Parallel Time Methods for Computing a Satellite's Trajectory

Nabil Nassif and Samah W. Karim

American University of Beirut

PinT Workshop, Jülich, May 20 2014

This work was supported by a two-year grant from the French-Lebanese project CEDRE

(January 2012 - December 2013)

Participants

Jocelyne Erhel
Noha Karam
Samah Karim
Nabil Nassif

1866 AUB American University of Beirut الجامعة الأميركية في بيروت

Outline

Introduction

- In many applications, one needs to account and predict as accurately as possible for the orbit of natural or manmade satellites
- This requires solving a system of second order differential equations to determine numerically the satellite's trajectory
- The end result is a mass of computations particularly on long time spans (a week, a month, a year, ...)

Introduction

- In 1998, S. Rault and J. Erhel [1], [2] proposed a multiple shooting method to solve this sort of equations in parallel based on previous work by Chartier and Philippe [5].
- Another parallel approach that has been devised in 2006 by [3] for reaction-diffusion on long-time, used in 2009 for a membrane problem, has been adapted to the satellite problem by J. Erhel, N. Karam and N. Nassif ,[4] is called: “Adaptive Parallel Time Integration (APTI)”

Introduction

Plan of this talk:

- Presentation of APTI and implementation the Satellite Problem
- A modified version of the Parareal algorithm implemented on the same problem
- A comparison on the satellite model between the 3 performances

Outline

Satellite Trajectory

Newton's second law

$\Downarrow$

General Equation of a Satellite's Trajectory:

$$\vec{F} = m\ddot{\vec{r}} \tag{1}$$

where:
m is the mass of the satellite
$\vec{r}$ is the position-vector of the satellite with respect to the center of the earth
$\ddot{\vec{r}}$ is the acceleration vector of the satellite
$\vec{F}$ is the force vector applied on the satellite

Satellite Trajectory

Keplerian Motion: In the proximity of the earth, the central gravitational attraction is considered the only force that is applied to the satellite:

$$\vec{F} = -GM\frac{m}{\|\vec{r}\|^3}\vec{r} = -\mu\frac{m}{\|\vec{r}\|^3}\vec{r} \tag{2}$$

where G is the universal gravitational constant, M the earth's mass and

$$\mu = GM = 3986005 \times 10^8 m^3/s^2 \tag{3}$$

Satellite Trajectory

Resulting Equation of Motion:

$$\ddot{\vec{r}} = -\frac{\mu}{\|\vec{r}\|^3}\vec{r} \tag{4}$$

- The satellite moves in a fixed plane along an ellipse having the center of the earth as one focus
- The motion of the satellite is periodic of period:

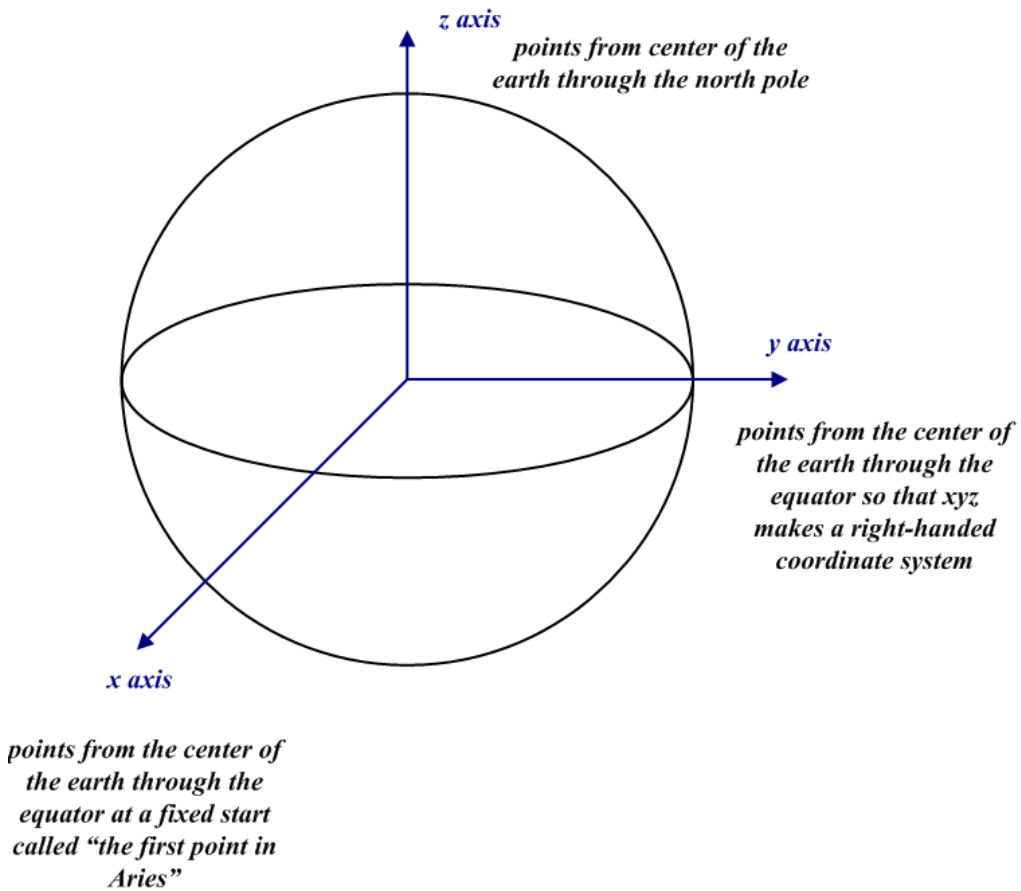
$$P = 2\pi\sqrt{\frac{a^3}{\mu}} \tag{5}$$

- Thus it is enough to solve the problem for 1 period

Reference Coordinate Systems

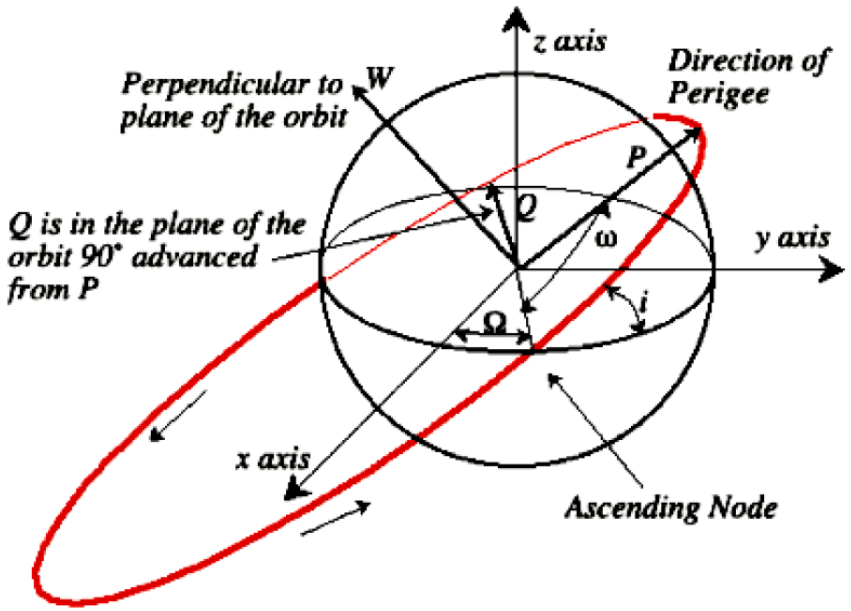
- To fully describe the trajectory of the satellite, 6 coordinates are needed
- Classical coordinate systems:
  - Earth Centered Inertial Coordinate Frame (ECI)
  - Perifocal Coordinate Frame (PQW)

