 & X_1 & 1 \end{pmatrix}. \tag{0.5.9}$$

Théorème .2

La matrice S est triangulaire inférieure. Elle peut être choisi à diagonale unité, auquel cas elle est de la forme matrice Toeplitz, *i.e.*

$$S = T(P_1,\dots,P_{M-1}),\quad ,\qquad P_i = p_1\dots p_i,\quad p_i = \frac{1}{1-\rho^i}. \tag{0.5.10}$$

La matrice l'inverse S^{-1} est alors la même forme:

$$S^{-1} = T(Q_1,\dots,Q_{M-1}),\qquad Q_i = q_1\dots q_i,\quad q_i = \frac{\rho^{i-1}}{\rho^i-1}. \tag{0.5.11}$$

Lemme .3

Pour $\rho = 1 + \varepsilon$, le comportement asymptotique du conditionnement de la matrice S (quelle que soit la norme) est

$$cond(S) \sim \frac{1}{((M-1)!)^2 \varepsilon^{2(M-1)}}.$$

Théorème .3

Le conditionnement de S ne diminue pas la précision du calcul de la méthode de produit tensoriel si et seulement si

$$Log(\tau) + Log(cond(S)) = Log\left(\left|\frac{Err(a,T,M,\rho)}{Err(a,T,M,1)}\right|\right). \tag{0.5.12}$$

où Log est en base 10 et τ est la précision de la machine. Cela peut être écrit asymptotiquement comme

$$Log(\tau) + Log(cond(S)) = Log(\alpha(a,T,M)\varepsilon^2), \tag{0.5.13}$$

où α est défini dans le théoreme précédant, et résolu en ε

$$\varepsilon_0(a,T,M) = \left(\frac{\tau}{\alpha(M-1)!^2}\right)^{\frac{1}{2M}}. \tag{0.5.14}$$

Les tests numériques en Matlab donnent les valeurs optimales de M, ρ et de η dans la figure suivante:

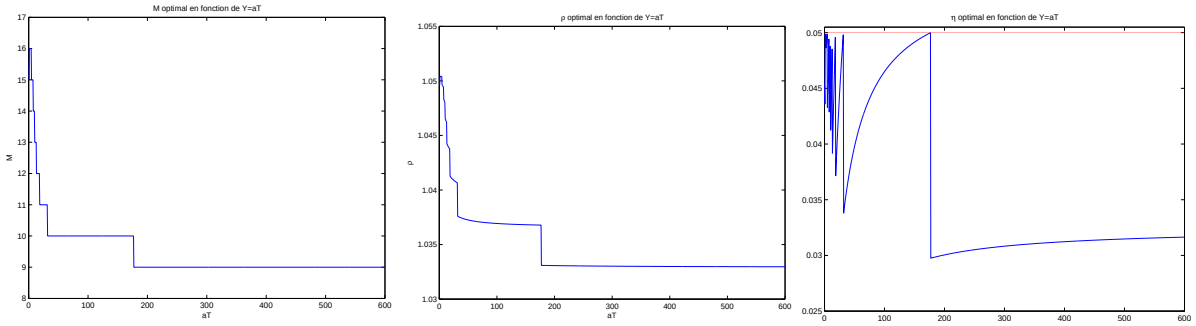

Figure 1: $\bar{M}, \bar{\rho}$ and $\bar{\eta}_{opt}$ en fonctions de $y = aT$ dans une tolérance de $\bar{\eta} = 5\%$

Application de la méthode au schéma de Newmark

En menant la même étude pour la méthode avec schéma de Newmark, nous avons les résultats suivants.

Théorème .4

La méthode Newmark est inconditionnellement stable pour l'équation de la chaleur si et seulement si $\gamma \geq \frac{1}{2}$ et $2\beta \geq \gamma$.

Quant à l'ordre de la méthode Newmark:

$$\begin{cases} \gamma = 2\beta : \text{ méthode l'ordre 2;} \\ \gamma \neq 2\beta : \text{ méthode l'ordre 1.} \end{cases}$$

Considérons l'EDO discrétisée par schéma Crank Nicolson

$$\frac{u^m - u^{m-1}}{\Delta t_m} + \frac{a}{2}(u^m + u^{m-1}) = 0,$$

qui est effectivement inconditionnellement stable et de deuxième ordre en temps.
Le propagateur d'erreur dans ce cas est

$$Err(a, T, (\Delta t_1, \cdots, \Delta t_M)) = \prod_{m=1}^{M} \frac{1 - a\frac{\Delta t_m}{2}}{1 + a\frac{\Delta t_m}{2}} - e^{-aT}$$

Théorème .5

Étant donné a, T and M, $\Delta t = T/M$, pour $\rho = 1 + \varepsilon$ quand ε est suffisant petit, nous avons

$$\begin{aligned} Err(a, T, M, 1 + \varepsilon) &= Err(a, T, M, 1)(1 + \alpha^N \varepsilon^2), \\ \alpha^N(a, T, M) &= -\frac{2x^3}{(1 - x^2)^2} \frac{M(M^2 - 1)}{12} \left(\frac{1 - x}{1 + x}\right)^M \frac{1}{Err(a, T, M, 1)} \end{aligned} \tag{0.5.15}$$

quand $x = \dfrac{a\Delta t}{2}$.

Théorème .6

Le produit tensoriel pour le schéma Newmark est

$$(B \otimes \mathbb{I}_x)U + (\mathbb{I}_x \otimes A)U = \tilde{F}, \tag{0.5.16}$$

$$\begin{aligned} &B = (B_2 + B_3 B_5^{-1} B_4)^{-1} B_1. \\ &\tilde{F} = F - (B_2 + B_3 B_5^{-1} B_4)^{-1}((B_3 B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 - \underline{f}_2). \end{aligned} \tag{0.5.17}$$

$$B_1 = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & & & \\ \Delta t_2 & 0 & & \\ & \ddots & \ddots & \\ & & \Delta t_M & 0 \end{pmatrix},$$

$$B_3 = \begin{pmatrix} \beta\Delta t_1^2 & & & \\ (\frac{1}{2}-\beta)\Delta t_2^2 & \beta\Delta t_2^2 & & \\ & \ddots & \ddots & \\ & & (\frac{1}{2}-\beta)\Delta t_M^2 & \beta\Delta t_M^2 \end{pmatrix},$$

$$B_4 = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \quad B_5 = \begin{pmatrix} \gamma\Delta t_1 & & & \\ (1-\gamma)\Delta t_2 & \gamma\Delta t_2 & & \\ & \ddots & \ddots & \\ & & (1-\gamma)\Delta t_M & \gamma\Delta t_M \end{pmatrix},$$

$$\underline{f}_1 = \begin{pmatrix} \dot{U}^0 + (1-\gamma)\Delta t_1 \ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \underline{f}_2 = \begin{pmatrix} U^0 + \Delta t_1 \dot{U}^0 + (\frac{1}{2}-\beta)\Delta t_1^2 \ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Théorème .7

La matrice B est diagonalisable quand tous les dti sont différents. La matrice des vecteurs propres S est triangulaire inférieure. Elle peut être choisi à diagonale unité, auquel cas elle est de la forme de la matrice Toeplitz, *i.e.*

$$S = T(P_1, \ldots, P_{M-1}), \qquad P_i = p_1 \ldots p_i, \quad p_i = \frac{1+\rho^i}{1-\rho^i}. \tag{0.5.18}$$

La matrice inverse S^{-1} est alors de la même forme:

$$S^{-1} = T(Q_1, \ldots, Q_{M-1}), \qquad Q_i = q_1 \ldots q_i, \quad q_i = -\rho\frac{1+\rho^{i-2}}{1-\rho^i}. \tag{0.5.19}$$

$$Q_n = \rho^{-n}\frac{(1+\rho)(1+1)(1+\rho^{-1})...(1+\rho^{2-n})}{(1-\rho^{-1})(1-\rho^{-2})...(1-\rho^{-n})} = \rho^{-n}\,\frac{\prod_{j=-1}^{n-2}(1+\rho^{-j})}{\prod_{j=0}^{n}(1-\rho^{-j})}$$

Lemme .4

Pour $\rho = 1+\varepsilon$, le comportement asymptotique du conditionnement de la matrice S est

$$cond(S) \sim \left(\frac{2^{M-1}}{(M-1)!\varepsilon^{M-1}}\right)^2.$$

Théorème .8

Le conditionnement de S ne diminue pas la précision du calcul de la méthode de produit tensoriel si et seulement si

$$Log(\tau) + Log(cond(S)) = Log\left(\left|\frac{|Err(a,T,M,\rho)|}{|Err(a,T,M,1)|}\right|\right). \tag{0.5.20}$$

où Log est en base 10 et τ est la précision de la machine. Cela peut être écrit asymptotiquement

$$Log(\tau) + Log(cond(S)) = Log(|\alpha^N(a,T,M)|\varepsilon^2),$$

où α est défini dans le théorème précédant, et résolu en ε

$$\varepsilon_0^N = \left(\frac{\tau 2^{2(M-1)}}{|\alpha^N(a,T,M)|(M-1)!^2}\right)^{\frac{1}{2M}}. \tag{0.5.21}$$

Les tests numériques en Matlab donnent les valeurs optimales de M, ρ et de η dans les figures suivantes

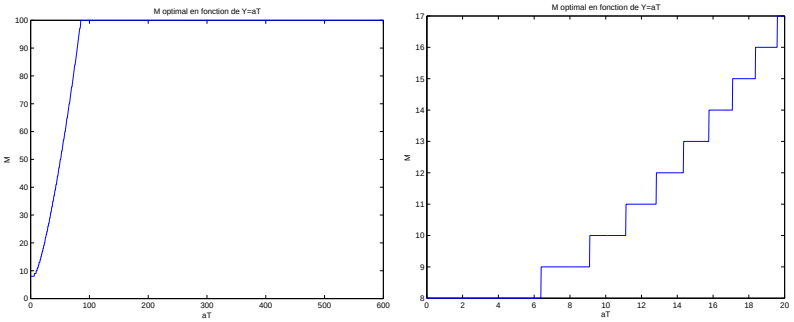

Figure 2: $\bar{M}$ en fonction de $y = aT$ dans une tolérance de 5%

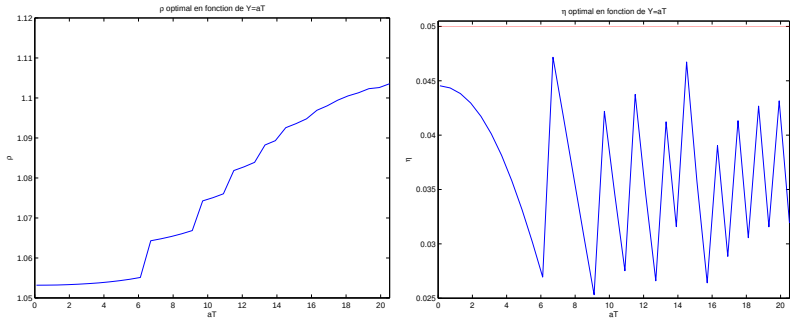

Figure 3: $\bar{\rho}$ et $\bar{\eta}_{opt}$ en fonctions de $y = aT$ dans une tolérance de 5%

0.5.2 Méthode Bloc

Prenons l'équation différentielle linéaire dans $\mathbb{R}^N$:

$$\begin{cases} y' = Ly + g(t), & t \in T = [t_0, t_1] \\ y(t_0) = y_0 \in \mathbb{R}^N. \end{cases} \tag{0.5.22}$$

Considérons un maillage grossier convenable de T

$$t_0 \equiv \tau_0 < \tau_1 < ... < \tau_p \equiv t_1.$$

Définissons les problèmes de $i = 1, ..., p$

$$\begin{cases} y' = L\,y + g(t) \quad t \in T_i = [\tau_{i-1}, \tau_i], \\ y(\tau_{i-1}) = y_0^i \in \mathbb{R}^N. \end{cases} \tag{0.5.23}$$

Chaque T_i est discrétisé par M pas de temps fins h_i

$$h_i = \frac{\tau_i - \tau_{i-1}}{M} \qquad i = 1, .., p.$$

Soit $y(t)$ est la solution de (0.5.22) et soit $Y^i(t), i = 1, ..., p$ les p solutions de (0.5.23). Posons

$$y_m^i = Y^i(\tau_{i-1} + mh_i) \in \mathbb{R}^N \qquad m = 0, ..., M \qquad i = 1, ..., p.$$

Afin d'avoir (0.5.22) équivalent à p problèmes (0.5.23), nous avons besoin

$$y_0^1 = y_0 \qquad y_0^i = y_M^{i-1} \qquad i = 2, ..., p.$$

Les approximations numériques pour les solutions de (0.5.23) peuvent être obtenues en résolvant les problèmes discrets sous la forme:

$$S_i\,Y^i = v_i\,y_0^i + G^i \quad Y^i = (y_1^i, ..., y_M^i)^T \in \mathbb{R}^{NM} \quad i = 1, ..., p.$$

òu $S_i \in \mathbb{R}^{NM\times NM}$, $v_i \in \mathbb{R}^{NM\times N}$, $G^i \in \mathbb{R}^{NM}$.

$$\mathbf{S}\,\mathbf{Y} \equiv \begin{pmatrix} I_N & & & & \\ -v_1 & S_1 & & & \\ & -V_2 & S_2 & & \\ & & \ddots & \ddots & \\ & & & -V_p & S_p \end{pmatrix} \begin{pmatrix} y_M^0 \\ Y^1 \\ Y^2 \\ \vdots \\ Y^p \end{pmatrix} = \begin{pmatrix} y_0 \\ G^1 \\ G^2 \\ \vdots \\ G^p \end{pmatrix}, \tag{0.5.24}$$

$$V_i = [0\,|\,v_i] \in \mathbb{R}^{NM\times NM} \quad i = 2, ..., p.$$

Considérons la factorisation:

$$\mathbf{S} = D\,W \equiv \begin{pmatrix} I_N & & & & \\ & S_1 & & & \\ & & S_2 & & \\ & & & \ddots & \\ & & & & S_p \end{pmatrix} \begin{pmatrix} I_N & & & & \\ -w_1 & I_{NM} & & & \\ & -W_2 & I_{NM} & & \\ & & \ddots & \ddots & \\ & & & -W_p & I_{NM} \end{pmatrix},$$

$$W_i = [0 \,|\, w_i] \in \mathbb{R}^{NM \times NM}, \quad w_i = S_i^{-1} v_i \in \mathbb{R}^{NM \times N}.$$

La méthode de Bloc peut être résumée en 4 étapes principales:

1. un solveur parallèle de
$$S_i Z^i = G^i$$
2. un solveur parallèle de
$$w_i = S_i^{-1} v_i$$
3. un solveur séquentiel de
$$y_0^1 = y_0, \qquad y_0^{i+1} = z_M^i + w_{M,i}\, y_0^i,$$
4. un mise-à-jour parallèle de
$$\hat{Y}^i = \hat{Z}^i + \hat{w}_i\, y_0^i$$

Notons que w_i, $v_i \in \mathbb{R}^{NM \times N}$ et $S_i \in \mathbb{R}^{NM \times NM}$, donc l'étape 2 est très coûteuse, par conséquent, ces 4 étapes peuvent être remplacées par:

1. un solveur parallèle de
$$S_i Z^i = G^i$$
2. une approximation séquentielle "moins-cher" de $w_{M,i}\, y_0^i$ fournir des initialisations y_0^i
3. un solveur parallèle de
$$S_i Y^i = G + v_i\, y_0^i$$

Application de la méthode Bloc avec le schéma Crank Nicolson pour l'équation de la chaleur

Résoudre l'équation de la chaleur dans $\Omega \times T$ = [0,1] x [0,1] x [0,.2] avec la solution exacte étant:

$$y(x,t) = sin(\pi x)\ sin(\pi y)\ (sin(\pi t) + e^{-2\pi^2 t})$$

et le second membre

$$f(x,t) = \pi sin(\pi x)\ sin(\pi y)\ (cos(\pi t) + 2\pi sin(\pi t))$$

$\Omega \times T$ est maillé par (N = Nx*Ny) points en espace et (M*p) points en temps, $p = 10$.

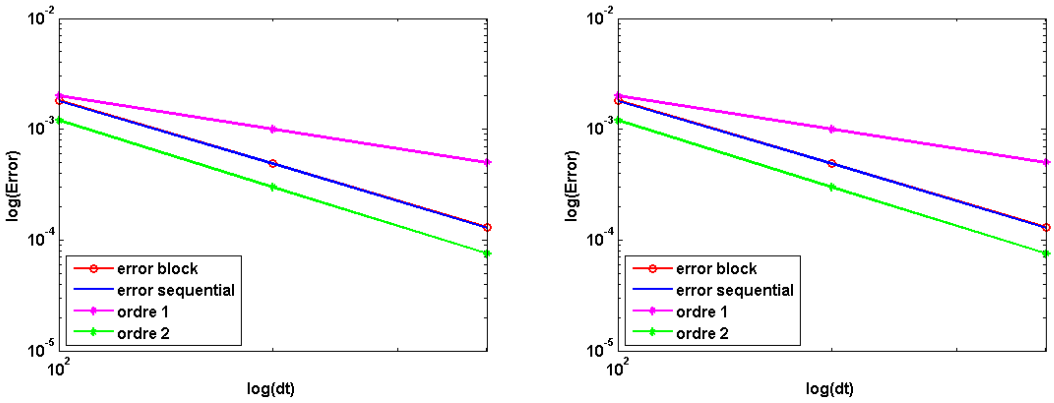

Figure 4: Ordre en temps de la méthode Bloc - CN avec le calcul plein de l'exponentielle à gauche, à droite approximation Arnoldi

N*M = (Nx*Ny)*M	10x10x10	20x20x20	40x40x40
Err. Seq	1.7913e-03	4.9045e-04	1.2860e-04
Err. Block FC	1.8088e-03	4.9483e-04	1.2969e-04
Err. Block H km = 1	1.8088e-03	4.9483e-04	1.2969e-04
Err. Block TA km = 5	12.7632e+07	2.2093e+34	5.6769e+61

Table 1: Erreurs pour le schéma de Crank-Nicolson

Ici, le Err. Seq est l'erreur de la méthode séquentielle, Err. Block FC est l'erreur de la méthode de Bloc avec Calcul plein de l'exponentielle d'une matrice (expm() dans Matlab), Err. Block H km est l'erreur de la méthode de Bloc avec l'approximation d' Arnoldi, Err. Block TA est l'erreur de la méthode de bloc avec calcul Taylor approximation de l'exponentielle de matrice de rang km, km est le paramètre mentionné dans les 2 techniques de moindre coût pour l'approximation de e^{Ldt_i}.

Application de la méthode Bloc pour équation de l'élasticité

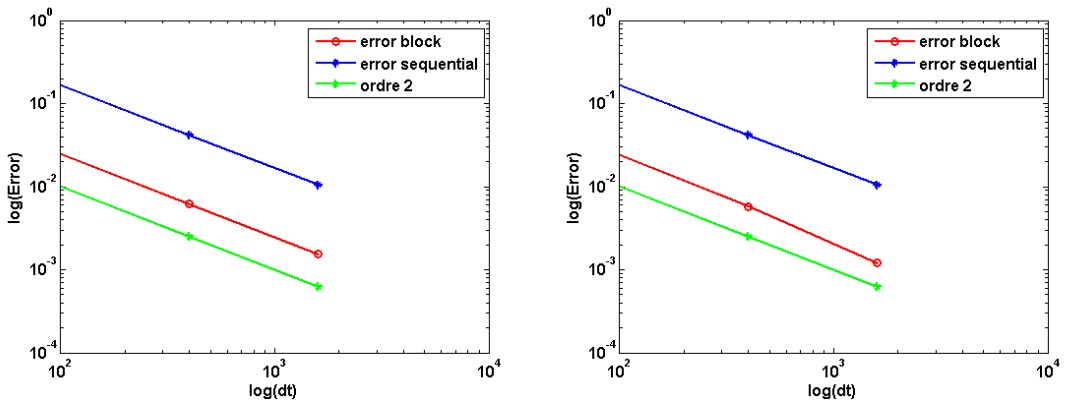

Figure 5: Ordre du temps de la méthode de Bloc - Euler Arnoldi rapprochement avec km = 2 sur la gauche, et Taylor rapprochement km = 2

2*Nx*Ny * Nt	20x10x10	40x20x40	80x40x160
Err. Seq	1.6631e-01	4.1770e-02	1.0443e-02
Err. Block FC of e	2.4396e-02	6.1595e-03	1.5332e-03
Err. Block H km = 1	6.3802e-01	6.3047e-01	6.2861e-01
Err. Block H km = 2	2.4396e-02	6.1595e-03	1.5332e-03
Err. Block TA of e, km = 1	9.0846e-02	7.3424e-02	6.9079e-02
Err. Block TA of e, km = 2	2.4047e-02	5.7981e-03	1.2019e-03

Table 2: Erreurs pour le schéma d'Euler

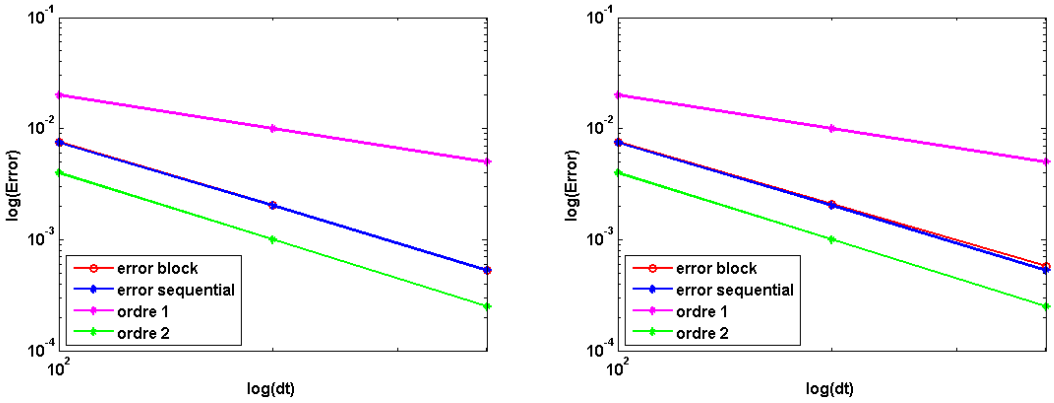

Figure 6: Ordre du temps de la méthode de Bloc - Crank-Nicolson Arnoldi rapprochement avec km = 2 sur la gauche, et Taylor rapprochement km = 2

2*Nx*Ny * Nt	20x10x10	40x20x40	80x40x160
Err. Seq	7.4817e-03	2.0269e-03	5.2838e-04
Err. Block H km = 1	6.3802e-01	6.3047e-01	6.2861e-01
Err. Block H km = 2	7.5318e-03	2.0405e-03	5.3192e-04
Err. Block H km = 3	7.5318e-03	2.0405e-03	5.3192e-04
Err. Block TA of e, km = 1	5.9542e-02	6.5452e-02	6.7079e-02
Err. Block TA of e, km = 2	7.7315e-03	2.2777e-03	8.5812e-04
Err. Block TA of e, km = 3	7.5743e-03	2.0834e-03	5.7499e-04
Err. Block TA of e, km = 4	7.5318e-03	2.0404e-03	5.3185e-04
Err. Block TA of e, km = 5	7.5318e-03	2.0405e-03	5.3191e-04

Table 3: Erreurs pour le schéma de Crank-Nicolson

0.6 Conclusion

En poursuivant le premier objectif de résoudre le système d'équation de la chaleur, j'ai analysé la méthode de produit tensoriel espace-temps avec le schéma d'Euler en temps. Ensuite, j'ai étudié l'erreur de la méthode numérique pour trouver , pour une tolérance d'erreur donnée, le parallélisme optimal de la méthode. Cela a été validé par les applications numériques. Ensuite, je présente la méthode de produit tensoriel espace-temps avec le schéma de Crank Nicolson en temps, ce qui n'a jamais été fait avant pour l'équation de la chaleur et montré qu'elle est stable et possède la précision de l'ordre 2. J'ai également défini le parallélisme optimal de la méthode tenseur - Newmark pour l'équation de la chaleur et le nombre optimal de processeurs

parallèles .
Les résultats numériques montrent que la méthode de produit tensoriel espace-temps fonctionne bien pour le schéma d'Euler et de Crank Nicolson en temps (il peut fonctionner pour n'importe quel schéma décentré), mais en raison du conditionnement de la matrice des vecteurs propres S , le nombre optimal de processeurs n'est pas très élevé: 9 processeurs pour le schéma d'Euler et 8 pour le schéma de Newmark pour une tolérance donnée de l'erreur de 5% .
Cela signifie que si nous voulons avoir un "speed-up" de la méthode , tout en préservant la précision (i.e la série de pas de temps proche du pas de temps fixé (ρ proche à 1), le nombre optimal de processeurs parallèle seront définis , d'où la méthode de produit tensoriel ne peut être appliquée à n'importe quel nombre de processeurs. Toutefois, sachant que la méthode de produit tensoriel résout l' EDP totalement en parallèle, chaque processeur parallèle résout un problème d'espace similaire , on peut toujours introduire une méthode de parallélisation dans l'espace , en combinaison avec la parallélisation en temps pour résoudre l'EDP en ajoutant ainsi une autre dimension au processus de mise en parallèle par le biais des sous-domaines de l'espace-temps complètement parallèles. En outre, en tirant parti des propriétés particulières de la série des pas de temps (avec un taux de croissance en fonction de ρ), cette méthode peut fonctionner de manière très efficace pour des problèmes physiques qui doivent avoir de très petits pas de temps au début et des grand pas de temps à la fin.
Dans la deuxième partie, j'ai présenté la méthode Bloc avec le schéma d'Euler et de Cranck-Nilcolson en temps. Ensuite, j'ai appliqué la méthode de Bloc pour résoudre deux équations aux dérivées partielles différentes: l'équation de la chaleur et l'équation d'élasticité. Les applications en 2D des deux équations aux dérivées partielles donnent de bons résultats validant les applications de la méthode à différentes équations aux dérivées partielles. Différente de la méthode de produit tensoriel, la méthode de Bloc n'a pas de contrainte sur les séries des pas de temps donc c'est une méthode très évolutive qui peut fonctionner pour n'importe quel schéma de temps. Ayant la même structure que la méthode "Pararéel" mais étant une méthode directe rend la méthode de Bloc extensible en utilisant la méthode du produit tensoriel à l'intérieur de chaque pas de temps grossier pour résoudre les problèmes raffinés. La méthode Bloc peut étonnamment fournir une meilleure condition initiale surtout pour les schémas d'ordre faible ainsi qu'une amélioration de la méthode séquentielle. Pour un système d'ordre supérieur, il ne détériore pas l'ordre du schéma et la précision. La combinaison de la méthode Bloc et la méthode produit tensoriel permettra une bonne accélération, ouvrant ainsi avec succès une autre dimension à la parallélisation.

