Algorithme parallèle pour le calcul d'orbites

Parallélisation à travers le temps

Jocelyne Erhel — Stéphanie Rault

INRIA-IRISA
Campus de Beaulieu
35042 Rennes

erhel@irisa.fr

RÉSUMÉ. *L'exploitation de satellites en orbite implique une évaluation précise et fréquente de leur position. Pour calculer la trajectoire du satellite, il faut résoudre un système d'équations différentielles du second ordre. La résolution de tels systèmes est réalisée à l'aide de méthodes d'intégration numérique qui permettent d'atteindre la précision souhaitée. Le but de notre étude est de proposer une méthode numérique de calcul d'orbite qui soit précise et rapide. Pour réduire le temps de calcul, nous proposons un algorithme parallèle de calcul d'orbites. Le parallélisme consiste à résoudre simultanément l'équation différentielle sur autant de sous-intervalles que de processeurs puis à décaler la fenêtre des sous-intervalles. Un procédé itératif de correction permet d'obtenir la solution finale. Une étude sur la convergence de la méthode, ainsi que la prise en compte de la géométrie du problème, nous permettent ensuite d'optimiser les performances de notre méthode. Nous proposons un algorithme de découpage automatique de l'intervalle de temps qui minimise la durée des calculs dans notre algorithme. Enfin, nous présentons les résultats numériques des différents algorithmes exécutés sur une machine de 64 processeurs.*

ABSTRACT. *The control of satellites needs accurate and frequent evaluations of their positions. Satellite trajectories are computed by solving a second order system of differential equations. In order to get a high accuracy, numerical integrators are considered to solve the problem. The aim of our study is to establish a parallel method in order to reduce the computational time. The method splits the integration interval into several chunks and solves independently on each subinterval. The parallelization is performed by computing simultaneously as many subintervals as processors. After each correction we move forward the window of intervals and leave the subintervals where convergence is obtained. This window is moved until global convergence. We propose an algorithm for automatic splitting of interval time which allows to reduce the computational time. Numerical results on a parallel computer of 64 processors exhihit performances of each algorithm.*

MOTS-CLÉS : *Equations différentielles, schéma de Cowell, parallélisation "à travers le temps", mémoire virtuelle partagée, évaluation de performances.*

KEYWORDS: *Differential equations, Cowell's scheme, parallelism "across the time", Distributed Shared Memory, performance evaluation.*

1. Introduction

Les véhicules spatiaux, au même titre que les véhicules terrestres et maritimes, ont recours à des systèmes de localisation. On doit en effet, à tout moment de leur mission, être en mesure de pointer sur eux des antennes de communication ou de localisation, prévoir l'évolution ultérieure de l'orbite, contrôler l'orbite en cas de besoin. La mécanique spatiale a besoin de méthodes d'orbitographie de plus en plus fiables, en particulier avec l'évolution de la précision des mesures d'observation. On utilise généralement pour chaque mission spatiale un modèle d'orbite adapté qui permet d'extrapoler la trajectoire.

Le but de notre étude est de concevoir une méthode numérique de calcul d'orbites, qui soit à la fois précise et rapide. L'utilisation des machines parallèles permet de diminuer le temps de calcul. C'est pourquoi, nous avons axé notre étude sur la construction d'algorithmes parallèles pour le calcul d'orbites.

Le calcul de la trajectoire d'un satellite en orbite se ramène à l'intégration d'un système d'équations différentielles du second ordre sur un long intervalle de temps. Ce système peut s'écrire comme un système du premier ordre tel que :

$$\left\{ \begin{array}{l} y'(t) = F(t, y(t)) \\ y(T_0) = y_0 \quad t \in [T_0, T] \end{array} \right. \qquad [1]$$

où $y \in \mathbb{R}^6$ est la position et la vitesse du satellite et F contient la force agissant sur le satellite. On peut définir la position du satellite sur l'orbite de deux façons : soit par ses coordonnées cartésiennes (ou rectangulaires), $x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3$, les coordonnées du vecteur position et du vecteur vitesse, soit par ses coordonnées képlériennes (ou éléments orbitaux), $(a, e, i, \omega, \Omega, M)$, où a est le demi-grand axe, e l'excentricité, i l'inclinaison, ω et Ω sont des angles et M l'anomalie moyenne.

Au voisinage de la terre, on considère qu'un objet n'est soumis qu'à une seule force, l'attraction terrestre supposée centrale. La trajectoire est alors képlérienne, $(a, e, i, \omega, \Omega)$ décrit une ellipse autour de la terre et M représente la position du satellite sur cette ellipse. En réalité, bien d'autres forces viennent perturber le mouvement du satellite :

1. le frottement atmosphérique : $\overrightarrow{F_a}$,
2. le potentiel luni-solaire : $\overrightarrow{F_l}$ et $\overrightarrow{F_s}$,
3. la pression solaire : $\overrightarrow{F_p}$,
4. la force $\overrightarrow{F_t}(nz, ntd)$ qui dérive du potentiel terrestre u tel que :

$$\begin{aligned} u = \frac{\mu}{r} \Big\{ & 1 + \sum_{k=1}^{nz} \left(\frac{r_{eq}}{r}\right)^k (J_k P_k(\sin\varphi)) \\ & + \sum_{n=1}^{ntd} \sum_{m=2}^{n} \left(\frac{r_{eq}}{r}\right)^n (C_{n,m} \cos(m\lambda) + S_{n,m} \sin(m\lambda)) P_{n,m}(\sin\varphi) \Big\} \end{aligned} \qquad [2]$$

où :

– μ est la constante de gravitation universelle,

– r est la distance centre de la terre-satellite,
– r_{eq} est le rayon équatorial,
– P_k est le polynôme de Legendre de degré k,
– $P_{n,m}$ est la fonction de Legendre de degré n et d'ordre m,
– J_k est l'harmonique zonal d'ordre k (coefficients donnés,$J_1 = 0$),
– $C_{n,m}$ et $S_{n,m}$ sont les harmoniques tesseraux de degré n et d'ordre m.
– nz, ntd sont les valeurs maximales de k et n.
– φ et λ sont respectivement la latitude et la longitude.

La force $\overrightarrow{F_t}$ dérive du potentiel terrestre (2) et s'écrit donc :

$$\overrightarrow{F_t}(nz, ntd) = \overrightarrow{grad}(u)$$

La force $\overrightarrow{F}$ qui agit sur le satellite s'écrit, alors, comme la résultante des forces perturbatrices décrites ci-dessus, c'est-à-dire :

$$\overrightarrow{F} = \overrightarrow{F_t} + \overrightarrow{F_l} + \overrightarrow{F_s} + \overrightarrow{F_a} + \overrightarrow{F_p}. \quad [3]$$

Dans ce cas le système (1) ne peut plus être intégré exactement. La méthode choisie par le CNES (Centre Nationale des Etudes Spatiales) utilise le schéma numérique de Cowell ([HAU 95]) qui est basé sur un schéma prédicteur-correcteur du type Adam-Moulton. La parallélisation de tels calculs peut s'effectuer de trois manières différentes ([GEA 86]) :

– à "travers le problème",
– à "travers le schéma",
– à "travers le temps",

La technique de parallélisation appelée à "travers le problème" par C. Gear ([GEA 86]) consiste à répartir les composantes du second membre du système sur les différents processeurs. Pour notre problème le second membre est intrinsèquement séquentiel et cette méthode ne donne pas de gain important ([BEA 95]). La méthode dite de parallélisation "à travers le schéma" consiste à remplacer le schéma d'intégration de Cowell par un schéma parallèle. L'ouvrage de K. Burrage ([BUR 95]) présente un large éventail des méthodes parallèles pour les équations différentielles. Parmi elles, les méthodes DIMSIM, développées par J.C. Butcher ([BUT 93a], [BUT 93b]) et les méthodes L-stables, développées par P. Chartier ([CHA 94]). Il existe aussi des méthodes parallèles basées sur un schéma d'extrapolation (voir [ITO 97]) et sur le schéma Stormer-Cowell ([HOU 98]) qui ont été construites pour les problèmes de calcul d'orbite. Même si ces schémas se parallélisent très bien, l'accélération obtenue par rapport au schéma de référence (prédiction-correction) est peu importante.