Earth Centered Inertial Coordinate Frame (ECI)



- This frame is a Cartesian coordinate system
- Centered on the Earth
- The X-axis goes from the center of the Earth through the earth's equator at the "the vernal equinox"
- The Z-axis is parallel to the Earth's axis of rotation
- The Y-axis is obtained by applying the right-hand rule

Perifocal Coordinate Frame (PQW)



- Cartesian coordinate system
- The X-axis points from the center of the Earth towards the perigee
- The Y-axis points from the center of the Earth, orthogonal to the X-axis in the Keplerian plane
- The Z-axis points from the center of the earth, orthogonal to the elliptical plane

Equivalent System of First Order ODE's for the Keplerian case

Using the components (x, y, z) of the position vector and the components $(\dot{x}, \dot{y}, \dot{z})$ of the velocity vector $\vec{r}$, one can rewrite the problem as a first-order initial value problem of dimension 6, of the general form (S), in which one seeks Y which satisfies:

$$(S) \begin{cases} \frac{dY}{dt} = F(Y) = F_K(Y), \quad t > 0, \\ \\ Y(0) = Y_0 \end{cases} \tag{6}$$

where: $Y = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}$

The case of a Perturbed Keplerian motion

- In actuality, the orbit of the satellite is not Keplerian
- Many other forces perturb the central gravitational attraction
- Their order of magnitude to that of the central attraction of the earth [2]:

The sun's gravitational attraction	:	10^{-8}
The moon's gravitational attraction	:	10^{-7}
Force due to the flattening of the earth	:	10^{-3}
Central gravitational attraction of the earth	:	1

- We only consider the force caused by the flattening of the earth since it is by far the most significant

The J_2-Perturbed Gravitational Potential of The Earth

$$u(J_2) = -\frac{\mu}{r}\left\{1 + \left(\frac{r_{eq}}{r}\right)^2 J_2 P_2(\sin\varphi)\right\} \tag{7}$$

where:

- $\mu = 3986005 \times 10^8 m^3/s^2$
- r_{eq} is the equatorial radius: $r_{eq} = 6378.137km$
- J_2 is the zonal harmonic: $J_2 = -11.10^{-4}$
- $r = \left\|\vec{r}\right\|_2$ is the norm of the position vector $\vec{r}$
- $P_2(\sin\varphi)$ is the Legendre polynomial of degree 2, in the variable $\sin\varphi$, where φ is the latitude: $P_2(\sin\varphi) = \frac{3}{2}\sin^2\varphi - \frac{1}{2}$

Principal Differential Equation for the J_2-model

- The force per unit mass deriving from the potential $u(J_2)$ is equal to $-\vec{\nabla}u(J_2)$
- The force applied to the satellite is expressed as: $\vec{F} = -m\vec{\nabla}u(J_2)$
- The general equation of motion (1) reduces to a second-order differential equation:

$$\ddot{\vec{r}} = -\vec{\nabla}u(J_2) \tag{8}$$

J_2-perturbed Motion as a System of ODE's

The differential equation (8) modelizing the J_2 problem along with a set of initial conditions, yields a second-order initial value problem, in which one seeks $\vec{r}$ where:

$$\begin{cases} \ddot{\vec{r}}(t) = \vec{f}\left(\vec{r}\right), & t > 0 \\ \vec{r}(0) = \vec{r_0}, \\ \dot{\vec{r}}(0) = \dot{\vec{r}}_0, \end{cases} \tag{9}$$

with: $\vec{f}\left(\vec{r}\right) = -\vec{\nabla} U$.

Equivalent System of First Order ODE's

$\Longrightarrow$ one can rewrite the problem as a first-order initial value problem of dimension 6, of the general form (S), in which one seeks Y which satisfies:

$$(S)\begin{cases} \frac{dY}{dt} = F(Y) = F_K(Y) + F_P(Y), \quad t > 0, \\ \\ Y(0) = Y_0 \end{cases} \tag{10}$$

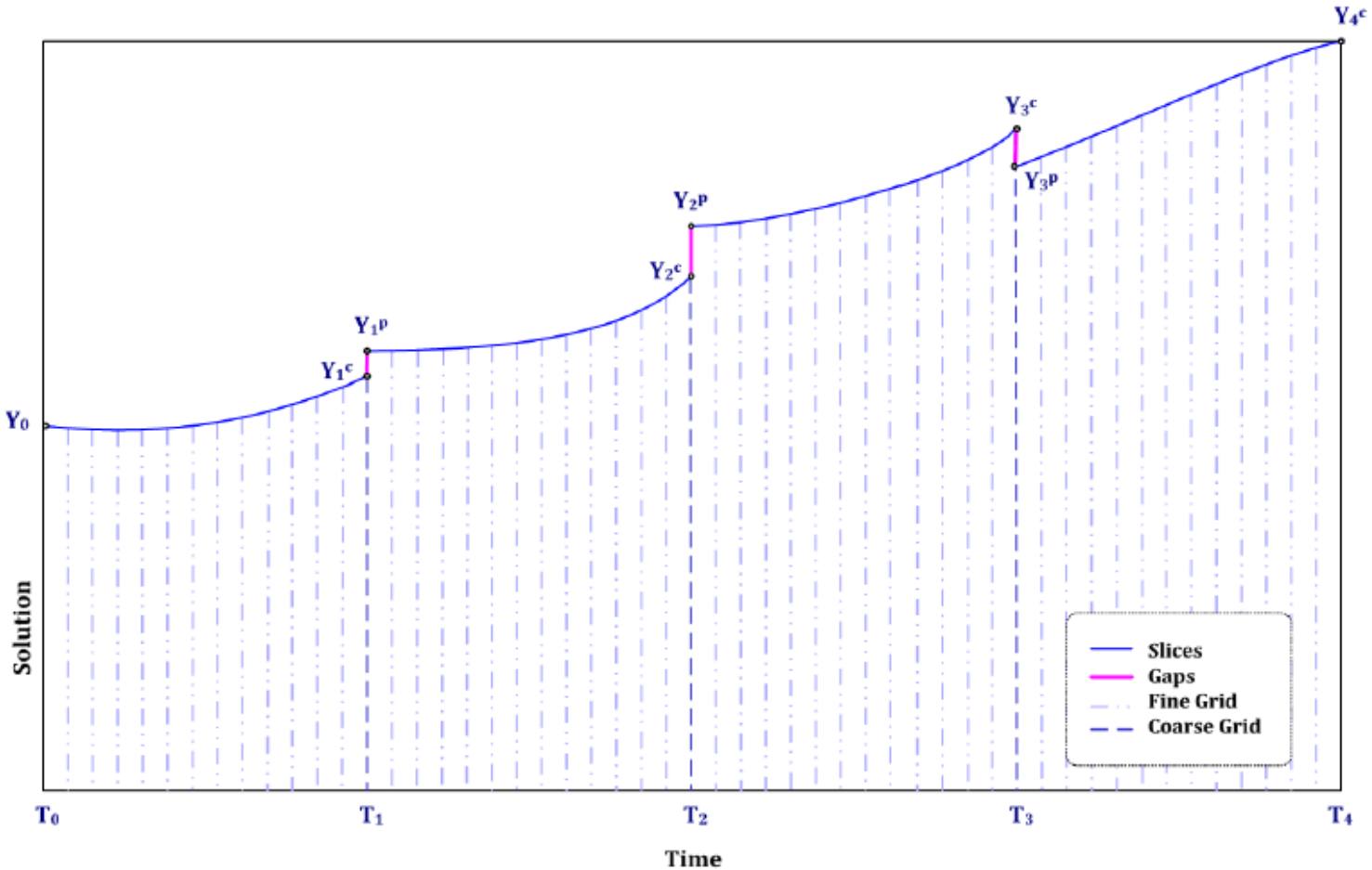
where: $Y = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}$