Perspectives

Une étude de la parallélisation optimale de la méthode de produit tensoriel pour l'équation d'élasticité à dérivées d'ordre supérieur est à réaliser.
La combinaison des deux méthodes temps-parallèles est à mettre en oeuvre: utiliser la méthode de Bloc pour résoudre les problèmes grossiers et de produit tensoriel pour résoudre les problèmes raffinés pour avoir une méthode de temps parallèle encore plus directe.

Enfin, une combinaison de la méthode de produit tensoriel ou de Bloc avec une méthode de décomposition en espace pour résoudre un problème complètement parallèle dans les sous-domaines espace-temps offrira une très bonne granularité idéale pour les calculs massivement sur les grands clusters.

Introduction

Context

The domain decomposition methods in space applied to Partial Differential Equations (PDEs) expanded considerably thanks to their effectiveness (memory costs, calculation costs, better local conditioned problems) and this related to the development of massively parallel machines. Domain decomposition in space-time brings an extra dimension to this optimization. Parallelization in time has been developing for the last thirty years with different time-parallel techniques such as a time multigrid method [28], a modified time stepping technique [57], the Parareal method [33], parallelization by space-time blocks [1, 2] or Tensor Product space-time method [42]. Among all these methods, this work is especially interested in the direct methods, and in particular, the Tensor Product space-time method which remains unexploited until now and the Blocks Method because of its similarity with Parareal method.

History

Partial Differential Equations (PDEs) are used to modelize more and more complicated physical phenomena. The invention of multi-core processors in computer science allowed big progress in industrial simulations. In order to use these large scale parallel computers (cluster) effectively, new parallel algorithms had to be developed.
Domain decomposition methods are naturally adapted to run in parallel and especially for complex modeling problems. The concept of these methods can be seen by their names: we structure the problems into subdomains, on each subdomain, all data and computations are handled by a processor of the parallel machine, and there are communications between processors through a special system (for example Message Passing Interface (MPI)). The resolution of the problem is then achieved by iterating between sub-domains and the sub-domains may even be in space-time.

Domain decomposition methods (DDM) in space was first known in 1870 as Schwarz [53] proposed a way of solving a given problem using subdomains with transmission conditions between the subdomains. In the late 80's this method has been re-explored and since then it has been intensively developed stimulated by the invention of parallel machines. DDM are used in many domains such as: automobile, meteorology, astrophysics, medicine, aerospace, etc..., or more complicated problems with different physical properties on different domains. Nearly a 100 years after Schwarz's proposition , Miller proposed numerical analogs to the Schwarz alternating procedure [45], based on using discretised Fourier series to compute simple domains

such as rectangles and disks. But each iteration of this method is expensive. Then in 1988, P. L. Lions proposed a brilliant extension of Schwarz's method [34, 35, 36], highly adapted to parallel computing on clusters and also proved the convergence of Schwarz's method for any number of subdomains. These articles gave a new interest to Schwarz's method, and hence, many developments have followed this path [10, 54, 49, 56]. In 2000, M. J. Gander, L. Halpern, F. Nataf introduced an Optimized Schwarz method [22] a new class of Schwarz methods with greatly enhanced convergence properties and with many applications for differents PDEs: Helmoltz equation [21, 20], Maxwell equation [11], wave equation [19], advection-diffusion equation [26], Navier Stokes equation [9], etc...
A second family of domain decomposition methods in space is based on a Schur decomposition, once called substructuring method.It was introduced by Przemieniecki in 1963 [48] in the context of aeronautical computations. A finite element problem is split into non-overlapping subdomains, and the unknowns in the interiors of the subdomains are eliminated. The remaining Schur complement system on the unknowns associated with subdomain interfaces is solved by the conjugate gradient method [54, 49, 10]. There are two benefits to this method. First, the elimination of the interior unknowns on the subdomains, that is the solution of Dirichlet problems, can be done in parallel. Second, passing to the Schur complement reduces condition number and thus tends to decrease the number of iterations. Three decades after, the Finite Element Tearing and Interconnect method, an outstanding Schur method, was proposed by F. X. Roux and C. Farhat [15, 17]. FETI contains two additional ingredients: natural coarse grid using floating subdomains and the condition number of FETI. This method immediately attracted a lot of attention of researchers and has been extended and used in massively parallel computation to solve PDEs in parallel [16, 31, 14].

Domain decomposition methods in space have now been so expanded that development is coming to a standstill when compared with the improvement of capacity of parallel machines. In order to exploit maximum massively parallel machines, we need to consider another dimension that is parallelization in time of time-dependant PDEs.

Domain decomposition in time was first known long after domain decomposition in space, but quickly developed by researchers. Mainly, there are two kinds of time parallel method: iterative and direct . The first time an iterative parallel method was presented is in the paper "Parallel Methods for Integrating Ordinary Differential Equations" presented by J.Nievergelt in 1964 [47] which eventually developed into the Multiple Shooting method for Boundary Value Problems [30]. Then, in 1967, W. Miranker and W. Liniger introduced "Parallel Methods for the Numerical Integration of Ordinary Differential Equations" [46] as a parallel time stepping method which was developed in 1990 by D. Womble into a time-stepping algorithm for parallel computers. But the potential of time domain decomposition became quite considerable when J-L. Lions, Y. Maday, G. Turinici presented the "Parareal algorithm in Time Discretization of PDEs" in [33] as

a numerical method to solve evolution equation problems in parallel. This method proposed to break the global problem of time evolution into a series of independent evolution problems on smaller time intervals, the method approximates the solution late in time before having full accurate approximations from earlier times. The iterative algorithm is based on a predictor corrector approach that generally converges quite fast, and leads, when very many processors are available, to real time solution procedures. In the flow of development of parallel machine, the Parareal method has received a lot of attention as soon as it appeared. Extensive experiments can be found for fluid and structure problems in [13], for Navier-Stokes equations in [18], and for reservoir simulation in [23]. Several variants of the method have been proposed in [6, 13] and further analysis and stability of the method have been done [43, 44, 55, 7].

Direct time parallel method was first known in 1963 as Tensor product analysis of PDEs in [38]. The idea was that certain multi-dimensional problems can be solved by solving a few one-dimensional problems. Then in 1964, R.Lynch, J.Rice and D.Thomas continued to exploit this method for the direct solution of PDEs [37]. The approach is natural and a classic one. If a problem is separable, then the solution can be expressed in terms of tensor products of solutions of lower dimensional problems. This implies that the matrix involved in the corresponding PDE can be expressed in terms of tensor product of lower order matrices hence much simpler to solve. This leads to a simple and direct method for the analysis of alternating direction implicit scheme [39]. Nevertheless, the fast tensor product solvers of Lynch, Rice and Thomas has limited applicability, but still they are very attractive when applicable.
However, this method wasn't studied much until 2008 when Y. Maday and E. M. Ronquist applied the fast tensor product solvers to problems of three dimensions in space [40]. In the same year, another application of Tensor product solvers to time-dependant PDES was published called Time-space tensor product method which solves a time dependant PDE in a parallel way at all time steps through an algebraic equations system with a constraint on time steps [42]. However, little analysis was available. A short time after, the Spectral discretization in time and space [41] was published using tensor product for a high order approximation in both time and space for the numerical solution of time-dependant PDEs which received a lot of interest.
Different from the Tensor-product time-space method which breaks a global problem into a series of independent evolution problems which can be solved directly and completely parallel, the Block method in [1] - derived from block Boundary Value Methods (BVMs)- has the same way of substructuring the time domain as the Parareal method, but the solution of the evolution problem in each interval of time (with refined time steps) can be solved directly from the algebraic system without doing any iteration as in the Parareal method. Only the application for heat equation with Gear second order backward method in time was presented.

Objective of this Work

In this thesis, two time-parallel methods will be studied with direct solvers: the Tensor-product Space-time method of Y. Maday and E. Rønquist [42] and the Block method of Amodio P. and Brugnano L. [3] to solve different PDEs parallel in time and also perform numerical experiments with Matlab and Fortran (MPI).
In the context that the Tensor-product Space-time method is still unexploited, a study of the stability and the error estimation of the numerical method with different time approximation methods such as implicit Euler backward, Newmark's β method [29] for the Heat equation and an Euler method and Newmark method for the elasticity equation [51] will be performed.
The second time parallel method is applied to solve the Heat equation with Euler and Crank-Nicolson time scheme [27] and the Elasticity equation [51] with Euler and Newmark for the time derivative approximation. The objective of this thesis is to study theoretically and numerically these time parallel method to evaluate and find optimal conditions of the methods for industrial applications in Aeroelastics and vibration acoustics (work in which the DTIM Department of ONERA Palaiseau is involved).

Plan of this Work

All over this work, finite differences are used to discretize the solution of the PDEs, in both space and time, and the applications presented concern simple geometries with Cartesian meshes.
- In the first part is introduced the Tensor product space-time method for the heat equation. Then, we use a first order approximation of the time derivative such as the implicit Euler backward method with the Tensor product space-time method to solve the heat equation. Due to the condition of the Tensor-product method , all time steps need to be different, this is why, in the following chapter, a study of the Error of the sequential method is performed for the ODE. Then, we show that having a condition on the series of time steps makes the eigenvector matrix S of the linear system (due to the diagonalization of the time matrix) a nearly singular matrix and this causes the error of the numerical method. So with the estimation of the sequential error and taking into account errors due to the matrix S, we find a condition on the time series of time steps to ensure that the final error of the numerical method is optimal. In the end of the chapter, numerical applications are shown to validate the proposed optimization algorithm as well as the effectiveness of the Tensor-product space-time method.

The second chapter of this part is dedicated to the Newmark method, which is normally used for elasticity equation, and applied to the heat equation to obtain a second order approximation of the time derivative. This chapter will follow the same structure as the previous chapter, with stability and accuracy of the Crank Nicolson method for the heat equation. Then error due to the diagonalization in the numerical time-parallel method will be studied and the numerical applications in the end will confirm the optimal parameters.

- The second part of this thesis is the application of the Block method. First, we present the Block method for ODEs, then its application to the heat equation using Euler and Crank-Nicolson method to discretize the time derivative. Next, the method will be used to solve the higher order time derivative - elasticity equation using two different time approximation method Euler and Newmark. This direct block method is based on an approximation of initial conditions of the type exponential of a matrix. Three different techniques will be compared: a full computation using the Padé method [4], an Arnoldi -Krylov approximation consisting in projecting the matrix onto a reduced Krylov space and a limited Taylor development of the exponential. Numerical experiments are presented showing the effectiveness of the method.

Part I

Tensor product method for the heat equation

1

Introduction

The tensor-product method has been introduced in [42] as an algebraic method to gain parallelism in time. Suppose a partial differential equation has been discretized in time and space, with dimensions $M \times N$. Suppose the discrete vector of all unknowns in time and space is solution of a large system

$$(B \otimes I_x + I_t \otimes A)U = F. \tag{1.0.1}$$

The vector U is ordered as $U = (U^1, \cdots, U^M)$, $U^i \in \mathbb{R}^N$.
If the matrix B is diagonalizable, with $B = SDS^{-1}$, then the previous equation can be decomposed into

$$\begin{array}{ll} (1) & (S \otimes I_x)G = F, \\ (2) & (\lambda_m + A)V^m = G^m, \quad 1 \leq m \leq M, \\ (3) & U = (S \otimes I_x)V. \end{array}$$

M equations in space can thus be solved independently on the processors. The idea seems very attractive, but the application requires some care:

1. First, obtaining the form (1.0.1) from any scheme is not an easy task (see application to the Newmark scheme below).
2. Then matrix B is not diagonalizable, unless the mesh steps in time are all different.
3. In that case, the precision of the scheme is usually affected compared to the equidistant configuration.
4. Therefore it is better to keep the time steps close to equidistant.
5. Then the condition number of matrix S increases exponentially with M, deteriorating the results of steps (1) and (3).

The purpose of this work is to provide a rigorous method to determine the number of time steps M and the time steps $\Delta t_i = t_{i+1} - t_i$ as a geometric sequence $\Delta t_i = \rho^{i-1} \Delta t_i$. We set an optimization process on the scalar equation $\partial_t u + au = 0$, with $a > 0$. It consists in writing a Taylor series for $\rho = 1 + \varepsilon$ of the error, and of the condition number of S, and equilibrating them. This process will be given in detail for the heat equation with the Euler scheme, and the Crank-Nicolson scheme.

Consider the heat equation with constant thermal diffusivity $\kappa > 0$,

$$\frac{\partial u}{\partial t} - \kappa \Delta u = f \quad \text{in } \Omega \times (0, T] \tag{1.0.2}$$

Dirichlet boundary conditions $g(t)$ are enforced on the boundary Γ of Ω. along with initial condition

$$u(x, 0) = u_0 \text{ in } \Omega.$$

In a semi-discretized in space setting, approximate $u(x, t)$ by $u_h(t)$, and $-\kappa \Delta u$ by Au_h, and obtain the system

$$\begin{cases} \dfrac{\partial u_h}{\partial t} + Au_h = f_h \quad \text{in } \times (0, T], \\ u(x, 0) = u_0. \end{cases} \tag{1.0.3}$$

To be practical, suppose Ω to be an interval in $\mathbb{R}$ of length L, and use a finite difference scheme with mesh $h = L/(N+1)$. The vector u_h of dimension N is an approximation of $(u(h, t), \ldots, u(L-h, t))$ .The matrix A of dimension N is

$$A = \frac{\kappa}{h^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \tag{1.0.4}$$

The right-hand side f_h is computed from f and the boundary conditions as the right hand side below.

$$f_h(1, t) = f(h, t) + \frac{g_l(t)}{h^2}, \quad f_h(N, t) = f(L - h, t) + \frac{g_r(t)}{h^2}. \tag{1.0.5}$$

2

The backward Euler scheme

2.1 Presentation

We first present the method for the implicit Euler scheme, since the form (1.0.1) is then easy to obtain. Introducing a mesh consisting of nodes $0 = t_0 < t_1 < t_2 < \cdots < t_M = T$, $\Delta t_m = t_m - t_{m-1}$, the Euler scheme is

$$\frac{U^m - U^{m-1}}{\Delta t_m} + AU^m = F^m, \quad m = 1, \ldots, M, \tag{2.1.1}$$

where the vector $U^m \in \mathbb{R}^N$ denotes the numerical approximation of the solution at time t_m. The vector $F^m \in \mathbb{R}^N$ represents the given data at the internal grid points at time t_m.

To rewrite the system (2.1.1) as a global system on the concatenated solution vector $U = (U^1, \ldots, U^M) \in \mathbb{R}^{M\times N}$, we rewrite the first iterate as

$$\frac{U^1}{\Delta t_1} + AU^1 = F^1 + \frac{U^0}{\Delta t_1},$$

define the new first right hand side as

$$\tilde{F}^1 = F^1 + \frac{U^0}{\Delta t_1},$$

and the concatenated right-hand side $F = (\tilde{F}^1, F^2, \ldots, F^M) \in \mathbb{R}^{M\times N}$.

Now, we define matrix B to be

$$B = \begin{pmatrix} \frac{1}{\Delta t_1} & & & & \\ \frac{-1}{\Delta t_2} & \frac{1}{\Delta t_2} & & & \\ & \frac{-1}{\Delta t_3} & \frac{1}{\Delta t_3} & & \\ & & \ddots & \ddots & \\ & & & \frac{-1}{\Delta t_M} & \frac{1}{\Delta t_M} \end{pmatrix}. \tag{2.1.2}$$

We also define I_t to be the identity matrix of dimension M (associated with the time domain) and I_x to be the identity matrix of dimension N (associated with the spatial domain). With this notation, we can write equation (2.1.1) in tensor product form as

$$(B \otimes I_x + I_t \otimes A)U = F. \tag{2.1.3}$$

There are several ways to solve this global problem, for example multigrid in time and space (see [25]). Here, having in mind to parallelize in time, we want to diagonalize the matrix B. This is provided by the following lemma.

Lemma I.1

The matrix B is diagonalizable if and only if the step sizes Δt_i are all different. In that case

$$B = S\Lambda S^{-1}, \quad \text{with } \Lambda = diag(1/\Delta t_1, \cdots, 1/\Delta t_M). \tag{2.1.4}$$

Proof : If all time steps Δt_i are different, it easy to see that the matrix B is diagonalizable.

When B is diagonalizable, we suppose that there exists 2 time steps: $\Delta t_i = \Delta t_j$ and $j > i \neq 1$. Now we will compute the eigenvectors that correspond to these eigenvalues: $\frac{1}{\Delta t_i}$ and $\frac{1}{\Delta t_i}$. For any eigenvector v_i of the matrix B associated to eigenvalue Δt_i, we have:

$$Bv_i = \frac{1}{\Delta t_i} v_i.$$

$$\Leftrightarrow \begin{cases} \frac{1}{\Delta t_1} v_{i,1} = \frac{1}{\Delta t_i} v_{i,1}, \\ (\frac{1}{\Delta t_k} - \frac{1}{\Delta t_i}) v_{k,1} = \frac{1}{\Delta t_k} v_{i,k-1}, \ k = 2, \ldots, M. \end{cases} \tag{2.1.5}$$

Because of $\Delta t_i = \Delta t_j$, from (2.1.5), we get

$$\begin{cases} v_{i,k} = v_{j,k} = 0, \ k = 1, \ldots, j-1, \\ v_{i,j} \text{ and } v_{j,j} \text{ are free variable,} \\ v_{i,k} = \frac{\Delta t_k}{\Delta t_j - \Delta t_k} v_{i,j}, \ k = j+1, \ldots, M \\ vj,k = \frac{\Delta t_k}{\Delta t_j - \Delta t_k} v_{j,j}, \ k = j+1, \ldots, M. \end{cases} \tag{2.1.6}$$

Hence vectors v_i and v_j are dependant, therefore the matrix B is not diagonalizable, contradiction. ■

Thus if the time matrix is diagonalizable, then we can rewrite (2.1.1) as

$$(S \otimes I_x)(\Lambda \otimes I_x + I_t \otimes A)(S^{-1} \otimes I_x)U = F \tag{2.1.7}$$

The solution of (2.1.7) can be obtained in 3 steps:

$$\text{Solve } (S \otimes I_x)G = F, \tag{2.1.8}$$

$$\text{Solve } (\Lambda_m + A)W^m = G^m, \text{ for } m = 1, \ldots, M \tag{2.1.9}$$

$$\text{Compute } U = (S \otimes I_x)W. \tag{2.1.10}$$

The process is the same in 2 or 3-D, only the matrix A is changed, and N becomes $N1 \times N2$ or $N1 \times N2 \times N3$. If the matrix S can be computed explicitly, the cost of (2.1.8), (2.1.10) is negligible, and (2.1.9) will be computed in parallel.

By a Fourier transform in space, we can see that an error analysis on a $0 - D$ equation, for $a > 0$ will be relevant. we get the ODE:

$$\frac{d\,u}{d\,t} + au = 0 \tag{2.1.11}$$

We will first compare the sequential error for fixed or variable timestep, and quantify their relative difference. We then study the matrix S, its form and its condition number. We will set a criterion, for which the condition of S does not affect the error.

2.2 Error of the sequential method for the ODE

The ordinary differential equation (2.1.11) is approximated by an implicit backward Euler scheme on the interval $(0, T)$:

$$\frac{u_{n+1} - u_n}{\Delta t_n} + au_{n+1} = 0. \tag{2.2.1}$$

For fixed time step $\Delta\, t = T/M$, the solution of equation (2.2.1) after M iterations is

$$u_M = (1 + a\Delta t)^{-M}\, u_0. \tag{2.2.2}$$

For different time steps $\Delta\, t_m$, the solution of the equation after M iterations is

$$v_M = \prod_{m=1}^{M} (1 + a\Delta\, t_m)^{-1}\, u_0, \tag{2.2.3}$$

and the solution of (2.1.11) at time $T = M\Delta t$ is $u(T) = e^{-aT} u_0$.

For given a and T, define the error propagator

$$Err(a, T, (\Delta\, t_1, \cdots, \Delta\, t_M)) = \prod_{m=1}^{M} (1 + a\Delta\, t_m)^{-1} - e^{-aT}. \tag{2.2.4}$$

Note that $Err > 0$ for all aT and all M, when $\Delta\, t_i = T/M$ for all i.

Lemma I.2

For any a, T and M, the minimum of Err over all partitions $\Delta = \{\Delta t_m\}_{1\le m\le M}$ of $(0,T)$ is obtained when all time steps are equal, ie $\Delta t_m = \frac{T}{M}$.

Proof : This is a minimization problem for $\Phi(\Delta) = \prod_{m=1}^{M}(1+a\Delta t_m)^{-1}$ as a function of $\Delta \in \mathbb{R}^M$, with M affine inequality constrains $\Delta t_m \ge 0$ and one affine equality constraint $\sum_m \Delta t_m = T$. Compute the derivatives of Φ,

$$\frac{\partial\Phi}{\partial\Delta t_i^2}(\Delta) = -\frac{a}{1+a\Delta t_i}\Phi(\Delta).$$

$$\frac{\partial^2\Phi}{\partial\Delta t_i}(\Delta) = \frac{2a^2}{(1+a\Delta t_i)^2}\Phi(\Delta), \quad \frac{\partial^2\Phi}{\partial\Delta t_i\Delta t_j}(\Delta) = \frac{a^2}{(1+a\Delta t_i)(1+a\Delta t_j)}\Phi(\Delta).$$

To show that that Φ is convex, we compute

$$\begin{aligned}\sum_{i,j}\frac{\partial^2\Phi}{\partial\Delta t_i\Delta t_j}(\Delta)X_iX_j &= 2\sum_{i\neq j}\frac{a^2}{(1+a\Delta t_i)(1+a\Delta t_j)}X_iX_j\Phi(\Delta) + 2\sum_i\frac{a^2}{(1+a\Delta t_i)^2}X_i^2\Phi(\Delta)\\ &= \left(\sum_i\frac{aX_i}{1+a\Delta t_i}\right)^2\Phi(\Delta) + \sum_i\left(\frac{aX_i}{1+a\Delta t_i}\right)^2\Phi(\Delta) > 0\end{aligned}$$

Therefore the Kuhn Tucker theorem holds, and the only minimum is such that there exists a Lagrange multiplyer p with

$$\Phi'(\Delta) + p = 0,$$

or equivalently

$$-\frac{a}{1+a\Delta t_i}\Phi(\Delta) + p = 0,$$

which is equivalent to

$$\Delta t_i = \Delta t = T/M,\ \forall\, i = 1,\dots,M, \quad p = a(1+a\Delta t)^{-M+1}.$$

■

Now, consider different time steps:

$$\Delta t_m = \rho^{m-1}\Delta t_1,\ i = 1,\dots,M, \tag{2.2.5}$$

and write

$$Err(a,T,M,\rho) := Err(a,T,(\Delta t_1,\cdots,\Delta t_M)).$$

In order for the error not to be too large, according to Lemma I.2 we need to keep ρ close to 1. Suppose therefore that

$$\rho = 1+\varepsilon. \tag{2.2.6}$$

The following theorem gives an expansion of Err for small ε.

Theorem I.1

Given a, T and M, $\Delta t = T/M$, for $\rho = 1 + \varepsilon$ with ε small enough, we have

$$\begin{aligned} Err(a, T, M, 1 + \varepsilon) &= Err(a, T, M, 1)(1 + \alpha\varepsilon^2 + o(\varepsilon^2)), \\ \alpha(a, T, M) &= \frac{(1 + a\Delta t)^{-M}}{(1 + a\Delta t)^{-M} - e^{-aT}} \frac{b^2 M(M^2 - 1)}{24} \end{aligned} \tag{2.2.7}$$

with $b = \dfrac{a\Delta t}{1 + a\Delta t}$.

