Time Parallel Time Integration

Part I

Multiple Shooting Type Methods

Martin J. Gander
martin.gander@unige.ch

University of Geneva

CEMRACS, July 2016
Solving Evolution Problems in Parallel?

Heat equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f$, $u(x, y, t_0) = u_0(x, y)$
Solving Evolution Problems in Parallel?

Heat equation \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f \), \( u(x, y, t_0) = u_0(x, y) \)
Solving Evolution Problems in Parallel?

Heat equation \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f \), \( u(x, y, t_0) = u_0(x, y) \)
Solving Evolution Problems in Parallel?

Heat equation: \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f, \quad u(x, y, t_0) = u_0(x, y) \)
Solving Evolution Problems in Parallel?

Heat equation \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f, \ u(x, y, t_0) = u_0(x, y) \)
Solving Evolution Problems in Parallel?

Heat equation \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f \), \( u(x, y, t_0) = u_0(x, y) \)
Solving Evolution Problems in Parallel?

Heat equation \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f \), \( u(x, y, t_0) = u_0(x, y) \)

Simpler: \( \frac{du}{dt} = f(u) \), \( u(t_0) = u_0 \), discretized by Forward Euler

\[ u_{n+1} = u_n + \Delta t f(u_n). \]
Solving Evolution Problems in Parallel?

Heat equation \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f, \ u(x, y, t_0) = u_0(x, y) \)

Simpler: \( \frac{du}{dt} = f(u), \ u(t_0) = u_0, \) discretized by Forward Euler

\[ u_{n+1} = u_n + \Delta t f(u_n). \]

Triangular solve in the linear case \( u' = au + f(t): \)

\[
\begin{pmatrix}
1 \\
-1 - a\Delta t & 1 \\
-1 - a\Delta t & 1 & \ddots
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
u_3 \\
\vdots
\end{pmatrix}
= 
\begin{pmatrix}
\Delta t f(t_0) + (1 + a\Delta t)u_0 \\
\Delta t f(t_1) \\
\Delta t f(t_2) \\
\vdots
\end{pmatrix}
\]
Time Parallel Methods Over the Course of Time

1960

small scale

ITERATIVE
Picard Lindeloef 1893/4

DIRECT
Nievergelt 1964

1970

large scale

Miranker Liniger 1967

Shampine Watts 1969

small scale

1980

Lelarasmee Ruehli Sangiovanni–Vincentelli

Hackbusch 1984

Lubich Ostermann 1987

Gear 1988

Axelsson Verwer 1985

Jackson Norsett 1986

1990

Horten Vandewalle 1995

Chartier Philippe 1993

Worley 1991

Hairer Norsett Wanner 1992

Womble 1990

Bellen Zennaro 1989

Gander 1996

Burrage 1995

2000

Saha Stadel Tremaine 1996

Gander Halpern Nataf 1999

Sheen Sloan Thomee 1999

2010

Gander Kwok Mandal 2013

Gander Neumueller 2014

Emmett Minion 2010/2012

Gander Guettel 2013

Maday Ronquist 2008

Christlieb Macdonald Ong 2010

Sheen Sloan Thomee 1999
Time Parallel Methods Over the Course of Time

1960

1970

1980

1990

2000

2010

1960 - small scale

1970 - large scale

1980 - small scale

1990 - small scale

2000 - small scale

2010 - small scale

ITERATIVE

DIRECT

Picard Lindelof 1893/4

Nievergelt 1964

Miranker Liniger 1967

Shampine Watts 1969

Lelarasmee Ruehl Sangiovanni–Vincentelli 1982

Hackbusch 1984

Lubich Ostermann 1987

Axelson Verwer 1985

Jackson Norsett 1986

Womble 1990

Bellen Zennaro 1989

Chartier Philippe 1993

Gear 1988

Worley 1991

Hairer Norsett Wanner 1992

Horton Vandewalle 1995

Saha Stadel Tremaine 1996

Gander 1996

Burage 1995

Gander Halpern Nataf 1999

Sheen Sloan Thomee 1999

Lions Maday Turinici 2001

Emmett Minion 2010/2012

Maday Ronquist 2008

Christlieb Macdonald Ong 2010

Gander Kwok Mandal 2013

Gander Guettel 2013

Gander Neumueller 2014

50 Years of time parallel time integration (G, 2015)

“For the last 20 years, one has tried to speed up numerical computation mainly by providing ever faster computers. Today, as it appears that one is getting closer to the maximal speed of electronic components, emphasis is put on allowing operations to be performed in parallel. In the near future, much of numerical analysis will have to be recast in a more 'parallel' form.”
“Today” of Nievergelt precisely 40 years later