$F(Y)$ can be split into a sum of two terms, a Keplerian term and a perturbing term:

$$F(Y) = \begin{pmatrix} Y_4 \\ Y_5 \\ Y_6 \\ -\mu \frac{1}{R^3} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{3\mu J_2 r_{eq}^2}{2} \left(1 - 5\frac{Y_3^2}{R^2}\right) \frac{Y_1}{R^5} \\ \frac{3\mu J_2 r_{eq}^2}{2} \left(1 - 5\frac{Y_3^2}{R^2}\right) \frac{Y_2}{R^5} \\ \frac{3\mu J_2 r_{eq}^2}{2} \left(3 - 5\frac{Y_3^2}{R^2}\right) \frac{Y_3}{R^5} \end{pmatrix} \quad (11)$$

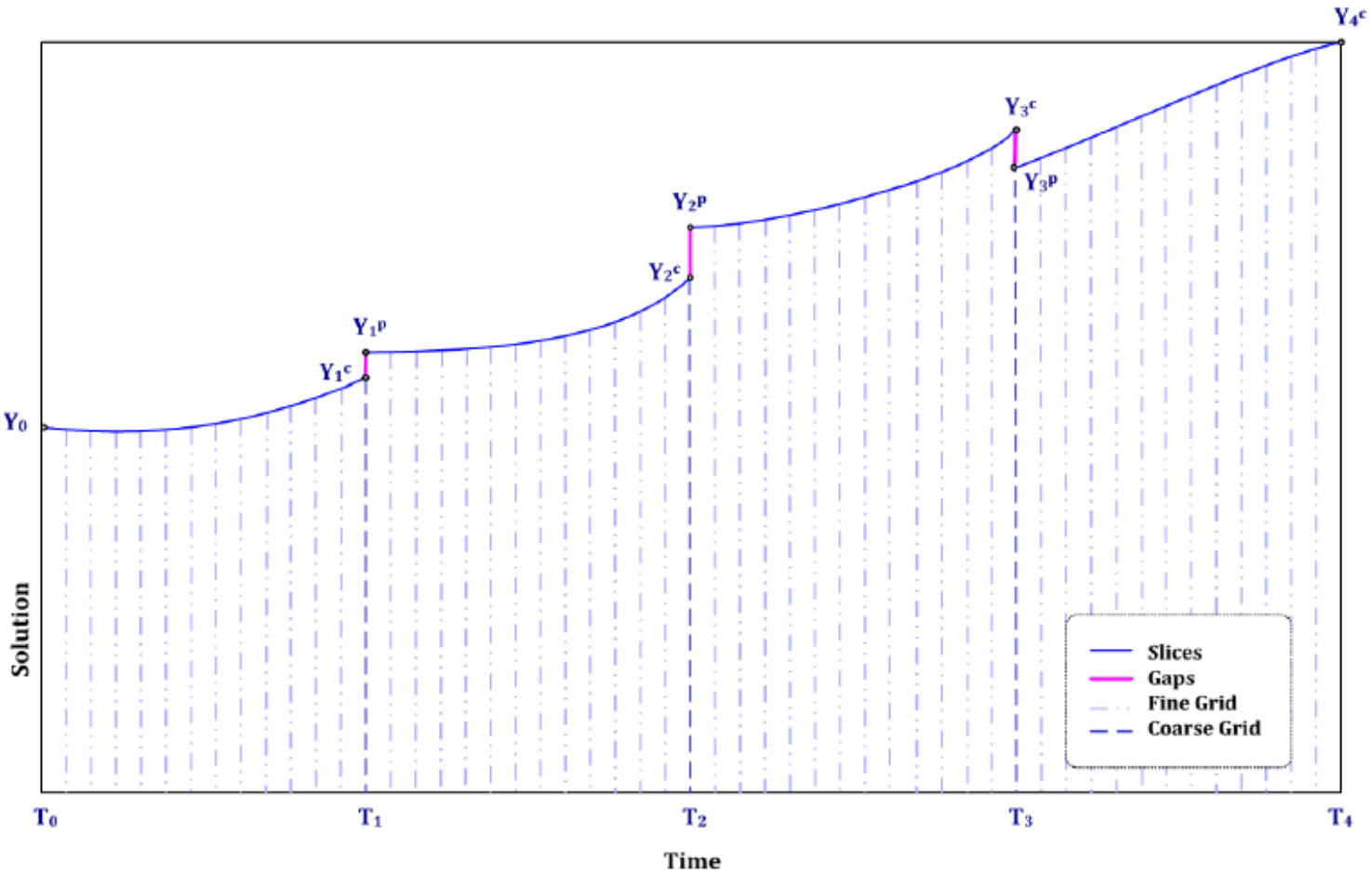
with $R = r = \sqrt{Y_1^2 + Y_2^2 + Y_3^2}$

Time Parallelism



General Steps of Multiple-Shooting Methods

1. Coarse grid discretization: The interval of integration $[T_0, T]$ is decomposed into N time slices $\Longrightarrow$ coarse time grid $[T_0 \ldots T_N]$
2. Initial Prediction: of the solution values $\{Y_n^p\}$ at every time grid point
3. Iterative process: *Do until convergence* of all slices:
   (a) *Parallel integration:* on each slice $[T_{n-1}, T_n]$, using a fine grid to solve the independent initial value sub-problems:
   $\frac{dY}{dt} = F(Y), \quad T_{n-1} < t \leq T_n, \quad Y(T_{n-1}) = Y_{n-1}^p.$
   ==> computed value Y_n^c at the end of each slice $[T_{n-1}, T_n]$.
   (b) *Check for convergence:* on every slice by evaluating the "gaps" between predicted values $\{Y_n^p\}$ and computed values $\{Y_n^c\}$:
   If $\forall n \frac{||Y_n^p - Y_n^c||}{||Y_n^p||} \leq \epsilon_{tol}$: convergence achieved and iterative process *stops*
   Else: iterations continue
   (c) *Corrective process:* update the predicted values at the onset of each slice