Proof : We start by rewritting the time steps as

$$\Delta t_m = \rho^{m-1} \Delta t_1 = \frac{\rho^{m-1}}{\sum_{j=1}^{M} \rho^{j-1}} T = \frac{\rho^m}{\sum_{j=1}^{M} \rho^j} T \tag{2.2.8}$$

We next need the Taylor developpment of $Err(a, T, M, \rho)$ in ε, to do that, we need first replace $\rho = 1 + \varepsilon$ then find the expansion of Δt_m in ε which is given by following lemma:

Lemma I.3

The time step Δt_m has for ε small the expansion

$$\Delta t_m = \Delta t(1 + \alpha_m \varepsilon + \beta_m \varepsilon^2 + o(\varepsilon^2)),$$

with

$$\alpha_m = m - \frac{M + 1}{2}, \quad \beta_m = m(m - M - 2) + \frac{(M + 1)(M + 5)}{6}.$$

They have the properties

$$\sum_m \alpha_m = \sum_m \beta_m = 0, \quad \sum_m \alpha_m^2 = \frac{M(M - 1)(M + 1)}{12}.$$

Proof : Because of (2.2.8), we first need these expansions:

$$\begin{aligned} \rho^m &= 1 + m\varepsilon + \tfrac{m(m-1)}{2}\varepsilon^2 + o(\varepsilon^2), \\ \sum_{j=1}^{M} \rho^j &= M\left(1 + \tfrac{M+1}{2}\varepsilon + \tfrac{(M+1)(M-1)}{6}\varepsilon^2\right) + o(\varepsilon^2), \\ \frac{1}{\sum \rho^j} &= \tfrac{1}{M}\left(1 + \tfrac{M+1}{2}\varepsilon + \tfrac{(M+1)(M-1)}{6}\varepsilon^2 + o(\varepsilon^2)\right)^{-1}. \\ &= \tfrac{1}{M}\left(1 - \tfrac{M+1}{2}\varepsilon + \tfrac{(M+1)(M+5)}{12}\varepsilon^2\right) + o(\varepsilon^2). \end{aligned} \tag{2.2.9}$$

Hence,

$$\begin{aligned}\frac{\rho^m}{\sum \rho^j} &= \frac{1}{M}\left(1 + m\varepsilon + \frac{m(m-1)}{2}\varepsilon^2 + o(\varepsilon^2)\right)\left(1 - \frac{M+1}{2}\varepsilon + \frac{(M+1)(M+5)}{12}\varepsilon^2 + o(\varepsilon^2)\right) \\ &= \frac{1}{M}\left(1 + m\varepsilon + \frac{m(m-1)}{2}\varepsilon^2 - \frac{M+1}{2}\varepsilon(1+m\varepsilon) + \frac{(M+1)(M+5)}{12}\varepsilon^2\right) + o(\varepsilon^2). \\ &= \frac{1}{M}\left(1 + (m - \frac{M+1}{2})\varepsilon + (m(m-M-2) + \frac{(M+1)(M+5)}{6})\frac{\varepsilon^2}{2}\right) + o(\varepsilon^2) \\ &= \frac{1}{M}(1 + \alpha_m\varepsilon + \beta_m\varepsilon^2) + o(\varepsilon^2),\end{aligned} \tag{2.2.10}$$

where

$$\alpha_m \;=\; m - \frac{M+1}{2}, \text{ and, } \beta_m \;=\; m(m-M-2) + \frac{(M+1)(M+5)}{6}. \tag{2.2.11}$$

That gives us some properties of α_m and β_m

$$\begin{aligned}\sum_{i=1}^{M}\alpha_i &= \sum_{i=1}^{M}(i - \frac{M+1}{2}) \\ &= \frac{M(M+1)}{2} - \frac{M(M+1)}{2} = 0.\end{aligned} \tag{2.2.12}$$

Then

$$\begin{aligned}\sum_{i=1}^{M}\alpha_i^2 &= \sum_{i=1}^{M}(i - \frac{M+1}{2})^2 \\ &= \sum_{i=1}^{M}(i^2 - i(M+1) + \frac{(M+1)^2}{4}) \\ &= (\frac{M(M+1)(2M+1)}{6} - \frac{M(M+1)^2}{2} + \frac{M(M+1)^2}{4}) \\ &= \frac{M(M+1)}{2}(\frac{2M+1}{3} - \frac{M+1}{2}) \\ &= \frac{M(M+1)}{2}\frac{(M-1)}{6} = \frac{M(M^2-1)}{12}.\end{aligned} \tag{2.2.13}$$

We also have

$$\begin{aligned}\sum_{i=1}^{M}\beta_i &= \frac{1}{2}\sum_{i=1}^{M}\left(i(i-1) - i(M+1) + \frac{(M+1)(M+5)}{6}\right) \\ &= \frac{1}{2}\left(\frac{M(M+1)(2M+1)}{6} - \frac{M(M+1)}{2} - \frac{M(M+1)^2}{2} + \frac{M(M+1)(M+5)}{6}\right) \\ &= 0.\end{aligned} \tag{2.2.14}$$

From (2.2.10), we get

$$\Delta t_m = \frac{1}{M}(1 + \alpha_m\varepsilon + \beta_m\varepsilon^2) + o(\varepsilon^2)T = \Delta t\,(1 + \alpha_m\varepsilon + \beta_m.\varepsilon^2) + o(\varepsilon^2)$$

■

Lemma I.4

The product term in Err from (2.2.4) has the expansion

$$\prod_{m=1}^{M}(1 + a\Delta\, t_m) = (1 + a\Delta t)^M(1 - \frac{b^2}{2}\sum \alpha_m^2\varepsilon^2 + o(\varepsilon^2))$$

with $b = \dfrac{a\Delta t}{1 + a\Delta t} = \dfrac{\frac{aT}{M}}{1 + \frac{aT}{M}}$.

Proof : We start by introducing the expansion of $\Delta\, t_m$,

$$\begin{aligned}\prod_{m=1}^{M}(1 + a\Delta\, t_m) &= \prod_{m=1}^{M}(1 + a\Delta t(1 + \alpha_m\varepsilon + \beta_m\varepsilon^2 + o(\varepsilon^2)))\\ &= (1 + a\Delta t)^M \prod_{m=1}^{M}(1 + b(\alpha_m\varepsilon + \beta_m\varepsilon^2 + o(\varepsilon^2)))\\ &= (1 + a\Delta t)^M\varphi,\end{aligned}$$

with

$$\varphi = \prod_{m=1}^{M}(1 + b\alpha_m\varepsilon + b\beta_m\varepsilon^2 + o(\varepsilon^2)).$$

We will write the Taylor expansion at order 2 of φ. To do so, we take the natural logarithm;

$$ln(\varphi) = \sum ln(1 + b\alpha_m\varepsilon + b\beta_m\varepsilon^2 + o(\varepsilon^2)),$$

and using that $ln(1 + x) = x - \frac{x^2}{2} + o(x^2)$, we get

$$\begin{aligned}ln(\varphi) &= \sum(b\alpha_m\varepsilon + b\beta_m\varepsilon^2 - \frac{b^2\alpha_m^2}{2}\varepsilon^2) + o(\varepsilon^2)\\ &= (\sum \alpha_m)b\varepsilon + (\sum \beta_m)b\varepsilon^2 - (\sum alpha_m^2)\frac{b^2}{2}\varepsilon^2) + o(\varepsilon^2).\end{aligned}$$

Using Lemma I.3, we obtain

$$ln(\varphi) = -\sum(\frac{b^2\alpha_m^2}{2}\varepsilon^2) + o(\varepsilon^2).$$

Then

$$\varphi = 1 - \frac{b^2}{2}\sum \alpha_m^2\varepsilon^2 + o(\varepsilon^2).$$

■

End of proof of the Theorem (forgetting the arguments a, T and M for simplicity), remind that

$$\begin{aligned}Err(\rho) = Err(1 + \varepsilon) &= \prod_{i=1}^{M}(1 + a\Delta\, t_i)^{-1} - e^{-aT},\\ Err(1+) &= (1 + a\Delta\, t_i)^{-M} - e^{-aT}.\end{aligned}$$

Using the lemma I.4 above, we get

$$\begin{aligned} Err(\rho) &= (1+a\Delta t)^{-M}(1+\frac{b^2}{2}\sum \alpha_m^2\varepsilon^2+o(\varepsilon^2))-e^{-aT} \\ &= Err(1)+(1+a\Delta t)^{-M}\frac{b^2M(M^2-1)}{24}\varepsilon^2+o(\varepsilon^2)) \\ &= Err(1)\left(1+\frac{(1+a\Delta t)^{-M}}{(1+a\Delta t_i)^{-M}-e^{-aT}}\frac{b^2M(M^2-1)}{24}\varepsilon^2+o(\varepsilon^2)\right) \\ &= Err(1)\left(1+\alpha\frac{b^2M(M^2-1)}{24}\varepsilon^2+o(\varepsilon^2)\right). \end{aligned}$$

■

2.3 Computation of the matrix S

Now, consider the time matrix B of M time steps defined in (2.2.5):

$$B=\begin{pmatrix} \frac{1}{\Delta t_1} & & & & \\ \frac{-1}{\Delta t_2} & \frac{1}{\Delta t_2} & & & \\ & \frac{-1}{\Delta t_3} & \frac{1}{\Delta t_3} & & \\ & & \ddots & \ddots & \\ & & & \frac{-1}{\Delta t_M} & \frac{1}{\Delta t_M} \end{pmatrix}=\frac{1}{\Delta t_1}\begin{pmatrix} 1 & & & & \\ \frac{-1}{\rho} & \frac{1}{\rho} & & & \\ & \frac{-1}{\rho^2} & \frac{1}{\rho^2} & & \\ & & \ddots & \ddots & \\ & & & \frac{-1}{\rho^{M-1}} & \frac{1}{\rho^{M-1}} \end{pmatrix}. \tag{2.3.1}$$

The matrix B is diagonalizable, so $B = S\Lambda S^{-1}$, where Λ is a diagonal matrix contains the eigenvalues of the matrix B, and S is the matrix of eigenvectors of B. It is easy to see that

$$\Lambda=\begin{pmatrix} \frac{1}{\Delta t_1} & & & \\ & \frac{1}{\Delta t_2} & & \\ & & \ddots & \\ & & & \frac{1}{\Delta t_M} \end{pmatrix}. \tag{2.3.2}$$

We now compute the matrix S and its inverse. It turns out that they belong to a special family of lower triangular matrices with unit diagonal (idempotent matrices), and furthermore they are Toeplitz, *i.e.* the family of matrices defined by a vector $(X_1,\ldots,X_{M-1})$, given by

$$T(X_1,\ldots,X_{M-1})=\begin{pmatrix} 1 & & & \\ X_1 & \ddots & & 0 \\ X_2 & \ddots & 1 & \\ \vdots & \ddots & \ddots & \ddots \\ X_{M-1} & & X_2 & X_1 & 1 \end{pmatrix}. \tag{2.3.3}$$

Theorem I.2

The matrix S is lower triangular. It can be chosen with unit diagonal, in which case it is of the form (2.3.3), *i.e.*

$$S = T(P_1, \dots, P_{M-1}), \quad , \qquad P_i = p_1 \dots p_i, \quad p_i = \frac{1}{1-\rho^i}. \tag{2.3.4}$$

The inverse S^{-1} is then also of the same form:

$$S^{-1} = T(Q_1, \dots, Q_{M-1}), \qquad Q_i = q_1 \dots q_i, \quad q_i = \frac{\rho^{i-1}}{\rho^i - 1}. \tag{2.3.5}$$

Proof : Since all eigenvalues are simple, the matrix is diagonalizable. We are going to construct the basis of eigenvectors. We search for a vector $X^{(i)}$ such that $BX^{(i)} = \frac{1}{\Delta t_i} X^{(i)} = \frac{1}{\Delta t_1 \rho^{i-1}} X^{(i)}$:

$$X_1^{(i)} = \frac{1}{\rho^{i-1}} X_1^{(i)}, \ \frac{1}{\rho}(X_2^{(i)} - X_1^{(i)}) = \frac{1}{\rho^{i-1}} X_2^{(i)}, \ \cdots, \ \frac{1}{\rho^{i-2}}(X_{i-1}^{(i)} - X_{i-2}^{(i)}) = \frac{1}{\rho^{i-1}} X_{i-1}^{(i)}, \tag{2.3.6}$$

$$\frac{1}{\rho^{i-1}}(X_i^{(i)} - X_{i-1}^{(i)}) = \frac{1}{\rho^{i-1}} X_i^{(i)}, \tag{2.3.7}$$

$$\frac{1}{\rho^{i+j}}(X_{i+j+1}^{(i)} - X_{i+j}^{(i)}) = \frac{1}{\rho^{i-1}} X_{i+j+1}^{(i)}, \ 0 \le j \le M-i-1. \tag{2.3.8}$$

From (2.3.6) we deduce that $X_1^{(i)} = \cdots = X_{i-1}^{(i)} = 0$, from (2.3.7) that $X_i^{(i)}$ is free, and will be chosen equal to 1. Then from (2.3.8) we get that $X_{i+j+1}^{(i)} = p_{j+1} X_{i+j}^{(i)}$, which proves that $X_{i+j}^{(i)} = p_1 \dots p_j$. This gives the required form of S.

The form of S^{-1} relies on the use of the $\rho-$analogue of the binomial formula. For all these notations and for the theory of $\rho-$ analogues, we refer to the book of Gasper and Rahman, *Basic Hypergeometric series* [24].

Define the $\rho-$binomial coefficients as follows:

The $\rho-$analogue of $x \in \mathbb{R}$ is $[x] = \frac{1-\rho^x}{1-\rho}$.

The $\rho-$analogue of $n!$ is $\quad [n!] = \prod_{i=1}^{n} [i] = \prod_{i=1}^{n} \frac{1-\rho^i}{1-\rho}$.

The $\rho-$rising factorial is $\quad (a;\rho)_k = \prod_{i=0}^{k-1} (1-\rho^i a)$, $(\rho)_n = (\rho;\rho)_n$, and $(a;\rho)_\infty = \prod_{i=0}^{\infty} (1-a\rho^i)$.

The $\rho-$analogue of $\binom{n}{k}$ is $\quad \begin{bmatrix} n \\ k \end{bmatrix} = \frac{(1-\rho^n)(1-\rho^{n-1}) \cdots (1-\rho^{n-k+1})}{(1-\rho^k)(1-\rho^{k-1}) \cdots (1-\rho)}$.

With this notation, we can find the form of S^{-1}. First it is easy to see that S^{-1} is unipotent Toeplitz. To establish the form in the theorem is equivalent to prove that

$$\text{For } 1 \le n \le M-1, \sum_{i=0}^{n} P_i Q_{n-i} = 0, \text{with the convention } P_0 = Q_0 = 1. \tag{2.3.9}$$

Insert now the p_j and q_j into sum in (2.3.9)

$$\sum_{i=0}^{n} P_i Q_{n-i} = p_1 \cdots p_n + p_1 \cdots p_{n-1} q_1 + p_1 \cdots p_{n-2} q_1 q_2 + \cdots \\ + p_1 \cdots p_{n-k} q_1 \cdots q_k + \cdots + p_1 q_1 \cdots q_{n-1} + q_1 \cdots q_n.$$

Divide by $p_1 \cdots p_n$:

$$\sum_{i=0}^{n} P_i Q_{n-i} = p_1 \cdots p_n \left(1 + \frac{q_1}{p_n} + \cdots + \frac{q_1 \cdots q_k}{p_{n-k+1} \cdots p_n} + \cdots + \frac{q_1 \cdots q_{n-1}}{p_2 \cdots p_n} + \frac{q_1 \cdots q_n}{p_1 \cdots p_n}\right) \quad (2.3.10)$$

Replace the requested values for p_j and q_j,

$$\sum_{i=0}^{n} P_i Q_{n-i} = p_1 \cdots p_n \left(1 - \frac{1-\rho^n}{1-\rho} + \rho \frac{(1-\rho^n)(1-\rho^{n-1})}{(1-\rho)(1-\rho^2)} \cdots \right. \\ \left. + (-1)^k \rho^{1+\cdots+k-1} \frac{(1-\rho^n)\cdots(1-\rho^{n-k+1})}{(1-\rho)\cdots(1-\rho^k)} + \cdots + (-1)^n \rho^{1+\cdots+n-1}\right).$$

Notice that $1 + \cdots + k - 1 = \binom{k}{2}$, and use the $\rho-$analogue of the binomial coefficient to rewrite

$$\sum_{i=0}^{n} P_i Q_{n-i} = p_1 \cdots p_n \left(1 - \begin{bmatrix} n \\ 1 \end{bmatrix} + \rho \begin{bmatrix} n \\ 2 \end{bmatrix} \cdots + (-1)^k \rho^{\binom{k}{2}} \begin{bmatrix} n \\ k \end{bmatrix} + \cdots + (-1)^n \rho^{\binom{n}{2}}\right)$$

$$\sum_{i=0}^{n} P_i Q_{n-i} = p_1 \cdots p_n \sum_{k=0}^{n} (-1)^k \rho^{\binom{k}{2}} \begin{bmatrix} n \\ k \end{bmatrix}. \quad (2.3.11)$$

This last formula resembles very much the binomial formula, and indeed

Theorem I.3 (*$\rho-$binomial theorem*)

For $|\rho| < 1$, for any positive n, for any $z \in \mathbb{R}$,

$$\sum_{k=0}^{n} (-1)^k \rho^{\binom{k}{2}} \begin{bmatrix} n \\ k \end{bmatrix} z^k = (-z, \rho)_n.$$

Choose now $z = -1$, then for $n \geq 1$, $(-z, \rho)_n = 0$, and we have proved that $\sum_{i=0}^{n} P_i Q_{n-i} = 0$ for $|\rho| < 1$. ■

In the step of resolution of (2.1.8, 2.1.10), the condition number of S has a drastic influence.

2.4 Error due to diagonalization

We recall the 3 main steps of the Tensor product method in (2.1.8), (2.1.9) and (2.1.10):

$$\begin{aligned}&\text{Solve } (S \otimes I_x)G = F,\\ &\text{Solve } (\Lambda_m + A)W^m = G^m, \text{ for } m = 1, \ldots, M\\ &\text{Compute } U = (S \otimes I_x)W.\end{aligned}$$

We can see from the first equation that error of G depends on condition of the matrix S (note that $cond(S \otimes I_x) = cond(S)$)given by the following theorems (See Chapter 3.4, "Analyse numerique matricielle appliquee a l'art de l'ingenieur", P.Lascaux R.Theodor,[32]):

Theorem I.4

Let S be a regular matrix. Let x and $\Delta x + x$ be solutions of linear system:

$$S\,x = b \qquad S\,(x + \Delta x) = b + \Delta b,$$

then

$$\frac{||\Delta x||}{||x||} \leq cond(S)\frac{||\Delta b||}{||b||}.$$

Since errors due to the 2nd and 3rd linear equations in the 3 steps are negligible, then the error of the solution U due to the diagonalization comes from the condition of matrix S.
Therefore, the next time step is to study condition of matrix S. We do this again for $\rho = 1 + \varepsilon$ and ε small.

Lemma I.5

For $\rho = 1 + \varepsilon$, the asymptotic behavior of condition of matrix S delivered in (2.3.4)(in any norm) is

$$cond(S) \sim \frac{1}{((M-1)!)^2\varepsilon^{2(M-1)}}.$$

Proof : We use here the L^1 condition number for simplicity, but the result would be the same with L^2.
We obtain

$$||S||_1 = 1 + |p_1| + |p_1\,p_2| + \cdots + |p_1\,p_2 \ldots p_{M-1}|,$$

with

$$p_i = \left|\frac{1}{1-\rho^i}\right| \sim \frac{1}{i\,\varepsilon}$$

$$|p_1 \ldots p_n| \sim \frac{1}{n!\varepsilon^n}\,.$$

Hence, we get

$$||S||_1 \sim |p_1 \ldots p_{M-1}| \sim \frac{1}{(M-1)!\varepsilon^{M-1}}.$$

Doing the same way for the matrix S^{-1}, we have:

$$\|S^{-1}\|_1 = 1 + |q_1| + |q_1\, q_2| + \cdots + |q_1\, q_2 \ldots q_{M-1}|,$$

with

$$\begin{aligned} q_i &= \left|\frac{\rho^{i-1}}{\rho^i - 1}\right| = \left|\frac{1}{\rho - \frac{1}{\rho^{i-1}}}\right| \\ &\sim \left|\frac{1}{1+\varepsilon - \frac{1}{1+(i-1)\varepsilon}}\right| \sim \left|\frac{1}{1+\varepsilon-(1-(i-1)\varepsilon)}\right| \\ &\sim \frac{1}{i\,\varepsilon}. \end{aligned}$$

$$\Rightarrow |q_1 \ldots q_n| \sim \frac{1}{n!\varepsilon^n}\,.$$

$$\Rightarrow \|S^{-1}\|_1 \sim |q_1 \ldots q_{M-1}| \sim \frac{1}{(M-1)!\varepsilon^{M-1}}.$$

Therefore we obtain for ε small the estimation

$$cond(S) \sim \frac{1}{\left((M-1)!\,\varepsilon^{M-1}\right)^2}$$

■

Knowing that the machine precision of the computer is τ (10^{-16} for double precision in Matlab), as we said in Theorem I.4, using the Tensor-product space-time method, the higher cond(S) is, the more we loose in terms of precision on W, and so we will loose the same precision in the numerical solution U.
Now, having that $\alpha\varepsilon^2$ is the asymptotic precision between the sequential method with dt fixed and sequential method with Δt_i different, see Theorem I.1 , we want that after solving the equation with Tensor product method for Δt_i different we still have the same precision as sequential method with Δt_i different. To obtains that, we need the next Theorem.

Theorem I.5

The condition number of S does not decrease the precision of the computation in the tensor product method if and only if

$$Log(\tau) + Log(cond(S)) = Log\left(\left|\frac{Err(a,T,M,\rho)}{Err(a,T,M,1)}\right|\right). \tag{2.4.1}$$

where the Log is in base 10 and τ is the machine precision. This can be written asymptotically as

$$Log(\tau) + Log(cond(S)) = Log(\alpha(a,T,M)\varepsilon^2), \tag{2.4.2}$$

where α is defined in (2.2.7), and solved in ε as

$$\varepsilon_0(a,T,M) = \left(\frac{\tau}{\alpha(M-1)!^2}\right)^{\frac{1}{2M}}. \tag{2.4.3}$$

Proof : Use the asymptotic values written before, and write

$$Log(\tau) - Log(\,\alpha(M-1)!^2\varepsilon^{2M}) = 0 \iff \frac{\tau}{\alpha(M-1)!^2\varepsilon^{2M}} = 1.$$

This provides the value of ε in (3.6.3). ■

In our Matlab computation, τ is equal to 10^{-16}. In Figure 2.1 on the left we show the left and right hand side in equation (2.4.2) for $a = T = 1$ and $M = 9$ as a function of ε, together with the exact value ε_0. On the right we show the variation of ε_0 a a function of M.

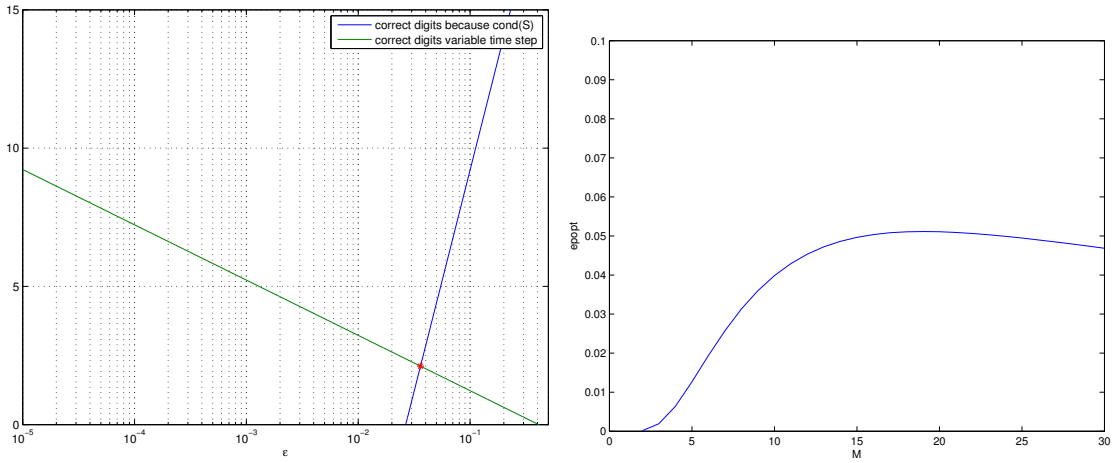

Figure 2.1: Optimization of ε: left and right handsides of (2.4.2) VS variation of $\varepsilon_0(M)$

Considering equation (2.2.7), we see that α is actually a function of the two variables $y = aT$ and M only. Define now

$$\alpha(y,M) = \frac{(1+y/M)^{-M}}{(1+y/M)^{-M} - e^{-y}} \frac{b^2M(M^2-1)}{24},\ \eta(y,M) = \alpha(y,M)\varepsilon_0(y,M)^2,$$
$$\overline{Err}(y,M,\rho) = Err(y,M,1)(1+\eta(y,M)),$$

with $b = \dfrac{y/M}{1+y/M}$.

The following results have been observed with Matlab but not mathematically proved.

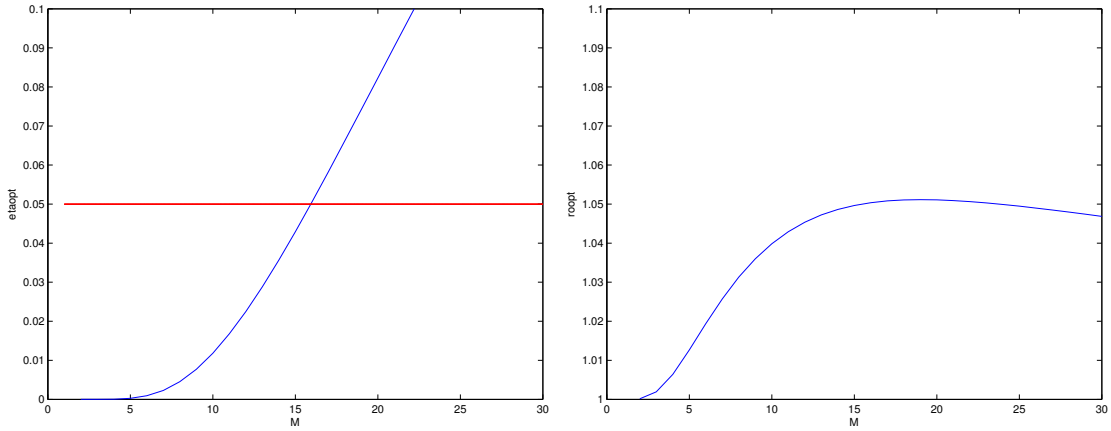

Figure 2.2: Variations of $\eta = \alpha(y, M)\varepsilon_0(y, M)^2$ on the left, of $\bar{\rho} = 1 + \varepsilon_0(y, M)$ on the right, for $y = 1$, as a function of M. In red the reference value $\bar{\eta} = 0.05$

Given y , $\eta(y, M) = \alpha(y, M)\varepsilon_0(y, M)^2$ is an increasing function of M. We thus define, for a given tolerance $\bar{\eta}$, $\bar{M}(y)$ as the largest M such that $\eta(y, M) < \bar{\eta}$. In turn, this gives functions $\bar{\varepsilon}(y) = \varepsilon_0(y, \bar{M}(y))$ and $\bar{\eta}_{opt}(y) = \eta(y, \bar{M}(y))$.

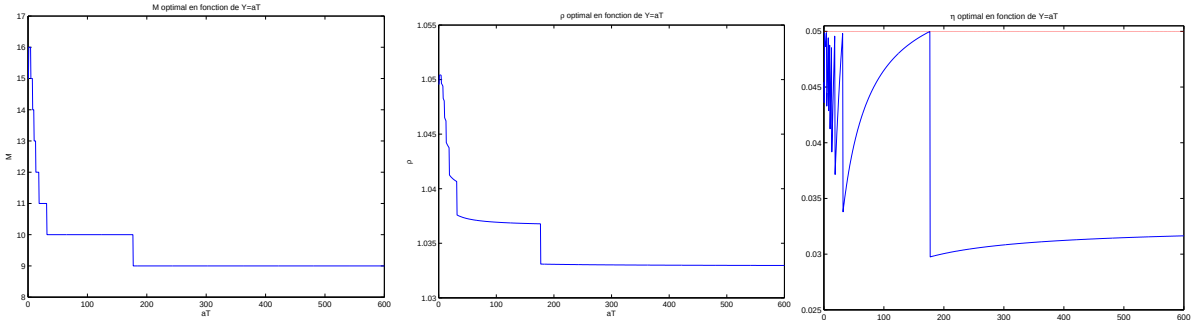

Figure 2.3: $\bar{M}, \bar{\rho}$ and $\bar{\eta}_{opt}$ in functions of $y = aT$ for a tolerance of $\bar{\eta} = 5\%$

We see from Figure 2.3 the optimal $\bar{M}$ (the biggest M, which sastifies $\bar{\eta}_{opt} = \eta(y, \bar{M}) < \bar{\eta}$) is constant over longer and longer range, and it is almost the case for ρ too. They are both constant “at infinity“.