Model problem treated:

\[ y' = f(y), \quad y(x_0) = y_0 \]

“As an example, a method is proposed for ‘parallelizing’ the numerical integration of an ordinary differential equation, which process, by all standard methods, is entirely serial”
Nievergelt’s Method from 1964

The idea is to divide the integration interval \([a, b]\) into \(N\) equal subintervals \([x_{i-1}, x_i]\), \(x_0 = a, \ x_N = b, \ i = 1, 2, \cdots, N\), to make a rough prediction \(y_i^0\) of the solution \(y(x_i)\), to select a certain number \(M_i\) of values \(y_{ij}\), \(j = 1, 2, \cdots, M_i\) in the vicinity of \(y_i^0\), \(i = 1, 2, \cdots, N\), and then to integrate *simultaneously* with an accurate integration method \(\mathcal{M}\) all the initial value problems.

![](image)

The connection between the branches is now brought about by interpolating the end value of the unique branch in \([x_0, x_1]\).

“In addition to the two types of parallelism mentioned above, we wish to isolate a third which is analogous to what Gear has more recently called parallelism across the time. Here it is more appropriately called parallelism across the steps. In fact, the algorithm we propose is a realization of this kind of parallelism. Without discussing it in detail here, we want to point out that the idea is indeed that of multiple shooting and parallelism is introduced at the cost of redundancy of computation.”
Bellen and Zennaro’s Method from 1989

Consider the difference equation

\[ y_{n+1} = F_{n+1}(y_n), \quad y_0 \text{ known.} \]

With \( y := (y_0, y_1, \ldots, y_n, \ldots) \) this represents a fixed point problem of the form

\[ y = \Phi(y), \]

where \( \Phi(y) = (y_0, F_1(y_0), F_2(y_1), \ldots, F_n(y_{n-1}), \ldots) \).

Steffensen’s method applied to the fixed point problem gives

\[ y^{k+1} = \Phi(y^k) + \Delta \Phi(y^k)(y^{k+1} - y^k) \]

where \( \Delta \Phi \) is an approximation to the differential \( D\Phi \), and Bellen and Zennaro chose \( y_n^0 = y_0 \).
Properties of the Algorithm

Steffensen’s method for $f(x) = 0$:

$$x_{k+1} = x_k - g(x_k)^{-1}f(x_k)$$

$$g(x_k) := \frac{f(x_k + f(x_k)) - f(x_k)}{f(x_k)}.$$ 

Bellen and Zennaro’s Results:

1. each iteration gives one more exact value, so convergence is guaranteed
2. convergence is locally quadratic
3. corrections can be computed in parallel
4. numerically estimated speedups of 29-53 with for 400 steps
Philippe Chartier and Bernard Philippe 1993


“In this paper, we study different modifications of a class of parallel algorithms, initially designed by A. Bellen and M. Zennaro for difference equations and called 'across the steps' methods by their authors, for the purpose of solving initial value problems in ordinary differential equations (ODE’s) on a massively parallel computer.”

“It is indeed generally admitted that the integration of a system of ordinary differential equations in a step-by-step process is inherently sequential.”