Outline

Adaptive **P**arallel **T**ime **I**ntegration (**APTI**)

The novelty in this approach is brought about by:

- Automatically generating a coarse grid of unequally sized time-slices
- Allowing predictions and corrections at the beginning of each of the coarse grid slices without using a sequential Finite Difference type method as in Parareal

Equivalent Initial Value Shooting Problems

Let $E : \mathbb{R}^k \to \mathbb{R}$ be the **E**nd **O**f **S**lice(EOS) function, which is problem dependent and chosen such that $E(Y(t)) = 0$ is satisfied infinitely many times.

IVP $\Rightarrow$ is equivalent to a sequence of Initial Value Shooting Problems where one seeks on the i^{th} slice the solution Y_i and the time T_i:

$$(S_i) \quad \begin{cases} \frac{dY}{dt} = F(Y), & T_{i-1} < t \leq T_i \\ Y(T_{i-1}) = Y_{i-1}, & \\ E(Y(T_i)) = 0, & \text{and } \forall t \in (T_{i-1}, T_i), E(Y(t)) \neq 0 \end{cases} \tag{12}$$

Core of the method

Main features:

- The (non-uniform) coarse grid $\{[T_{i-1}, T_i] \,|i = 1, 2, ...N\}$ is automatically generated by the End of Slice condition
- We seek relations between $Y_{i-1} = Y(t_{i-1})$ and $Y_i = Y(t_i)$, $i = 1, 2, ...N$:
  - For the Keplerian motion: $Y_i = Y_{i-1}, \forall i$
  - For the perturbed motion: $Y_i - Y_{i-1} = D_i Z_i$:
    $D_i = diag(\alpha_i)$,
    $\alpha_i(j) = Y_{i-1}(j)$ if $Y_{i-1}(j) \neq 0$ and
    $\alpha_i(j) = 1$ if $Y_{i-1}(j) = 0$

Change of variables

$\{Y,t\} \Rightarrow \{Z,s\}$:

$$\begin{cases} t = T_{i-1} + \beta_i s, \quad \beta_i > 0 \\ Y(t) = Y_{i-1} + D_i Z(s), \end{cases} \tag{13}$$

Where β_i is a time rescaling factor and $D_i = diag(\alpha_i) \in \mathbb{R}^{K \times K}$ is an invertible diagonal matrix. For example, for a 3 dimensional problem the matrix D_i is given by:

$$\begin{pmatrix} Y_{i-1}(1) & 0 & 0 \\ 0 & Y_{i-1}(2) & 0 \\ 0 & 0 & Y_{i-1}(3) \end{pmatrix} \text{ or } \begin{pmatrix} Y_{i-1}(1) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & Y_{i-1}(3) \end{pmatrix}$$

respectively, if $Y_{i-1}(j) \neq 0$, $j = 1,2,3$ and if $Y_{i-1}(j) \neq 0$, $j = 1,3$ and $Y_{i-1}(2) = 0$.

Resulting Rescaled Systems

Original IVP (S) $\Leftrightarrow$ sequence of rescaled Initial Value Shooting Problems where one seeks $Z_i(s_i)$ and s_i on the slice $[T_{i-1}, T_i]$:

$$(S_i') \quad \begin{cases} \frac{dZ}{ds} = G_i(Z), & 0 < t \leq s_i \\ Z(0) = 0, & \\ H_i(Z(s_i)) = 0, & \text{and } \forall s < s_i, H_i(Z(s)) \neq 0 \end{cases} \tag{14}$$

where $G_i(Z) = \beta_i D_i^{-1} F(Y_{i-1} + D_i Z)$. Letting: $Z(s_i) = Z_i$, then:

$$Y_i - Y_{i-1} = D_i Z_i,$$

i.e., the study of the behavior of $\{Y_i\}_i$ is equivalent to that of $\{Z_i\}_i$ or equivalently if for $\{Y_i(j) \neq 0, \forall i \forall j\}$, we define the **ratio vectors** $\{R_i\}_i$, $R_i(j) = \frac{Y_i(j)}{Y_{i-1}(j)}$ then:

$$Z_i = R_i - e, \text{ with } e = (1, 1, ..., 1)^T.$$

Main Tasks of the APTI Approach

- **1. Initialization** Sequential procedure determines n_s slices, such that: $\{Z_n\}_n \{R_n\}_n$ stabilize at the n_s^{th} slice: $n_s << N$, N: total number of slices N of the coarse mesh.
- **2. A prediction procedure**: Use extrapolation on the calculated solutions in terms of $\{Z_i|, i = 1, 2, ..n_s\}$ or $\{R_i|, i = 1, 2, ..n_s\}$ on the first n_s slices to predict sequence $\{Z_i|\, i > n_s\}$ and $\{Y_i^p|\, i > n_s\}$):

$$Y_i(j) = Y_I(j) * (R_{I+1}(j)) * (R_{I+2}(j)) ... * (R_i(j)),\ I = n_s$$

- **3. Parallel Runs and Gaps Computation**: Slices numbered i: $i > n_s$ are assigned to available processors based on a cyclic distribution. Obtain $n_{conv} > n_s$ the number of slices at which the method converges.
- **4. Correction** if $n_{conv} < N$, then update n_s by n_{conv}; repeat 2, 3, 4 else **Stop**.

APTI Algorithm I

- **Step 1: Preliminary Sequential Run**
  - Solve the first n_s slices of the rescaled shooting value problems
  - computes the successive rescaled solutions $\{Z_i\}$ which can be equivalent to ratio vectors $\{R_i\} = \{Y_i./Y_{i-1}\}$
  - keeps going until the detection of a ratio property up to a certain tolerance ϵ_R
- **Step 2: Iterative Process**
  - **2.1:** Predict
    To predict on the slices $i > n_s$, every processor:
    - fits the last few ratios into an appropriate mathematical model
    - then extrapolates in order to compute predicted solution values Y_i^p

APTI Algorithm II

- **2.2:** Parallel Run on the Remaining Slices
  - The remaining slices (slices numbered i, where $i > n_s$) are assigned to the processors available based on a cyclic distribution
  - For instance, if there are two available processors, one takes on the computations of the odd numbered slices and the other takes the even numbered slices
  - Every processor will solve the rescaled problem on the i^{th} slice starting with the predicted solution value Y_{i-1}^{p} to get a calculated value Y_i^c

