Parareal Methods

Scott Field

December 2, 2009
Outline

- The parareal algorithm
- Properties: Convergence, Stability, and Parameters
- Matlab example
- Conclusion: Advantages, disadvantages, and survey of usage in literature
Proposed by Lions, Maday, Turinici in 2001
- Parareal = “Parallel in time” ODE solver
- Low order accurate solution obtained via serial computation to a final time
  - e.g. forward Euler
- Corrections to low order solution done in parallel
  - e.g. on a finer temporal grid
Notation and Problem Statement

- \( u' = f(t, u) \) on coarse mesh \( t^n = n\Delta t \). \( n = 0, 1, \ldots, N \)
- IC \( u^0 = u(t^0) \)

- Three flavors of solution operator
  - Analytic: \( u(t^{n+1}) = g(t^n, u(t^n)) \)
  - Numerical, coarse with order m: \( u^{n+1} = g_{\Delta t}(t^n, u^n) \)
  - Numerical, fine: \( u^{n+1} = g_{\text{fine}}(t^n, u^n) \)

- One might choose the fine solution operator such that \( \Delta t/100 \), or use a method with an order of accuracy higher than m.

- \( \delta g^n(u) = g_{\text{fine}}(t^n, u) - g_{\Delta t}(t^n, u) \)

- Introduce a correction iteration label \( u_k^{n+1} \), where \( k = 1, 2, \ldots \). 
  k will denote the number of refinements, and \( u_1^{n+1} = g_{\Delta t}(t^n, u^n) \).
Algorithm

After choosing a temporal discretization and schemes $g_{\Delta t}$ and $g_{\text{fine}}$, the following iterative procedure comprises the parareal algorithm:

1. compute $u_1^{n+1} = g_{\Delta t}(t^n, u^n)$ in serial
2. compute the corrections $\delta g^n(u_1^n) = g_{\text{fine}}(t^n, u_1^n) - g_{\Delta t}(t^n, u_1^n)$ in parallel
3. add the prediction and correction terms as $u_2^{n+1} = g_{\Delta t}(t^n, u_2^n) + \delta g^n(u_1^n)$
4. repeat steps 2 and 3, incrementing the iteration label and using $u_{k+1}^0 = u^0$ as the initial condition.

Or more compactly

$$u_{k+1}^{n+1} = g_{\Delta t}(t^n, u_{k+1}^n) + [g_{\text{fine}}(t^n, u_k^n) - g_{\Delta t}(t^n, u_k^n)] \quad k = 1, 2, ...$$
A Time Domain Decomposition

\[
\begin{align*}
\delta g_1 &\quad \delta g_2 \\
\text{Processor 1} &\quad \text{Processor 2} \\
g_{1\{\text{fine}\}} &\quad g_{2\{\text{fine}\}} \\
0 &\quad D0 \quad 1 \quad D1 \quad 2 \\
g_{\{\text{coarse}\}} &
\end{align*}
\]
Comments on the algorithm

\[ u_{k+1}^{n+1} = g_{\Delta t}(t^n, u_{k+1}^n) + [g_{\text{fine}}(t^n, u_k^n) - g_{\Delta t}(t^n, u_k^n)] \]

▶ Optimally we will have \( N \) processors.
▶ Example: if \( k = 1 \) we recover the order \( m \) scheme. If, say \( k = 3 \), we have an order \( 3m \) scheme requiring 3 coarse computations in serial, and 2 correction calculations in parallel.
▶ as \( k \to N \) the parareal algorithm gives \( u_{k+1}^n = u_k^n \), producing a solution with accuracy of \( g_{\text{fine}} \).
▶ One would like to take large steps with \( g_{\Delta t} \). Choosing an appropriate implicit method is a popular choice.