“In diesem Artikel studieren wir verschieden Versionen einer Klasse paralleler Algorithmen, die ursprünglich von A. Bellen und M. Zennaro für Differenzengleichungen konzipiert und von ihnen 'across the steps’ Methode genannt worden ist.”
Multiple Shooting for Initial Value Problems

To solve the initial value problem

\[ u' = f(u), \quad u(0) = u^0, \quad x \in [0, 1] \]

by **multiple shooting**, one splits the time interval into subintervals \([0, \frac{1}{3}], [\frac{1}{3}, \frac{2}{3}], [\frac{2}{3}, 1]\), and then solves on each subinterval

\[
\begin{align*}
  u'_0 &= f(u_0), & u'_1 &= f(u_1), & u'_2 &= f(u_2), \\
  u_0(0) &= U_0, & u_1\left(\frac{1}{3}\right) &= U_1, & u_2\left(\frac{2}{3}\right) &= U_2,
\end{align*}
\]

together with the matching conditions

\[
U_0 = u^0, \quad U_1 = u_0\left(\frac{1}{3}, U_0\right), \quad U_2 = u_1\left(\frac{2}{3}, U_1\right)
\]

\[\iff \quad F(U) = \left( \begin{array}{c} U_0 - u^0 \\ U_1 - u_0\left(\frac{1}{3}, U_0\right) \\ U_2 - u_1\left(\frac{2}{3}, U_1\right) \end{array} \right) = 0, \quad U = (U_0, U_1, U_2)^T. \]
Using Newton’s Method

\[
F(U) = \begin{pmatrix}
U_0 - u^0 \\
U_1 - u_0(\frac{1}{3}, U_0) \\
U_2 - u_1(\frac{2}{3}, U_1)
\end{pmatrix} = 0
\]

\[
\begin{pmatrix}
U_{0}^{k+1} \\
U_{1}^{k+1} \\
U_{2}^{k+1}
\end{pmatrix} = \begin{pmatrix}
U_{0}^{k} \\
U_{1}^{k} \\
U_{2}^{k}
\end{pmatrix} - \begin{bmatrix}
1 & & \\
-\frac{\partial u_0}{\partial U_0}(\frac{1}{3}, U_0^k) & 1 & \\
-\frac{\partial u_1}{\partial U_1}(\frac{2}{3}, U_1^k) & 1 &
\end{bmatrix}^{-1} \begin{pmatrix}
U_{0}^{k} - u^0 \\
U_{1}^{k} - u_0(\frac{1}{3}, U_0^k) \\
U_{2}^{k} - u_1(\frac{2}{3}, U_1^k)
\end{pmatrix}
\]

Multiplying through by the matrix, we find the recurrence

\[
U_{0}^{k+1} = u^0,
\]

\[
U_{1}^{k+1} = u_0(\frac{1}{3}, U_0^k) + \frac{\partial u_0}{\partial U_0}(\frac{1}{3}, U_0^k)(U_{0}^{k+1} - U_{0}^{k}),
\]

\[
U_{2}^{k+1} = u_1(\frac{2}{3}, U_1^k) + \frac{\partial u_1}{\partial U_1}(\frac{2}{3}, U_1^k)(U_{1}^{k+1} - U_{1}^{k}).
\]

General case with \(N\) intervals:

\[
U_{n+1}^{k+1} = u_n(t_{n+1}, U_n^k) + \frac{\partial u_n}{\partial U_n}(t_{n+1}, U_n^k)(U_{n}^{k+1} - U_{n}^{k}).
\]
Example: first iteration

\[ U_0, U_1, U_2 \]
Example: second iteration
Example: third iteration

The diagram illustrates a function $u(t)$ with points $U_0$, $U_1$, and $U_2$ at $t = 0$, $t = 1/3$, and $t = 2/3$, respectively.
Results of Chartier and Philippe

- The algorithm converges locally quadratically
- Global convergence is proved for dissipative systems

Non-dissipative example $y' = \cos x \sin y^2, \ y(0) = 1$

Speedup for a dissipative scalar ODE and a system of 3 ODEs

“We describe how long-term solar system orbit integration could be implemented on a parallel computer. The interesting feature of our algorithm is that each processor is assigned not to a planet or a pair of planets, but to a time-interval. Thus, the 1st week, 2nd week, ..., 1000th week of an orbit are computed concurrently. The problem of matching the input to the \((n+1)\)-st processor with the output of the \(n\)-th processor can be solved efficiently by an iterative procedure. Our work is related to the so-called waveform relaxation methods...”.
The Idea of Saha, Stadel and Tremaine