APTI Algorithm III

- **2.3:** Calculate the Gaps Each processor:
  - reaches the end of every slice by calculating Y_i^c
  - calculates the relative gaps between Y_i^c and the predicted value Y_i^p , for $i > n_s$ as follows:

$$G_i = \frac{Y_i^c - Y_i^p}{max(|Y_i^c|, |Y_i^p|)} \tag{15}$$

  - checks if $||G_i||_\infty \leq \epsilon_G$ where ϵ_G is some chosen tolerance on the relative gaps
  - proceeds to solve the next slice assigned to it

APTI Algorithm IV

- **2.4:** End of iteration:
  - Whenever at an assigned slice, a processor reaches $||G_i||_\infty > \epsilon_G$ it has then reached its own n_last
  - Every processor sends n_{last} to a master processor
- **2.5:** Update n_s
  - Based on $\{n_{last}\}$, master processor calculates n_{conv}, updates n_s by n_{conv} and broadcasts the new n_s to all processors.
  - Step (2) is repeated until convergence occurs for all slices

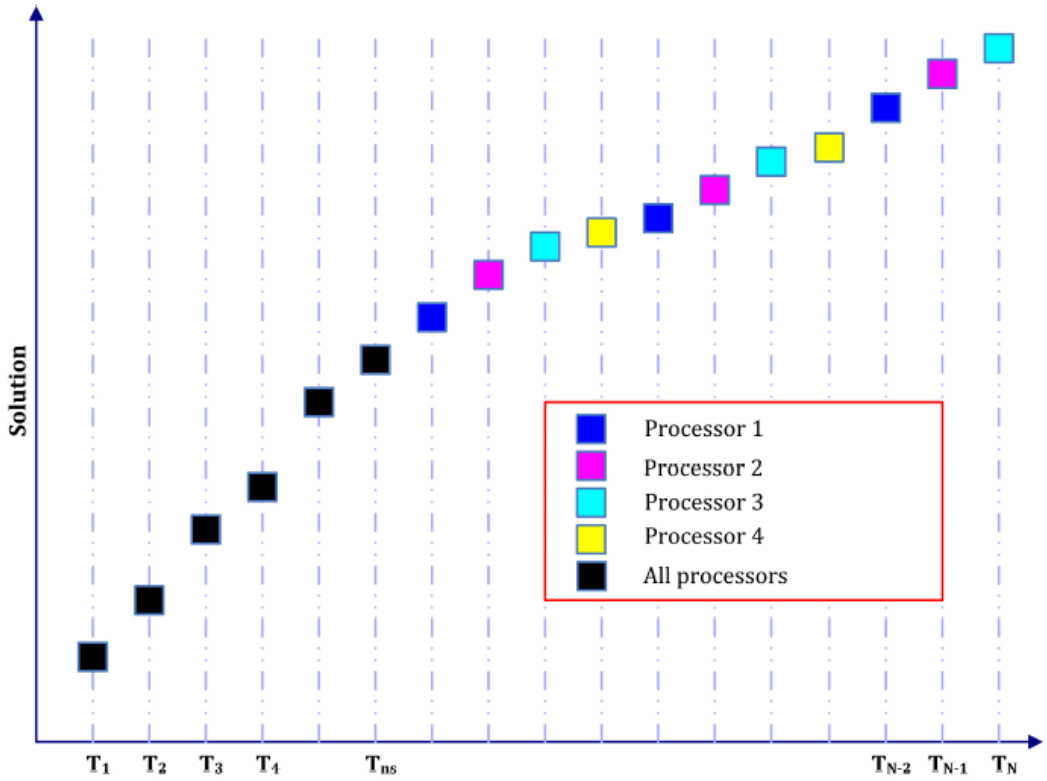


Figure : Cyclic Distribution on 4 processors

Increase Speed-Up through a Duplication Approach I

In the "classical" implementation of APTI:

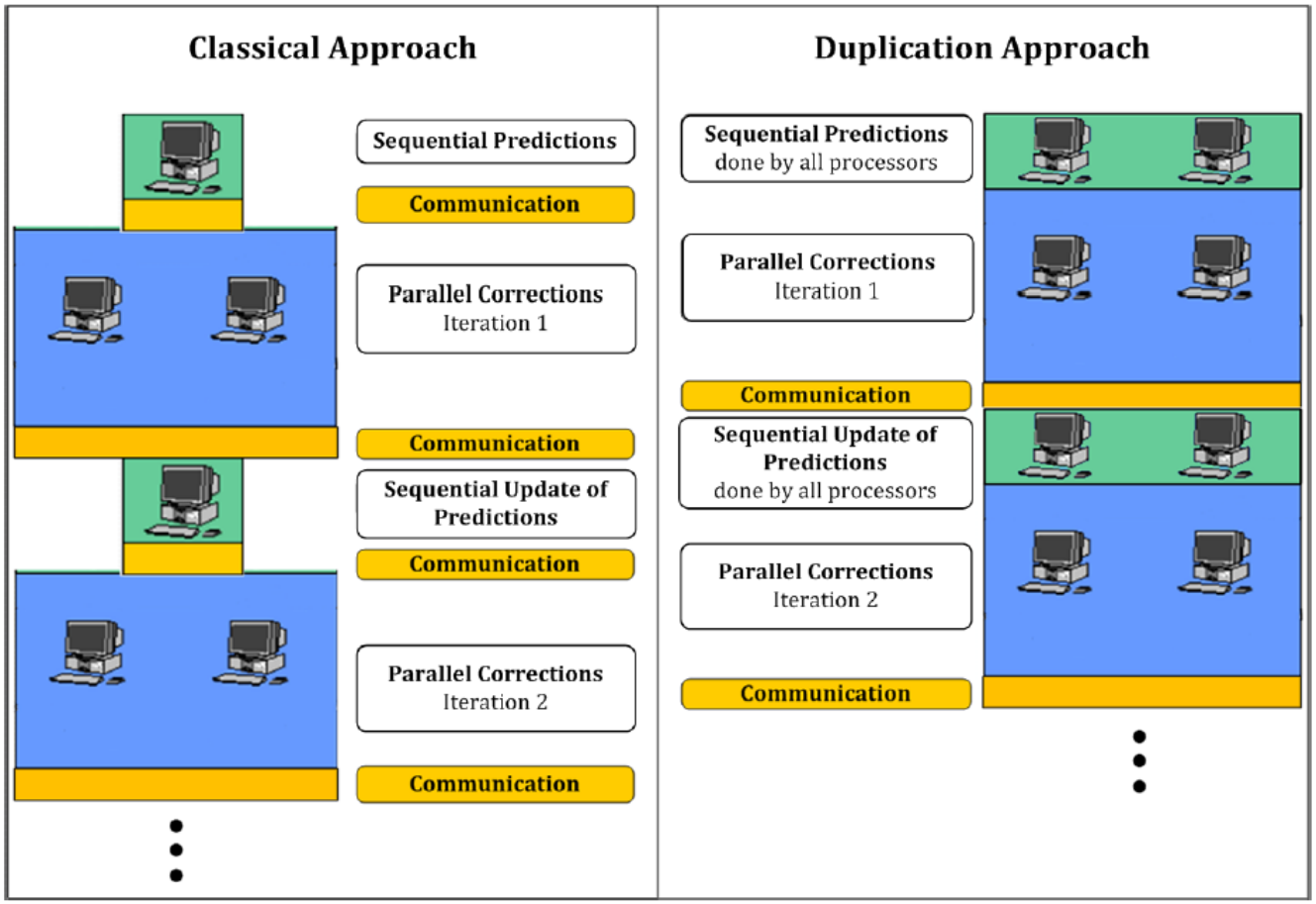
1. One processor finds the predicted values Y_i^p then sends them to all other processors
2. Then every processor solves in parallel the slices assigned to it in, to get the computed values Y_i^c
3. All the processors send the computed values to the master processor
4. All processors wait until the master computes the new predictions and sends these predictions back to them
5. This process is repeated until convergence of all slices