The following tables obtained by Matlab(see Matlab code of "Calcul des parametres optimaux") give the values of $\bar{M}$ and $\bar{\rho}$ by subintervals.

y	$[0, 1.27[$	$[1.27, 4.39[$	$[4.39, 7.48[$	$[7.48, 10.06[$	$[10.06, 13.19[$
M	15	16	15	14	13
ρ	1.0495	1.0504	1.0497	1.0483	1.0466

(2.4.4)

y	$[13.19, 18.33[$	$[18.33, 31.59[$	$[31.59, 177[$	$[177, +\infty[$
M	12	11	10	9
ρ	1.0442	1.0413	1.0376	1.0331

(2.4.5)

2.5 Application in 1D and 2D

Consider now the PDE

$$\frac{\partial u}{\partial t} - \kappa \Delta u = 0.$$

and use a Fourier transform in space

$$\frac{\partial \hat{u}}{\partial t} + \kappa |\xi|^2 \hat{u} = 0.$$

We can apply the results of the preceding section with $a = \kappa|\xi|^2$. Since M and ε are increasing functions of aT, we can proceed as follows. The discrete frequencies are smaller than $\Xi = \pi^2/h_1^2 + \pi^2/h_2^2$ (h_1 and h_2 are space steps), therefore we compute $a_{max} = \kappa\Xi$ and define ε and ρ by choosing in the tables above for $a = a_{max}$. In general, the space step $h \leq 10^{-1}$, then $y = a_{max}T$ usually greater than 177 therefore the relevant values will be $M = 9$ and $\rho = 1.0331$ (see (2.4.5)).

2.5.1 The one-dimensional problem

Consider the exact solution of the heat equation in one dimension ($\kappa = 1$):

$$u(x,t) = sin(\pi x)(sin(\pi t) + \exp^{-\pi^2 t}), \;\; x \in [0, L], \; t \in [0, T = 0.2],$$

with righthand side

$$f(x,t) = \pi^2(sin(\pi t)sin(\pi x)) + \pi sin(\pi x)cos(\pi t)).$$

Denote by N the number of discretized points in space. Then the biggest discrete frequency is

$$a_{max} = \frac{\pi^2}{h^2} = \pi^2 \times L^2 \times (N+1)^2.$$

Then solving the heat equation with an optimal number of time step and optimal ρ optimal from 2.4.5, we have

N	$a_{max}T$	M	ρ	Eseqf	Eseqv	Etensor	η
5	71.0612	10	1.0376	0.0262	0.0261	0.0261	0.0038
10	238.44	9	1.0331	0.0208	0.0208	0.0208	0.0032
100	2013.6	9	1.0331	0.0179	0.0178	0.0178	0.0036

Table 2.1: Results for Euler scheme in 1D

In the Table 2.1, $Eseqf$ is the error in the 2 norm of the numerical solution with the sequential Euler method for the heat equation with constant time steps, $Eseqv$ is the error in the 2 norm of the numerical solution with the sequential Euler method for the heat equation for different time steps and $Etensor$ is the error in the 2 norm of the numerical solution with Tensor product method for different time steps, and η is defined by

$$\eta = |\frac{Etensor}{Eseqf} - 1|.$$

2.5.2 The two-dimensional problem

Consider the exact solution of the heat equation in two dimensions ($\kappa = 1$):

$$u(x,t) = sin(\pi x_1)sin(\pi x_2)(sin(\pi t) + \exp^{-2\pi^2 t}),\ \ x \in [0,1.], \times [0,1.]\, t \in [0, T = 0.2];$$

with right hand side

$$f(x,t) = sin(\pi x_1)sin(\pi x_2)(\pi cos(\pi t) + 2\pi^2 sin(\pi t)).$$

Denote by $N := N_1 = N_2$ the number of points in the x_1 and x_2 direction, then the largest discrete frequency is

$$a_{max} = \frac{\pi^2}{h_1^2} + \frac{\pi^2}{h_2^2} = 2\pi^2(N+1)^2.$$

Then solving the heat equation with an optimal number of time step and optimal ρ optimal from 2.4.4 and 2.4.5, we have

$N := N_1 = N_2$	$a_{max}T$	M	ρ	Eseqf	Eseqv	Etensor	η
5	142.12	10	1.0376	0.0136	0.0135	0.0135	0.0033
10	477.68	9	1.0331	0.0100	0.0099	0.0099	0.0053
100	4027.2	9	1.0331	0.0081	0.0081	0.0081	0.0042

Table 2.2: Results for Euler scheme in 2D

Then the values of η from the Table 2.2 are all inferior to 5% at the optimal point of M and ρ that validates our study of error of the Tensor product method with Euler backward scheme in time.
We can see the validation of this optimal point better in Figure 2.4 where we show the value of η in the neighboorhood of the optimal M and ρ.

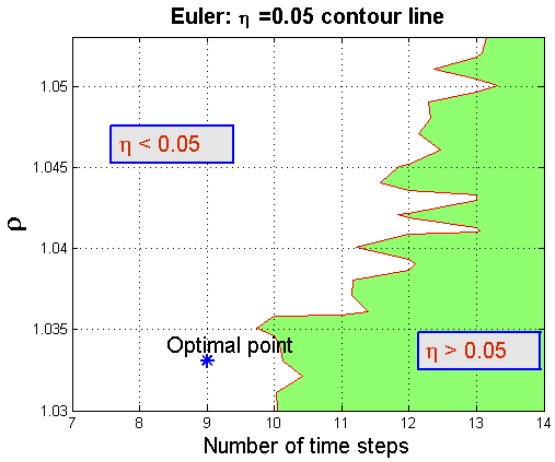

Figure 2.4: $\bar{\eta}$ of Euler as function of ρ and M

In the previous part, we used a first order scheme in time for the heat equation with the Tensor product method. Next, we will use a higher order scheme in time to solve the heat equation. We will study the application of the Newmark's method to the heat equation and then study the error for a particular case of the Newmark's method i.e. the Crank Nicolson method.

3

Newmark method for the heat equation

3.1 Introduction

The Newmark method (named after Nathan M. Newmark [1], former professor of Civil Engineering at the University of Illinois, who developed it in 1959 for use in Structural dynamics) is a method of numerical integration used to solve the wave equation. It's use for the heat equation is unusual, but it will help us it to write high order tensor product methods.
We start from (1.0.3), and introduce three unknowns: U^m denotes the numerical approximation of $u_h(t_m)$, $\dot{U}^m$ is the numerical approximation of the velocity at time t_m, and $\ddot{U}^m$ is the numerical approximation of the acceleration at time t_m. The method uses two parameters β and γ in $[0, 1]$. Two equations are obtained by writing the Taylor series of u and $\dot{U}$ at time t_m, followed by an affine approximation of the second derivative. The third one is the same heat equation dealt with in the previous chapter, with A is the Laplacian matrix:

$$\dot{U}^m = \dot{U}^{m-1} + \Delta t_m \left((1-\gamma)\, \ddot{U}^{m-1} + \gamma\, \ddot{U}^m\right), \tag{3.1.1}$$

$$U^m = U^{m-1} + \Delta t_m\, \dot{U}^{m-1} + \frac{\Delta t_m^2}{2}\left((1-2\beta)\ddot{U}^{m-1} + 2\beta\ddot{U}^m\right), \tag{3.1.2}$$

$$\dot{U}^m + AU^m = F^m. \tag{3.1.3}$$

Note that for $\gamma = 2\beta$, the mean acceleration $(1-\gamma)\ddot{U}^m + \gamma\ddot{U}^{m-1}$ can be eliminated from (3.1.2) and (3.1.3), yielding

$$U^m - U^{m-1} = \frac{\Delta t_m}{2}(\dot{U}^{m-1} + \dot{U}^m),$$

which gives the Crank-Nicolson scheme

$$U^m - U^{m-1} + \frac{\Delta t_m}{2} A(U^m + U^{m-1}) = \frac{1}{2}(F^m + F^{m-1}). \tag{3.1.4}$$

In general, the Newmark scheme is an implicit system of equations. It can be solved as follows. We can replace the velocities in (3.1.1) and (3.1.2) using (3.1.3) at times $m-1$ and m, yielding

$$AU^m + \Delta t_m \gamma \ddot{U}^m - AU^{m-1} + \Delta t_m (1-\gamma) \ddot{U}^{m-1} = F^m - F^{m-1} \tag{3.1.5}$$

$$-U^m + \beta \Delta t_m^2 \ddot{U}^m + (I - \Delta t_m A) U^{m-1} + \frac{\Delta t_m^2}{2}(1-2\beta)\ddot{U}^{m-1} = -\Delta t_m F^{m-1}. \tag{3.1.6}$$

This is the form we use in the stability analysis below. For implementation, $\ddot{U}^m$ can easily be eliminated from (3.1.5,3.1.6) , which gives a single equation for U^m:

$$(\gamma I + \beta \Delta t_m A) U^m = (\gamma I + (\beta - \gamma) A \Delta t_m) U^{m-1} - (\beta - \gamma/2) \Delta t_m^2 \ddot{U}^{m-1} + \Delta t_m (\beta F^m - (\beta - \gamma) F^{m-1}). \tag{3.1.7}$$

This problem is well-conditioned and can be solved by usual methods, such as the Gauss method. Then $\dot{U}^m$ can be computed from (3.1.3) and finally $\ddot{U}^m$ can be computed from (3.1.1) . It remains to give initial conditions. They are

$$U^0 = u_0, \quad \dot{U}^0 = -A\,U^0 + F(0,:), \ \ddot{U}^0 = -A(A\,U^0 + F(0,:)) + \dot{F}(0,:), \tag{3.1.8}$$

with $\dot{F}(0) = \frac{\partial f}{\partial t}(x,0)$.

3.2 Stability and accuracy of the Newmark method for the heat equation

Since the Newmark method has been designed and analyzed for the wave equation, there was no known proof of stability of the Newmark's method for the heat equation , so it is worthwhile analyzing it for the heat equation.

Let's start with the stability of the Newmark's method.

Theorem I.6

The Newmark method is unconditionally stable for the heat equation if and only if $\gamma \geq \frac{1}{2}$ and $2\beta \geq \gamma$.

Proof : We use the displacement/acceleration form in (3.1.5, 3.1.6). By Fourier transform in space, the scheme is unconditionally stable if and only if, for all frequencies ω, the solution with zero right-hand side is decaying.

$$\dot{U} + \omega^2 U = 0. \tag{3.2.1}$$

Replace (3.2.1) into the 2 equations of the Newmark's method (3.1.1) and (3.1.2) at time steps m and $m+1$, we can rewrite the scheme as:

$$\begin{cases} U^{m+1} - \beta dt^2 \ddot{U}^{m+1} &= (1 - dt\omega^2)U^m + (\frac{1}{2} - \beta)dt^2\ddot{U}^m, \\ \omega^2 U^{m+1} + \gamma dt\ddot{U}^{m+1} &= \omega^2 U^m + (\gamma - 1)dt\ddot{U}^m. \end{cases} \tag{3.2.2}$$

This is equivalent to

$$\begin{pmatrix} 1 & -\beta dt^2 \\ \omega^2 & \gamma dt \end{pmatrix} \times \begin{pmatrix} U^{m+1} \\ \ddot{U}^{m+1} \end{pmatrix} = \begin{pmatrix} 1 - dt\omega^2 & (\frac{1}{2} - \beta)dt^2 \\ \omega^2 & (\gamma - 1)dt \end{pmatrix} \times \begin{pmatrix} U^m \\ \ddot{U}^m \end{pmatrix}. \tag{3.2.3}$$

For simplicity, we put $\underline{F} = 0$.
In equation (3.2.3), all unknown vectors U and $\ddot{U}$ at time t_{m+1} depend on the known vectors at time t_m, hence we can consider U and $\ddot{U}$ together as an unknown vector $\underline{U}$:

$$\underline{U} = \begin{pmatrix} U \\ \ddot{U} \end{pmatrix},$$

and we aslo put

$$A_1 = \begin{pmatrix} 1 & -\beta dt^2 \\ \omega^2 & \gamma dt \end{pmatrix},$$

and

$$A_2 = \begin{pmatrix} 1 - dt\omega^2 & (\frac{1}{2} - \beta)dt^2 \\ \omega^2 & (\gamma - 1)dt \end{pmatrix}.$$

Then, equation (3.2.3) can be written by

$$A_1\underline{U}^{m+1} = A_2\underline{U}^m. \tag{3.2.4}$$

The Newmark method for the heat equation is stable when, for all ω, the eigenvalues of matrix $A_1^{-1}A_2$ sastify:

$$\begin{cases} \lambda_1(\omega) \neq \lambda_2(\omega) : \ \|\lambda_1(\omega)\| \leq 1, \ \|\lambda_2(\omega)\| \leq 1; \\ \lambda_1(\omega) = \lambda_2(\omega) : \ \|\lambda_1(\omega)\| = \|\lambda_2(\omega)\| < 1. \end{cases} \tag{3.2.5}$$

This is equivalent with solutions λ of the characteristic equation associated to $Det(A_1^{-1}A_2 - \lambda I) = 0$ that satisfy condition 3.2.5,

$$Det(A_1^{-1}A_2 - \lambda I) = 0 \ \Leftrightarrow \ det(A_2 - \lambda A_1) = 0. \tag{3.2.6}$$

$$\Leftrightarrow (\gamma + \beta dt\omega^2)\lambda^2 + (1 - 2\gamma + (\frac{1}{2} + \gamma - 2\beta)dt\omega^2)\lambda - 1 + \gamma + (\frac{1}{2} - \gamma + \beta)dt\omega^2 = 0. \tag{3.2.7}$$

Put

$$P(\lambda) = a\lambda^2 + b\lambda + c, \tag{3.2.8}$$

where

$$\begin{aligned} a &= \gamma + \beta y, \\ b &= 1 - 2\gamma + (\frac{1}{2} + \gamma - 2\beta)y, \\ c &= -1 + \gamma + (\frac{1}{2} - \gamma + \beta)y, \\ y &= dt\omega^2. \end{aligned} \tag{3.2.9}$$

We define

$$\hat{P} = c\lambda^2 + b\lambda + a. \tag{3.2.10}$$

We want to have the condition on β and γ that for all $y > 0$, the two roots of $P(\lambda)$ are either different and less than or equal to one or the same with modulus less than one.
Remind that a polynomial that has all its roots lying in the open unit disk is a Schur polynomial.
We recall a well-known result on Schur polynomials [50]

Theorem I.7

P is a Schur polynomial if and only if
- $|\hat{P}(0)| > |P(0)|$,
- $P_1 = \frac{1}{\lambda}\left[\hat{P}(0)P(\lambda) - P(0)\hat{P}(\lambda)\right]$ is a Schur polynomial.

Now we will find the condition on β and γ that for all $y > 0$ $P(\lambda)$ is a Schur polynomial by using Theorem I.7.

First, we have

$$P(0) = c, \quad \hat{P}(0) = a.$$

Then we compute

$$\begin{aligned} P_1 &= \frac{1}{\lambda}\left[a(a\lambda^2 + b\lambda + c) - c(c\lambda^2 + b\lambda + a)\right] \\ &= (a^2 - c^2)\lambda + b(a - c) = (a - c)\left((a + c)\lambda + b\right) \end{aligned} \tag{3.2.11}$$

so, after Theorem I.7, P is a Schur polynomial if

$$\forall\, y \begin{cases} |a| > |c|, \\ |b| < |a + c|. \end{cases} \tag{3.2.12}$$

We start by solving the first inequation of (3.2.12)

$$|a| > |c|.$$

$$\Leftrightarrow\ -a < c < a.$$

Replace a, b, c by the terms in (3.2.9), we have

$$\Leftrightarrow\ -(\gamma + \beta y) < y(\beta - \gamma + \frac{1}{2}) + \gamma - 1 < \gamma + \beta y.$$

$$\Leftrightarrow \begin{cases} y(2\beta - \gamma + \frac{1}{2}) + 2\gamma - 1 > 0 \\ y(\frac{1}{2} - \gamma) - 1 < 0. \end{cases} \tag{3.2.13}$$

this is true for any y nonzero, if and only if

$$\begin{cases} 2\gamma - 1 > 0 \\ 2\beta - \gamma + \frac{1}{2} > 0 \\ \frac{1}{2} - \gamma < 0. \end{cases} \tag{3.2.14}$$

or

$$\Leftrightarrow \begin{cases} \gamma > \frac{1}{2} \\ 2\beta - \gamma + \frac{1}{2} > 0. \end{cases} \tag{3.2.15}$$

We suppose that this is true from now on. So $a + c > 0$.

Then we solve the second inequation of (3.2.12)

$$|b| < |a + c|.$$

$$\Leftrightarrow \ -a - c < b < a + c$$

Replace a, b, c in in (3.2.9), we get

$$-(\gamma + \beta y + y(\beta - \gamma + \frac{1}{2}) + \gamma - 1) < (\frac{1}{2} - \gamma + 2\beta)y + 1 - 2\gamma < \gamma + \beta y + y(\beta - \gamma + \frac{1}{2}) + \gamma - 1.$$

$$\Leftrightarrow \begin{cases} 2(\gamma - 2\beta)y + 2(1 - 2\gamma) < 0 \\ y > 0 \end{cases} \tag{3.2.16}$$

Then (3.2.16) is true for all y nonzero if and only if

$$\begin{cases} \gamma - 2\beta < 0 \\ 1 - 2\gamma < 0 \end{cases} \tag{3.2.17}$$

$$\begin{cases} \gamma < 2\beta, \\ \frac{1}{2} < \gamma. \end{cases} \tag{3.2.18}$$

Therefore, from (3.2.12), (3.2.15) and (3.2.18), the polinomial $P(\lambda)$ is a Schur polynomial if and only if

$$\frac{1}{2} < \gamma < 2\beta, \tag{3.2.19}$$

That means the two eigenvalues of the matrix $A_1^{-1}A_2$ have modulus less than one for all $y > 0$ if and only if we have (3.2.19).

Now, the question is what happens when $y = 0$. If $y = 0$, we have

$$\begin{aligned} P(\lambda) &= \gamma\lambda^2 + (1 - 2\gamma)\lambda + \gamma - 1 \\ &= \gamma(\lambda - 1)^2 + \lambda - 1 \\ &= (\lambda - 1)(\gamma(\lambda - 1) + 1). \end{aligned} \tag{3.2.20}$$

It's easy to see that $P(\lambda) = 0$ has 2 distinct solution 1 and $1 - \frac{1}{\gamma}$ which are in the unit disk if and only if $\gamma > \frac{1}{2}$. Therefore, the condition (3.2.19) is true for any y.

Now we will consider the limit cases of Theorem I.6.

First, if $\gamma = \frac{1}{2}$, and $2\beta \geq \frac{1}{2}$, then

$$P(\lambda) = (\beta y + \frac{1}{2})\lambda^2 + y(1 - 2\beta)\lambda - \frac{1}{2} + \beta y.$$

Hence, the discriminant of equation $P(\lambda) = 0$ is

$$d \ = \ 1 - (4\beta - 1)y^2.$$

When $\beta > \frac{1}{4}$
For $y^2 < \frac{1}{4\beta-1}$, we have $d > 0$, then there are 2 roots of $P(\lambda)$
- If $y = 0$: $P(\lambda)$ has two roots are $+/-1$. It sastifies the condition of the method.
- If $y \neq 0$, we have

$$\lambda_1\lambda_2 = \frac{\beta y - \frac{1}{2}}{\beta y + \frac{1}{2}}$$

Then
If $\beta y > \frac{1}{2}$, $0 < \lambda_1\lambda_2 < 1$
If $\beta y < \frac{1}{2}$, $-1 < \lambda_1\lambda_2 < 0$
Hence $\lambda_1\lambda_2 \in (-1, 1)$, so it sastifies the stability condition of the method.
For $y^2 > \frac{1}{4\beta-1}$, we have two conjugated complex roots $\lambda_1\lambda_2 = \lambda^2$, so

$$0 < |\lambda^2| = \frac{\beta y - \frac{1}{2}}{\beta y + \frac{1}{2}}$$

When $\beta = \frac{1}{4}$

$$\lambda_1 = -1, \ \lambda = \frac{2 - y}{2 + y} \in]-1, 1]$$

(Note that the discretized frequencies are bounded.)
Second,if $\gamma = 2\beta, \beta \neq \frac{1}{4}, \beta > \frac{1}{4}$ then the eigenvalues are

$$\lambda_1 \ = \ -\frac{y-2}{y+2}, \lambda_2 \ = \ \frac{1}{2}\frac{2\beta - 1}{2\beta} \ \in \ (-\frac{1}{2}, \frac{1}{4})$$

$\in]-1, 1]$ sastify the condition of the Theorem I.6 so the method is stable. ■

As for the order of the Newmark method

$$\begin{cases} \gamma = 2\beta : \text{ second order method;} \\ \gamma \neq 2\beta : \text{ first order method.} \end{cases} \tag{3.2.21}$$

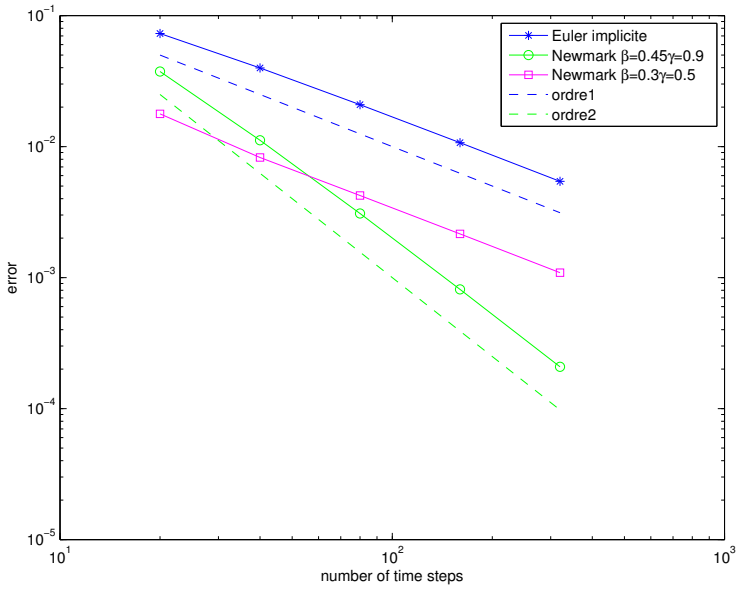

Figure 3.1: Order in time of the Newmark method for small space steps in one dimension

We now proceed to the same analysis as for the Euler scheme, for the special case of Newmark's method: Crank Nicolson scheme (*i.e.* $\gamma = 2\beta$), considering variable time steps with ρ close to 1.

3.3 Error of variable time step for the method Crank Nicolson

The reference equation is $\frac{du}{dt} + au = 0$, discretized by

$$\frac{u^m - u^{m-1}}{\Delta t_m} + \frac{a}{2}(u^m + u^{m-1}) = 0, \tag{3.3.1}$$

which is effectively unconditionally stable and second order in time.
We now redo the computation as in Euler scheme. We have

For a fixed time step $\Delta t = T/M$, the solution of equation (3.3.1) after M iterations is

$$u_M = \left(\frac{1 - a\frac{\Delta t}{2}}{1 + a\frac{\Delta t}{2}}\right)^M u_0 \tag{3.3.2}$$

For different time steps Δt_m, the solution of the equation after M iterations is

$$v_M = \prod_{m=1}^{M} \frac{1 - a\frac{\Delta t_m}{2}}{1 + a\frac{\Delta t_m}{2}} u_0 \tag{3.3.3}$$

and the solution of (2.1.11) at time $T = M\Delta t$ is $u(T) = e^{-aT}u_0$.
The error propagator is now

$$Err(a, T, (\Delta t_1, \cdots, \Delta t_M)) = \prod_{m=1}^{M} \frac{1 - a\frac{\Delta t_m}{2}}{1 + a\frac{\Delta t_m}{2}} - e^{-aT} \tag{3.3.4}$$

Lemma I.6

For any a, T and M, the only critical point of Err over all partitions $\Delta = \{\Delta t_m\}_{1 \le m \le M}$ of $(0, T)$ is obtained when all time steps are equal, ie $\Delta t_m = \frac{T}{M}$.

Proof : Let $\Delta = \{\Delta t_m\}_{1 \le m \le M}$, and $\tilde{\Delta} = \{\Delta t_m\}_{1 \le m \le M-1}$,
Define $\Delta t_M(\tilde{\Delta}) = T - \sum_{1}^{M-1} \Delta t_m$, and

$$\Phi(\tilde{\Delta}) = \prod_{m=1}^{M} \frac{1 - a\frac{\Delta t_m}{2}}{1 + a\frac{\Delta t_m}{2}}.$$

as a function of $\tilde{\Delta} \in \mathbb{R}^{M-1}$. Compute the partial derivatives of Φ,

$$\frac{\partial \Phi}{\partial \Delta t_j}(\tilde{\Delta}) = -\frac{a^3}{4} \prod_{m=1, m \ne j}^{M-1} \frac{1 - a\frac{\Delta t_m}{2}}{1 + a\frac{\Delta t_m}{2}} \frac{\Delta t_M^2 - \Delta t_m^2}{(1 + a\frac{\Delta t_m}{2})^2 (1 + a\frac{\Delta t_M}{2})^2}.$$

They all vanish if and only if all time steps are equal:

$$\Delta t_i = \Delta t = T/M,$$

■

However, the equal partition is not always a local minimum, as the asymptotic study will show. Consider time steps :

$$\Delta t_m = \rho^{m-1} \Delta t_1, \ m = 1, \ldots, M,$$

and write

$$Err(a, T, M, \rho) := Err(a, T, (\Delta t_1, \cdots, \Delta t_M))$$

Suppose

$$\rho = 1 + \varepsilon \tag{3.3.5}$$

Theorem I.8

Given a, T and M, $\Delta t = T/M$, for ε small enough, $\rho = 1 + \varepsilon$

$$Err(a, T, M, 1 + \varepsilon) = Err(a, T, M, 1)(1 + \alpha^N \varepsilon^2),$$

$$\alpha^N(a, T, M) = -\frac{2x^3}{(1 - x^2)^2} \frac{M(M^2 - 1)}{12} \left(\frac{1 - x}{1 + x}\right)^M \frac{1}{Err(a, T, M, 1)} \tag{3.3.6}$$

with $x = \dfrac{a\Delta t}{2}$.