Consider the system of ordinary differential equations

\[ \dot{y} = f(y), \quad y(0) = y_0, \]

or equivalently the set of quadratures

\[ y(t) = y(0) + \int_0^t f(y(s))ds. \]

Approximating the quadrature by sums gives

\[ y_n = y_0 + h \sum_{m=0}^{n-1} f\left(\frac{1}{2}(y_m + y_{m-1})\right), \quad n = 1, \ldots, N. \]

This is again a fixed point equation of the form

\[ y = F(y), \]

which can be solved by an iterative process.
Algorithm of Saha, Stadel and Tremaine

Algorithm for a Hamiltonian problem with a small perturbation

\[ \dot{p} = -H_q, \quad \dot{q} = H_p, \quad H(p, q, t) = H^0(p) + \epsilon H^1(p, q, t). \]

Denoting \( y := (p, q) \), \( f(y) := (-H_q(y), H_p(y)) \), they derive Newton’s method for the associated fixed point problem (as Chartier Philippe)

\[ Y_{n+1}^{k+1} = y^\epsilon_n(t_{n+1}, Y_n^k) + \frac{\partial y^\epsilon_n}{\partial Y_n}(t_{n+1}, Y_n^k)(Y_{n+1}^{k+1} - Y_n^k) \]

but now propose to approximate the derivative by a cheap difference for the unperturbed Hamiltonian

\[ Y_{n+1}^{k+1} = y^\epsilon_n(t_{n+1}, Y_n^k) + y^0_n(t_{n+1}, Y_{n+1}^{k+1}) - y^0_n(t_{n+1}, Y_n^k). \]

They argue that each iteration now improves the error by a factor \( \epsilon \), instead of quadratically.
Results for our solar system

Using for $H^0$ Kepler’s law, and $\epsilon H^1$ planetary perturbations

Maximum error in mean anomaly $M$ versus time, $h = 7\frac{1}{32}$ days, compared to results from the literature.
Possible Speedup

Iterations needed to converge to relative error $1e^{-15}$

Top linear scaling, and bottom logarithmic scaling
Lions, Maday, Turinici 2001

Résolution d’EDP par un schéma en temps “pararéel”.


“Elle a pour principale motivation les problèmes en temps réel, d’où la terminologie proposée de pararéel.”

\[ \dot{y} = -ay, \quad \text{on } [0, T], \quad y(0) = y_0. \]

First use Backward Euler on grid \( T_n \) with step \( \Delta T \)

\[ Y_{n+1}^1 - Y_n^1 + a\Delta T Y_{n+1}^1 = 0, \quad Y_0^1 = y_0. \]

Then compute on each interval \([T_n, T_{n+1}]\) exactly

\[ \dot{y}_n^1 = -ay_n^1, \quad y_n^1(T_n) = Y_n^1. \]

Iteration for \( k = 1, 2, \ldots : \)

1. Compute jumps \( S_n^k := y_{n-1}^k(T_n) - Y_n^k \)
2. Propagate jumps \( \delta_{n+1}^k - \delta_n^k + a\Delta T \delta_{n+1}^k = S_n^k, \quad \delta_0^k = 0 \)
3. Set \( Y_{n+1}^k := y_{n-1}^k(T_n) + \delta_n^k \) and solve in parallel

\[ \dot{y}_n^{k+1} = -ay_n^{k+1}, \quad \text{on } [T_n, T_{n+1}], \quad y_n^{k+1}(T_n) = Y_n^{k+1}. \]
“C’est alors un exercice que de montrer là:”

**Proposition:** The parareal scheme is order $k$, i.e. there exists $c_k$ s.t.

$$|Y_n^k - y(T_n)| + \max_{t \in [T_n, T_{n+1}]} |y_n^k(t) - y(t)| \leq c_k \Delta T^k.$$

**Parareal Algorithm in Modern Notation for** $u' = f(u)$

1. $G(t_2, t_1, u_1)$ is a rough approximation to $u(t_2)$ with initial condition $u(t_1) = u_1$,

2. $F(t_2, t_1, u_1)$ is a more accurate approximation of the solution $u(t_2)$ with initial condition $u(t_1) = u_1$.