Pour notre étude, nous avons choisi de conserver le schéma d'intégration de Cowell et nous nous intéressons à la méthode de parallélisation à "travers le temps". Cette méthode est basée sur les travaux de A. Bellen et M. Zennaro ([BEL 89]) et sur les travaux de P. Chartier et B. Philippe ([CHA 93a] et [CHA 93b]). Nous proposons ensuite une mise en œuvre parallèle de cette méthode sur une machine à mémoire virtuelle partagée (l'Origin 2000). Les résultats numériques montrent les performances de notre algorithme. L'étude théorique de la vitesse de convergence et la prise en compte de la géométrie du problème nous permettent ensuite de proposer un algorithme de découpage automatique de l'intervalle de temps. Enfin, nous présentons les résultats numériques obtenus à l'aide de l'algorithme ainsi optimisé.

2. Parallélisation à "travers le temps"

2.1. *Méthode générale*

On désire résoudre le système d'équations différentielles donné par (1) sous les hypothèses :

– $y \in \mathbb{R}^6$,
– F est lipschitzienne sur $[T_0, T] \times \mathbb{R}^6$,
– F est continûment différentiable sur $[T_0, T] \times \mathbb{R}^6$,
– $\frac{\partial F}{\partial y}$ est lipschitzienne sur $[T_0, T] \times \mathbb{R}^6$.

On veut construire une méthode qui soit parallèle à travers le temps. Pour cela, suivant [CHA 93b], on partitionne l'intervalle $[T_0, T]$ en N sous-intervalles $[T_i, T_{i+1}]$, pour $i = 0, N-1$ avec $T_N = T$, sur lesquels on pourra mener des intégrations indépendantes.

On note $y_0, y_1, ..., y_{N-1}$ les conditions initiales aux temps $T_0, T_1, ..., T_{N-1}$. Soit $\varphi(y_i, T_i, T_{i+1})$ la solution au temps T_{i+1} du problème donné en (4) de condition initiale y_i au temps T_i.

$$\left\{ \begin{array}{l} y' = F(t, y(t)), \\ y(T_i) = y_i \quad t \in [T_i, T_{i+1}]. \end{array} \right. \qquad i = 0, N-1 \qquad [4]$$

Alors, connaissant les conditions initiales $y_0, y_1, ..., y_{N-1}$, les N solutions $\varphi(y_0, T_0, T_1), ..., \varphi(y_{N-1}, T_{N-1}, T_N)$ peuvent être calculées en parallèle.
La figure 1 illustre les intégrations des N sous-intervalles en parallèle.

On recherche une suite $y_0, .., y_{N-1}$ telle que $y_i = \varphi(y_{i-1})$ pour $i = 1, ..N-1$ pour que les solutions des N intégrations puissent être vues comme des restrictions de la solution générale du problème (1).
Soit Φ la fonction telle que

$$\Phi : Y = (y_0, y_1, ..., y_{N-1})^T \longrightarrow \Phi(Y) = (y_0, \varphi(y_0), ..., \varphi(y_{N-2}))^T. \qquad [5]$$

Pour simplifier l'écriture, on note $\varphi(y_i) = \varphi(y_i, T_i, T_i + 1)$, pour $i = 0, N-2$. La solution de (1) est alors la solution d'un problème de point fixe donné par (6).

$$\Phi(Y) = Y. \qquad [6]$$

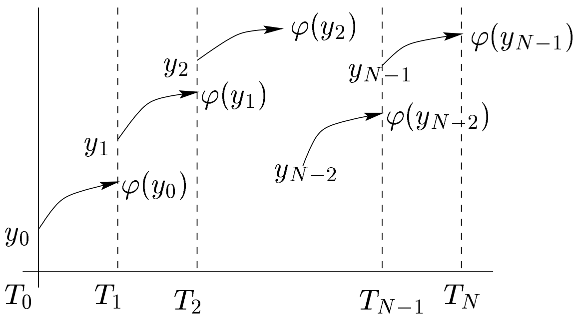

Figure 1. *Partition de l'intervalle de temps*

Ce problème qui s'écrit aussi $(I - \Phi)(Y) = 0$, peut se résoudre par une méthode itérative classique qui est la méthode de Newton. La k ième itération de Newton s'écrit :

$$(I - J_\Phi^{[k]})(Y^{[k+1]} - Y^{[k]}) = -Y^{[k]} + \Phi(Y^{[k]}), \qquad [7]$$

avec $J_\Phi^{[k]}$ le jacobien de Φ donné par (8) (voir [HAI 93]).

$$J_\Phi^{[k]} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ J_\varphi(y_0^{[k]}) & 0 & \cdots & 0 \\ 0 & J_\varphi(y_1^{[k]}) & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & J_\varphi(y_{N-2}^{[k]}) & 0 \end{pmatrix} \qquad [8]$$

En développant l'équation (7) avec l'expression du Jacobien donnée par (8), on obtient la formulation suivante :

$$y_i^{[k+1]} - y_i^{[k]} = -y_i^{[k]} + \varphi(y_{i-1}^{[k]}) + J_\varphi(y_{i-1}^{[k]})(y_{i-1}^{[k+1]} - y_{i-1}^{[k]}) \quad i = 1, N-1. \quad [9]$$

Afin d'appliquer l'itération (9), trois éléments sont à déterminer :

– la condition initiale $Y^{[0]} = (y_0, y_1^{[0]}, ..., y_{N-1}^{[0]})^T$,

– la solution $\varphi(y_{i-1}^{[k]}, T_{i-1}, T_i)$ de l'intégration sur $[T_{i-1}, T_i]$,

– le Jacobien de φ au point y_i, $J_\varphi(y_i^{[k]})$.

Dans le cas où φ et J_φ sont calculés exactement, le théorème suivant permet d'établir la convergence de la méthode (voir [CHA 93a]). Dans tout ce qui suit, $\|.\|$ désigne la norme euclidienne et la norme matricielle induite.

Théorème 2.1 *Soit F le second membre de l'équation différentielle, telle que F est continûment différentiable sur $\mathcal{R} = [T_0, T] \times \mathbb{R}^m$ et globalement lipschitzienne sur $\mathcal{R}$,* *et que $\frac{\partial F}{\partial y}$ soit également lipschitzienne sur le même domaine $\mathcal{R}$. Alors Y^*, la solution de l'équation (6) est un point fixe de Φ et l'itération (9) est localement quadratiquement convergente. c'est-à-dire, si*

$$e^{[k]} = Y^{[k]} - Y^*$$

avec $Y^{[k]}$ la solution à l'itération k de la méthode de Newton (9) et Y^* la solution du point fixe (6).
Alors il existe une constante $K > 0$ telle que :

$$||e^{[k]}|| \leq K||e^{[k-1]}||^2. \quad [10]$$

De plus (voir par exemple [KEL 95]), cette constante s'écrit :

$$K = \gamma||J_H^{-1}(Y^*)||, \quad [11]$$

et γ est la constante de Lipschitz de J_H (avec $H = I - \Phi$).

Preuve. J.Ortega (voir [ORT 70]) montre que $H = I - \Phi$ est continûment différentiable sur $\mathbb{R}^{m\times(N-1)}$, que $H'(Y^*)$ est non singulière, que $H(Y^*) = 0$ et que pour tout $Y \in \mathbb{R}^{m\times(N-1)}$, il existe un réel $\alpha > 0$ tel que, $||(I - \Phi')(Y) - (I - \Phi')(Y^*)|| \leq (\alpha + 1)||Y - Y^*||$. □

2.2. *Application au calcul de la trajectoire d'un satellite*

Reprenons maintenant les trois points précédents :

1. Pour le calcul de la condition initiale $Y^{[0]}$, nous utilisons le modèle analytique développé par Eckstein-Echler (voir [ECK 70]) qui approche la solution exacte. Ce modèle est basé sur une intégration en deux itérations des équations de Lagrange et correspond à un mouvement Képlérien perturbé par les six premiers termes zonaux du potentiel terretre. c'est-à-dire que le second membre $\overrightarrow{F} = \overrightarrow{F_t}(6,0)$ dérive du potentiel terrestre suivant :

$$u(J_6) = \frac{\mu}{r}\left\{1 + \sum_{k=1}^{6}\left(\frac{r_{eq}}{r}\right)^k J_k P_k(\sin\varphi)\right\}, \quad [12]$$

où $J_1 = 0$, $J_2 = -0.11 \times 10^{-2}$, $J_3 = 0.25 \times 10^{-5}$, $J_4 = 0.16 \times 10^{-5}$, $J_5 = 0.22 \times 10^{-6}$, $J_6 = -0.52 \times 10^{-6}$.