Choices (to be discussed throughout the talk) must be made for \( k \), \( \Delta t \), methods for \( g_{\text{fine}} \) and \( g_{\Delta t} \), and the number of processors \( P \).
Example: model equation

Our problem is \( u' = \lambda u \) s.t. \( \lambda < 0 \) and \( u^0 = 1 \).

Let the fine solution operator be exact \( g_{\text{fine}} = g \) and the coarse operator be a forward Euler scheme. Define \( z = \lambda \Delta t \), thus

\[
g_{\text{fine}} = e^z \\
g_{\Delta t} = 1 + z \Rightarrow u^n_1 = (1 + z)^n = e^z + O(\Delta t) \\
\delta g(u) = [e^z - (1 + z)] u
\]

For the \( k = 2 \) iteration we have...

\[
\begin{align*}
u^0_2 &= 1 \\
u^1_2 &= (1 + z) + \delta g(u^0_1) = e^z \quad \text{EXACT!} \\
u^2_2 &= (1 + z)^2 + (1 + z)\delta g(u^0_1) + \delta g(u^1_1) = e^{2z} + O((\Delta t)^2)
\end{align*}
\]
Example: model equation

\[ u_2^{n+1} = (1 + z)^{n+1} + \sum_{j=0}^{n} (1 + z)^{n-j} \delta g(u_1^j) \]

\[ = (1 + z)^{n+1} + \sum_{j=0}^{n} (1 + z)^{n-j} [e^z - (1 + z)] (1 + z)^j \]

\[ = (1 + z)^{n+1} + (n + 1)(1 + z)^n [e^z - (1 + z)] \]

\[ = e^{(n+1)z} + O((\Delta t)^2) \]

- From our first order FE, we now have a second order parareal method! This example is representative of the general theory...
Theory: Convergence, Stability, and Parameters
Assumptions

Assume the coarse operator $g_{\Delta t}$ is order m and Lipschitz:

$$|g_{\Delta t}(t^n, u) - g_{\Delta t}(t^n, v)| \leq (1 + L\Delta t)|u - v| \quad \forall t \in (0, t^N)$$

$$|u(t^N) - u_1^N| \leq C(\Delta t)^m|u_0|$$

It is also assumed that the function $u$ remains bounded on $(0, t^N)$. 
Assume the fine solution operator is sufficiently accurate approximation to the analytic operator so that we may replace $g_{\text{fine}} \rightarrow g$

**Theorem:** The order of accuracy of the parareal method with coarse solution operator $g_{\Delta t}$ and fine operator $g$ is mk. (G. Bal, www.columbia.edu/~gb2030)

**proof:** By induction

k=1: This is just the order m coarse operator.

Assume for k: $|u(t^N) - u_k^N| \leq C(\Delta t)^{mk}|u_0|$

now show $k \Rightarrow k+1$: 
Convergence

\[ k \Rightarrow k+1: \]

\[ |u(t^N) - u^N_{k+1}| = |g(u(t^{N-1})) - g_{\Delta t}(u^N_{k+1}) - \delta g(u^N_{k})| \]
\[ = |g_{\Delta t}(u(t^{N-1})) - g_{\Delta t}(u^N_{k+1}) - \delta g(u^N_{k}) + \delta g(u(t^{N-1}))| \]
\[ \leq |g_{\Delta t}(u(t^{N-1})) - g_{\Delta t}(u^N_{k+1})| + |\delta g(u^N_{k}) - \delta g(u(t^{N-1}))| \]
\[ \leq (1 + C\Delta t)|u(t^{N-1}) - u^N_{k+1}| + C(\Delta t)^{m+1}|u^N_{k} - u(t^{N-1})| \]
\[ \leq (1 + C\Delta t)|u(t^{N-1}) - u^N_{k+1}| + C(\Delta t)^{m(k+1)+1}|u_0| \]

\[ \therefore |u(t^N) - u^N_{k+1}| \leq C(\Delta t)^{m(k+1)}|u_0| \]
Parareal methods prescribe a means for combining ODE solvers. Thus a study of the stability region requires specifying $g_{\Delta t}$ and $g_{\text{fine}}$. Consider $u' = \lambda u$