Increase Speed-Up through a Duplication Approach II

While in the duplication approach:

- The sequential predictions are done by all the processors at the same time
- Only 1 communication step in each iteration rather than 2

Increase Speed-Up through a Duplication Approach III



Outline

Parareal Algorithm

- Parareal was a turning point in the time parallel solution of time-dependent differential equations
- It is a predictor corrector scheme
- It is an iterative procedure $\Rightarrow$ it consists of iterating until convergence is reached
- And in every iteration, predictions and corrections are done

Parareal

The Parareal algorithm is defined using two propagation operators:

- $F(T_2, T_1, u_1)$ provides an accurate approximation of the solution $u(T_2)$ given the initial condition $u(T_1) = u_1$
- $G(T_2, T_1, u_1)$ provides a less accurate approximation of the solution $u(T_2)$ given the initial condition $u(T_1) = u_1$, for example:
  - on a coarser grid
  - or using a lower order method
  - or even an approximation using a simpler model

Parareal

- The algorithm starts with the initial condition $U_0^0 = u_0$
- It then obtains an initial approximation at iteration 0 of $U_n^0, n = 0, \ldots, N$ at times $T_0, T_1, \ldots, T_N$ using the sequential computation of $U_{n+1}^0 = G(T_{n+1}, T_n, U_n^0)$
- It then performs for $k = 0, 1, \ldots$ the correction iteration:

$$U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, T_n, U_n^k) - G(T_{n+1}, T_n, U_n^k) \tag{16}$$

Parareal Algorithm

$U_0^0 = u_0$
//Iteration 0
//Initial prediction
for $n = 0$ to $N - 1$ **do**
  $\hat{\mathcal{G}}_{n+1}^0 = G(T_{n+1}, T_n, U_n^0)$
  $U_{n+1}^0 = \hat{\mathcal{G}}_{n+1}^0$
end for
//Parareal Iterations
for $k = 0$ to K_{max} **do**
  $U_0^{k+1} = u_0$
  //Parallel Step
  for $n = 0$ to $N - 1$ **do**
    $\hat{\mathcal{F}}_{n+1}^k = F(T_{n+1}, T_n, U_n^k)$
  end for
  //Sequential Step
  for $n = 0$ to $N - 1$ **do**
    // Predict
    $\hat{\mathcal{G}}_{n+1}^{k+1} = G(T_{n+1}, T_n, U_n^{k+1})$
    //Correct
    $U_{n+1}^{k+1} = \hat{\mathcal{G}}_{n+1}^{k+1} + \hat{\mathcal{F}}_{n+1}^k - \hat{\mathcal{G}}_{n+1}^k$
  end for
  //Check for convergence
  if $|U_{n+1}^{k+1} - U_{n+1}^k| < \epsilon \; \forall \; n$ **then**
    BREAK
  end if
end for

Implementation

In our implementation:

- We divided the time interval $[0, T]$ into N coarse intervals each of size $\delta T = \frac{T}{N}$
- We chose the two propagation operators such that:
  - The G operator is given by the classical fourth order Runge Kutta method with coarse time step $\delta T = \frac{T}{N}$
  - The F operator is given by the classical fourth order Runge Kutta method with fine time step $\tau = \frac{\delta T}{Nf} = \frac{T}{N*Nf}$, where N_f is the number of fine slices in every coarse slice

Modified Parareal

- In real applications, execution time is far more important than the number of iterations made before convergence
- Parareal suffers from a performance issue in its current form
- In every iteration, the parallel fine propagation step and the sequential coarse prediction and correction steps are performed on all the slices for $n = 0$ to $N - 1$
- When performing experiments, we noticed that some slices converge before the last iteration
- Hence, performing computations on the slices that have already converged is a redundant task

Modified Parareal

- We suggest a modified version of the Parareal algorithm
- Here computations at every iteration are done on the slices that have not converged yet
- The number of operations (both parallel and sequential) at every iteration will decrease significantly
- This version will decrease the execution time of the algorithm without changing the number of iterations the algorithm takes to converge

Modified Parareal: Algorithm

$U_0^0 = u_0$
//Iteration 0, Initial prediction
for $n = 0$ to $N - 1$ **do**
  $\hat{\mathcal{G}}_{n+1}^0 = G(T_{n+1}, T_n, U_n^0)$
  $U_{n+1}^0 = \hat{\mathcal{G}}_{n+1}^0$
end for
num_converged_slices = 0
//Parareal Iterations
for $k = 0$ to K_{max} **do**
  $U_0^{k+1} = u_0$
  //Parallel Step
  for $n =$ num_converged_slices to $N - 1$ **do**
    $\hat{\mathcal{F}}_{n+1}^k = F(T_{n+1}, T_n, U_n^k)$
  end for
  //Sequential Step
  for $n =$ num_converged_slices to $N - 1$ **do**
    // Predict
    $\hat{\mathcal{G}}_{n+1}^{k+1} = G(T_{n+1}, T_n, U_n^{k+1})$
    //Correct
    $U_{n+1}^{k+1} = \hat{\mathcal{G}}_{n+1}^{k+1} + \hat{\mathcal{F}}_{n+1}^k - \hat{\mathcal{G}}_{n+1}^k$
  end for
  //Update the number of converged slices
  if $|U_{n+1}^{k+1} - U_{n+1}^k| < \epsilon$ **then**
    num_converged_slices $+= 1$
  end if
  //Check for convergence
  if num_converged_slices $= N$ **then**
    BREAK
  end if
end for

Outline

Notations

N	Number of slices
N_f	Number of fine slices used by the fine propagator of Parareal
n_s	Number of sequential slices computed by APTI before the parallel computations start
τ	Time step (in sec) of integration used by the fine RK4 propagator
T_s	Time (in sec) needed to solve the problem sequentially
n_I	Number of iterations it takes the algorithm to converge
n_p	Number of MATLAB workers used
T_{n_p}	Time (in sec) needed to solve the problem in parallel using n_p workers
S_{n_p}	Speed-up achieved while using n_p workers
E_{rel}	Relative error of the parallel solution with respect to the sequential solution