On dit alors que $\overrightarrow{F}$ est une *force en* J_6 et que l'on résout un *problème en* J_6. De plus, on note $\varphi_1 : \mathbb{R}^6 \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^6$ la fonction qui exprime les coordonnées képlériennes de l'approximation de la solution du *problème en* J_6 en fonction du temps :
Si $y_i = (a, e, i, \omega, \Omega, M)$ est la condition initiale au temps T_i, alors
$\varphi_1(y_i, T_i, T_{i+1}) = (a + \Delta a, e + \Delta e, i + \Delta i, \omega + \Delta\omega, \Omega + \Delta\Omega, M + \Delta M)$,
est alors la solution au temps T_{i+1} de condition initiale y_i.

On peut trouver dans [ECK 70] la méthode qui permet d'obtenir ce modèle analytique. Il faut cependant remarquer que lors de la mise en œuvre, le calcul de la solution du *problème en* J_6 à l'aide du modèle d'Eckstein-Hechler nécessite la résolution d'une équation de point fixe.

2. Pour les intégrations sur les intervalles $[T_{i-1}, T_i]$, nous utilisons le schéma numérique de Cowell à l'ordre 8. On note $\tilde{\varphi}(y_{i-1}, T_{i-1}, T_i)$ la solution au temps T_i de condition initiale y_{i-1} au temps T_{i-1} calculée par ce schéma. $\tilde{\varphi}(y_{i-1}, T_{i-1}, T_i)$ correspond donc à $\varphi(y_{i-1}, T_{i-1}, T_i)$ aux erreurs d'approximation près. Pour ce calcul, le vecteur y est exprimé en coordonnées rectangulaires.

3. Le Jacobien $J_\varphi(y_i)$ est approché par le Jacobien $J_{\varphi_1}(y_i)$ du modèle analytique φ_1.

Le calcul de la fonction φ_1 nécessite la résolution d'une équation de point fixe. Nous n'avons donc pas d'expression analytique de φ_1 et nous ne pouvons donc pas dériver φ_1 pour obtenir explicitement le Jacobien correspondant. Pour calculer le Jacobien, une solution est d'utiliser une différentiation numérique. Une telle méthode serait assez coûteuse car elle nécessiterait plusieurs évaluations de la solution sur chaque sous-intervalle par le modèle en J_6. Une autre approche est d'utiliser la différentiation automatique.

La différentiation automatique ([GRI 89]) génère, à partir d'un code de calcul d'une fonction, le code de calcul de la dérivée en appliquant les règles de dérivation. La différentiation automatique ne suppose pas que la fonction à dériver doive pouvoir être représentée symboliquement mais suppose qu'une représentation au moyen d'un programme informatique soit disponible. Le programme modélisant le modèle en J_6 qui fait intervenir un processus itératif (on a un point fixe à résoudre) entre dans la classe des programmes pouvant être différentiés automatiquement (voir [GIL 92]). La différentiation automatique n'est pas une différentiation par différences divisées (ou différences finies), méthode par laquelle les dérivées partielles sont estimées. Au contraire, les dérivées obtenues par différentiation automatique sont exactes aux erreurs d'arrondi près ([GIL 91], [CHA 97], [FAU 97a], [FAU 96]). Pour cela, nous disposons d'un logiciel appelé ODYSSEE (voir [FAU 97b]) et développé à l'INRIA qui permet d'effectuer de la différentation automatique.

Enfin il reste à définir le critère de convergence qui est le suivant :

$$\text{si } \frac{||Y^{[k]} - Y^{[k-1]}||}{||Y^{[k]}||} < 10^{-10} \text{ alors } Y^{[k]} \text{ est la solution finale.} \qquad [13]$$

Nous avons donc écrit l'algorithme 1 qui met en œuvre la méthode de partition de l'intervalle de temps pour notre problème de calcul de trajectoire.

ALGORITHME 1: Algorithme général du calcul d'orbite

$y_0^{[0]} = y_0$

– Calcul de la condition initiale – boucle 1

pour i=1,N-1 **faire**

$y_i^{[0]} = \varphi_1(y_0, T_0, T_i)$
fin pour
pour k=0,1,.. jusqu'à convergence **faire**
– Intégration des sous-intervalles par la méthode de Cowell – boucle 2
pour i=k,N-1 **faire**
Résoudre (4) pour calculer $\tilde{\varphi}(y_i^{[k]}, T_i, T_{i+1})$
fin pour
– Calcul d'une approximation du Jacobien et correction – boucle 3
$y_k^{[k+1]} = y_k^{[k]}$
pour i=k,N-1 **faire**
calculer $J_{\varphi_1}(y_i^{[k]})$
$y_{i+1}^{[k+1]} - y_{i+1}^{[k]} = -y_{i+1}^{[k]} + \tilde{\varphi}(y_i^{[k]}) + J_{\varphi_1}(y_i^{[k]})(y_i^{[k+1]} - y_i^{[k]})$
fin pour
test de convergence donné par (13)
fin pour

Remarque 2.1 *A l'itération k, l'algorithme calcule la solution sur $[T_k, T_N]$. En effet à chaque itération le premier sous-intervalle converge obligatoirement puisqu'il est intégré à partir d'une condition initiale exacte (aux erreurs numériques près). Par suite, la méthode converge en au plus N itérations.*

Remarque 2.2 *Il est possible d'affiner le test de convergence pour contrôler combien d'intervalles ont convergé.*

3. Mise en œuvre parallèle

3.1. *Algorithme*

3.1.1. *Choix de la machine*

La mise en œuvre de notre algorithme s'effectue sur l'Origin 2000 (64 processeurs) de l'INRIA. L'Origin 2000 est un ensemble de nœuds reliés par plusieurs liens qui permettent à plusieurs processeurs de communiquer simultanément avec d'autres processeurs. Chaque nœud est composé de 2 processeurs R10000, d'une mémoire cache par processeur, le tout connecté à un HUB qui gère les communications *intranœud* (entre les nœuds du sous-système) et les communications *internœud* avec les HUB des autres nœuds. L'Origin 2000 utilise une mémoire DSM (Distributed Shared Memory), autrement dit mémoire virtuelle partagée. Dans ce cas, la mémoire principale est physiquement distribuée entre les processeurs mais accessible et partagée logiquement par tous les processeurs.

3.1.2. *Modèle de progammation*

Il existe plusieurs modèles de programmation pour exécuter en parallèle le programme source. Le code original du calcul de la trajectoire d'un satellite en orbite nous a été fourni par le CNES (Centre National des Etudes Spatiales). Ce code comprend le calcul des forces agissant sur le satellite ainsi que le calcul de l'éphéméride par le schéma de Cowell et est écrit en FORTRAN 77. Pour l'écriture du code parallèle nous avons utilisé une programmation à l'aide de directives que le compilateur interprète de façon à pouvoir exécuter le programme en parallèle. Ces directives portent essentiellement sur :

– les variables partagées : les variables sont par défaut partagées, à moins qu'une annotation ne spécifie qu'une variable est d'un accès privé pour le processeur qui exécute l'instruction.

– les boucles parallèles : les boucles sont par défaut séquentielles, à moins qu'une directive ne spécifie qu'une boucle est parallèle et qu'elle doit être distribuée sur les processeurs selon tel ou tel autre schéma.

Nous avons utilisé la directive C$DOACROSS permet d'exécuter en parallèle une boucle donnée. On peut distribuer les itérations sur les différents processeurs, tout en contrôlant la répartition du travail. Cette directive peut être utilisée avec des options telles que SHARE et LOCAL, qui spécifient respectivement les variables partagées par tous les processeurs et les variables locales à chaque processeur.

3.1.3. *Analyse des dépendances*

L'analyse des dépendances de données dans l'algorithme séquentiel 1 montre clairement que la boucle 2 (i=k,N-1) est parallèle. En effet, la fonction $\tilde{\varphi}$ qui représente l'intégration par la méthode de Cowell sur l'intervalle $[T_i, T_{i+1}]$ ne dépend que de $y_i^{[k]}$, T_i et T_{i+1}. La condition initiale $y_i^{[k]}$ est le résultat de l'itération de Newton précédente et est donc connue. La boucle 1 correspondant à l'initialisation est aussi parallèle, cependant il ne faut pas attendre de gain important de cette parallélisation car la charge de calcul est très petite. La boucle 3 est séquentielle puisqu'il s'agit d'une récurrence.

3.1.4. *Parallélisation de la boucle avec fenêtre glissante*

La méthode que nous avons développée consiste à intégrer NPROC sous-intervalles simultanément sur NPROC processeurs. Après chaque correction, la fenêtre des NPROC sous-intervalles glisse de ncv sous-intervalles, ncv étant le nombre de sous-intervalles qui ont convergé. La fenêtre se déplace ainsi jusqu'à ce que tous les sous-intervalles aient convergé. Le tableau 1 illustre la méthode pour NPROC=8 processeurs et pour N=15 sous-intervalles (pj est le processeur numéro j et CV signifie que l'on a convergé sur l'intervalle en question).