Proof : We use Lemmas I.3 and I.4 with $a = a/2$ et $a = -a/2$

$$\prod_{i=1}^{M}(1 + \frac{a\Delta\, t_i}{2}) = (1 + \frac{a\Delta t}{2})^M(1 - \frac{b_+^2}{2}\sum \alpha_m^2\varepsilon^2 + o(\varepsilon^2))$$
$$\prod_{i=1}^{M}(1 - \frac{a\Delta\, t_i}{2}) = (1 - \frac{a\Delta t}{2})^M(1 - \frac{b_-^2}{2}\sum \alpha_m^2\varepsilon^2 + o(\varepsilon^2))$$

$$b_\pm = \frac{\dfrac{a\Delta t}{2}}{1 \pm \dfrac{a\Delta t}{2}}.$$

Insert this into $Err(a, T, M, \rho)$:

$$Err(a, T, M, \rho) = Err(a, T, M, 1) + \left(\frac{1 - \dfrac{a\Delta t}{2}}{1 + \dfrac{a\Delta t}{2}}\right)^M \frac{b_+^2 - b_-^2}{2}\sum \alpha_m^2\varepsilon^2 + o(\varepsilon^2)$$

$$b_+^2 - b_-^2 = -4\frac{x^3}{(1 - x^2)^2}$$

$$Err(a, T, M, \rho) = Err(a, T, M, 1) - 2\frac{x^3}{(1 - x^2)^2}\left(\frac{1 - \dfrac{a\Delta t}{2}}{1 + \dfrac{a\Delta t}{2}}\right)^M \sum \alpha_m^2\varepsilon^2 + o(\varepsilon^2)$$

■

Note that unlike the Euler case, there is no sign to $Err(a, T, M, 1)$, since it depends on the sign of $1 - \dfrac{a\Delta\, t}{2}$. For fixed a, $\Delta\, t$ can be chosen so that it is positive, but this is quite different when applied to a PDE.

3.4 Tensor product with Newmark time scheme for heat equation

Now we will use the Newmark approach for the heat equation to have a global algebraic problem then we shall apply the tensor product method to solve it.
The main goal of this part is to use equations (3.1.1,3.1.2) to find $\dot{U}$ in function of U then replace it into the third equation (3.1.3) to have an algebraic global system as in the Euler method.

Theorem I.9

The tensor-product form of the Newmark scheme is

$$(B \otimes \mathbb{I}_x)U + (\mathbb{I}_x \otimes A)U = \tilde{F}, \tag{3.4.1}$$

$$\begin{aligned} B &= (B_2 + B_3 B_5^{-1} B_4)^{-1} B_1. \\ \tilde{F} &= F - (B_2 + B_3 B_5^{-1} B_4)^{-1}((B_3 B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_{-1} - \underline{f}_{-2}). \end{aligned} \tag{3.4.2}$$

$$B_1 = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & & & \\ \Delta t_2 & 0 & & \\ & \ddots & \ddots & \\ & & \Delta t_M & 0 \end{pmatrix},$$

$$B_3 = \begin{pmatrix} \beta\Delta t_1^2 & & & \\ (\frac{1}{2} - \beta)\Delta t_2^2 & \beta\Delta t_2^2 & & \\ & \ddots & \ddots & \\ & & (\frac{1}{2} - \beta)\Delta t_M^2 & \beta\Delta t_M^2 \end{pmatrix},$$

$$B_4 = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \quad B_5 = \begin{pmatrix} \gamma\Delta t_1 & & & \\ (1-\gamma)\Delta t_2 & \gamma\Delta t_2 & & \\ & \ddots & \ddots & \\ & & (1-\gamma)\Delta t_M & \gamma\Delta t_M \end{pmatrix},$$

$$\underline{f}_{-1} = \begin{pmatrix} \dot{U}^0 + (1-\gamma)\Delta t_1 \ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \underline{f}_{-2} = \begin{pmatrix} U^0 + \Delta t_1 \dot{U}^0 + (\frac{1}{2} - \beta)\Delta t_1^2 \ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Proof : Writing the first equation (3.1.1) at all time steps

$$\begin{cases} \dot{U}^1 &= \dot{U}^0 + (1-\gamma)\Delta t_1 \ddot{U}^0 + \gamma\Delta t_1 \ddot{U}^1, \\ \dot{U}^2 &= \dot{U}^1 + (1-\gamma)\Delta t_2 \ddot{U}^1 + \gamma\Delta t_2 \ddot{U}^2, \\ \vdots & \\ \dot{U}^M &= \dot{U}^{M-1} + (1-\gamma)\Delta t_M \ddot{U}^{M-1} + \gamma\Delta t_M \ddot{U}^M, \end{cases}$$

then

$$\begin{cases} \dot{U}^1 - \gamma\Delta t_1 \ddot{U}^1 &= \dot{U}^0 + (1-\gamma)\Delta t_1 \ddot{U}^0, \\ \dot{U}^2 - \dot{U}^1 - (1-\gamma)\Delta t_2 \ddot{U}^1 - \gamma\Delta t_2 \ddot{U}^2 &= 0, \\ \vdots & \\ \dot{U}^M - \dot{U}^{M-1} - (1-\gamma)\Delta t_M \ddot{U}^{M-1} - \gamma\Delta t_M \ddot{U}^M &= 0, \end{cases}$$

$$\begin{pmatrix} \dot{U}^1 & & & \\ -\dot{U}^1 & \dot{U}^2 & & \\ & \ddots & \ddots & \\ & & -\dot{U}^{M-1} & \dot{U}^M \end{pmatrix} - \begin{pmatrix} \gamma\Delta t_1\ddot{U}^1 & & & \\ (1-\gamma)\Delta t_2\ddot{U}^1 & \gamma\Delta t_2\ddot{U}^2 & & \\ & \ddots & \ddots & \\ & & (1-\gamma)\Delta t_M\ddot{U}^{M-1} & \gamma\Delta t_M\ddot{U}^M \end{pmatrix} = \begin{pmatrix} \dot{U}^0 + (1-\gamma)\Delta t_1\ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

or equivalently

$$\begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \begin{pmatrix} \dot{U}^1 \\ \dot{U}^2 \\ \vdots \\ \dot{U}^M \end{pmatrix} - \begin{pmatrix} \gamma\Delta t_1 & & & \\ (1-\gamma)\Delta t_2 & \gamma\Delta t_2 & & \\ & \ddots & \ddots & \\ & & (1-\gamma)\Delta t_M & \gamma\Delta t_M \end{pmatrix} \begin{pmatrix} \ddot{U}^1 \\ \ddot{U}^2 \\ \vdots \\ \ddot{U}^M \end{pmatrix}$$
$$= \begin{pmatrix} \dot{U}^0 + (1-\gamma)\Delta t_1\ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

which gives the first equation

$$(B_4 \otimes \mathbb{I}_x)\dot{U} - (B_5 \otimes \mathbb{I}_x)\ddot{U} = \underline{f}_1 \tag{3.4.3}$$

where

$$B_4 = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix},$$

$$B_5 = \begin{pmatrix} \gamma\Delta t_1 & & & \\ (1-\gamma)\Delta t_2 & \gamma\Delta t_2 & & \\ & \ddots & \ddots & \\ & & (1-\gamma)\Delta t_M & \gamma\Delta t_M \end{pmatrix}, \tag{3.4.4}$$

$$\underline{f}_1 = \begin{pmatrix} \dot{U}^0 + (1-\gamma)\Delta t_1\ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

$$\Rightarrow (B_5 \otimes \mathbb{I}_x)\ddot{U} = (B_4 \otimes \mathbb{I}_x)\dot{U} - \underline{f}_1$$

$$\Rightarrow \ddot{U} = B_5^{-1} \otimes \mathbb{I}_x((B_4 \otimes \mathbb{I}_x)\dot{U} - \underline{f}_1). \tag{3.4.5}$$

Now, writing the second equation 3.1.2 at all time steps we have:

$$\begin{cases} U^1 = U^0 + \Delta t_1\dot{U}^0 + (\frac{1}{2} - \beta)\Delta t_1^2\ddot{U}^0 + \beta\Delta t_1^2\ddot{U}^1, \\ U^2 = U^1 + \Delta t_2\dot{U}^1 + (\frac{1}{2} - \beta)\Delta t_2^2\ddot{U}^1 + \beta\Delta t_2^2\ddot{U}^2, \\ \vdots \\ U^M = U^{M-1} + \Delta t_M\dot{U}^{M-1} + (\frac{1}{2} - \beta)\Delta t_M^2\ddot{U}^{M-1} + \beta\Delta t_M^2\ddot{U}^M. \end{cases}$$

Equivalent with

$$\begin{cases} U^1 - \beta\Delta t_1^2\ddot{U}^1 = U^0 + \Delta t_1\dot{U}^0 + (\frac{1}{2} - \beta)\Delta t_1^2\ddot{U}^0, \\ U^2 - U^1 - \Delta t_2\dot{U}^1 - (\frac{1}{2} - \beta)\Delta t_2^2\ddot{U}^1 - \beta\Delta t_2^2\ddot{U}^2 = 0, \\ \vdots \\ U^M - U^{M-1} - \Delta t_M\dot{U}^{M-1} - (\frac{1}{2} - \beta)\Delta t_M^2\ddot{U}^{M-1} - \beta\Delta t_M^2\ddot{U}^M = 0. \end{cases}$$

i.e

$$\begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}\begin{pmatrix} U^1 \\ U^2 \\ \vdots \\ U^M \end{pmatrix} - \begin{pmatrix} 0 & & & \\ \Delta t_2 & 0 & & \\ & \ddots & \ddots & \\ & & \Delta t_M & 0 \end{pmatrix}\begin{pmatrix} \dot{U}^1 \\ \dot{U}^2 \\ \vdots \\ \dot{U}^M \end{pmatrix}$$
$$- \begin{pmatrix} \beta\Delta t_1^2 & & & \\ (\frac{1}{2} - \beta)\Delta t_2^2 & \beta\Delta t_2^2 & & \\ & \ddots & \ddots & \\ & & (\frac{1}{2} - \beta)\Delta t_M^2 & \beta\Delta t_M^2 \end{pmatrix}\begin{pmatrix} \ddot{U}^1 \\ \ddot{U}^2 \\ \vdots \\ \ddot{U}^M \end{pmatrix}$$
$$= \begin{pmatrix} U^0 + \Delta t_1\dot{U}^0 + (\frac{1}{2} - \beta)\Delta t_1^2\ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Therefore

$$(B_1 \otimes \mathbb{I}_x)U - (B_2 \otimes \mathbb{I}_x)\dot{U} - (B_3 \otimes \mathbb{I}_x)\ddot{U} = \underline{f}_2, \tag{3.4.6}$$

where

$$\begin{aligned} B_1 &= \begin{pmatrix} 1 & & \\ -1 & 1 & \\ & \ddots & \ddots \\ & -1 & 1 \end{pmatrix}, \\ B_2 &= \begin{pmatrix} 0 & & & \\ \Delta t_2 & 0 & & \\ & \ddots & \ddots & \\ & & \Delta t_M & 0 \end{pmatrix}, \\ B_3 &= \begin{pmatrix} \beta\Delta t_1^2 & & & \\ (\frac{1}{2} - \beta)\Delta t_2^2 & \beta\Delta t_2^2 & & \\ & \ddots & \ddots & \\ & & (\frac{1}{2} - \beta)\Delta t_M^2 & \beta\Delta t_M^2 \end{pmatrix}, \\ \underline{f}_2 &= \begin{pmatrix} U^0 + \Delta t_1\dot{U}^0 + (\frac{1}{2} - \beta)\Delta t_1^2\ddot{U}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \end{aligned} \tag{3.4.7}$$

Then the two first equations of the Newmark method can be written by

$$\begin{cases} \Rightarrow \ddot{U} = B_5^{-1} \otimes \mathbb{I}_x((B_4 \otimes \mathbb{I}_x)\dot{U} - \underline{f}_1), \\ (B_1 \otimes \mathbb{I}_x)U - (B_2 \otimes \mathbb{I}_x)\dot{U} - (B_3 \otimes \mathbb{I}_x)\ddot{U} = \underline{f}_2. \end{cases} \tag{3.4.8}$$

so

$$(B_1 \otimes \mathbb{I}_x)U - (B_2 \otimes \mathbb{I}_x)\dot{U} - (B_3 \otimes \mathbb{I}_x)\left(B_5^{-1} \otimes \mathbb{I}_x((B_4 \otimes \mathbb{I}_x)\dot{U} - \underline{f}_1)\right) = \underline{f}_2. \tag{3.4.9}$$

$$(B_1 \otimes \mathbb{I}_x)U - ((B_2 + B_3B_5^{-1}B_4) \otimes \mathbb{I}_x)\dot{U} + (B_3B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 = \underline{f}_2. \tag{3.4.10}$$

$$((B_2 + B_3B_5^{-1}B_4) \otimes \mathbb{I}_x)\dot{U} \ = \ (B_1 \otimes \mathbb{I}_x)U + (B_3B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 - \underline{f}_2. \tag{3.4.11}$$

hence

$$\dot{U} \ = \ ((B_2 + B_3B_5^{-1}B_4)^{-1}B_1 \otimes \mathbb{I}_x)U + (B_2 + B_3B_5^{-1}B_4)^{-1}((B_3B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 - \underline{f}_2). \tag{3.4.12}$$

Replace 3.4.12 into the heat equation, we get

$$((B_2 + B_3B_5^{-1}B_4)^{-1}B_1 \otimes \mathbb{I}_x)U + (B_2 + B_3B_5^{-1}B_4)^{-1}((B_3B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 - \underline{f}_2) + (\mathbb{I}_t \otimes A)U = F. \tag{3.4.13}$$

$$((B_2 + B_3B_5^{-1}B_4)^{-1}B_1 \otimes \mathbb{I}_x)U + (\mathbb{I}_t \otimes A)U = F - (B_2 + B_3B_5^{-1}B_4)^{-1}((B_3B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 - \underline{f}_2). \tag{3.4.14}$$

Put

$$\begin{aligned} B &= (B_2 + B_3B_5^{-1}B_4)^{-1}B_1. \\ \tilde{F} &= F - (B_2 + B_3B_5^{-1}B_4)^{-1}((B_3B_5^{-1} \otimes \mathbb{I}_x)\underline{f}_1 - \underline{f}_2). \end{aligned} \tag{3.4.15}$$

Then (3.4.14) turns into

$$(B \otimes \mathbb{I}_x)U + (\mathbb{I}_t \otimes A)U = \tilde{F}. \tag{3.4.16}$$

■

Suppose that this matrix B is diagonalizable, then $B = S\Lambda S^{-1}$, therefore we can apply the Tensor product method to solve 3.4.16 in parallel as in the Euler scheme.

We will now restrict ourselves to the Crank-Nicolson case, that is $\gamma = 2\beta$.

3.5 Computation of matrix S for the Crank Nicolson method

In the presentation of the tensor product with Newmark method for the heat equation, we constructed a matrix B based on matrices B_1, B_2,B_3,B_4,B_5 in 3.4.4 and 3.4.7.
Now, we rewrite the matrix with $dt_i = \rho^{i-1}\Delta t_1$,

$$
\begin{aligned}
B_1 = B_4 &= \begin{pmatrix} 1 & & \\ -1 & 1 & \\ & \ddots & \ddots \\ & -1 & 1 \end{pmatrix}, \\
B_2 &= \Delta t_1 \begin{pmatrix} 0 & & & \\ \rho & 0 & & \\ & \ddots & \ddots & \\ & & \rho^{M-1} & 0 \end{pmatrix}, \\
B_3 &= \tfrac{\Delta t_1^2}{2} \begin{pmatrix} 2\beta & & & \\ (1-2\beta)\rho^2 & 2\beta\rho^2 & & \\ & \ddots & \ddots & \\ & & (1-2\beta)\rho^{2M-2} & 2\beta\rho^{2M-2} \end{pmatrix}, \\
B_5 &= \Delta t_1 \begin{pmatrix} \gamma & & & \\ (1-\gamma)\rho & \gamma\rho & & \\ & \ddots & \ddots & \\ & & (1-\gamma)\rho^{M-1} & \gamma\rho^{M-1} \end{pmatrix}
\end{aligned}
\tag{3.5.1}
$$

Put

$$
B_5 = \Delta t_1 \overline{B}_5,\ B_2 = \Delta t_1 \overline{B}_2,\ B_3 = \frac{\Delta t_1^2}{2}\overline{B}_3.
$$

$$
\begin{aligned}
\overline{B}_3 &= \tfrac{1}{2} \begin{pmatrix} 1 & & \\ \rho^2 & \rho^2 & \\ & \ddots & \ddots \\ & \rho^{2M-2} & \rho^{2M-2} \end{pmatrix}, \\
\overline{B}_5 &= \tfrac{1}{2} \begin{pmatrix} 1 & & \\ \rho & \rho & \\ & \ddots & \ddots \\ & \rho^{M-1} & \rho^{M-1} \end{pmatrix}
\end{aligned}
$$

We have the following intermediate results

Lemma I.7

$$\overline{B}_5^{-1} = 2\begin{pmatrix} 1 & & & & & \\ -1 & \frac{1}{\rho} & & & & \\ 1 & -\frac{1}{\rho} & \frac{1}{\rho^2} & & & \\ -1 & \frac{1}{\rho} & -\frac{1}{\rho^2} & \frac{1}{\rho^3} & & \\ \vdots & \vdots & \vdots & & \ddots & \\ & & & & & \frac{1}{\rho^{M-1}} \end{pmatrix} \tag{3.5.2}$$

which allows to compute

Lemma I.8

Define

$$\overline{B}_3\overline{B}_5^{-1} = \begin{pmatrix} 1 & & \\ & \rho & \\ & & \ddots \\ & & \rho^{M-1} \end{pmatrix} \tag{3.5.3}$$

$$B_6 := \begin{pmatrix} & & \\ & & \\ & B_2 + B_3B_5^{-1}B_4 & \\ & & \end{pmatrix}$$

then $B_6 = B_5$.

Proof :

$$\overline{B}_3\overline{B}_5^{-1}B_4 = \begin{pmatrix} 1 & & & \\ -\rho & \rho & & \\ & -\rho^2 & \rho^2 & \\ & & \ddots & \ddots \end{pmatrix} \tag{3.5.4}$$

$$B_2 + B_3B_5^{-1}B_4 = \begin{pmatrix} & & & \\ & & & \\ \Delta t_1\overline{B}_2 + \frac{\Delta t_1^2}{2}\overline{B}_3 \times (\Delta t_1\overline{B}_5)^{-1}M-B_1 & & & \\ & & & \end{pmatrix}$$

$$= \Delta t_1(\overline{B}_2 + \frac{1}{2}\overline{B}_3\overline{B}_5^{-1}B_4).$$

$$\overline{B}_2 + \frac{1}{2}\overline{B}_3\overline{B}_5^{-1}B_4 = \frac{1}{2}\begin{pmatrix} 1 & & \\ \rho & \rho & \\ & \ddots & \ddots \\ & \rho^{M-1} & \rho^{M-1} \end{pmatrix} = \overline{B}_5.$$

■

We call

$$D = \begin{pmatrix} \Delta t_1 & & & \\ & \Delta t_2 & & \\ & & \ddots & \\ & & & \Delta t_M \end{pmatrix}$$

then we get by Lemma above

$$B_3B_5^{-1} = \frac{1}{2}D.$$

We can rewrite 3.4.14 as

$$\begin{aligned} &(B_5^{-1}B_1 \otimes \mathbb{I}_x)U + (\mathbb{I}_x \otimes A)U = \tilde{F}. \\ &\tilde{F} = F - (B_5^{-1} \otimes \mathbb{I}_x)(\tfrac{1}{2}(D \otimes \mathbb{I}_x)\underline{f}_{-1} - \underline{f}_{-2}) \end{aligned} \tag{3.5.5}$$

Lemma I.9

$$B = B_5^{-1}B_1 = \frac{1}{\Delta t_1}\overline{B}. \tag{3.5.6}$$

with

$$\overline{B} = 2\begin{pmatrix} 1 & & & & \\ -(1+\frac{1}{\rho}) & \frac{1}{\rho} & & & \\ (1+\frac{1}{\rho}) & -\frac{1}{\rho}(1+\frac{1}{\rho}) & \frac{1}{\rho^2} & & \\ -(1+\frac{1}{\rho}) & \frac{1}{\rho}(1+\frac{1}{\rho}) & -\frac{1}{\rho}^2(1+\frac{1}{\rho}) & \frac{1}{\rho^3} & \\ \vdots & \vdots & \vdots & \ddots & \\ & & & & \frac{1}{\rho^{M-1}} \end{pmatrix}$$

Theorem I.10

The matrix B is diagonalizable, provided all Δt_i are different. The matrix S of eigenvectors is lower triangular. It can be chosen with unit diagonal, in which case it is a unipotent Toeplitz matrix of the form (2.3.3),

$$S = T(P_1, \ldots, P_{M-1}), \qquad P_i = p_1 \ldots p_i, \quad p_i = \frac{1+\rho^i}{1-\rho^i}. \tag{3.5.7}$$

S^{-1} is of the same form:

$$S^{-1} = T(Q_1, \ldots, Q_{M-1}), \qquad Q_i = q_1 \ldots q_i, \quad q_i = -\rho\frac{1+\rho^{i-2}}{1-\rho^i}. \tag{3.5.8}$$

$$Q_n = \rho^{-n}\frac{(1+\rho)(1+1)(1+\rho^{-1})...(1+\rho^{2-n})}{(1-\rho^{-1})(1-\rho^{-2})...(1-\rho^{-n})} = \rho^{-n}\,\frac{\prod_{j=-1}^{n-2}(1+\rho^{-j})}{\prod_{j=0}^{n}(1-\rho^{-j})}$$

Proof : The eigenvector associated to the first eigenvalue is

$$\begin{pmatrix} 1 \\ \frac{1+\rho}{1-\rho} \\ \frac{(1+\rho)(1+\rho^2)}{(1-\rho)(1-\rho^2)} \\ \vdots \\ \frac{(1+\rho)\ldots(1+\rho^{M-1})}{(1-\rho)\ldots(1-\rho^{M-1})} \end{pmatrix}$$

the other eigenvectors are similar. The proof of the form of S^{-1} relies on proving the formula

$$P_N + P_{N-1}Q_1 + ... + P_1Q_{N-1} + Q_N = 0. \tag{3.5.9}$$

The source of the proof is an identity of Heine (1847) which is the ρ analogue of a remarkable identity on the binomial coefficients due to Chu and Vandermonde, (see [12]):

$$\binom{n+m}{k} = \sum_{k=0}^{r} \binom{n}{k}\binom{m}{r-k}$$

In addition to the notations in section 2.3, we define a $\rho-$ analogue of a hypergeometric series

$$ {}_2\varphi_1(a_1, a_2; b; \rho; x) = \sum_{0}^{\infty} \frac{(a_1;\rho)_n (a_2;\rho)_n}{(\rho;\rho)_n (b;\rho)_n} x^n, \tag{3.5.10}$$

where it is implied that b is not an integer negative power of ρ. The Heine's identity is stated as follows, for $|b/a_1a_2| < 1$,

$$ {}_2\varphi_1(a_1, a_2; b; \rho; b/a_1a_2) = \frac{(b/a_1;\rho)_\infty (b/a_2;\rho)_\infty}{(b;\rho)_\infty (b/a_1a_2;\rho)_\infty}, \tag{3.5.11}$$

or equivalently for $|x| < 1$

$$ {}_2\varphi_1(a_1, a_2; xa_1a_2; \rho; x) = \frac{(xa_1;\rho)_\infty (xa_2;\rho)_\infty}{(x;\rho)_\infty (xa_1a_2;\rho)_\infty}. \tag{3.5.12}$$

Let N be an integer larger than 1; setting $a_1 = \rho^{-N}$ and $a_2 = a$, yields the ρ-Chu-Vandermonde's formula:

$$ {}_2\varphi_1(\rho^{-N}, a; b; \rho; \frac{b}{a}\rho^N) = \frac{(b/a;\rho)_N}{(b;\rho)_N}. \tag{3.5.13}$$

Since $(\rho^{-N};\rho)_n = 0$ as soon as $n \geq N+1$, the series on the left is a finite sum, and we obtain

$$ \sum_{n=0}^{n=N} \frac{(\rho^{-N};\rho)_n (a;\rho)_n}{(\rho;\rho)_n (b;\rho)_n} \frac{b^n}{a^n} \rho^{nN} = \frac{(b/a;\rho)_N}{(b;\rho)_N}. \tag{3.5.14}$$

This is an equality between rational fractions in ρ, thus valid for all ρ different from 0, 1, all $a \neq 0$ and all b which is not an integer negative power of ρ.

Choose now $a = -u\rho$ et $b = -u\rho^{-N+2}$, to get

$$ (b/a;\rho)_N = (\rho^{-N+1};\rho)_N = (1-\rho^{-N+1})...(1-\rho^{-N+1}\rho^{N-1}) = 0, \tag{3.5.15}$$

which gives

$$ \sum_{n=0}^{n=N} \frac{(-u\rho;\rho)_n (\rho^{-N};\rho)_n}{(\rho;\rho)_n (-u\rho^{-N+2};\rho)_n} \rho^n = 0. \tag{3.5.16}$$

Multiply on the left by$(-u\rho;\rho)_N/(\rho^N(\rho^{-N};\rho)_N)$ and obtain vient

$$ 0 = \sum_{n=0}^{n=N} \frac{(-u\rho;\rho)_n}{(\rho;\rho)_n} \frac{(1-\rho^{-N})...(1-\rho^{-(N-n)-1})}{(1+u\rho^{-N+2})...(1+u\rho^{-(N-n)+1})} \frac{(1+u\rho^{-N+2})...(1+u\rho)}{(1-\rho^{-N})...(1-\rho^{-1})} \rho^{n-N} \tag{3.5.17}$$

$$ = \sum_{n=0}^{n=N} \frac{(-u\rho;\rho)_n}{(\rho;\rho)_n} \rho^{n-N} \frac{(1+u\rho)...(1+u\rho^{-(N-n)+2})}{(1-\rho^{-1})...(1-\rho^{-(N-n)})}. \tag{3.5.18}$$

Choosing $u = 1$, we recognize on the right $\sum_{n=0}^{n=N} P_n Q_{N-n}$, and (3.5.9) is proved. ■

Remark I.1

Using $u = 0$ in (3.5.18), we recover the case of the Euler scheme.