Let $g_{\text{fine}}(t^n, u^n) = \bar{g}_{\text{fine}} u^n$ and $g_{\Delta t}(t^n, u^n) = \bar{g}_{\Delta t} u^n$

As shown in *Stability of the Parareal Algorithm* by Staff et al. the parareal method becomes

$$u^n_k = \left( \sum_{j=0}^{k} \binom{n}{j} (\bar{g}_{\text{fine}} - \bar{g}_{\Delta t})^j \bar{g}_{\Delta t}^{n-j} \right) u_0 = H(\bar{g}_{\Delta t}, \bar{g}_{\text{fine}}, n, k, \lambda) u_0$$

Stability $\Rightarrow \max_{n,k}|H| \leq 1$
The authors go on to show that when $\lambda \leq 0$ and real,

$$|H| \leq \sum_{j=0}^{n} \binom{n}{j} |\bar{g}_{\text{fine}} - \bar{g}_{\Delta t}|^j |\bar{g}_{\Delta t}|^{n-j}$$

$$= (|\bar{g}_{\text{fine}} - \bar{g}_{\Delta t}| + |\bar{g}_{\Delta t}|)^n \leq 1$$

$$\Rightarrow |\bar{g}_{\text{fine}} - \bar{g}_{\Delta t}| + |\bar{g}_{\Delta t}| \leq 1$$

The conditions are:
1. $|\bar{g}_{\text{fine}}| \leq 1 \rightarrow$ this is the usual stability requirement.
2. $|\bar{g}_{\text{fine}} - 2\bar{g}_{\Delta t}| \leq 1$

Example: $\bar{g}_{\Delta t} = (1 - \lambda \Delta t)^{-1}$ and $\bar{g}_{\text{fine}} = (1 + \lambda \frac{\Delta t}{10})^{10}$
Stability

- Parareal methods work best when there is (numerical or analytic) dissipation. Consider the $k^{th}$ term in $H$
  \[
  \binom{n}{k} |\bar{g}_{\text{fine}} - \bar{g}_{\Delta t}|^k |\bar{g}_{\Delta t}|^{n-k}.
  \]
- For $k << n$, $\binom{n}{k} < n^k$ is a good bound.
- Thus a desirable property would be
  \[
  n^k |\bar{g}_{\text{fine}} - \bar{g}_{\Delta t}|^k |\bar{g}_{\Delta t}|^{n-k} \leq 1
  \]
- Terms 2 and 3 must compensate for the presence of $n^k$. We must have
  1. $|\bar{g}_{\Delta t}| \leq (1 + c\Delta t)e^{-\gamma[\min(|\lambda\Delta t|^\beta,1)]}$
  2. $|\bar{g}_{\text{fine}} - \bar{g}_{\Delta t}| \leq c\min(|\lambda\Delta t|^{m+1},1)$
- $\gamma > 0$ and $\beta \geq 1$ chosen to satisfy $e^{-\gamma n^k} \leq 1$ and $|\lambda\Delta t|^{k(m+1)} n^k e^{-n\gamma|\lambda\Delta t|^\beta} \leq 1$
Stability

Guillaume Bal: “The parareal algorithm [...] may generate instabilities.”

2 stage, 3 third order RK-Radau method (A-stable)

Fig. 3. Stability plots using Radau3 for both $G_{\Delta T}$ and $F_{\Delta T}$. The x-axis is $\text{Re}(\mu \Delta T)$ and the y-axis is $\text{Im}(\mu \Delta T)$. The dark regions represent the regions in the complex plane where (6) is satisfied. Here, $N = 1000$, and $s = 10$ (left) and $s = 1000$ (right).
Choosing the parameters wisely