Pour l'algorithme parallèle, il faut donc introduire un nouveau paramètre (noté NCV) qui compte le nombre de sous-intervalles ayant convergé. Une directive permet, après chaque correction, d'intégrer les sous-intervalles par le même processeur qu'à l'itération précédente. Sur l'exemple du tableau 1, le sous-intervalle i reste sur le processeur i modulo NPROC. La directive AFFINITY(i) = THREAD(j) affecte le sous-intervalle numéro i au processeur pj, où $j = Mod(i-1, nproc) + 1$ (Mod est le

sous-intervalles	1	2	3	4	5	6	7	8	9	10
itération 1	p1	p2	p3	p4	p5	p6	p7	p8		
itération 2	**cv**	p2	p3	p4	p5	p6	p7	p8	p1	
itération 3	cv	**cv**	p3	p4	p5	p6	p7	p8	p1	p2
itération 4	cv	cv	**cv**	**cv**	**cv**	**cv**	**cv**	**cv**	**cv**	**cv**

Tableau 1. *Exemple de parallélisation à l'aide de la deuxième méthode*

modulo). La commande MP_NUMTHREADS() permet de connaître le nombre de processeurs utilisés. L'algorithme est alors le suivant :

ALGORITHME 2: Algorithme parallèle du calcul d'orbite

TOL donné
y_0 donné
$y_0^{[0]} = y_0$
– initialisation du nombre d'itérations –
$k = 0$
– initialisation du nombre de sous-intervalles ayant convergé –
NCV=0
– nombre de processeurs –
NPROC=MP_NUMTHREADS()
– calcul de la condition initiale – boucle 1
C$ DOACROSS LOCAL(i,$y_i^{[0]}$)
pour i=1,N-1 **faire**
    $y_i^{[0]} = \varphi_1(y_0^{[0]}, T_0, T_i)$
fin pour
– itération de Newton –
tant que NCV $<$ N **faire**
    – intégration par la méthode de Cowell – boucle 2 parallèle
C$ DOACROSS LOCAL(i,$y_i^{[k]}$), AFFINITY(i)=THREAD(Mod(i-1,nproc)+1)
    pour i=NCV,max(N-1,NCV+NPROC-1) **faire**
        calcul de $\tilde{\varphi}(y_i^{[k]}, T_i, T_{i+1})$
    fin pour
    – approximation du Jacobien et correction – boucle 3 séquentielle
    $y_{ncv}^{[k+1]} = y_{ncv}^{[k]}$
    pour i= NCV,max(N-1,NCV+NPROC-1) **faire**
        calcul de $J_{\varphi_1}(y_i^{[k]})$
        $y_{i+1}^{[k+1]} - y_{i+1}^{[k]} = -y_{i+1}^{[k]} + \tilde{\varphi}(y_i^{[k]}) + J_{\varphi_3}(y_i^{[k]})(y_i^{[k+1]} - y_i^{[k]})$
    fin pour
    – test de convergence –
    NCV=NCV+1
    pour i=NCV,max(N-1,NCV+NPROC-1) **faire**

si $\frac{||y_i^{[k+1]} - y_i^{[k]}||}{||y_i^{[k]}||} < \text{TOL}$ **alors**
NCV=NCV+1
$y_{ncv+nproc-1}^{[k+1]} = y_{ncv+nproc-1}^{[0]}$
fin si
fin pour
k=k+1
fin tant que
'Convergence en k itérations'

3.2. *Evaluation des performances*

On considère les notations suivantes :

– NPROC : le nombre de processeurs,
– N : le nombre de sous-intervalles,
– NITER : le nombre d'itérations nécessaires pour obtenir la convergence globale.
– $\tau_{init}^1 = \gamma(N-1)$: le temps d'initialisation de l'algorithme 2 qui correspond au temps d'initialisation de l'algorithme 1,
– β : le temps de correction d'un sous-intervalle de l'algorithme 2,
– τ_{cnes} : le temps d'exécution de l'algorithme de référence (méthode de Cowell),
– τ_{seq}^2 : l'exécution sur un processeur de l'algorithme 2,
– $\tau_{par}^2(\text{NPROC})$: l'exécution en parallèle de l'algorithme 2 sur NPROC processeurs,
– $S_a^2(nproc) = \frac{\tau_{seq}^2}{\tau_{par}^2(nproc)}$: le speed-up *apparent*,
– $S_e^2(nproc) = \frac{\tau_{seq}^2}{\tau_{cnes}}$: le speed-up *effectif*.

Nous pouvons estimer théoriquement le speed-up apparent et le speed-up effectif par la proposition suivante :

Proposition 3.1 *Soient les notations suivantes :* $\rho_1 = \frac{\gamma}{\tau_{cnes}}$ *et* $\rho_2 = \frac{\beta}{\tau_{cnes}}$.

Si $\frac{\rho_1 N(N-1)}{\text{NPROC}\times\text{NITER}} << 1$ *et si* $\rho_2 N \times \text{NPROC} << 1$ *alors :*

$$S_e^2(\text{NPROC}) \simeq \frac{N}{\text{NITER}}.$$

Si de plus $\rho_1(N-1) << 1$ *alors :*

$$S_a^2(\text{NPROC}) \simeq \frac{N}{\text{NITER}}.$$

Preuve. voir [RAU 98] □

3.3. *Résultats numériques*

La figure 2 représente les speed-up apparents théoriques (en trait pointillé) et réels (en trait plein) pour trois durées d'éphéméride (10, 20 et 40 jours). On remarque que les speed-up réels sont plus petits que les speed-up théoriques lorsque le nombre de processeurs augmente. Il est par exemple de 4 au lieu de 6 pour une durée de 10 jours et de 2 au lieu de 2.5 pour une durée de 40 jours. Le grand écart entre les deux speed-up dans le cas d'un trajectoire de 10 jours provient du fait que l'intervalle de temps n'est pas très grand et donc sur 30 processeurs le temps de correction n'est plus négligeable par rapport au temps d'exécution en parallèle. Or ce temps de correction a été négligé dans le calcul du speed-up théorique (voir 3.1).

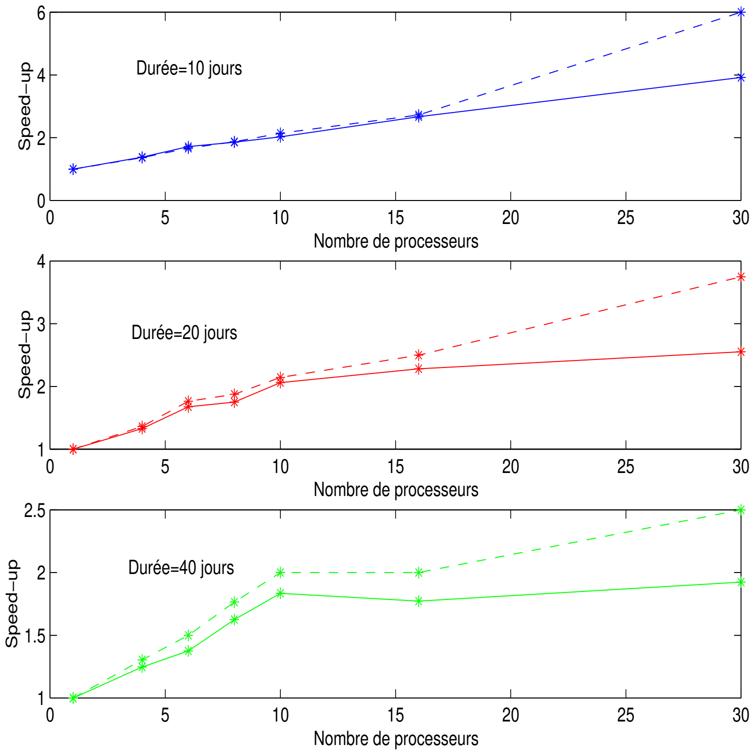

Figure 2. *Speed-up apparent de l'algorithme 2 pour N=30*

La figure 3 représente les speed-up réels effectifs pour les même durées d'éphéméride. Ceux-ci sont plus petits que les speed-up apparents de la figure 2. Par exemple, pour une durée d'éphéméride de 10 jours, le speed-up effectif vaut 3 alors que l'on a un speed-up apparent de 4.

