A time-parallel ordinary differential equation solver with an adaptive step size: performance assessment

Eugeniy Kazakov¹, Dmitry Efremenko², Viacheslav Zemlyakov¹, Jiexing Gao¹

¹ Huawei Technologies Co., Ltd, Russian Research Institute, Moscow, Russia
² Remote Sensing Technology Institute (IMF), German Aerospace Center (DLR), Oberpfaffenhofen, Germany
Content

• Motivation

• Classical parareal ODE algorithm

• Parallel ODE solvers with adaptive step

• Numerical results

• Summary
Motivation

• There is a need for speeding up the ODE solvers on large time domains (with fine resolution).

• In a view of cheap multicore CPUs, it is very tempting to design parallel ODE solvers

• parallelization across the time direction is challenging due to a causality principle, namely, the ODE solution later in time depends on the solution earlier in time.

• Nevertheless, in 1950s, Nievergelt proposed parallel-in-time ODE solver - PARAREAL)

• The goals of the study are
  a. to assess the possible benefits of PARAREAL
  b. to find some ways to improve PARAREAL
Parallel-in-time ODE solver

- Parareal is based on coarse and fine ODE solvers:
- The coarse solver provides initial points across the domain to the fine solver.
- Fine solvers are performed in parallel, while the coarse solver is sequential.

\[ U_n^0 = \text{initial coarse integrator solution} \]
\[ U_n^k = G(U_n^k) + \left( F(U_{n-1}^{k-1}) - G(U_{n-1}^{k-1}) \right) \]

\[ U_n^k \quad \text{– current solution,} \]
\[ G(U_{n-1}^{k-1}) \quad \text{– coarse integrator (e.g. Euler method),} \]
\[ F(U_{n-1}^{k-1}) \quad \text{– fine integrator (e.g. Runge-Kutta method),} \]

\( k \) – iteration number,
\( n \) – solution point number.
Theoretical speedup factor for the Parareal algorithm:

\[ S = \left( \frac{k}{M} + (k + 1) \left( 1 - \frac{k}{2M} \right) \frac{W_G}{W_F} \right)^{-1} \]

- \( k \) is the number of iterations,
- \( M \) – number of parallel subdomains,
- \( W_G, W_F \) - the total computational fine and coarse solver times.

\( S \) is large when \( W_G/W_F \ll 1 \) (which is a common assumption in papers about Parareal)
Iterations

- If the prediction of the fine solver is far from that of the coarse solver, then the whole solution is wrong and has to be corrected (empirically).

\[
y^{k+1}_{n(j+1)} = G \left( y^{k+1}_{n(j)}, t_n, t_{n(j+1)} \right) + F \left( y^{k}_{n(j)}, t_n, t_{n(j+1)} \right) - G \left( y^{k}_{n(j)}, t_n, t_{n(j+1)} \right)
\]

- E.g., in Gparareal: the correction is performed using the “history” of previous iterations involving Gaussian processes (a little improvement as compared to the classical Parareal)

- The corrections are not fail-safe -> for \( M \) subdomains, \( M-1 \) iterations might be required -> no gain in performance
Ways to improve Parareal

• The majority of studies of Parareal are concerned with the performance of the fine solvers and the correction procedure, sticking to the Euler solver as the coarse one [1,2].

• In our examples, we did not find a case where this approach would be more efficient than a single-threaded ODE solver.

• A possible solution: improve the accuracy of the coarse solver -> minimize the number of iterations


Algorithm and implementation

Begin:
Setup input parameters.
MPI_Init:
   \textit{Begin iteration cycle:}
   \textit{In master process:}
   \textit{Coarse solver iteration cycle:}
      Run G solver,
   \textit{End coarse solver iteration cycle,}
   MPI_Send: obtained initial values another processes,
   MPI_Recv: solutions of non-zero processes,
   Error estimation.
   \textit{In other processes:}
      MPI_Recv: initial value,
      Run F solver,
      MPI_Send: solution to 0 process.
   \textit{End iteration cycle.}
   MPI_Finalize.
End

Applying adaptive step

\[ \frac{dy}{dt} = \alpha - \beta y, \]
\[ y(0) = 1, \]
\[ \alpha = 520, \beta = 2. \]  

Solution after one iteration for Eq. (1):
(a) with the fix step coarse integrator;
(b) with the adaptive step coarse integrator.
Efficiency of Parareal

Solution after one and two iteration for Eq. (1), 1e7 points.

- Unlike in theoretical papers about Parareal, in real applications, iterations may nullify performance enhancement of Parareal;
  - \( W_G/W_F \sim 1 \);
  - Formula for \( S \) does not take into account overhead due to data transfer;

<table>
<thead>
<tr>
<th>Number of CPUs</th>
<th>Speedup</th>
<th>Implementation</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>2</td>
<td>C++ MPI</td>
<td>Current work</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>MATLAB parfor + Gaussian predictor</td>
<td>[1]</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>FORTRAN OpenMP/MPI</td>
<td>[2]</td>
</tr>
</tbody>
</table>

Results for FitzHugh-Nagumo model

(a) Solutions for Eq. (2) obtained with Parareal on 32 CPUs and serial ode45 solver; (b) Absolute errors of the Parareal solution.

\[
\frac{dy_1}{dt} = c \left( y_1 - \frac{y_1^3}{3} + y_2 \right),
\]
\[
\frac{dy_2}{dt} = -\frac{1}{c} \left( y_1 - a + by_2 \right),
\]
\[a = 0.2, b = 0.2, c = 3.\]
(c) Average solution time for the serial ode45 solver and Parareal as a function of CPUs for 1e7 points; 
(d) The speedup factor provided by Parareal as a function of points for 32 CPUs.
Solution of Rossler system (stiff ODE)

\[
\begin{align*}
\frac{dy_1}{dt} &= -y_2 - y_3, \\
\frac{dy_2}{dt} &= y_1 + ay_2, \\
\frac{dy_3}{dt} &= b + y_3(y_1 - c), \\
\end{align*}
\]

(a) Solutions obtained with Parareal on 32 CPUs and serial ode45 solver; 
(b) Absolute errors of the Parareal solution.

\(a = 0.2, b = 0.2, c = 5.7\).
Conclusions

• For our implementation of Parareal, two-fold speedup is achieved on 16 CPUs;
• A classical implementation of Parareal algorithm does not lead to a significant gain in performance (given an optimized ODE solver as a reference solver);
• Parareal algorithm with an adaptive step coarse solver is more robust and requires less iterations than the classical Parareal algorithm. Gain in speed is larger than the loss in performance due to the adaptive step;
• The gain in performance is pronounced for sufficiently large number of points – a subject for fine tuning of data transfer between the processes;
• In the view of the overhead due to data transfer, the efficiency of the coarse solver is just as important as that of the fine solver.
Thank you.
Performance assessment

Schematic comparison solution time of computational load, number of points.
Problems

Solution scalable and matrix equations

Matrix Riccati equation

\[
\frac{d}{dt} y(t) = B + Ay(t) + y(t)A + y(t)By(t),
\]