It is now easy to compute asymptotic values of the condition number of S

Lemma I.10

For $\rho = 1 + \varepsilon$, the asymptotic behavior of condition of matrix S (in any norm) is

$$cond(S) \sim \left(\frac{2^{M-1}}{(M-1)!\varepsilon^{M-1}} \right)^2 .$$

Proof : We use here the L^1 condition number for simplicity of notations, but the result would be the same with L^2

$$\|S\|_1 = 1 + |p_1| + |p_1\, p_2| + \cdots + |p_1\, p_2 \ldots p_{M-1}| \sim |p_1 \ldots p_{M-1}| \sim \frac{2^{M-1}}{(M-1)!\varepsilon^{M-1}}$$

the same holds for S^{-1}, and finally gives the result in the Lemma. ■

It remains now to extend the analysis of (3.6.1) in Theorem I.5.

3.6 Balancing errors

The analogue of Theorem I.5 is below. Notice that α has been replaced by $|\alpha|$.

Theorem I.11

The condition number of S does not decrease the precision of the computation in the tensor product method if and only if

$$Log(\tau) + Log(cond(S)) = Log\left(\left| \frac{|Err(a, T, M, \rho)|}{|Err(a, T, M, 1)|} \right| \right). \tag{3.6.1}$$

This can be written asymptotically as

$$Log(\tau) + Log(cond(S)) = Log(|\alpha^N(a, T, M)|\varepsilon^2), \tag{3.6.2}$$

where α^N denotes the α coefficient for Crank Nicolson defined in (3.3.6), and solved in ε as

$$\varepsilon_0^N = \left(\frac{\tau 2^{2(M-1)}}{|\alpha^N(a, T, M)|(M-1)!^2} \right)^{\frac{1}{2M}} . \tag{3.6.3}$$

Proof : The proof is exactly the same as in the Euler case. ■

In figure 3.2 on the left we show the left and right hand side in equation (3.6.2) for $a = T = 1$ and $M = 9$ as a function of ε, together with the exact value ε_0^N.

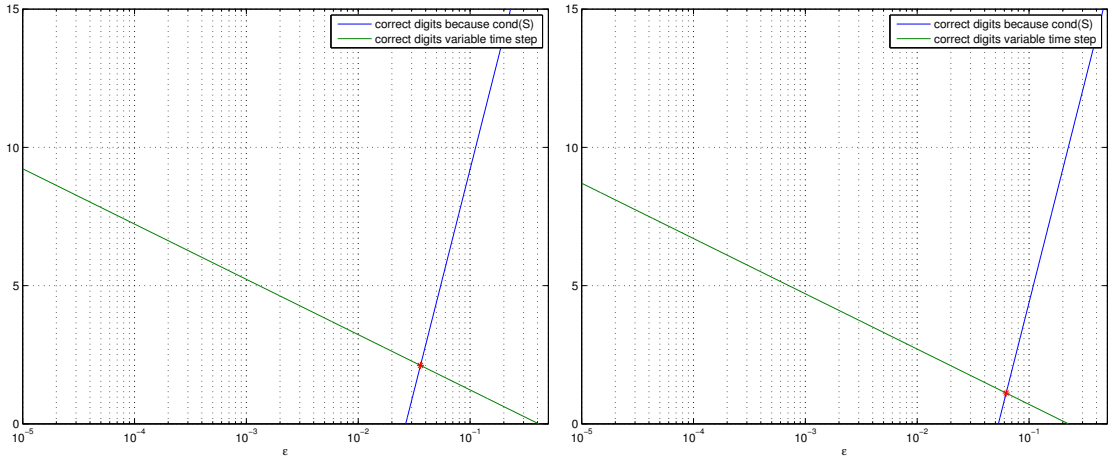

Figure 3.2: Optimization of ε for Euler on the left, Crank Nicolson on the right

Note that the error is much smaller for $Crank - Nicolson$:

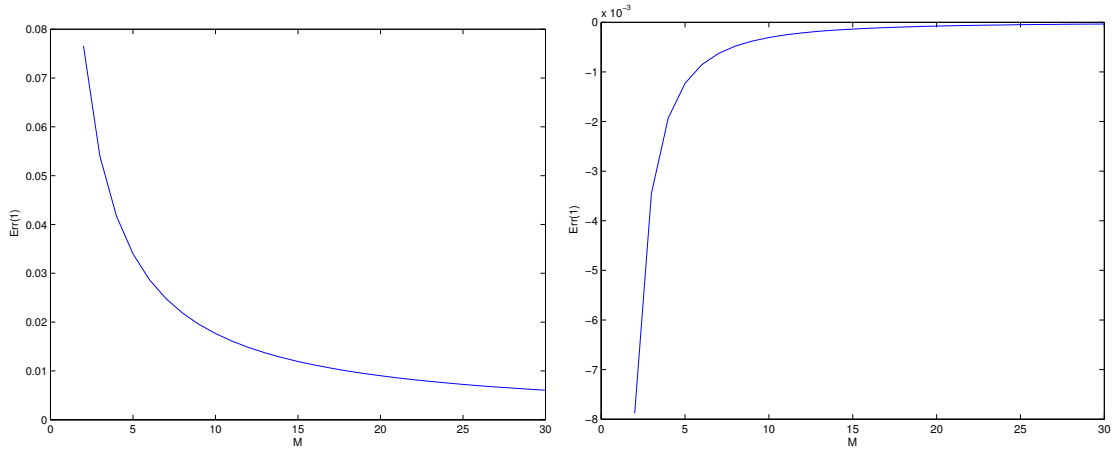

Figure 3.3: Error as a function of M for Euler on the left, Crank Nicolson on the right

Below we show the variation of ε_0 a a function of M.

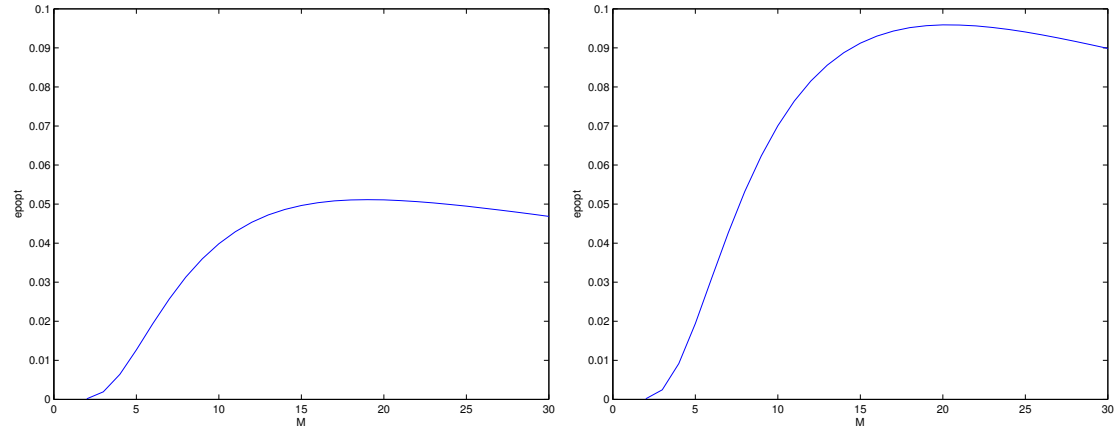

Figure 3.4: Optimized ε_0 as a function of M for Euler on the left, Crank Nicolson on the right

Again, α^N is actually a function of the two variables $y = aT$ and M only:

$$\alpha^N(y, M) = -\frac{2x^3}{(1-x^2)^2}\frac{M(M^2-1)}{12}\frac{\left(\frac{1-x}{1+x}\right)^M}{\left(\frac{1-x}{1+x}\right)^M - e^{-y}},$$
$$\eta^N(y, M) = \alpha^N(y, M)\varepsilon_0^N(y, M)^2,$$
$$\overline{Err}^N(y, M) = Err^N(a, T, M, 1)(1 + \eta(y, M)),$$

with $x = \dfrac{y}{2M}$.

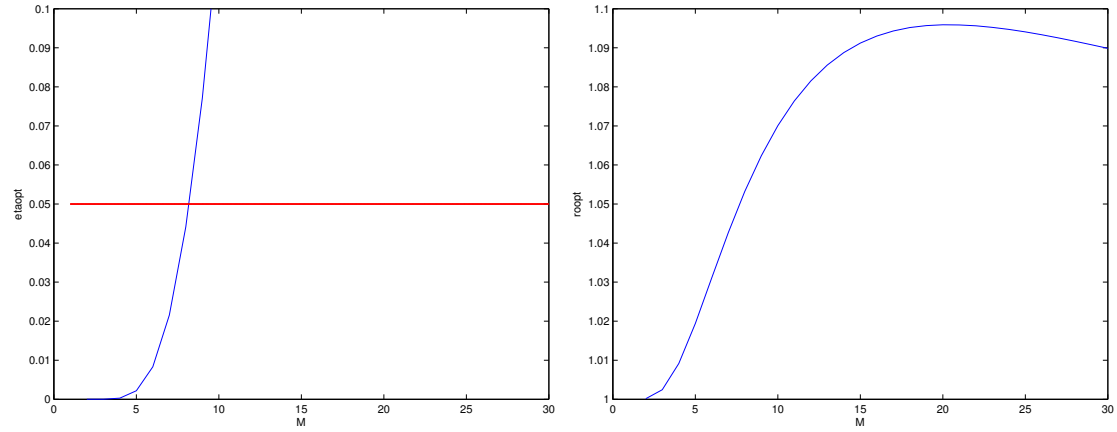

Figure 3.5: Variations of $\eta = \alpha(y, M)\varepsilon_0(y, M)^2$ on the left, of $\bar{\rho} = 1 + \varepsilon_0(y, M)$ on the right, for $y = 1$, as a function of M. In red the reference value $\bar{\eta} = 0.05$

Given y , $\eta(y, M) = \alpha(y, M)\varepsilon_0(y, M)^2$ is an increasing function of M. We thus define, for given tolerance $\bar{\eta}$, $\bar{M}(y)$ as the larger M such that $\eta(y, M) < \bar{\eta}$. In turn, it gives functions $\bar{\varepsilon}(y) = \varepsilon_0(y, \bar{M}(y))$ and $\bar{\eta}_{opt}(y) = \eta(y, \bar{M}(y))$. But the behavior is surprisingly different from the Euler case. The next figure shows the optimized $\bar{M}(y)$, which is now an increasing function of y. It is clear on the zoom on the right that the computations above have no meaning for large values of y.

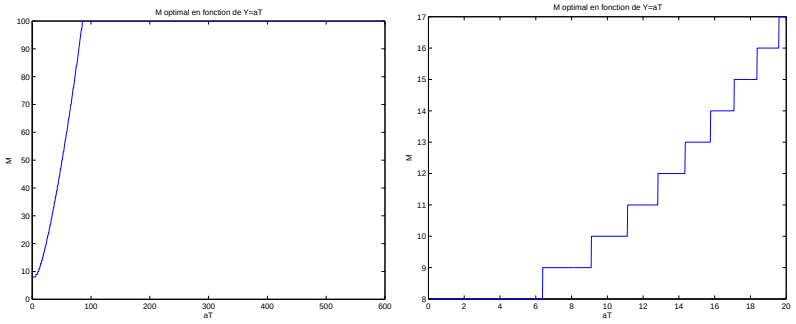

Figure 3.6: $\bar{M}$ in function of $y = aT$ for a tolerance of 5%

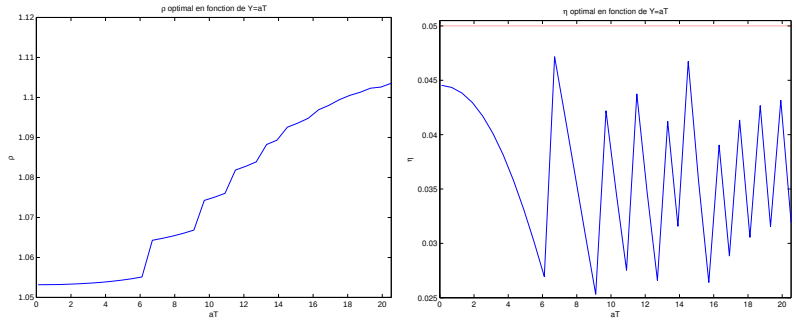

Figure 3.7: $\bar{\rho}$ and $\bar{\eta}_{opt}$ in function of $y = aT$ for a tolerance of 5%

The following tables give the values of $\bar{M}$ and $\bar{\rho}$ by subintervals.

y	$[0, 6.38[$	$[6.38, 9.11[$	$[9.11, 11.12[$	$[11.12, 12.82[$	$[12.82, 14.34[$
M	8	9	10	11	12
ρ	1.0554	1.0669	1.0764	1.0841	1.0902

(3.6.4)

y	$[14.34, 15.76[$	$[15.76, 17.09[$	$[17.09, 18.37[$	$[18.37, 19.60[$
M	13	14	15	16
ρ	1.0949	1.0984	1.1010	1.1028

(3.6.5)

3.7 Application in 1D and 2D

Consider now the PDE

$$\frac{\partial u}{\partial t} - \kappa \Delta u = 0$$

and Fourier transform in space

$$\frac{\partial \hat{u}}{\partial t} + \kappa |\xi|^2 \hat{u} = 0$$

We can apply the results of the proceeding section with $a = \kappa|\xi|^2$. Since M and eps are increasing functions of aT, we can proceed as follows. The discrete frequencies are smaller than $\Xi = \pi^2/h_1^2 + \pi^2/h_2^2$, therefore we compute $a_{max} = \kappa\Xi$ and define ε and ρ by choosing in the tables above for $a = a_{max}$. In general the relevant values will be $M = 8$ and $\rho = 1.0554$.

Consider the same exact solution as above

$$u(x,t) = sin(\pi x_1) sin(\pi x_2)(sin(\pi t) + \exp^{-2\pi^2 t}),\ x \in [0, 1.], \times [0, 1.]\, t \in [0, T = 0.2];$$

Then solving heat equation with number of time step optimal and ρ optimal taken from 3.6.4, we have

$N := N_1 = N_2$	$a_{max}T$	M	ρ	Eseqf	Eseqv	Etensor	$\bar{\eta}$
5	3.9478	8	1.0554	0.0053	0.0052	0.0052	0.0091
10	3.9478	8	1.0554	0.0009	0.0009	0.0009	0.000001
100	3.9478	8	1.0554	0.00078	0.0008	0.0008	0.0162

In this table, Eseqf is the error of norm 2 of the numerical solution with sequential Euler method for heat equation with time steps fixed, Eseqv is the error of norm 2 of the numerical solution with sequential Euler method for different time steps and Etensor is the error of norm 2 of the numerical solution with Tensor product method for different time steps . And η is defined by

$$\bar{\eta} = |\frac{Etensor}{Eseqf} - 1|.$$

Then the values of η from the table above are all less than 5% at the optimal point of M and ρ which validate our study of error of the Tensor product method with Crank Nicolson scheme in time.
We can see the validation of this optimal point better with the following figure show the value of $\bar{\eta}$ in the neighboor of M and ρ optimal.

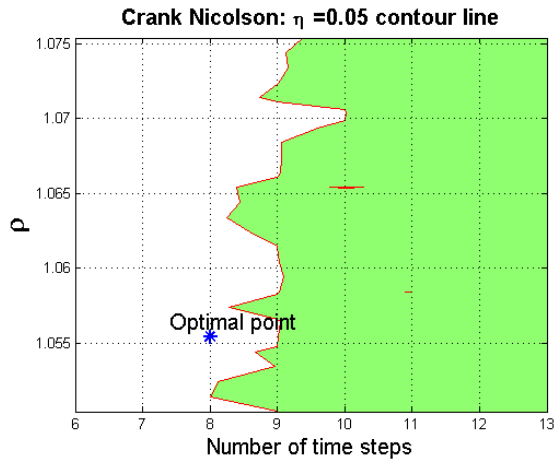

Figure 3.8: $\bar{\eta}$ of Crank Nicolson in function of ρ and M

3.7.1 Comparison of Crank Nicolson and Euler methods for heat equation

Solving the heat equation with the same exact solution and right hand side as in previous section for the space mesh: $N = 200$ for interval of time $(0, T = 0.2]$.
For the Euler method, we use the corresponding optimal point of biggest frequency which is $M = 9$ and $\rho = 1.0331$, and for Crank Nicolson method we use the corresponding optimal point of smallest frequency which is $M_{CN} = 8$ and $\rho_{CN} = 1.0554$, we get:

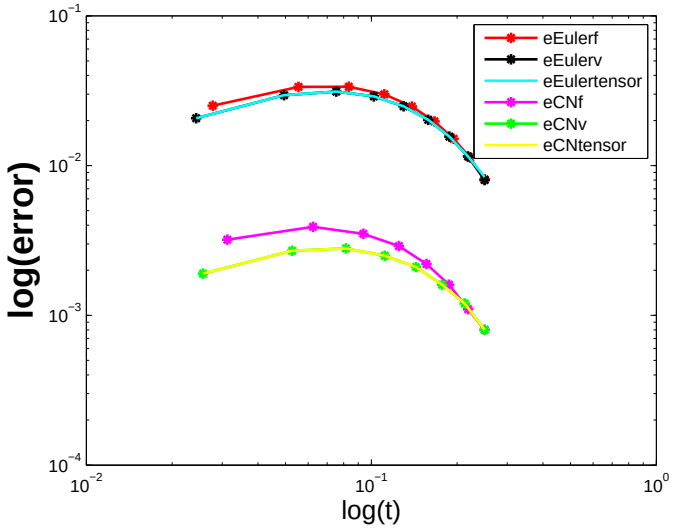

Figure 3.9: Error of Euler and Crank Nicolson methods for heat equation

We can see from figure 3.9 that error of Crank Nicolson method is nearly 10 times smaller than error from Euler method as predicted by the order of the methods.
To get an idea of the efficiency of the Crank Nicolson scheme versus Euler scheme within the Tensor product context, we look at, for a given time interval, for the Crank Nicolson scheme and 1 time window with optimal M and ρ, how many time windows do wee need for the Euler scheme with optimal M and ρ to have the same accuracy.
Numerical results show that, solving the heat equation with the Crank Nicolson Tensor product with optimal M and ρ for a time interval of $[0, 1.25]$, the error of the method at time 1.25 is:

$$e_{CN} = 0.0019.$$

Using the Euler Tensor product method with optimal M and ρ, we need 5 time windows for the time interval of $[0, 1.25]$ to achieve a similar accuracy:

$$e_E = 0.0021.$$

This means that for the Euler method, we solve 45 times the equation

$$(\Lambda_m + A)W^m = G^m$$

to find the solution of the heat equation at time $t = 1.25$, but for Crank Nicolson method, we need to solve only 8 times that equation to get the solution at time $t = 1.25$ and to have the same accuracy. Hence, Tensor product with Newmark method is $\frac{45}{8}$ times more efficient than Euler method. Therefore, one can see that using Crank Nicolson Tensor product method to solve heat equation not only improves in order of accuracy but also in efficiency.

4

Matlab codes for optimization

```
function C=condS(M,Iflag)
% [S,C]=condS(ep,M,Iflag) donne les  coefficients de la matrice S Toeplitz
% triangulaire inferieure et son conditionnement  C evalue
% asymptotiquement,
% i.e. cond(S)=C/eps^2(M-1)
% Iflag =1 pour Euler, 2 pour Newmark
if Iflag ==1
    % p=1./(1-ro.^(1:M-1));
    % q=-ro.^((1:M-1)-1).*P;
    norm1S=1./factorial(M-1);
    C=norm1S.^2;
else
    % p=(1+ro.^(1:M-1))./(1-ro.^(1:M-1));
    norm1S=1./factorial(M-1);
    C=(2.^(M-1).*norm1S).^2;
end



% Calcul des parametres optimaux
CS=16;
Mmax=30;
IM=(2:Mmax);
%Iflag =1, Euler, Iflag=2, Newmark
ind={'E','N'}
for Iflag=1:2
    axis([0 Mmax 0 0.1])
% on cherche pour y fixe le M tel que etaopt=|etab|
tol=0.05;
clear Y Mm rom r1 etaopt
Y=linspace(0.01,600,1000);
for k=1:length(Y)
    y=Y(k);
    for M=2:Mmax
```

```
        [al(M),epopt(M),etaopt(M),El(M)]=alphaopt(y,M,CS,Iflag);

        if  etaopt(M)  > tol &etaopt(M-1) > tol
            [k M-1];
            Mm(k)=M-2;
            epm(k)=epopt(M-2);
            etam(k)=etaopt(M-2);
            break
        end
        Mm(k)=IM(end);
        epm(k)=epopt(end);
        etam(k)=etaopt(end);

    end
    rom=1+epm;
end
Mm(1:length(Y))
rom(1:length(Y))
```

Part II

The Block Method

In 1997, Pierluigi Amodio and Luigi Brugnano presented 2 papers ([1],[2]) on parallel solutions of initial value problems for ordinary differential equations (ODE-IVPs) based on a block technique. In 2008 , they proposed an extension in [3] and showed connections with several approaches such as "Parareal" (Yvon Maday, J.L.Lions, G. Turinici, 2001, [33]). The main common idea between the block method and Parareal is to split the time interval into sub-intervals (coarse time step) and compute in each subinterval independently with a fine time step. The main problem is how to initialize each subinterval. Parareal initialization process is based on an iterative technique based on a predictor-corrector scheme over the coarse grid. The block method initializes with a direct technique based on the approximation of a matrix exponential.

1

Methodology of the Block Method

First we recall the method as presented by Amodio and Brugnano in [3].

1.1 Presentation

Consider the linear Ordinary Differential Equation in $\mathbb{R}^N$:

$$\begin{cases} y' = L\,y + g(t), & t \in T = [t_0, t_1] \\ y(t_0) = y_0 \in \mathbb{R}^N. \end{cases} \tag{1.1.1}$$

Consider a suitable coarse mesh of T

$$t_0 \equiv \tau_0 < \tau_1 < ... < \tau_p \equiv t_1.$$

Define the p problems $i = 1, ..., p$

$$\begin{cases} y' = L\,y + g(t) \quad t \in T_i = [\tau_{i-1}, \tau_i], \\ y(\tau_{i-1}) = y_0^i \in \mathbb{R}^N. \end{cases} \tag{1.1.2}$$

Each T_i is discretized with M fine time steps h_i

$$h_i = \frac{\tau_i - \tau_{i-1}}{M} \qquad i = 1, .., p.$$

Let $y(t)$ be the solution of (1.1.1) and let $Y^i(t), i = 1, ..., p$ be the p solutions of (1.1.2). Set

$$y_m^i = Y^i(\tau_{i-1} + mh_i) \in \mathbb{R}^N \qquad m = 0, ..., M \qquad i = 1, ..., p.$$

In order to have (1.1.1) equivalent to the p problems (1.1.2), we require

$$y_0^1 = y_0 \qquad y_0^i = y_M^{i-1} \qquad i = 2, ..., p.$$

To generalize the writing to the left interval , we set

$$y_0^1 = y_M^0.$$

Numerical approximations to the solutions of (1.1.2) can be obtained by solving discrete problems in the form:

$$S_i\, Y^i = v_i\, y_0^i + G^i \quad Y^i = (y_1^i, ..., y_M^i)^T \in \mathbb{R}^{NM} \quad i = 1, ..., p.$$

with $S_i \in \mathbb{R}^{NM\times NM}$, $v_i \in \mathbb{R}^{NM\times N}$, $G^i \in \mathbb{R}^{NM}$.

$$\mathbf{S}\,\mathbf{Y} \equiv \begin{pmatrix} I_N & & & & \\ -v_1 & S_1 & & & \\ & -V_2 & S_2 & & \\ & & \ddots & \ddots & \\ & & & -V_p & S_p \end{pmatrix} \begin{pmatrix} y_M^0 \\ Y^1 \\ Y^2 \\ \vdots \\ Y^p \end{pmatrix} = \begin{pmatrix} y_0 \\ G^1 \\ G^2 \\ \vdots \\ G^p \end{pmatrix}, \tag{1.1.3}$$

$$V_i = [0\,|\,v_i] \in \mathbb{R}^{NM\times NM} \quad i = 2, ..., p.$$

In the following I_X stands for the identity matrix of dimension X.