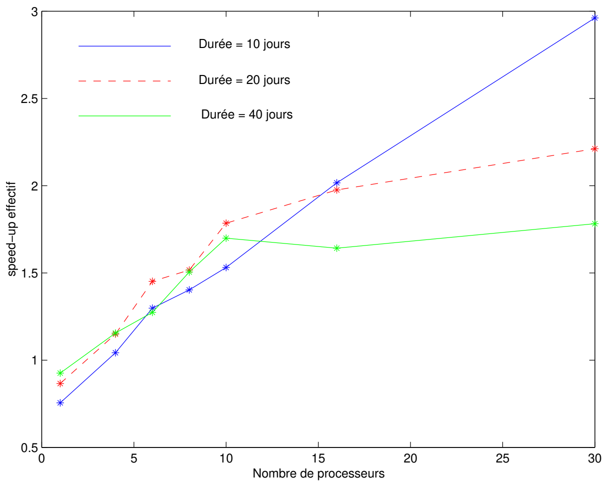

Figure 3. *Speed-up effectif de l'algorithme 2 pour N=30*

Cette différence est due à la partie correction de l'algorithme 2. En effet, plus le nombre de sous-intervalles est grand et plus il y a de corrections à effectuer. Ce surcoût de la partie séquentielle n'existe pas dans l'algorithme de référence. Le modèle de la proposition 3.1 et les résultats montrent que les performances dépendent essentiellement du nombre d'itérations NITER. Nous étudions maintenant les variations de ce paramètre.

4. Etude théorique de la convergence

4.1. *Vitesse de convergence*

Dans le cas où φ et J_φ sont exacts la convergence locale est quadratique (voir le théorème 2.1). Notre algorithme ne calcule pas exactement φ ni J_φ donc la vitesse de convergence n'est plus quadratique. Le but de ce paragraphe est d'étudier les paramètres qui influencent la vitesse de convergence.

Dans la mesure où on approche la fonction φ par $\tilde{\varphi}$ correspondant au schéma numérique de Cowell, on approche aussi Φ par $\tilde{\Phi}$ qui s'écrit :

$$\tilde{\Phi} : Y = (y_0, y_1, ..., y_{N-1})^T \longrightarrow \tilde{\Phi}(Y) = (y_0, \tilde{\varphi}(y_0), ..., \tilde{\varphi}(y_{N-2}))^T.$$

avec, pour simplifier les notations, $\tilde{\varphi}(y_i) = \tilde{\varphi}(y_i, T_i, T_{i+1})$.
Soit ε_i (i=0,N-2), l'erreur due au pas de discrétisation du schéma de Cowell, alors on a :

$$\tilde{\varphi}(y_i) = \varphi(y_i) + \varepsilon_i(y_i), \quad i = 0, N-2,$$

et on a :

$$\tilde{\Phi} = \Phi + \epsilon,$$

avec

$$\epsilon = (0, \varepsilon_0(y_0), \varepsilon_1(y_1), ..., \varepsilon_{N-2}(y_{N-2}))^T.$$

Pour le calcul d'orbite, nous appliquons donc la méthode de Newton à $\tilde{H} = H + \epsilon$, au lieu de H.

En ce qui concerne le Jacobien, nous utilisons le Jacobien du modèle analytique φ_1 qui approche lui aussi φ. Soit δ_i (i=0,N-2), la perturbation sur le Jacobien, c'est-à-dire telle que :

$$J_{\varphi_1}(y_i) = J_\varphi(y_i) + \delta_i(y_i), \quad i = 0, N-2.$$

Soit Δ la matrice :

$$\Delta = \begin{pmatrix} 0 & 0 & & \cdots & 0 \\ \delta_0(y_0) & 0 & & \cdots & 0 \\ 0 & \delta_1(y_1) & 0 & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & & \delta_{N-2}(y_{N-2}) & 0 \end{pmatrix}.$$

Alors, l'itération de Newton pour notre problème est calculée avec le Jacobien $(J_H + \Delta)$ au lieu du Jacobien J_H.
L'itération de Newton s'écrit alors en fonction de H et J_H :

$$\tilde{Y}^{[k+1]} = \tilde{Y}^{[k]} - (J_H + \Delta)^{-1}(\tilde{Y}^{[k]})(H + \epsilon)(\tilde{Y}^{[k]})$$

On note :

$$\tilde{e}^{[k]} = \tilde{Y}^{[k]} - Y^*.$$

On a alors le théorème suivant :

Théorème 4.1 *Sous les hypothèses du théorème 2.1, il existe $\eta > 0$ et $\delta > 0$ et $K > 0$ tels que si $||\tilde{e}^{[k+1]}|| \leq \delta$ et $||\Delta(\tilde{Y}^{[k]})|| \leq \eta$, alors $\tilde{H}$ est inversible et :*

$$||\tilde{e}^{[k]}|| \leq \tilde{K} \left(||\tilde{e}^{[k-1]}||^2 + ||\Delta^{[k-1]}|| \quad ||\tilde{e}^{[k-1]}|| + ||\epsilon^{[k-1]}|| \right),$$

avec

$$\tilde{K} = K + 16||J_H^{-1}(Y^*)||^2 ||J_H(Y^*)|| + 4||J_H^{-1}(Y^*)||, \qquad [14]$$

K étant la constante de l'inégalité (10).

Preuve. La preuve de ce théorème est donnée par exemple dans [KEL 95]. □

La convergence n'est donc plus localement quadratique mais elle est linéaire.

On peut aussi exprimer l'erreur locale dans le cas particulier du calcul d'orbite. Cette erreur est donnée par la proposition suivante :

Proposition 4.1 *Si pour tout i =0,N-2* $||J_\varphi(y_i^*)|| \neq 1$ *alors :*

$$||J_H^{-1}(Y^*)|| \leq \frac{\max_{i=0,N-2} ||J_\varphi(y_i^*)||^N - 1}{\max_{i=0,N-2} ||J_\varphi(y_i^*)|| - 1} \leq \frac{||J_\Phi(Y^*)||^N - 1}{||J_\Phi(Y^*)|| - 1}.$$

De plus, on a :

$$||J_H(Y^*)|| \leq 1 + \max_{i=0,N-2} ||J_\varphi(y_i^*)|| = 1 + ||J_\Phi(Y^*)||.$$

et aussi :

$$||\Delta^{[k-1]}|| = \max_{i=0,N-2} ||\delta_i(\tilde{y_i}^{[k-1]})||,$$

$$||\epsilon^{[k-1]}|| \leq \max_{i=0,N-2} ||\varepsilon_i(\tilde{y_i}^{[k-1]})||.$$

Preuve. voir [RAU 98] □

D'après la proposition 4.1, la convergence dépend du $\max_{i=0,N-2} ||\delta_i||$, $\max_{i=0,N-2} ||\varepsilon_i||$ et du $\max_{i=0,N-2} ||J_\varphi(y_i^*)||$.

On peut obtenir une majoration du $||\epsilon^{[k]}||$ en fonction du pas h. La méthode de Cowell utilisée pour notre problème étant d'ordre 8, on a la majoration suivante :

$$\exists C > 0 \text{ telle que } ||\tilde{\varphi}(y_i) - \varphi(y_i)|| = ||\varepsilon_i(y_i)|| \leq Ch^8 \qquad i = 0, N-2$$

Donc l'erreur $||\varepsilon_i(y_i)||$ est relativement petite. La convergence dépend essentiellement de la perturbation sur le Jacobien, $||\Delta^{[k]}||$ ainsi que de la norme du Jacobien $||J_\Phi||$. Nous étudions maintenant ces deux paramètres.

4.2. *Etude de la qualité du Jacobien*

Dans cette partie, nous étudions la qualité du Jacobien J_{φ_1} obtenu par différentiation automatique de la fonction φ_1. Nous nous plaçons dans le cas de la résolution d'un problème où la force $\overrightarrow{F}$ agissant sur le satellite s'écrit :

$$\overrightarrow{F} = \overrightarrow{F}_t(nz, ntd) + A\overrightarrow{F}_a + B\overrightarrow{F}_p + C\overrightarrow{F}_l + D\overrightarrow{F}_s,$$

avec $A, B, C, D \in \{0, 1\}$. Le cas où $\overrightarrow{F}$ est telle que $nz = ntd = 30$ et $A = B = C = D = 1$, est appelé un *problème complet* et $\overrightarrow{F}$ est dite une *force complète*.