Starting with a coarse approximation $U^0_n$ at the time points $t_1, t_2, \ldots, t_N$, parareal performs for $k = 0, 1, \ldots$ the correction iteration

$$U_{n+1}^{k+1} = F(t_{n+1}, t_n, U_n^k) + G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k).$$

**G, Vandevalle 2007:** Parareal is multiple shooting with the Jacobian approximated by differences on a coarser grid.
Precise Convergence Estimate for Parareal

Theorem (G, Hairer 2007)

Let $F(t_{n+1}, t_n, U^k_n)$ denote the exact solution at $t_{n+1}$ and $G(t_{n+1}, t_n, U^k_n)$ be a one step method with local truncation error bounded by $C_1 \Delta T^{p+1}$. If

$$|G(t + \Delta T, t, x) - G(t + \Delta T, t, y)| \leq (1 + C_2 \Delta T)|x - y|,$$

then

$$\max_{1 \leq n \leq N} |u(t_n) - U^k_n| \leq \frac{C_1 \Delta T^{k(p+1)}}{k!}(1 + C_2 \Delta T)^{N-k} \prod_{j=1}^{k} (N-j) \max_{1 \leq n \leq N} |u(t_n) - U^0_n|$$

$$\leq \frac{(C_1 T)^k}{k!} e^{C_2 (T-(k+1)\Delta T)} \Delta T^{pk} \max_{1 \leq n \leq N} |u(t_n) - U^0_n|.$$
Results for the Lorenz Equations


\[
\begin{align*}
\dot{x} &= -\sigma x + \sigma y \\
\dot{y} &= -xz + rx - y \\
\dot{z} &= xy - bz
\end{align*}
\]

Parameters: $\sigma = 10$, $r = 28$ and $b = \frac{8}{3}$ \(\implies\) chaotic regime.

Initial conditions: $(x, y, z)(0) = (20, 5, -5)$

Simulation time: $t \in [0, T = 10]$

Discretization: Fourth order Runge Kutta, $\Delta T = \frac{T}{180}$, $\Delta t = \frac{T}{1800}$. 
Overview

Shooting Methods

Nievergelt
Bellen Zennaro
Chartier Philippe
Saha, Stadel, Tremaine
Lions, Maday, Turinici

Experiments

Conclusions
Time Parallel Methods Part I
Shooting Methods

Martin J. Gander

Overview

Shooting Methods
Nievergelt
Bellen Zennaro
Chartier Philippe
Saha, Stadel,
Tremaine
Lions, Maday, Turinici

Experiments

Conclusions
Overview

Shooting Methods

Nievergelt
Bellen
Chartier
Saha, Stadel,
Lions, Maday, Turinici

Experiments

Conclusions
Overview

Shooting Methods
Nievergelt
Bellen Zennaro
Chartier Philippe
Saha, Stadel, Tremaine
Lions, Maday, Turinici
Experiments

Conclusions
Numerical experiment: Arenstorf orbit

Let's try the parareal algorithm with 250 processors.
Iteration 3
Iteration 4

The image contains two subplots. The left subplot is a phase portrait with the axes labeled as $x_1$ on the x-axis and another axis labeled $y$. The right subplot is a log-log plot with the axes labeled as $t$ on the x-axis and $X$ on the y-axis. The plots show some complex behavior with curves that intersect and diverge.
Iteration 5
Iteration 6
Conclusions Part I: Shooting Type Method

- The idea of parallelizing the solution of ODEs in the time direction goes back to Nievergelt (1964)

- Multiple shooting methods for initial value problems were investigated by Bellen and Zennaro (discrete case, 1989) and Chartier and Philippe (continuous case, 1993)

- Approximating the Jacobian by a simpler model in multiple shooting for initial value problems was proposed by Saha, Stadel and Tremaine (1997)

- The parareal algorithm by Lions, Maday and Turinici (2001) uses an approximation of the Jacobian on a coarse grid

Preprints are available at www.unige.ch/~gander