This problem can be solved sequentially, by means of the iteration:

$$y_M^0 = y_0, \quad S_i\, Y^i = G^i + v_i\, y_M^{i-1}, \quad i = 1, ...p.$$

Let us consider the factorization:

$$\mathbf{S} = D\,W \equiv \begin{pmatrix} I_N & & & & \\ & S_1 & & & \\ & & S_2 & & \\ & & & \ddots & \\ & & & & S_p \end{pmatrix} \begin{pmatrix} I_N & & & & \\ -w_1 & I_{NM} & & & \\ & -W_2 & I_{NM} & & \\ & & \ddots & \ddots & \\ & & & -W_p & I_{NM} \end{pmatrix},$$

$$W_i = [0\,|\,w_i] \in \mathbb{R}^{NM\times NM}, \quad w_i = S_i^{-1}\, v_i \in \mathbb{R}^{NM\times N}.$$

First, we can solve in parallel $DZ = G$ that is , the systems:

$$S_i\, Z^i = G^i \qquad Z^i = (z_1^i, ..., z_M^i)^T,\ i = 1, ...p. \tag{1.1.4}$$

Then we update sequentially the local solutions

$$\begin{aligned} &Y^1 = Z^1 + w_1\, y_M^0, \\ &Y^i = Z^i + W_i\, Y^{i-1} = Z^i + w_i\, y_M^{i-1}\ (= Z^i + w_i\, y_0^i) \qquad i = 2, ..., p. \end{aligned} \tag{1.1.5}$$

However, it still has much parallelism. Indeed, if we consider

$$Y^i = \begin{pmatrix} \hat{Y}^i \\ y_M^i \end{pmatrix}, \quad Z^i = \begin{pmatrix} \hat{Z}^i \\ z_M^i \end{pmatrix}, \quad w_i = \begin{pmatrix} \hat{w}_i \\ w_{M,i} \end{pmatrix}, \quad w_{M,i} \in \mathbb{R}^{N\times N},$$

then (1.1.5) can be written as $Z = \mathcal{W}\mathcal{Y}$ where

$$Z = \begin{pmatrix} y_0 \\ Z^1 \\ Z^2 \\ \vdots \\ Z^p \end{pmatrix}, \qquad Y = \begin{pmatrix} y_M^0 \\ Y^1 \\ Y^2 \\ \vdots \\ Y^p \end{pmatrix}$$

and

$$\mathcal{W} \equiv \begin{pmatrix} I_N & & & & \\ -w_1 & I_{NM} & & & \\ & -W_2 & I_{NM} & & \\ & & \ddots & \ddots & \\ & & & -W_p & I_{NM} \end{pmatrix}.$$

Because

$$y_M^i = y_0^{i+1} \qquad i = 0, ..., p-1,$$

it is equivalent to solve globally and sequentially the *reduced system*

$$\begin{pmatrix} I_N & & & \\ -w_{M,1} & I_N & & \\ & \ddots & \ddots & \\ & & -w_{M,p-1} & I_N \end{pmatrix} \begin{pmatrix} y_0^1 \\ y_0^2 \\ \vdots \\ y_0^p \end{pmatrix} = \begin{pmatrix} y_0 \\ z_M^1 \\ \vdots \\ z_M^{p-1} \end{pmatrix} \tag{1.1.6}$$

i.e.,

$$y_0^1 = y_0, \qquad y_0^{i+1} = z_M^i + w_{M,i}\, y_0^i, \quad i = 1, ..., p-1. \tag{1.1.7}$$

After that, we can perform the p parallel updates

$$\hat{Y}^i = \hat{Z}^i + \hat{w}_i\, y_0^i, \qquad i = 1, ..., p-1, \qquad Y^p = Z^p + w_p\, y_0^p. \tag{1.1.8}$$

Note: solution of (1.1.6) consists in computing the right initial values y_0^i for each of the p problems (1.1.2).

To summarize: To solve (1.1.1) using a block method thus solving the p problems (1.1.2) requires the following steps:

1. a parallel solve of
$$S_i Z^i = G^i$$
2. a parallel solve of
$$w_i = S_i^{-1} v_i$$
3. a sequentiel solve of
$$y_0^1 = y_0, \qquad y_0^{i+1} = z_M^i + w_{M,i}\, y_0^i,$$
4. a parallel update
$$\hat{Y}^i = \hat{Z}^i + \hat{w}_i\, y_0^i$$

This method has perfect parallel speedup provided that the cost of step 2 and step 3 are small which is so if p/N is large enough.

When the linear system (1.1.1) arises from a space discretization of a PDE, if the number of space discretization points N is important which is usually the case for many PDEs, these steps can become very costly as v_i has N columns requiring N times S_i solves. Amodio and Brugnano proposed the following extension.

Should step 2 (and especially only $w_{M,i}$) be approximated in a fairly "cheap way", then the 4 steps above can be replaced by:

1. a parallel solve of
$$S_i Z^i = G^i$$
2. a sequentiel "cheap" approximation of $w_{M,i}\, y_0^i$ providing initializations y_0^i
3. a parallel solve of
$$S_i Y^i = G + v_i\, y_0^i$$

1.2 Approximating $w_{M,i}\, y_0^i$

Let us look at the case where an Implicit Euler scheme is used.

$$v_i = \begin{pmatrix} I_N \\ 0_N \\ \vdots \\ 0_N \end{pmatrix} \text{ and } w_i = \begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{M,i} \end{pmatrix},$$

I_N is the Identity matrix of dimension N, 0_N the null matrix of dimension N.
$S_i\, w_i = v_i$ becomes

$$\begin{pmatrix} I_N - L*h_i & & & \\ -I_N & I_N - L*h_i & & \\ & \ddots & \ddots & \\ & & -I_N & I_N - L*h_i \end{pmatrix} \begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{M,i} \end{pmatrix} = \begin{pmatrix} I_N \\ 0_N \\ \vdots \\ 0_N \end{pmatrix}.$$

Thus

$$w_{M,i} y_0^i = (I_N - L*h_i)^{-M} \, y_0^i. \tag{1.2.1}$$

Recalling that $M * h_i = dt_i$, then (1.2.1) is a 1^{st} order approximation of $e^{Ldt_i} \, y_0^i$

If a Crank-Nicolson scheme is used

$$\begin{pmatrix} I_N - \frac{L*h_i}{2} & & & \\ -I_N - \frac{L*h_i}{2} & I_N - \frac{L*h_i}{2} & & \\ & \ddots & \ddots & \\ & & -I_N - \frac{L*h_i}{2} & I_N - \frac{L*h_i}{2} \end{pmatrix} \begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{M,i} \end{pmatrix} = \begin{pmatrix} I_N + \frac{L*h_i}{2} \\ 0_N \\ \vdots \\ 0_N \end{pmatrix}$$

and $w_{M,i} y_0^i = [(I_N - \frac{L*h_i}{2})^{-1}(I_N + \frac{L*h_i}{2})]^M \, y_0^i$ which is a 2^{nd} order approximation of $e^{Ldt_i} \, y_0^i$

Whatever the time scheme

$$w_{M,i} y_0^i \approx e^{Ldt_i} \, y_0^i.$$

So the problem remains how to cheaply approximate e^{Ldt_i}.

The two following techniques will be tested

1. if dt_i small, $e^{Ldt_i} = \sum_{k=0}^{km} \frac{Ldt_i^k}{k!}$, the value of km to be tested
2. if L can be factorized as VHV^T where V is an orthonormal matrix of dimension (N,km) (i.e.: $V^T V = I_N$) and H is a matrix of dimension (km,km), then

$$e^{Ldt_i} = \sum_{k=0}^{\infty} \frac{Ldt_i^k}{k!} = \sum_{k=0}^{\infty} \frac{(VHV^T)^k dt_i^k}{k!} = \sum_{k=0}^{\infty} \frac{(VHV^T)(VHV^T)\cdots(VHV^T)dt_i^k}{k!}$$

$$e^{Ldt_i} = V(\sum_{k=0}^{\infty} \frac{(Hdt_i)^k}{k!})V^T = Ve^{Hdt_i}V^T.$$

If the value of km is small, e^{Hdt_i} is cheap to compute using a Padé approximation,(see [4], [8]).

Such a factorization occurs when an Arnoldi-Krylov factorization is made (see [52]).

In the following, we recall the Arnoldi-Krylov factorization.

The Arnoldi-Krylov factorization

Let A be a matrix of size (n,n), b a vector of size n. The Krylov space $K_m(A, b)$ is defined as :

$$K_m(A, b) = span\{b, Ab, A^2b, ..., A^{m-1}b\}.$$

Arnoldi [5] presented this method in 1951 as a means of reducing a dense matrix into a Hessenberg form. It consists in an orthogonal projection of A onto $K_m = K_m(A, b)$.

The algorithm is with the following notations:
$V = (v_j)_{j=1,m}$ is a matrix of size(n,m)
$H = (h_{i,j})_{i=1,m+1,j=1,m}$ is a Hessenberg matrix of size(m+1,m)

$v_1 = b/||b||$
For j=2, 3, ..., m

1. Compute $h_{i,j} = (Av_j, v_i)$ for i=1,2,..., j
2. Compute $w_j = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$
3. $h_{j+1,j} = ||w_j||$, If $h_{j+1,j} = 0$, stop
4. $v_{j+1} = w_j/h_{j+1,j}$

EndDo

Let H_m be the matrix obtained from H deleting its last row. Equality in step 2 combined with equality in step 4 can be written as:

$$\begin{aligned} AV &= VH_m + w_m e_m^T, \\ V^TAV &= H_m, \end{aligned}$$

$e_m^T = (0, 0, ..., 0, 1$ is a vector of size(1,m).

Thus

$$A = VH_mV^T.$$

2

Application to the Heat equation

2.1 Block method for the Heat equation: Euler and Cranck-Nicolson

Let us consider the heat equation

$$\begin{aligned} &\frac{\partial y(x,t)}{\partial t} - \Delta y(x,t) = g(x,t), \qquad (x,t) \in \mathbb{R}^{N\times M}, \\ &y(x,0) = y_0(x). \end{aligned} \tag{2.1.1}$$

Using the 1^{st} order implicit Euler method or the 2^{nd} order Cranck-Nicolson scheme for time and 2^{nd} order finite differences in space, we obtain

$$\mathbf{S}\,\mathbf{Y} \equiv \begin{pmatrix} I_N & & & & \\ -v_1 & S_1 & & & \\ & -V_2 & S_2 & & \\ & & \ddots & \ddots & \\ & & & -V_p & S_p \end{pmatrix} \begin{pmatrix} y_M^0 \\ Y^1 \\ Y^2 \\ \vdots \\ Y^p \end{pmatrix} = \begin{pmatrix} y_0 \\ G^1 \\ G^2 \\ \vdots \\ G^p \end{pmatrix}, \tag{2.1.2}$$

$$v_M = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^M,$$

$$S_i = B \otimes I_N + h_i J_M \otimes L, \quad v_i = v_M \otimes I_N.$$

– Matrix L is the Finite difference Laplace matrix for 2 D (similarly with cases above).
– Matrix B is the time matrix ,

$$B = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}.$$

– If the scheme is Euler, $J_M = I_M$ is the identity matrix.

– If the scheme is the Crank-Nicolson scheme,

$$J_M = \begin{pmatrix} 0 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1/2 & 0 & 1/2 \end{pmatrix}.$$

Right hand side

$$G^i = h_i(g_1^i, ..., g_M^i)^T, \qquad g_m^i \in \mathbb{R}^N$$

g_m^i is the discretized approximation of $g(x, t_m)$ at time t_m .

Then we apply the block method and solve the problem in parallel.

2.2 Numerical Results

This equation will be solved in $\Omega \times T$ = [0,1] x [0,1] x [0,.2] as before with the exact solution being :

$$y(x,t) = sin(\pi x)\ sin(\pi y)\ (sin(\pi t) + e^{-2\pi^2 t})$$

and right hand side

$$f(x,t) = \pi sin(\pi x)\ sin(\pi y)\ (cos(\pi t) + 2\pi sin(\pi t))$$

$\Omega \times T$ is meshed with (N = Nx*Ny) points in space and (M*p) points in time.

2.2.1 Block method with Crank-Nicolson

On Fig.2.1, is shown the order of the block method with p=10 for the Crank-Nicolson scheme (CN) compared with the sequential scheme with computational parameters in table 2.1.

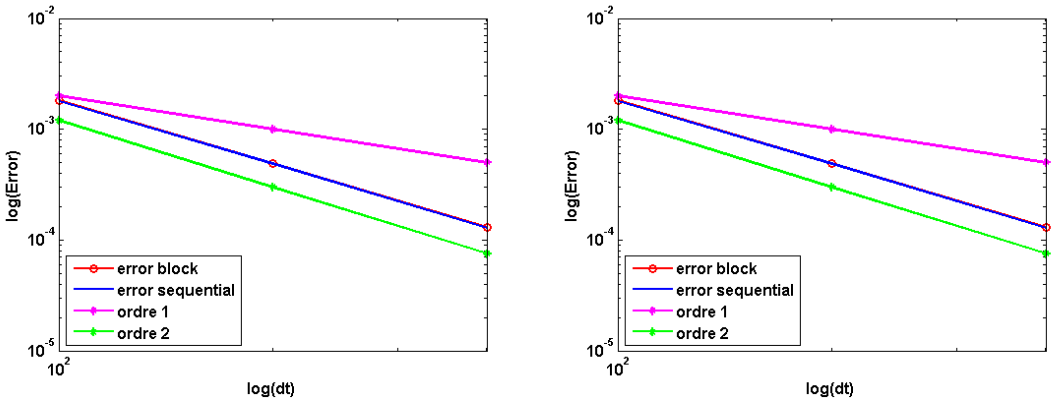

Figure 2.1: Order in time of the Block method - CN with full computation of the exponential on the left , Arnoldi approximation on the right

N*M = (Nx*Ny)*M	10x10x10	20x20x20	40x40x40
Err. Seq	1.7913e-03	4.9045e-04	1.2860e-04
Err. Block FC	1.8088e-03	4.9483e-04	1.2969e-04
Err. Block H km = 1	1.8088e-03	4.9483e-04	1.2969e-04
Err. Block TA km = 5	12.7632e+07	2.2093e+34	5.6769e+61

Table 2.1: Errors for the Crank-Nicolson scheme

In table 2.1, the Err. Seq. stands for error of the sequential method, Err. Block FC is the error of the block method with full computation of the exponential of matrix (expm() in Matlab), Err. Block H km is the error of the block method with Arnoldi's approximation, Err. Block TA is the error of the block method with Taylor Approximation computation of the exponential of matrix of rank km, km is the parameter mentioned in the 2 techniques to cheaply approximate e^{Ldt_i}.

For a mesh of 40x40, p=10, M=40, the ratio of computing costs between the sequential computation and Block computation with Arnoldi approximation (km =1) is of 5. The Taylor approximation does not converge as norm(dt*L) $\gg$ 1 explaining why results are totally wrong .

2.2.2 Block method with Euler

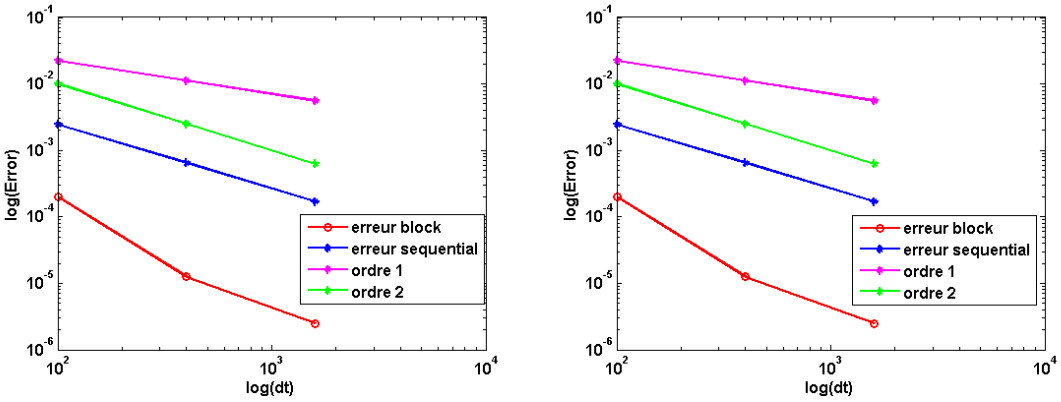

Figure 2.2: Order in time of the Block method - Euler with full computation of the exponential on the left , Arnoldi approximation on the right

N*M = (Nx*Ny)*M	10x10x10	20x20x40	40x40x160
Err. Seq	2.4551e-03	6.5282e-04	1.6895e-04
Err. Block FC of e	1.9825e-04	1.2542e-05	2.4912e-06
Err. Block H km = 1	1.9825e-04	1.2542e-05	2.4912e-06
Err. Block TA of e, km = 5	4.6147e+15	5.4400e+28	5.2526e+55

Table 2.2: Errors for the Euler scheme

The Block method with an Euler scheme is actually better than the sequential scheme and can be seen as a preconditioner. Initializing each time block with an approximation of the exponential globally improves the accuracy of the scheme .

For a mesh of 40x40, p=10, M=40, the ratio of computing costs between the sequential computation and Block computation with Arnoldi approximation (km =1) is of 5 while accuracy is similar.

3

Application to the Elasticity equation

3.1 The Elasticity equation

Consider the elasticity equation with a velocity equal to 1, which is a second order in time PDE:

$$\frac{\partial^2 u}{\partial t^2} - \Delta u = f$$

with initial data $u(\cdot, 0) = u_0$, $\frac{\partial u}{\partial t}(\cdot, 0) = u_1$. It can be written as a first order system, setting $v = u' = \frac{\partial u}{\partial t}$:

$$\begin{pmatrix} v' \\ u' \end{pmatrix} - \begin{pmatrix} 0 & \Delta \\ I & 0 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}.$$

In a condensed way with $U = (v, u)^T$ and $\mathcal{L} = \begin{pmatrix} 0 & \Delta \\ I & 0 \end{pmatrix}$,

$$\begin{aligned} &\frac{\partial U}{\partial t} - \mathcal{L}U = 0, \\ &U(\cdot, 0) = U_0 = (u_0, u_1)^T \end{aligned} \tag{3.1.1}$$

and block development is similar to what has been done in the preceding chapter.

3.1.1 Elasticity and Euler

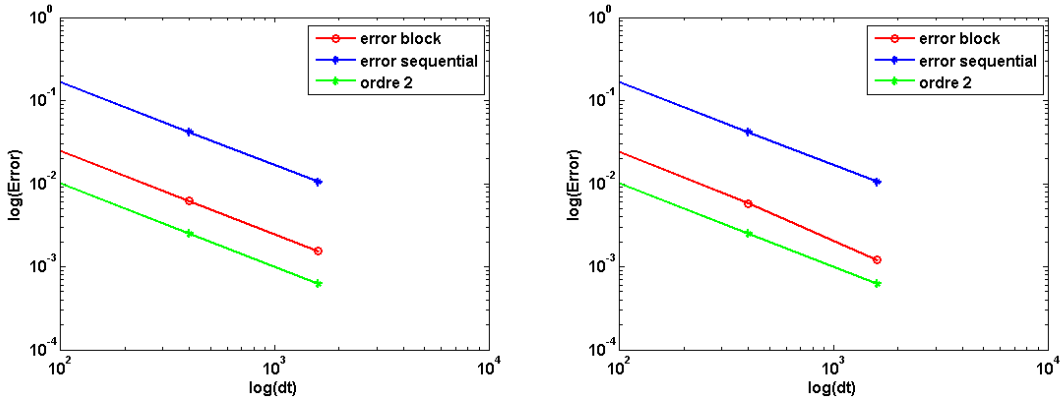

Figure 3.1: Order in time of the Block method - Euler with Arnoldi approximation on the left km=2, and Taylor approximation km=2

2*Nx*Ny * Nt	20x10x10	40x20x40	80x40x160
Err. Seq	1.6631e-01	4.1770e-02	1.0443e-02
Err. Block FC of e	2.4396e-02	6.1595e-03	1.5332e-03
Err. Block H km = 1	6.3802e-01	6.3047e-01	6.2861e-01
Err. Block H km = 2	2.4396e-02	6.1595e-03	1.5332e-03
Err. Block TA of e, km = 1	9.0846e-02	7.3424e-02	6.9079e-02
Err. Block TA of e, km = 2	2.4047e-02	5.7981e-03	1.2019e-03

Table 3.1: Errors for the Euler scheme

Using an Arnoldi approximation with km = 2 or a full computation of $e^{\mathcal{L}dt}$ provides identical results, but the ratio of computing time between the two is of 600 for a 80x40 mesh. Thus in the following results , only the Arnoldi computation is computed. The new factor for this PDE is that a Taylor approximation of $e^{\mathcal{L}dt}$ for the same value of km as Arnoldi provides a solution which is slightly better than the Arnoldi one.

As seen for the Heat equation , the block method provides a better solution that the sequential one.

3.1.2 Elasticity and Crank-Nicolson

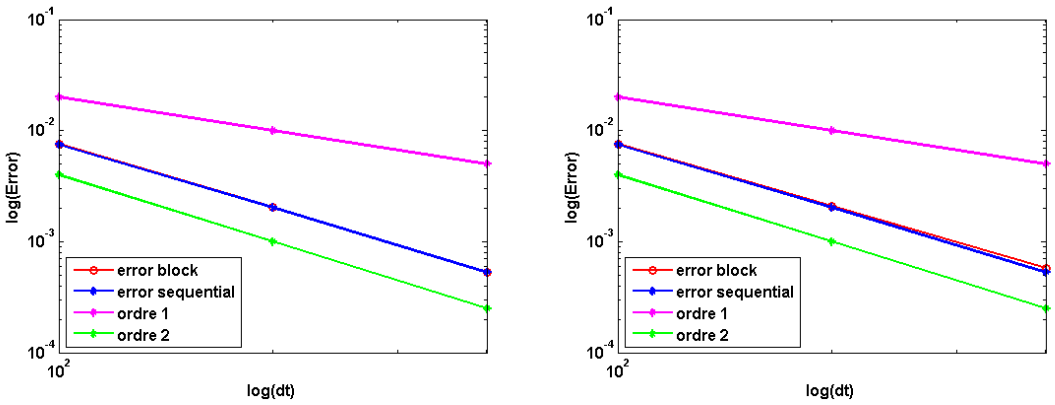

Figure 3.2: Order in time of the Block method - Crank-Nicolson with Arnoldi approximation on the left km=2, and Taylor approximation km=2

2*Nx*Ny * Nt	20x10x10	40x20x40	80x40x160
Err. Seq	7.4817e-03	2.0269e-03	5.2838e-04
Err. Block H km = 1	6.3802e-01	6.3047e-01	6.2861e-01
Err. Block H km = 2	7.5318e-03	2.0405e-03	5.3192e-04
Err. Block H km = 3	7.5318e-03	2.0405e-03	5.3192e-04
Err. Block TA of e, km = 1	5.9542e-02	6.5452e-02	6.7079e-02
Err. Block TA of e, km = 2	7.7315e-03	2.2777e-03	8.5812e-04
Err. Block TA of e, km = 3	7.5743e-03	2.0834e-03	5.7499e-04
Err. Block TA of e, km = 4	7.5318e-03	2.0404e-03	5.3185e-04
Err. Block TA of e, km = 5	7.5318e-03	2.0405e-03	5.3191e-04

Table 3.2: Errors for the Crank-Nicolson scheme

In this case, the Arnoldi method is slightly better for km = 2. Results do not change for a higher value of km. For the Taylor approximation, results vary no longer after km=4. For these optimal values of km, results are of the same order of accuracy as the sequential ones.

4

Matlab codes for the block method

Arnoldi Procedure

```
function [ y ] = prodexp(km,dt,L,x )

N = length(x) ;

u = zeros(N,km) ;
H = zeros(km+1, km+1) ;
usn = norm(x);
u(:,1) = x/usn ;

for j=1:km
    v = L*u(:,j) ;
    for i=1:j
        H(i,j) = u(:,i)' * v ;
        v = v - H(i,j)*u(:,i) ;
    end
    H(j+1,j)= norm(v) ;
    u(:,j+1) = v/ H(j+1,j) ;
end
e1 = zeros(km,1) ;
e1(1) = 1 ;
y = usn * u(:,1:km)*(expm(dt*H(1:km,1:km))*e1) ;

end
```

4.1 Heat equation and Euler

```
function u=HeatEblock(Pb,AL, AU, AA, L)
```

```
%% solve heat equation with Block EUler
[Ntc,Nt, Nx, Ny, Nxy,T,Lx, Ly,hx, hy,x,y,dt, hi,tc] = recupstruct(Pb);

hx2 = hx*hx ;
hy2 = hy*hy ;

for i=1:Ntc
    g(i) = struct('xy', zeros(Nx*Ny,Nt));
    ff(i) = struct('fxy', zeros(Nx,Ny,Nt));
end

%Rhs = .5*(fn + fn+1)
t = (0:hi:dt) ;
ss = sm(x,y,t, tc) ;
I0 = (1:Nt) ;
I1 = (2:Nt+1) ;

for i=1:Ntc
    ff(i).fxy(:,:,I0) = ss(i).uxy(:,:,I1) ;
end
clear ss

% Space boundary conditions
t = (0:hi:dt) ;
xx = [0 Lx ] ;
yy = [0 Ly] ;
sx = uxy(xx,y,t, tc) ;
sy = uxy(x,yy,t, tc) ;

for i=1:Ntc
    % Horizontal
    ff(i).fxy(1,:,I0) = ff(i).fxy(1,:,I0) + ...
        sx(i).uxy(1,:,I1) /hx2 ;
    ff(i).fxy(Nx,:,I0) =ff(i).fxy(Nx,:,I0)+ ...
        sx(i).uxy(2,:,I1) /hx2 ;
    % Vertical
    ff(i).fxy(:,1,I0) = ff(i).fxy(:,1,I0) + ...
        sy(i).uxy(:,1,I1) /hy2 ;
    ff(i).fxy(:,Ny,I0) =  ff(i).fxy(:,Ny,I0) + ...
      sy(i).uxy(:,2,I1) /hy2 ;
end

clear sx sy
```

```
for i=1:Ntc
    g(i).xy = hi*reshape(ff(i).fxy, Nx*Ny,Nt);
end

%----------------------------------------
%   Initialisation
%----------------------------------------
ss = uxy(x,y,0., tc) ;
u0 = reshape(ss(1).uxy(:,:,1), Nxy,1) ;  % Sol. Init.


uu=zeros(Nxy,Ntc);
vv=zeros(Nxy,Ntc);

%-----------------
%Etape 1
%-----------------

for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:Nxy,j) = u ;
end

%-----------------
%Etape 2
%-----------------
vv(:,1) = u0 ;

version = 2 ;
if(version == 1)
    % Version totale
    EA = expm( dt*L) ;
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end
if(version ==2)
    %  Version Arnoldi
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
```

```
    km = 1 ;
    for j=2:Ntc
        ww = prodexp(km,dt,L,vv(:,j-1) ) ;
        vv(:,j) = uu(:,j-1) +ww ;

    end
end

if(version ==3)
    %  Version approx exponentielle
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
    km = 5 ;
    EA = speye(Nx*Ny) ;
    L0 = EA ;
    for k=1:km
        L0 = L0*L*dt ;
        EA = EA + L0/factorial(k) ;
    end
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end

%-----------------
%Etape 3
%-----------------

for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:Nxy,j) = u ;
end

%-----------------
%Fin Block algorithm
%-----------------

u=reshape(uu,Nx,Ny,Ntc);
```