Soient les notations suivantes :

– d : la durée de l'éphéméride,
– e : l'excentricité,
– N : le nombre de sous-intervalles,
– $\Delta J = J_{\Phi_1} - J_{Co(nz,ntd,A,B,C,D)}$,

d	e	(17)	(18)	(19)
1 jour	0.0005	0.050	0.046	0.114
10 jours	0.0005	0.043	0.209	0.287
1 jour	0.05	0.184	0.181	0.178
10 jours	0.05	0.231	0.192	0.196

Tableau 2. *Qualité de l'approximation et influence des forces perturbatrices*

avec :

$$J_{\Phi_1} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ J_{\varphi_1}(y_0) & 0 & \cdots & 0 \\ 0 & J_{\varphi_1}(y_1) & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & J_{\varphi_1}(y_{N-2}) & 0 \end{pmatrix}, \qquad [15]$$

et

$$J_{Co(nz,ntd,A,B,C,D)} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ J_{\tilde{\varphi}}(y_0) & 0 & \cdots & 0 \\ 0 & J_{\tilde{\varphi}}(y_1) & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & J_{\tilde{\varphi}}(y_{N-2}) & 0 \end{pmatrix}. \qquad [16]$$

Rappelons que J_Φ est le Jacobien exact donné par (8).

On définit les erreurs relatives suivantes, qui approchent $\frac{||\Delta||}{||J_\Phi||}$ pour trois problèmes :

$$\frac{||J_{\Phi_1} - J_{Co(6,0,0,0,0,0)}||_2}{||J_{Co(6,0,0,0,0,0)}||_2} \qquad [17]$$

$$\frac{||J_{\Phi_1} - J_{Co(6,0,1,1,1,1)}||_2}{||J_{Co(6,0,1,1,1,1)}||_2} \qquad [18]$$

$$\frac{||J_{\Phi_1} - J_{Co(30,30,1,1,1,1)}||_2}{||J_{Co(30,30,1,1,1,1)}||_2} \qquad [19]$$

Le tableau suivant donne ces erreurs en fonction de la durée de l'éphéméride et de l'excentricité, pour N=10.

L'erreur par rapport à *un problème en* J_6 (17) est plus petite dans le cas d'une trajectoire de faible excentricité. De plus, la précision se détériore légèrement si la durée d'intégration augmente.

Les forces perturbatrices (18) détériorent la précision du Jacobien dans le cas de faible excentricité et pour une durée d'éphéméride de 10 jours.

d	e	$\|\|J_{Co(30,30,1,1,1,1)}\|\|_2$
1 jour	0.0005	1.42×10^3
10 jours	0.0005	1.07×10^4
1 jour	0.05	89.31
10 jours	0.05	144.27

Tableau 3. *Evaluation de la norme* $||J_\Phi(Y^*)||$

d	e	nz	ntd	A	B	C	D	Nb d'itérations
1 jour	0.0005	6	0	0	0	0	0	8
1jour	0.0005	30	30	1	1	1	1	8
0.1 jour	0.0005	2	0	0	0	0	0	8
1 jour	0.05	6	0	0	0	0	0	3
1 jour	0.05	6	0	1	1	1	1	3
1 jour	0.05	30	30	1	1	1	1	3
10 jours	0.05	6	0	0	0	0	0	4
10 jours	0.05	6	0	1	1	1	1	4
10 jours	0.05	30	30	1	1	1	1	4

Tableau 4. *Convergence de la méthode de Newton en fonction de la durée et de l'excentricité*

L'erreur par rapport à un *problème complet* (19) devient nettement plus grande dans le cas d'une éphéméride de un jour et d'une excentricité de 0.0005.

Nous étudions maintenant le deuxième facteur $||J_\Phi||$ que nous approchons par J_{co}.

Nous nous intéressons au calcul de la solution dans le cas où toutes les forces sont prises en compte. Nous calculons donc $||J_{Co(30,30,1,1,1,1)}||_2$ pour les différentes durées d'éphéméride et excentricités utilisées dans les exemples précédents.

Le tableau 3 montre que la norme $||J_{Co(30,30,1,1,1,1)}||_2$ est plus petite pour une excentricité de 0.05.

Il apparaît clairement que l'erreur $||\Delta||$ sur le Jacobien est suffisament petite dans tous les cas. Par contre, la norme $||J_\Phi||$ influence fortement la vitesse de convergence.

C'est donc dans le cas de forte excentricité que notre méthode converge le plus vite. De plus, cette norme est plus grande lorsque la durée d'éphéméride est plus longue ce qui explique la convergence moins rapide dans ce cas.

Nous vérifions maintenant expérimentalement l'influence de $\frac{||\Delta||}{||J_\Phi||}$, et de $||J_\Phi||$ sur la convergence de notre algorithme. Nous avons utilisé l'algorithme 1 pour obtenir un nombre d'itérations indépendant du nombre de processeurs.

Le tableau 4 donne le nombre d'itérations nécessaires pour converger en fonction de la durée de l'éphéméride, de l'excentricité et pour des problèmes considérant différentes forces perturbatrices.

Dans le cas de faible excentricité, la convergence est très lente (8 itérations) quelles que soient les forces perturbatrices prises en compte. De plus la diminution de la durée d'intégration (d=0.1 jour) n'améliore pas cette convergence. Dans ce cas le mouvement du satellite est quasi-circulaire et un changement de variable pourrait améliorer la convergence.

Pour une excentricité un peu plus grande (e=0.05), la convergence de la méthode est rapide (3 itérations) lorsque la durée d'intégration est courte (d=1 jour). De plus, le nombre d'itérations nécessaires pour converger reste le même quelle que soit la force perturbatrice prise en compte. Enfin, lorsque la durée d'intégration est plus longue (d=10 jours) la méthode converge un peu moins vite (4 itérations).

5. Optimisation

Grâce à la proposition 3.1 nous pouvons, dans un premier temps, étudier le speed-up théorique de notre algorithme en faisant varier plusieurs paramètres (nombre de processeurs, longueur des sous intervalles). La vitesse de convergence dépend de la norme du Jacobien associé à la méthode numérique de Cowell. Nous étudions donc le comportement de cette norme en fonction de la position des points sur l'ellipse. A partir des résultats de ces deux études, nous donnons un algorithme qui permet d'utiliser la méthode de parallélisation à travers le temps (de l'algorithme 2) avec un découpage automatique de l'intervalle de temps. Enfin, nous présentons les résultats numériques obtenus avec ce nouvel algorithme.

5.1. *Découpage automatique*

La figure 4 représente le speed-up théorique $N/NITER$ en fonction du nombre de pas par sous-intervalles pour une trajectoire d'excentricité e=0.05. Le speed-up est plus élevé lorsque les sous-intervalles ne sont pas trop longs. Le speed-up théorique est supérieur à 2.5 sur 16 processeurs pour des sous-intervalles contenant moins de 1000 pas d'intégration, c'est-à-dire pour des durées d'environ 0.7 jour (1000*h sec, h=60 sec).

La figure 5 représente la même courbe que celle de la figure 4 mais pour des durées de sous-intervalles inférieures à 0.7 jours. On remarque que certains découpages donnent un meilleur speed-up théorique.

On s'intéresse maintenant à la norme du Jacobien $J_{Co(30,30,1,1,1,1)}$ en fonction de l'anomalie vraie v. L'anomalie vraie est l'angle que fait le satellite par rapport au demi-grand axe de la trajectoire. La figure 6 nous permet de remarquer que les oscillations sont beaucoup plus petites pour des valeurs de l'anomalie proches de 0 ou de

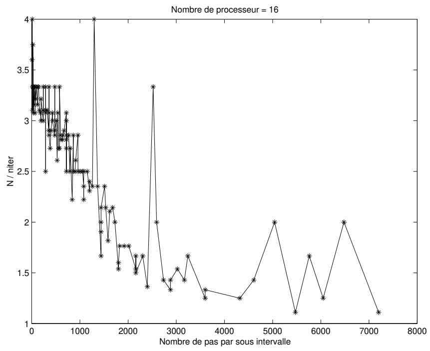

Figure 4. *Speed-up théorique sur 16 processeurs*

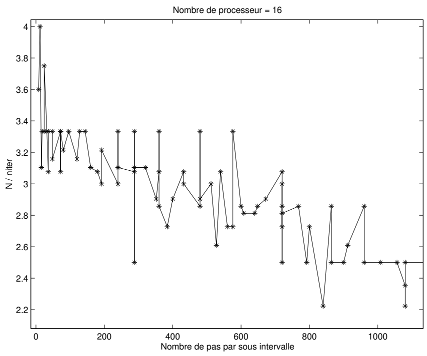

Figure 5. *Speed-up théorique : zoom de la figure 4*

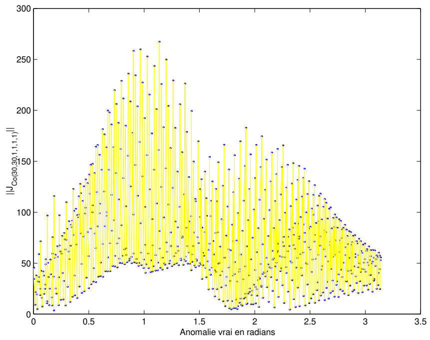

Figure 6. $||J_{Co(30,30,1,1,1,1)}||$ *en fonction de l'anomalie vraie*

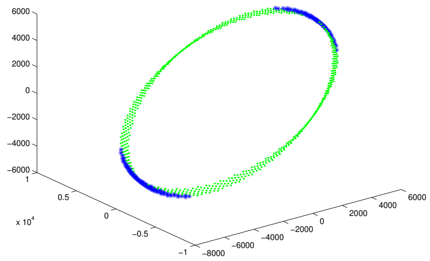

Figure 7. *Points de la trajectoire où la norme du Jacobien est petite*

PI, c'est-à-dire pour des points qui se situent près du demi-grand axes de la trajectoire. Par exemple, les points dont l'anomalie vraie est supérieure à 2.8 ou inférieure à 0.3 sont représentés en gris foncé sur la figure 7. Lors du découpage de l'intervalle de temps $T - T_0$, il y a donc intérêt à prendre les points correspondant au début des sous-intervalles proches du demi-grand axe. Ainsi la norme du Jacobien est plus petite et la convergence de la méthode est meilleure.