In Bal *Parallelization In Time of ODEs*, the author attempts to optimize

- Speedup $S = \frac{\text{full fine resolution}}{\text{parallel algorithm}}$
- Efficiency $E = S/P$, where $P =$ processors. Best case $E = 1$

Assuming an order 1 course and fine solution operator, the main points are as follows

- $E \leq (k - 1)^{-1}$
- $S$ can be unbounded at the expense of $E$

Proposes a “mult-level” parareal method to improve $S$ and $E$ (essentially applies $k=2$ case hierarchically).
Matlab Example
Problem and Code

Consider the model problem, with a BE coarse solution operator and the exact operator use as the fine operator. The Matlab code is

```matlab
lambda = -1; TF=1; nsteps=2e3;
h=TF/nsteps; dts = 0.0:h:TF;
y=100;

solution=zeros(1,nsteps+1);
correction = zeros(1,nsteps+1);
solution(1,1)=y;
coarse = (1-h*lambda)^-1;
fine = exp(lambda*h);
corrector = fine - coarse;

tic
for k=1:10
    for ii=1:nsteps
        y = coarse*y + correction(ii);
solution(1,ii+1)=y;  %save solution
    end
    %compute corrections at each coarse step
    correction = corrector*solution;
y=solution(1);
    error(k) = solution(end)-100*exp(lambda*TF)
end
toc
```
Δt convergence for K=2

Convergence plot

\[ y = -1.9x + 0.54 \]
k convergence

\[ |u_{\text{exact}} - u_{\text{numerical}}| \text{ With } N=10 \]

\[ \log_{10}(\text{Error}) \]

K iterative steps

\[ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \]

\[ -14 \quad -12 \quad -10 \quad -8 \quad -6 \quad -4 \quad -2 \quad 0 \quad 2 \]
Conclusion
What have other people used parareal for?

- Martin Gander: Fourier transformed heat and wave equations. For the latter an exact solution operator was used in place of a fine operator.
- Guillaume Bal: Exponential function, harmonic oscillator, Brownian motion. Typical speedup and efficiency

we obtain for $M = 1$ that

$$dT = 7.21 \times 10^{-9}, \quad \Delta T = 9.67 \times 10^{-5}, \quad P = 10341, \quad S = 3987, \quad E = 0.40,$$

and for $M = 20$ that

$$dT = 7.21 \times 10^{-9}, \quad \Delta T = 4.35 \times 10^{-4}, \quad P = 114.9, \quad S = 112.3, \quad E = 0.98.$$

$M =$ number of parareal algorithms used to get to $T_{final}$
What have other people used parareal for?

- Bruce Boghosian, Paul Fischer, Frederic Hecht, Yvon Maday: Navier-Stokes equations when diffusion dominant. Speed up 10-20.
  - It turns out that even when the coarse and fine solution operators are symplectic, their sums are not necessarily. So these methods require one to somehow express the parareal algorithm as a composition of symplectic operators. It is not known what the best way to do this is.
Advantages and Disadvantages

Advantages

- Allows speed up ODE solver (compared to coarse scheme with similar accuracy).
- Given a coarse and fine scheme, straightforward to implement.

Disadvantages

- Ideally the number of processors should scale $N_{\text{coarse}}$.
- Stability region is not simply related to that of $g_{\text{coarse}}$.
- Requires one to save the solutions history, or at least coordinate the corrector step appropriately.
- Requires a good understanding of the eigenvalues and stability regions on a case by case basis
  - Staff: “No multistage scheme has been found that makes the parareal algorithm stable for all [pure imaginary] eigenvalue”
The parareal algorithm is relatively new, and is an active area of research. Applications to PDEs and ODE systems with conserved quantities are two developing areas.

Basic theory is known: order is mk, and stability can be cumbersome or (worst) unstable.

The standard algorithm allows a time-domain decomposition, whereby the high accurate corrections can be done in parallel.

Numerous extension and modifications are possible.

Speed up for ODEs appears to be a good example of usefulness: 10-1000x