4.2 Heat equation and Crank-Nicolson

```
function u=HeatCNblock(Pb,AL, AU, AA, L)


%% solve heat equation with Block Crank nicolson
[Ntc,Nt, Nx, Ny, Nxy,T,Lx, Ly,hx, hy,x,y,dt, hi,tc] = recupstruct(Pb);

hx2 = hx*hx ;
hy2 = hy*hy ;

for i=1:Ntc
    g(i) = struct('xy', zeros(2*Nxy,Nt));
    ff(i) = struct('fxy', zeros(2*Nx,Ny,Nt));
end

%Rhs = .5*(fn + fn+1)
t = (0:hi:dt) ;
ss = sm(x,y,t, tc) ;
I0 = (1:Nt) ;
I1 = (2:Nt+1) ;
Ix = (1:Nx) ;

for i=1:Ntc
    ff(i).fxy(:,:,I0) = .5*(ss(i).uxy(:,:,I0) + ss(i).uxy(:,:,I1) ) ;
end
clear ss

% Space boundary conditions
t = (0:hi:dt) ;
xx = [0 Lx ] ;
yy = [0 Ly] ;
sx = uxy(xx,y,t, tc) ;
sy = uxy(x,yy,t, tc) ;

for i=1:Ntc
    % Horizontal
    ff(i).fxy(1,:,I0) = ff(i).fxy(1,:,I0) + ...
        .5*(sx(i).uxy(2+1,:,I0) + sx(i).uxy(2+1,:,I1) )/hx2 ;
    ff(i).fxy(Nx,:,I0) =ff(i).fxy(Nx,:,I0)+ ...
        .5*(sx(i).uxy(2+2,:,I0) + sx(i).uxy(2+2,:,I1) )/hx2 ;
    % Vertical
    ff(i).fxy(Ix,1,I0) = ff(i).fxy(Ix,1,I0) + ...
        .5*(sy(i).uxy(Nx+Ix,1,I0) + sy(i).uxy(Nx+Ix,1,I1) )/hy2 ;
    ff(i).fxy(Ix,Ny,I0) =  ff(i).fxy(Ix,Ny,I0) + ...
```

```
        .5*(sy(i).uxy(Nx+Ix,2,I0) + sy(i).uxy(Nx+Ix,2,I1) )/hy2 ;
end

clear sx sy

for i=1:Ntc
    for j=1:Nt
        g(i).xy(1:Nxy,j) = hi*reshape(ff(i).fxy(1:Nx,:,j), Nxy,1);
        g(i).xy(Nxy+1:2*Nxy,j) = hi*reshape(ff(i).fxy(Nx+1:2*Nx,:,j), Nxy,1);
    end
end

%----------------------------------------
%   Initialisation
%----------------------------------------
ss = uxy(x,y,0., tc) ;
%SOL. Init
u0(1:Nxy,1) = reshape(ss(1).uxy(1:Nx,:,1), Nxy,1) ;
u0(Nxy+1:2*Nxy,1) = reshape(ss(1).uxy(Nx+1:2*Nx,:,1), Nxy,1) ;

uu=zeros(2*Nxy,Ntc);
vv=zeros(2*Nxy,Ntc);

%-----------------
%Etape 1
%-----------------
%  v = reshape(ss(1).uxy(:,:,1), Nxy,1) ;

for j=1:Ntc
    %      v = reshape(ss(j).uxy(:,:,1), Nxy,1) ;
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:2*Nxy,j) = u ;
end

%-----------------
%Etape 2 computing the exponential
%-----------------
vv(:,1) = u0 ;

version = 3 ;
```

```
if(version == 1)
    % Version totale
    EA = expm( dt*L) ;
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end
if(version ==2)
    %  Version Arnoldi
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
    km = 3 ;
    for j=2:Ntc
        ww = prodexp(km,dt,L,vv(:,j-1) ) ;
        vv(:,j) = uu(:,j-1) +ww ;

    end
end

if(version ==3)
    %  Version approx Taylor exponentielle
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
    km =3 ;
    EA = speye(2*Nx*Ny) ;
    L0 = EA ;
    for k=1:km
        L0 = L0*L*dt ;
        EA = EA + L0/factorial(k) ;
    end
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end

%-----------------
%Etape 3
%-----------------

for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
```

```
    end
    uu(1:2*Nxy,j) = u ;
end

%-----------------
%Fin Block algorithm
%-----------------

clear u
u = zeros(2*Nx,Ny,Ntc) ;
for j=1:Ntc
    u(1:Nx,1:Ny,j)=reshape(uu(1:Nxy,j),Nx,Ny);
    u(Nx+1:2*Nx,1:Ny,j)=reshape(uu(Nxy+1:2*Nxy,j),Nx,Ny);
end
```

4.3 Elasticity equation and Euler

```
function u=ElastEblock(Pb,AL, AU, AA, L)


%% solve Elast equation with Euler Block
[Ntc,Nt, Nx, Ny, Nxy,T,Lx, Ly,hx, hy,x,y,dt, hi,tc] = recupstruct(Pb);

hx2 = hx*hx ;
hy2 = hy*hy ;

for i=1:Ntc
    g(i) = struct('xy', zeros(2*Nxy,Nt));
    ff(i) = struct('fxy', zeros(2*Nx,Ny,Nt));
end

%Rhs = fn+1
t = (0:hi:dt) ;
ss = sm(x,y,t, tc) ;
I0 = (1:Nt) ;
I1 = (2:Nt+1) ;
Ix = (1:Nx) ;

for i=1:Ntc
    ff(i).fxy(:,:,I0) =ss(i).uxy(:,:,I1)  ;
end
```

```
clear ss

% Space boundary conditions
t = (0:hi:dt) ;
xx = [0 Lx ] ;
yy = [0 Ly] ;
sx = uxy(xx,y,t, tc) ;
sy = uxy(x,yy,t, tc) ;

for i=1:Ntc
    % Horizontal
    ff(i).fxy(1,:,I0) = ff(i).fxy(1,:,I0) + ...
       sx(i).uxy(2+1,:,I1) /hx2 ;
    ff(i).fxy(Nx,:,I0) =ff(i).fxy(Nx,:,I0)+ ...
       sx(i).uxy(2+2,:,I1) /hx2 ;
    % Vertical
    ff(i).fxy(Ix,1,I0) = ff(i).fxy(Ix,1,I0) + ...
      sy(i).uxy(Nx+Ix,1,I1) /hy2 ;
    ff(i).fxy(Ix,Ny,I0) =  ff(i).fxy(Ix,Ny,I0) + ...
       sy(i).uxy(Nx+Ix,2,I1) /hy2 ;
end

clear sx sy

for i=1:Ntc
    for j=1:Nt
        g(i).xy(1:Nxy,j) = hi*reshape(ff(i).fxy(1:Nx,:,j), Nxy,1);
        g(i).xy(Nxy+1:2*Nxy,j) = hi*reshape(ff(i).fxy(Nx+1:2*Nx,:,j), Nxy,1);
    end
end

%----------------------------------------
%   Initialisation
%----------------------------------------
ss = uxy(x,y,0., tc) ;
%SOL. Init
u0(1:Nxy,1) = reshape(ss(1).uxy(1:Nx,:,1), Nxy,1) ;
u0(Nxy+1:2*Nxy,1) = reshape(ss(1).uxy(Nx+1:2*Nx,:,1), Nxy,1) ;

uu=zeros(2*Nxy,Ntc);
vv=zeros(2*Nxy,Ntc);

%-----------------
%Etape 1
%-----------------
```

```
for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:2*Nxy,j) = u ;
end

%-----------------
%Etape 2
%-----------------
vv(:,1) = u0 ;

version = 2 ;
if(version == 1)
    % Version totale
    EA = expm( dt*L) ;
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end
if(version ==2)
    %  Version Arnoldi
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
    km = 2 ;
    for j=2:Ntc
        ww = prodexp(km,dt,L,vv(:,j-1) ) ;
        vv(:,j) = uu(:,j-1) +ww ;

    end
end

if(version ==3)
    %  Version approx exponentielle
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
    km = 2 ;
    EA = speye(2*Nx*Ny) ;
    L0 = EA ;
    for k=1:km
        L0 = L0*L*dt ;
```

```
        EA = EA + L0/factorial(k) ;
    end
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end

%-----------------
%Etape 3
%-----------------

for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:2*Nxy,j) = u ;
end

%-----------------
%Fin Block algorithm
%-----------------

clear u
u = zeros(2*Nx,Ny,Ntc) ;
for j=1:Ntc
    u(1:Nx,1:Ny,j)=reshape(uu(1:Nxy,j),Nx,Ny);
    u(Nx+1:2*Nx,1:Ny,j)=reshape(uu(Nxy+1:2*Nxy,j),Nx,Ny);
end
```

4.4 Elasticity and Crank-Nicolson

```
function u=ElastCNblock(Pb,AL, AU, AA, L)


%% solve Elast equation with Crank-Nicolson -Block
[Ntc,Nt, Nx, Ny, Nxy,T,Lx, Ly,hx, hy,x,y,dt, hi,tc] = recupstruct(Pb);

hx2 = hx*hx ;
```

```
hy2 = hy*hy ;

for i=1:Ntc
    g(i) = struct('xy', zeros(2*Nxy,Nt));
    ff(i) = struct('fxy', zeros(2*Nx,Ny,Nt));
end

%Rhs = .5*(fn + fn+1)
t = (0:hi:dt) ;
ss = sm(x,y,t, tc) ;
I0 = (1:Nt) ;
I1 = (2:Nt+1) ;
Ix = (1:Nx) ;

for i=1:Ntc
    ff(i).fxy(:,:,I0) = .5*(ss(i).uxy(:,:,I0) + ss(i).uxy(:,:,I1) ) ;
end
clear ss

% Space boundary conditions
t = (0:hi:dt) ;
xx = [0 Lx ] ;
yy = [0 Ly] ;
sx = uxy(xx,y,t, tc) ;
sy = uxy(x,yy,t, tc) ;

for i=1:Ntc
    % Horizontal
    ff(i).fxy(1,:,I0) = ff(i).fxy(1,:,I0) + ...
        .5*(sx(i).uxy(2+1,:,I0) + sx(i).uxy(2+1,:,I1) )/hx2 ;
    ff(i).fxy(Nx,:,I0) =ff(i).fxy(Nx,:,I0)+ ...
        .5*(sx(i).uxy(2+2,:,I0) + sx(i).uxy(2+2,:,I1) )/hx2 ;
    % Vertical
    ff(i).fxy(Ix,1,I0) = ff(i).fxy(Ix,1,I0) + ...
        .5*(sy(i).uxy(Nx+Ix,1,I0) + sy(i).uxy(Nx+Ix,1,I1) )/hy2 ;
    ff(i).fxy(Ix,Ny,I0) =  ff(i).fxy(Ix,Ny,I0) + ...
        .5*(sy(i).uxy(Nx+Ix,2,I0) + sy(i).uxy(Nx+Ix,2,I1) )/hy2 ;
end

clear sx sy

for i=1:Ntc
    for j=1:Nt
        g(i).xy(1:Nxy,j) = hi*reshape(ff(i).fxy(1:Nx,:,j), Nxy,1);
        g(i).xy(Nxy+1:2*Nxy,j) = hi*reshape(ff(i).fxy(Nx+1:2*Nx,:,j), Nxy,1);
```

```
    end
end

%----------------------------------------
%   Initialisation
%----------------------------------------
ss = uxy(x,y,0., tc) ;
%SOL. Init
u0(1:Nxy,1) = reshape(ss(1).uxy(1:Nx,:,1), Nxy,1) ;
u0(Nxy+1:2*Nxy,1) = reshape(ss(1).uxy(Nx+1:2*Nx,:,1), Nxy,1) ;

uu=zeros(2*Nxy,Ntc);
vv=zeros(2*Nxy,Ntc);

%-----------------
%Etape 1
%-----------------

for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:2*Nxy,j) = u ;
end

%-----------------
%Etape 2
%-----------------
vv(:,1) = u0 ;

version = 3 ;
if(version == 1)
    % Version totale
    EA = expm( dt*L) ;
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end
if(version ==2)
    %  Version Arnoldi
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
```

```
    km = 3 ;
    for j=2:Ntc
        ww = prodexp(km,dt,L,vv(:,j-1) ) ;
        vv(:,j) = uu(:,j-1) +ww ;

    end
end

if(version ==3)
    %  Version approx exponentielle
%     pct = .1 ;
    % km = max(1,floor(Nxy/100*pct))
    km =3 ;
    EA = speye(2*Nx*Ny) ;
    L0 = EA ;
    for k=1:km
        L0 = L0*L*dt ;
        EA = EA + L0/factorial(k) ;
    end
    for j=2:Ntc
        vv(:,j) = uu(:,j-1) + EA * vv(:,j-1) ;
    end
end

%-----------------
%Etape 3
%-----------------

for j=1:Ntc
    v = vv(:,j) ;
    for i=1:Nt
        k=(j-1)*Nt+i;
        u=AU\( AL\( AA*v+g(j).xy(:,i) ) );
        v = u ;
    end
    uu(1:2*Nxy,j) = u ;
end

%-----------------
%Fin Block algorithm
%-----------------

clear u
u = zeros(2*Nx,Ny,Ntc) ;
for j=1:Ntc
```

```
    u(1:Nx,1:Ny,j)=reshape(uu(1:Nxy,j),Nx,Ny);
    u(Nx+1:2*Nx,1:Ny,j)=reshape(uu(Nxy+1:2*Nxy,j),Nx,Ny);
end
```

Conclusion

Throughout this work, we have been delving with the construction and analysis of schemes arising from the Tensor-product space-time method using Euler and Crank Nicolson scheme in time to solve the heat equation and the construction and application of the Block method using Euler and Crank-Nicolson applied to the heat equation and elasticity equation.

I draw here a summary of the main contributions of this research.
In pursuing the first objective of solving the system of the Heat equation, I analyzed the Tensor-product space-time method with the Euler scheme in time. Then I studied the error of the numerical method to find, for any given error tolerance, the optimal parallelism of the method. This was validated by the numerical applications. Then, I presented the Tensor-product space-time method with the Newmark scheme in time, which was never done before for the Heat equation and showed that is stable and has second order accuracy. I also defined the optimal parallelism of the Tensor-Newmark method for heat equation and the optimal number of parallel processors. The numerical results show that the Tensor-product space-time method works well for the Euler and the Crank Nicolson scheme in time (it can work for any backward scheme) but because of the condition number of the eigenvectors matrix S, the optimal number of processors is not very high: 9 processors for Euler scheme and 8 for Crank Nicolson scheme for a given tolerance of error 5%. This means, if we want to have a good speed up of the method, while preserving accuracy (i..e the series of time steps close to the fixed time step (ρ closes to 1), the optimal number of parallel processors will be defined, hence the Tensor product method cannot be applied for any number of processors. However, knowing that the Tensor-product method solves PDEs totally in parallel, each parallel processor solves a similar space problem, one can always introduce a method of parallelization in space in combination with the time-parallel method to solve the PDE thus adding another dimension to the parallelization process through a completely parallel time-space subdomains. In addition, taking advantage of the special properties of the time steps series (with a growth rate depending on ρ), this method can work very efficiently for physical problems which need to have very small time steps at the beginning and large time steps at the end.

In the second part, I introduced the Block method with Euler and Cranck-Nilcolson scheme in time. Then I applied the Block method to solve two different PDEs: the Heat equation and the Elasticity equation. Applications in two dimension of the two PDEs give good results validating the applications of the method to different PDEs. Different from the Tensor-product method, the Block method doesn't have any constraint on the time series therefore it is a very scalable method which can work for any time scheme. Having the same structure as the "Parareal" method but being a direct method makes the Block method extendable by using the Tensor product method inside each coarse time step to solve the refined problems. The Block method can surprisingly provide a better initial condition especially for low order schemes thus an improvement of the sequential method. For a higher order scheme it does not deteriorate the scheme order and accuracy. Combination of Block and Tensor-product method will allow for a good speed-up, thus opening successfully another dimension to parallelization.

Perspectives

A study of the optimal parallelization of the tensor product method for higher order time derivative elasticity equation is to be made. Combination of the two time-parallel methods is to be implemented: using Block method to solve the coarse problems and Tensor-product to solve the refined problems to have an even better direct time-parallel method.
Finally , a combination of Tensor-product - Block method with a space domain decomposition method solving a problem completely parallel in time-space subdomains will offer a very good granularity ideal for large size computations on massively large clusters.

Bibliography

[1] P. Amodio and L. Brugnano, *Parallel implementation of block boundary value methods for ODEs*, Journal of Computational and Applied Mathematics **78** (1997), 197–211 (English).

[2] _____, *Parallel ODE solvers based on block BVMs*, Applied Numerical Mathematics **7** (1997), no. 1-2, 5–26 (English).

[3] _____, *Parallel solution in time of ODEs:some achievements and perspectives*, Applied Numerical Mathematics **59** (2008), no. 3-4, 424–435 (English).

[4] M. Arioli, B. Codenotti, and C. Fassino, *The Padé method for computing the matrix exponential*, Linear Algebra and its Applications **240** (1996), 111–130 (English).

[5] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, The Quarterly of Applied Mathematics **9** (1951), 17–29 (English).

[6] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zerah, *Parallel-in-time molecular-dynamics simulations*, Phys. Rev. E **66** (2002), 057706–1–4 (English).

[7] G. Bal, *On the convergence and stability of the parareal algorithm to solve partial differential equations*, The 15th International Domain Decomposition Conference (Berlin) (R. H. W. Hoppe, J. Periaux, O. Pironneau, J. Xu, and Olof Widlund, eds.), Springer, 2002, pp. 426–432.

[8] E. Bertolazzi, *Matrix exponential: Integration lectures for the course: Numerical methods for dynamical system and control*, Tech. report, UNITN, 2009.

[9] M. Borel, L. Halpern, and J. Ryan, *Euler/navier-stokes couplings for multiscale aeroacoustic problems*, Computing Fluid Dynamics (2010), 427–433 (English).

[10] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numerica 1994, Cambridge University Press, 1994, pp. 61–143.

[11] V. Dolean, M. Gander, and L. Gerardo-Giorda, *Optimized Schwarz methods for Maxwell equations*, SIAM Journal on Scientific Computing **31** (2009), no. 3, 2193–2213 (English).

[12] J.-P. Fang, *Extensions of q–Chu–Vandermonde's identity*, Journal of Mathematical Analysis and Applications **339** (2008), no. 2, 845 – 852.

[13] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: Theory and feasibility studies for fluid, structure and fluid-structure applications*, Internat, J. Numer. Methods Engrg. **58** (2003), 1397–1434 (English).

[14] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, *FETI-DP: a dual-primal unified FETI method-part I: A faster alternative to the two-level FETI method*, International Journal for Numerical Methods in Engineering **50** (2001), no. 7, 1523–1544 (English).

[15] C. Farhat and F. X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Meths. Engrg. (1991), no. 32, 1205–1227 (English).

[16] C. Farhat and F.-X. Roux, *Non-overlapping domain decomposition methods in structural mechanics*, Archives of Computational Methods in Engineering **13** (2006), no. 4, 515–572 (English).

[17] C. Farhat, F. X. Roux, and J. Mandel, *Optimal convergence properties of the FETI domain decomposition method*, Comput. Meth. Appl. Mech. Engrg. (1994), no. 115, 365–385 (English).

[18] F. Fischer, F. Hecht, and Y. Maday, *A parareal in time semi-implicit approximation of the Navier-Stokes equations*, The 15th International Domain Decomposition Conference (Berlin) (R. H. W. Hoppe, J. Periaux, O. Pironneau, J. Xu, and Olof Widlund, eds.), Springer, 2003, pp. 433–440.

[19] M. J. Gander and L. Halpern, *Absorbing boundary conditions for the wave equation and parallel computing*, Math. Comp. **74** (2005), 153–176 (English).

[20] M. J. Gander, L. Halpern, and F. Magoulès, *An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation*, Int. J. Numer. Meth. Fluids **55** (2007), 163–175 (English).

[21] M. J. Gander, F. Magoulès, and F. Nataf, *Optimized Schwarz Methods without Overlap for the Helmholtz Equation*, SIAM J. Sci. Comput. **24** (1994), no. 1, 38–60 (English).

[22] Martin J. Gander, Laurence Halpern, and Frédéric Nataf, *Optimized Schwarz methods*, Twelfth International Conference on Domain Decomposition Methods, Chiba, Japan (Bergen) (Tony Chan, Takashi Kako, Hideo Kawarada, and Olivier Pironneau, eds.), Domain Decomposition Press, 2001, pp. 15–28 (English).

[23] I. Garrido, M. S. Espedal, and G. E. Fladmark, *A convergence algorithm for time parallelization applied to reservoir simulation*, The 15th International Domain Decomposition Conference (Berlin) (R. H. W. Hoppe, J. Periaux, O. Pironneau, and Olof Widlund J. Xu, eds.), Springer, 2003, pp. 469–476.

[24] G. Gasper and M. Rahman, *Basic hypergeometric series*, second ed., Encyclopedia of Mathematics and its Applications, vol. 96, Cambridge University Press, Cambridge, 2004, With a foreword by Richard Askey.

[25] W. Hackbusch, *Fast numerical solution of time-periodic parabolic problems by a multigrid method*, SIAM Journal on Scientific and Statistical Computing **2** (1981), no. 2, 198–206.

[26] L. Halpern and F. Hubert, *A new finite volume Schwarz algorithm for advection-diffusion equations*, 21st International Conference on Domain Decomposition Methods, Rennes, Springer, 2013, To appear.

[27] C. Hirsch, *Numerical computation of internal and external flows*, second ed., Elsevier, 2007, Fundamentals of Computational Fluid Dynamics.

[28] G. Horton, *The time-parallel multigrid method*, Communications in Applied Numerical Methods **8** (1992), no. 9, 585–595.

[29] T. J. R. Hughes, *The finite element method*, Prentice Hall Inc., Englewood Cliffs, NJ, 1987, Linear static and dynamic finite element analysis, With the collaboration of Robert M. Ferencz and Arthur M. Raefsky.

[30] H. B. Keller, *Numerical methods for two-point boundary-value problems*, Blaisdell Publishing Co. Ginn and Co., Waltham, Mass.-Toronto, Ont.-London, 1968 (English).

[31] A. Klawonn, O. B. Widlund, and M. Dryja, *Dual-Primal FETI Methods for Three-Dimensional Elliptic Problems with Heterogeneous Coefficients*, SIAM J. Numer. Anal. **40** (2000), no. 1, 159–179 (English).

[32] P. Lascaux and R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur. Tome 2*, second ed., Masson, Paris, 1994, Méthodes itératives.

[33] J. L. Lions, Y. Maday, and G. Turinici, *A parareal in time discretization of PDE's*, C.R. Acad. Sci. Paris, Serie I **332** (2001), 661–668.

[34] P. L. Lions, *On the Schwarz alternating method I*, First International Symposium on Domain Decomposition Methods for Partial Differential Equations (Philadelphia, PA) (Roland Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, eds.), SIAM, 1988, pp. 1–42.

[35] _____, *On the Schwarz alternating method II: Stochastic interpretation and orders properties*, Domain Decomposition Methods (Philadelphia, PA) (T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds.), SIAM, 1989, pp. 47–70.

[36] _____, *On the Schwarz alternating method III: A variant for nonoverlapping subdomains*, Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, held in Houston, Texas, March 20-22, 1989 (Philadelphia, PA) (T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds.), SIAM, 1990, pp. 202–223.

[37] R. E. Lynch, J. R. Rice, and D. H. Thomas, *Direct solution of partial difference equations by tensor product methods*, Numerische Mathematik **6** (1964), no. 1, 185–199 (English).

[38] _____, *Tensor product analysis of partial difference equations*, Bull. Amer. Math. Soc. **70** (1964), no. 3, 378–384 (English).

[39] _____, *Tensor product analysis of alternating direction implicit methods*, Journal of the Society for Industrial and Applied Mathematics **13** (1965), no. 4, 995–1006 (English).

[40] Y. Maday, T. Bjøntegaar, and E. Rønquist, *Fast tensor product solvers. Part I: Partially deformed three-dimensional domains*, Journal of Scientific Computing **39** (2009), no. 1, 28–48 (English).

[41] Y. Maday and E. M. Rønquist, *Fast tensor product solvers. Part II: Spectral discretization in space and time*, Tech. Report 7-9, Laboratoire Jacques-Louis Lions, 2007.

[42] _____, *Parallelization in time through tensor-product space-time solvers*, C. R. Math. Acad. Sci. Paris **346** (2008), no. 1-2, 113–118.

[43] Y. Maday and G. Turinici, *A parallel-in-time procedure for the control partial differential equations*, C.R. Math. Acad. Sci. Paris. Ser. I Math **335** (2002), 387–391 (English).

[44] _____, *The parareal in time iterative solver: a further direction to parallel implementation*, Proceedings of the 15th international domain decomposition conference (R. Kornhuber, R. H. W. Hoppe, J. Périaux, O. Pironneau, O. B. Widlund, and J. Xu, eds.), Springer LNCSE, 2003, pp. 441–448.

[45] K. Miller, *Numerical analogs to the Schwarz alternating procedure*, Numer. Math. **7** (1965), 91–103.

[46] W. Miranker and W. Liniger, *Parallel Methods for the Numerical Integration of Ordinary Differential Equations*, Math. Comp. **91** (1967), 303–320 (English).

[47] J. Nievergelt, *Parallel Methods for Integrating Ordinary Differential Equations*, Comm. of the ACM **7** (1964), no. 12, 731–733 (English).

[48] J. S. Przemieniecki, *Matrix structural analysis of substructures.*, Am. Inst. Aero. Astro. J. **1** (1963), 138–147.

[49] A. Quarteroni and A. Valli, *Domain decomposition methods for partial differential equations*, Oxford Science Publications, 1999.

[50] Robert D. Richtmyer and K. W. Morton, *Difference methods for initial-value problems*, Second edition. Interscience Tracts in Pure and Applied Mathematics, No. 4, Interscience Publishers John Wiley & Sons, Inc., New York-London-Sydney, 1967.

[51] M. H. Saad, *Elasticity: Theory, Applications and Numerics*, Elsevier, Amsterdam, 2009.

[52] Y. Saad, *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.

[53] H. Schwarz, *Über einen Grenzübergang durch alternierendes Verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich **15** (1870), 272–286.

[54] B. F. Smith, P. E. Bjørstad, and W. Gropp, *Domain decomposition: Parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, 1996.

[55] G. Staff and E. Rønquist, *Stability of parareal algorithm*, Proceedings of the 15th international domain decomposition conference (Berlin) (Ralf Kornhuber, Ronald H. W. Hoppe, Jacques Périaux, Olivier Pironneau, Olof B. Widlund, and Jinchao Xu, eds.), Springer, 2003, pp. 449–456.

[56] A. Toselli and O. Widlund, *Domain decomposition methods - algorithms and theory*, Springer Series in Computational Mathematics, vol. 34, Springer, 2005.

[57] D. E. Womble, *A time stepping algorithm for parallel computers*, SIAM J.Sci. Statist. Comput. **11** (1990), 824–837 (English).