Etant donné un nombre de processeurs NPROC et un intervalle de temps d'une durée $T - T_0$, le nombre de sous-intervalles N et le temps de début des sous-intervalles T_i (i=1,N-1) sont choisis tels que :

1. Le nombre de sous-intervalles N est supérieur au nombre de processeurs NPROC et N est le plus petit possible pour minimiser le sur-coût de la partie séquentielle (voir proposition 3.1).

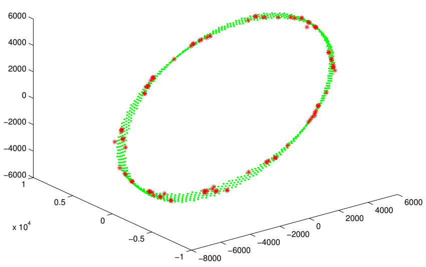

Figure 8. *Répartition des points avec l'algorithme 2 : N= 30 et d=10 jours*

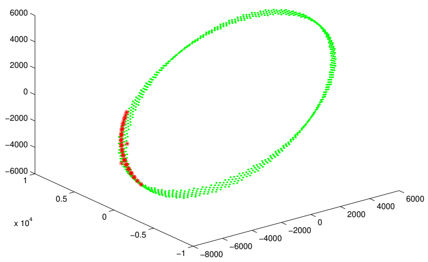

Figure 9. *Répartition des points avec l'algorithme 3 avec d=10 jours et* NPROC=*16*

2. les sous-intervalles ont une durée inférieure à 0.7 jour , c'est-à-dire

$$T_{i+1} - T_i < 0.7 \text{ jour} \quad \text{pour } i = 0, N-2.$$

3. les points y_i au temps T_i sont proches du demi-grand axe de l'ellipse.

L'algorithme suivant donne le découpage automatique ($y_0(1)$ est la première composante de y_0, v(y) est l'anomalie vraie de y et E(x) est la partie entière de x) :

ALGORITHME 3: Algorithme de découpage automatique

y_0 donné
$T_{periode} = 2\pi\sqrt{\frac{y_0(1)}{\mu}}$
T_1 tel que $v(y_1^{[0]}) \approx 0$
– Calcul du nombre de périodes par sous-intervalle–
$q = E(\frac{T-T_0}{T_{periode}} \frac{1}{\max(NPROC, \frac{T-T_0}{0.7})})$
– Calcul de N et T_2, T_3,...,T_{N-1} –
k=1
tant que $T_k < T$ **faire**
  $T_{k+1} = \min(T_k + qT_{periode}, T)$
  k=k+1
fin tant que
N=k

La figure 8 représente la répartition des points y_i aux temps T_i sur la trajectoire pour une durée d'éphéméride de 10 jours et un découpage de N=30 sous-intervalles. La figure 9 donne la nouvelle répartition des points y_i due au découpage automatique et pour 16 processeurs. La figure 9 montre que les points y_i aux temps T_i sont bien regroupés au niveau du demi-grand axe. On observe cependant un décalage pour certains points qui provient du fait que la période utilisée est celle d'une trajectoire calculée pour un *problème de Kepler* et non d'un *problème complet* (qui n'est pas périodique).

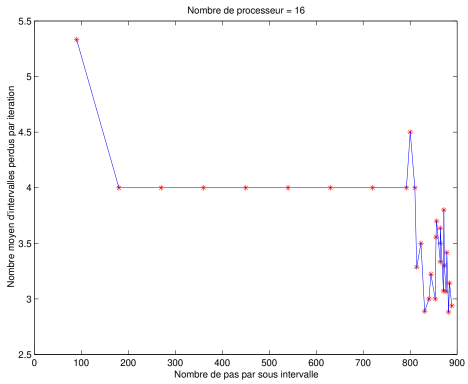

Figure 10. *Speed-up théorique sur 16 processeurs avec découpage automatique*

La figure 10 représente le nouveau speed-up théorique obtenu par découpage automatique et correspondant à celui de la figure 5. Ce nouveau speed-up théorique est maintenant plus régulier et meilleur que celui obtenu avec l'algorithme 2. Par exemple, considérons le découpage en N=30 sous-intervalles pour une trajectoire de 10 jours (exemple de la figure 8). La convergence s'effectue en NITER=11 itérations sur NPROC=16 processeurs et le speed-up théorique est donc de 30/11=2.72. Le découpage automatique sur cet exemple (voir fig 9) donne sur 16 processeurs, N=19 sous-intervalles et la convergence s'effectue en NITER=4 itérations d'où un speed-up théorique de 19/4=4.75 nettement meilleur que celui obtenu avec l'algorithme 2.

5.2. *Résultats numériques*

La figure 11 donne les speed-up apparents théoriques (en trait pointillé) et réels (en trait plein) pour des durées de 30, 40 et 45 jours.

Le speed-up réel est similaire au speed-up théorique jusqu'à 36 processeurs pour 30 jours et 48 processeurs pour 40 et 45 jours. Au-delà, l'écart se creuse entre les speed-up réels et théoriques. Cela est dû au surcoût de la partie correction de l'algorithme 2. En effet, le speed-up théorique est calculé à l'aide de la proposition 3.1 et la condition $\rho_2 N \times$ NPROC $<< 1$ n'est plus satisfaite dans ce cas car le nombre de sous-intervalles N et le nombre de processeurs NPROC augmentent. La proposition 3.1 n'est alors plus valable et le gain de la partie parallélisée est annihilé par l'augmentation de la partie séquentielle. Le meilleur speed-up apparent réel que nous obtenons est de 5 pour 60 processeurs et pour une durée de 45 jours et 40 jours.

La figure 12 donne le speed-up effectif pour les différentes durées considérées dans les exemples précédents. Les speed-up sont légèrement inférieurs aux speed-up apparents notamment dans le cas où les trajectoires ne sont pas très longues. En effet,

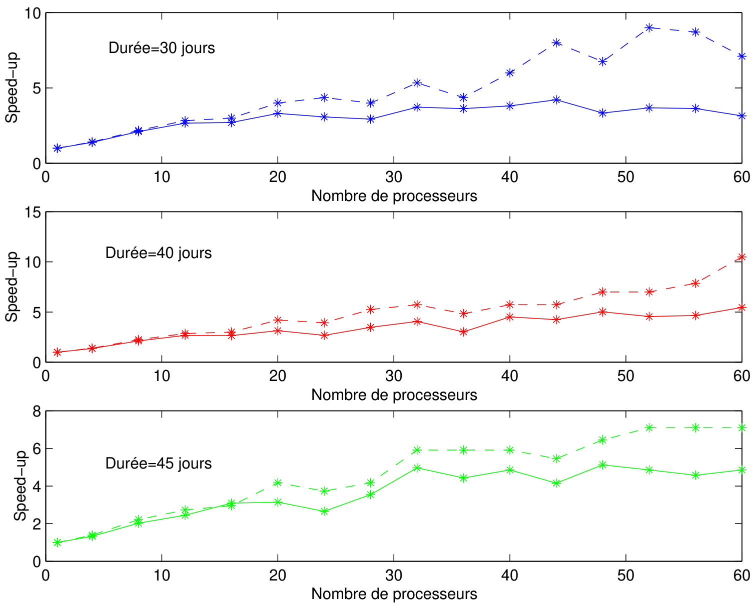

Figure 11. *Speed-up apparent de l'algorithme 2 avec découpage automatique*

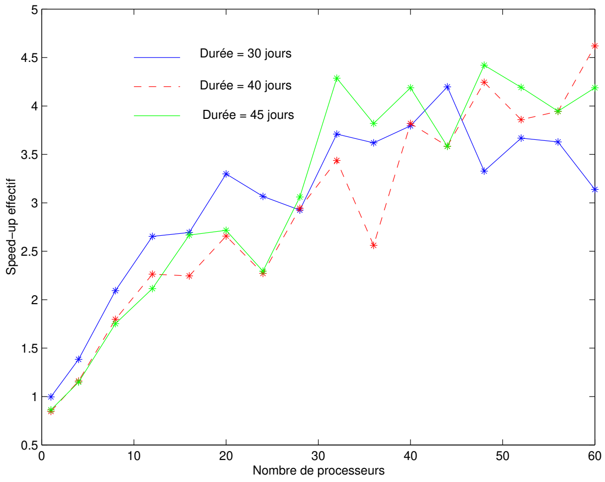

Figure 12. *Speed-up effectif de l'algorithme 2 avec découpage automatique*

lorsque la durée est relativement courte, la partie correction de notre méthode (qui n'est pas utilisée dans l'algorithme de référence) prend une part plus importante par rapport à la partie intégration.