Definitions

- The parallel speed-up S_{n_p} is evaluated as the ratio of the sequential execution time to the parallel execution time while using n_p MATLAB workers:

$$S_{n_p} = \frac{T_s}{T_{n_p}} \tag{17}$$

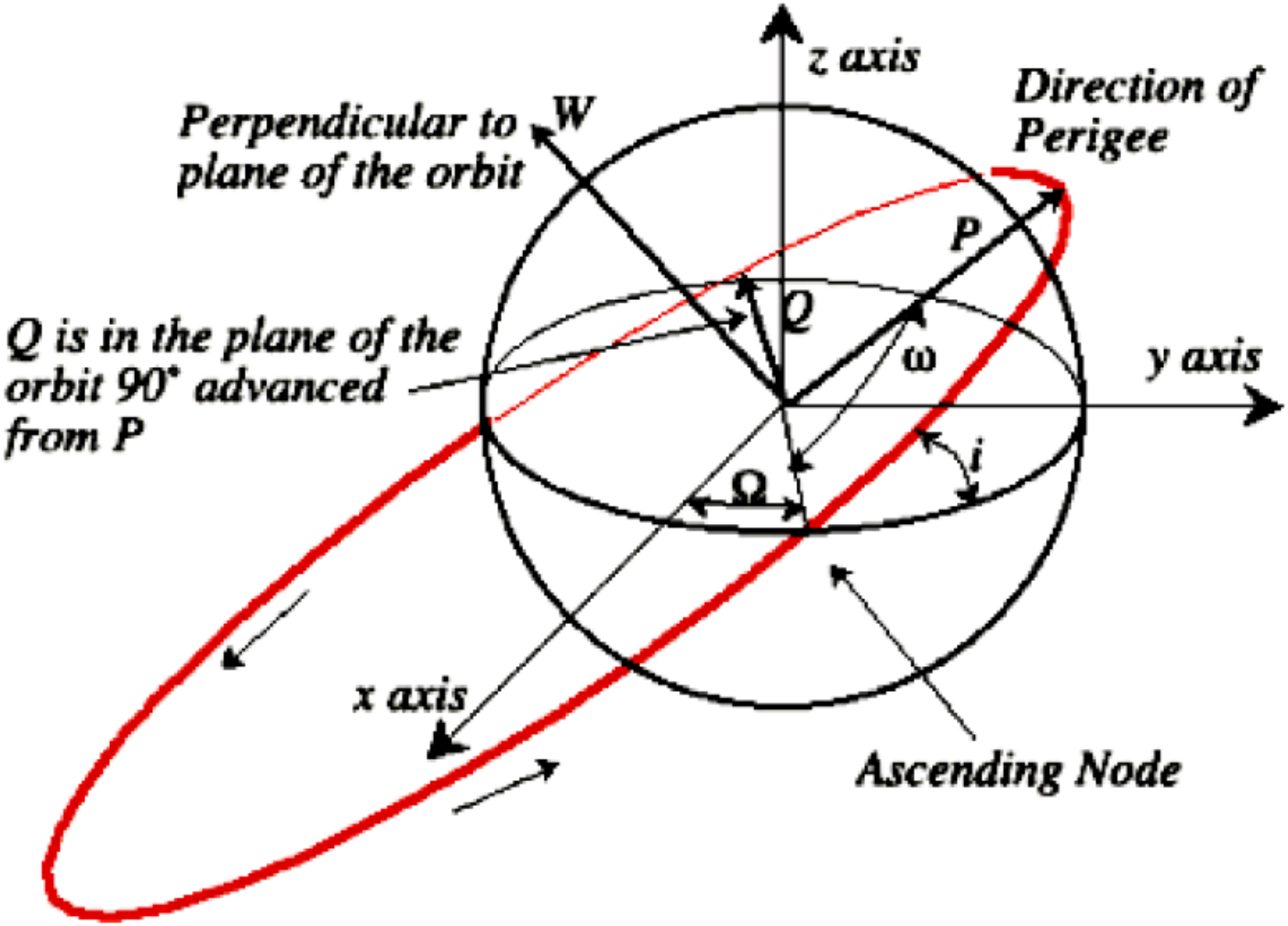
- The relative error is a measure of the difference between the solution given by the parallel solver Y^P and the solution given by the sequential solver Y^S:

$$E_{rel} = \frac{||Y^P - Y^S||}{||Y^P||} \tag{18}$$

Choice of an EOS condition for the Satellite Problem

- In a J_2 perturbed motion,the satellite will not follow a closed elliptic path
- It will follow a trajectory that is always tangential to an instantaneous ellipse: the osculating ellipse
- All of the instantaneous ellipses have the center of the earth at one focus
- $\Longrightarrow$ The satellite will certainly pass through the initial Keplerian plane $\Longrightarrow$ Introduce a new coordinate frame, the **IPQW-frame**: the **PQW-frame** corresponding to the initial conditions

IPQW frame


z axis
Direction of Perigee
Perpendicular to plane of the orbit
W
P
Q
Q is in the plane of the orbit 90° advanced from P
ω
y axis
i
Ω
x axis
Ascending Node

Choice of an EOS for APTI

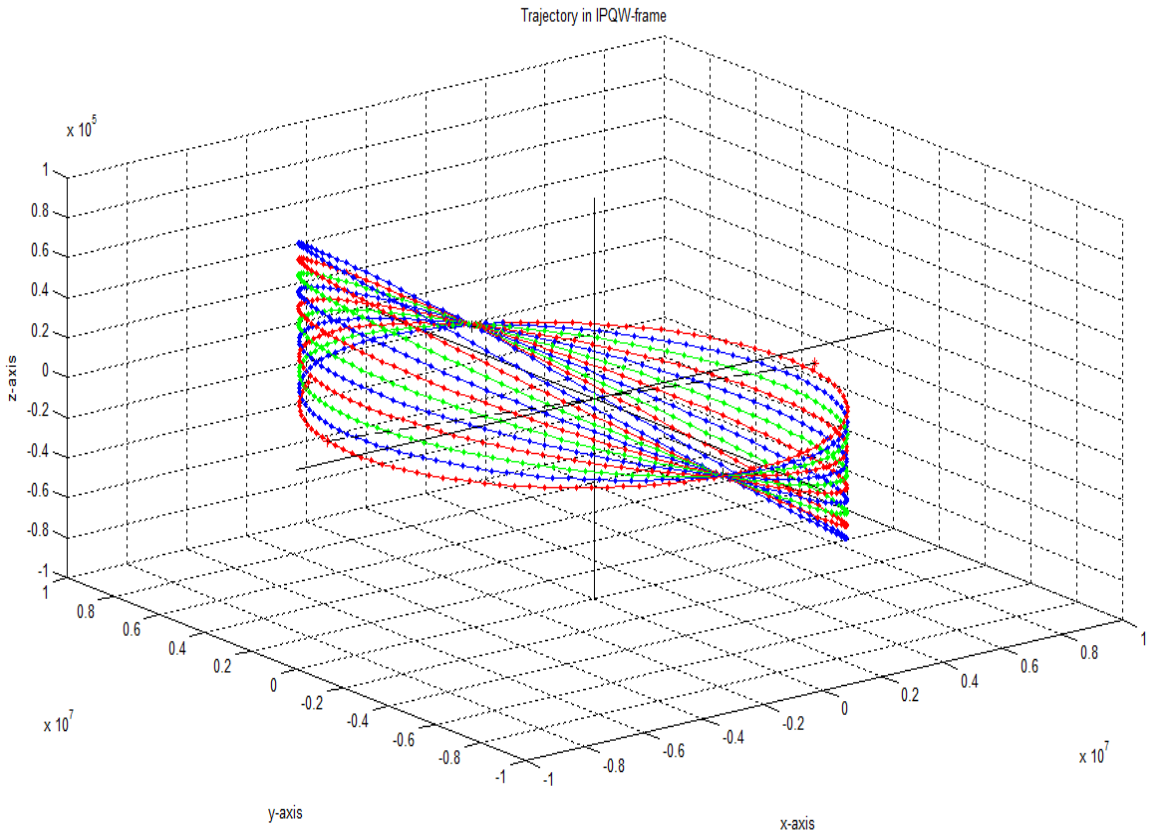
$\Longrightarrow$ EOS condition: End the i^{th} slice when the satellite crosses the xy-plane of the IPQW frame, after completing a rotation, i.e.:

$$Y_{i,3} = 0 \tag{19}$$

for the second time, after being satisfied at the previous slice $Y_{i-1,3} = 0$.

P.S. Having crossed the xy-plane at the end of the $i-1$ *slice, the satellite will cross this plane twice in order to make an almost full rotation about the center of the earth*

Orbit of a satelitte in a J_2 perturbed motion (11 rotations)



Results: Period of 1 day

- We applied APTI, Parareal and Modified Parareal algorithms to the satellite problem
- Time period of 1 day
- Set of 6 initial conditions, with:
  - e ranging from 0.0005(nearly circular) to 0.5(eccentric)
- Used 2,4 and 8 MATLAB workers

Period of 1 day

	APTI	Parareal 1	Parareal 2	Parareal 3	Parareal 4	Parareal 5	Parareal 6
N	14	24	48	144	864	1728	2880
Number of Iterations	7	24	48	13	2	2	2
S_2	1.44	0.05	0.02	0.08	0.63	0.63	1.22
S_4	1.6	0.07	0.04	0.12	1.04	1.05	1.84
S_8	1.54	0.08	0.04	0.12	1.02	0.93	1.67
Rel Error	E-15	E-12	E-12	E-8	E-10	E-11	E-11

Average performance results for a period of 1 day

	APTI	**Parareal**	**Modified Parareal**
N	14	2880	2880
n_s	4	-	-
n_l	7	2	2
E_{rel}	e-15	0	0
S_2	1.44	1.04	1.26
S_4	1.60	1.88	2.29
S_8	1.54	3.15	3.80

- All 3 algorithms achieved speed-up even when using 2 workers
- Parareal achieved speed-up higher than APTI

- APTI is at a disadvantage because for this relatively short period of time, the chosen EOS condition required choosing 14 total slices, a very small number of slices compared to 2880 slices for Parareal
- APTI needed to compute 4 slices sequentially before starting the parallel process $\Longrightarrow$ 30% of the slices were computed sequentially $\Longrightarrow$ thus the relatively bad performance of APTI
- Modified Parareal outperforms classic Parareal since it achieves on average speed-up of 1.21 versus Parareal

Period of 20 days

	APTI	**Parareal**	**Modified Parareal**
N	270	5760	5760
n_s	35	-	-
n_l	8	19	19
E_{rel}	e-05	e-05	e-05
S_2	2.60	0.12	0.20
S_4	3.47	0.20	0.35
S_8	3.26	0.31	0.52

- Only APTI achieved speed-up versus sequential
- Parareal and Modified Parareal fail to do so
- APTI outperforms Parareal since the total time of integration is relatively large $\Longrightarrow$ enforcing a reasonable total number of slices

- APTI has faster convergence rate since it required 8 iterations to converge while Parareal needed 19 iterations
- Modified Parareal outperforms classic Parareal since it takes on average half of the execution time $\Longrightarrow$ achieving almost twice the speed-up

Note: The speed-up for Parareal increased as the number of workers increased, but we could not test it with more workers due to a licence for very few MATLAB workers

Period of 43 days

	APTI	Parareal	Modified Parareal
N	579	12384	12384
n_s	46	-	-
n_l	10	38	38
E_{rel}	e-05	e-04	e-04
S_2	2.86	0.05	0.09
S_4	4.29	0.08	0.15
S_8	4.85	0.11	0.21

- Only APTI achieved speed-up versus sequential
- APTI has 4 times better convergence rate based on n_l
- Modified Parareal took almost $\frac{1}{2}$ of the execution time of Parareal $\Longrightarrow$ almost 2 times the speed-up

Period of 108 days

N	1500
n_s	46
n_l	21
E_{rel}	e-03
S_2	3.14
S_4	5.12
S_8	6.19

- We evaluated APTI on a set of 8 initial conditions
- Average performance results are very promising since:
  - APTI shows fast convergence rate: it converges after 21 iterations which is negligible compared to the number of slices 1500
  - Speed-up is significant while using 2,4 and 8 workers

Outline

References I

[1] J. Erhel and S. Rault, “Algorithme parallèle pour le calcul d’orbites: Parallélisation à travers le temps,” *TSI. Technique et science informatiques*, vol. 19, no. 5, pp. 649–673, 2000.

[2] J. Erhel and S. Rault, “Algorithme parallèle pour le calcul d’orbites,” 1998.

[3] N. R. Nassif, N. M. Karam, and Y. Soukiassian, “A new approach for solving evolution problems in time-parallel way,” in *Computational Science–ICCS 2006*, pp. 148–155, Springer, 2006.

[4] N. M. Karam, N. Nassif, and J. Erhel, “An adaptive parallel-in-time method with application to a membrane problem,” 2013.

References II

[5] P. Chartier and B. Philippe, “A parallel shooting technique for solving dissipative ode’s,” *Computing*, vol. 51, no. 3-4, pp. 209–236, 1993.

[6] H. Goldstein, *Classical Mechanics*. Reading, MA: Addison-Wesley Publishing Company, 2nd ed., 1980.

Outline

Conclusion

The results have shown that:

- Parareal outperforms APTI for relatively small integration periods
  - since the chosen EOS condition enforced choosing a relatively small number of slices
  - The small total number of slices was not significant enough to get a valuable speed-up

Conclusion

- APTI outperforms Parareal (in execution time and n_l) for relatively large integration periods
  - when the total number of slices becomes much larger
- We tested APTI's efficiency for predicting the orbit of a satellite in a J_2 perturbed motion for up to 108 days
- We also tested that modified Parareal takes less execution time than Parareal $\Longrightarrow$ doubling speed-up