6. Conclusion

Nous avons construit une méthode parallèle pour le calcul d'orbites ainsi qu'un modèle de temps d'exécution et une évaluation théorique du speed-up de cette méthode. Les résultats numériques et le modèle de temps montrent l'importance du découpage de l'intervalle de temps dans la mise en œuvre parallèle de notre méthode. Ainsi, le nombre d'itérations nécessaires pour atteindre la convergence dépend directement du nombre de sous-intervalles et du nombre de processeurs considérés. De plus, nous avons vu que l'augmentation du nombre de sous-intervalles ou du nombre de processeurs entraîne un surcoût de la partie séquentielle de notre algorithme.

Nous avons alors construit un algorithme de découpage automatique de l'intervalle de temps qui permet de minimiser le nombre d'itérations et donc d'optimiser les temps d'exécution en parallèle. Les résultats numériques montrent que ce découpage automatique de l'intervalle de temps a permis d'améliorer nettement les performances de notre algorithme. En effet, le speed-up effectif est passé de 1.8 sur 30 processeurs pour une durée de 40 jours à 3.5 (voir figures 3 et 12).

Même si cette accélération paraît petite par rapport au nombre de processeurs utilisés, elle est intéressante car il n'existe pas actuellement de méthode permettant d'atteindre ce niveau d'accélération. De plus, si on place le calcul de trajectoires dans le cadre de la restitution d'orbite ce facteur permet de réduire considérablement les temps de calcul. L'autre intérêt de cette méthode est son indépendance vis-à-vis du schéma numérique d'intégration. Avec un schéma de référence plus rapide, l'accélération de notre méthode par rapport à ce nouveau schéma reste la même.

7. Bibliographie

[BEA 95] BEAUGENDRE P., « Parallélisation du code Mimosa », Rapport de stage, 1995, Institut de Recherche en Informatique et Systèmes Aléatoires (I.R.I.S.A).

[BEL 89] BELLEN A., ZENNARO M., « Parallel algorithms for initial value problems for difference and differential equations. », *Journal of Computational and Applied Mathematics*, vol. 25, 1989, p. 341-350.

[BUR 95] BURRAGE K., *Parallel and sequential methods for ordinary differential equations.*, Clarendon press . Oxford, 1995.

[BUT 93a] BUTCHER J., « Diagonally-implicit multi-stage integration methods », *Applied Numerical Mathematics*, vol. 11, 1993, p. 347-363.

[BUT 93b] BUTCHER J., « General linear methods for the parallel solution of ordinary differential equations. », *World Scientific series in Applicable Analysis*, vol. 2, 1993, p. 99-111.

[CHA 93a] CHARTIER P., « Parallélisme dans la résolution numérique des problèmes de valeur initiale pour les équations différentielles ordinaires et algébriques », Thèse de doctorat, Université de Rennes 1, 1993.

[CHA 93b] CHARTIER P., PHILIPPE B., « A parallel Shooting Technique for Solving Dissipative ODE's », *Computing*, vol. 51, n° 3-4, 1993, p. 209-236.

[CHA 94] CHARTIER P., « L-stable parallel one-block methods for ordinary differential equations. », *SIAM Journal on Numerical Analysis*, vol. 31, n° 2, 1994, p. 552-571.

[CHA 97] CHARPENTIER I., GHEMIRES M., « Génération automatique de code adjoint: Statégies d'utilisation pour le logiciel odyssée, application au code météorologique meso-nh », rapport n° 3251, 1997, INRIA.

[ECK 70] ECKSTEIN M., HECHLER F., « A reliable derivation of perturbation due to any zonal and tesseral harmonics of geo potential for nearly circular satellite », rapport n° ESRO/SR/13, Juin 1970.

[FAU 96] FAURE C., « Documentation succinte d'Odyssée version 1.6, Manuel d'utilisation », rapport, 1996, INRIA.

[FAU 97a] FAURE C., « Development and Application of Odyssée an Adjoint Generator for Fortran Programs », *Identification and optimization of complex models by sensitivity analysis*, Dresdes, Juin 1997.

[FAU 97b] FAURE C., PAPEGAY Y., « Odyssée Version 1.6, The language reference manual », rapport n° 211, 1997, INRIA.

[GEA 86] GEAR C., « Parallel methods for ordinary differential equations », rapport n° R-87-1369, 1986, University of Illinois.

[GIL 91] GILBERT J., LE VEY G., MASSE J., « La différentiation automatique de fonctions représentées par des programmes », rapport n° 1557, 1991, INRIA.

[GIL 92] GILBERT J., « Automatic differentiation and iterative processes », *Optimization Methods and Software*, vol. 1, 1992, p. 13-21.

[GRI 89] GRIEWANK A., « On automatic Differentiation », M. IRI K. T., Ed., *Mathematical Programming: Recent Developments and Applications*, Kluwer Academic, 1989, p. 83-108.

[HAI 93] HAIRER E., NORSETT S., WANNER G., *Solving Ordinary Differential Equations I*, Springer-Verlag, second Revised Edition, 1993.

[HAU 95] HAUTESSERRES D., « Extrapolation de trajectoires spatiales, intégrateur de Cowell », rapport n° CT/TI/MS/MN/95-050, 1995, CNES (Centre National d'Etudes Spatiales).

[HOU 98] VAN DER HOUWEN P., MESSINA E., DE SWART J., « Parallel Stormer-Cowell Method for High-Precision Orbit Computations », *Modelling, Analysis and Simulation*, , 1998.

[ITO 97] ITO T., FUKUSHIMA T., « Parallelized extrapolation method and its application to orbital dynamics », *The Astronomical Journal*, vol. 114, n° 3, 1997.

[KEL 95] KELLEY C., *Iterative Methods for Linear and Nonlinear Equations*, vol. 16, SIAM, Frontieres in applied mathematics, 1995.

[ORT 70] ORTEGA J., RHEINBOLT W., *Iterative solution of nonlinear equations in several variables*, Academic Press, 1970.

[RAU 98] RAULT S., « Algorithmes parallèles pour le calcul d'orbites », Thèse de doctorat, Université de Rennes 1, septembre 1998.

Jocelyne Erhel *est directrice de recherche à l'INRIA, unité de recherche de Rennes, et responsable du projet Aladin. Ses activités de recherche concernent la conception et la programmation d'algorithmes numériques, qui soient à la fois rapides et précis, par exemple grâce au parallélisme.*

Stéphanie Rault *a effectué sa thèse dans le projet Aladin, sous la direction de J. Erhel, et a soutenu en 1998. Elle est maintenant fonctionnaire au Trésor Public.*

Annexe pour le service de fabrication

Article pour la revue :
Technique et science informatiques

Auteurs :
Jocelyne Erhel — Stéphanie Rault

Titre de l'article :
Algorithme parallèle pour le calcul d'orbites

Titre abrégé :
Calcul parallèle d'orbites

Traduction du titre :
Parallel algorithm for computing orbits :
parallelism through time

Date de cette version :
20th June 2000

Coordonnées des auteurs :

– téléphone : 02 99 84 73 39
– télécopie : 02 99 84 25 27
– Email : erhel@irisa.fr

Logiciel utilisé pour la préparation de cet article :
LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.3 du 26/09/1999.

Formulaire de copyright :
Joindre le formulaire de copyright signé, récupéré sur le web à l'adresse
`http://www.hermes-science.com`