Data-driven time parallelism with application to reduced order models

Lukas Brencher1, Kevin Carlberg2, Bernard Haasdonk1, Andrea Barth1
1 IANS, University of Stuttgart, 2 Sandia National Laboratories, Livermore

Motivation
Model order reduction lowers the computational costs of dynamical systems in cores \times hours, but does not generate speedup regarding the wall-time as spatial parallelism is quickly saturated. Therefore, one can apply time-parallelism to reduced order models (ROMs). A new method is presented [1, 2], which employs the time-evolution data collected during the offline (training) stage of the ROM to compute an accurate coarse propagator of the Parareal framework.

Time parallelism
In parallel-in-time methods, the time interval of the problem is divided into sub-intervals. To construct sub-problems, which can be computed in parallel, initial conditions are introduced to each sub-interval. This parallel computation yields to jumps in the final solution, which are corrected iteratively.

In the Parareal framework [4], two propagation operators are used for the computation, which differ in the used time discretization:
- \( G(T_{m-1}, T_m, X_m) \) provides a coarse approximation (serial)
- \( F(T_{m-1}, T_m, X_m) \) provides a fine approximation (parallel)

For Parareal iteration \( k \) do the correction iteration for the final solution \( X_{m+1}^{k+1} \) until convergence:

\[
X_{m+1}^{k+1} = F(T_{m-1}, T_m, X_{m}) + G(T_{m-1}, T_m, X_{m+1}^{k}) - G(T_{m-1}, T_m, X_{m})
\]

Forecasting
The forecasting method introduced in [3] aims to predict the unknown state at future time steps. Therefore, it employs the data of the ROM offline stage and the previously computed time steps to forecast the unknown state. After a time-evolution basis is computed via (thin) SVD of the ROM training snapshots, the forecast coefficients are obtained by solving the least-squares problem:

\[
z_i = \text{argmin}_{z} |Z(m, n)\mathbf{z} - \text{Z}(m, n)\mathbf{h}(x)|
\]

Here, \( n \) is the number of previous time steps used for the computation (Memory), \( Z \) is the sampling matrix, which extracts entries of a given vector and \( h \) unrolls the time according to its discretization. This procedure is applied locally on each sub-interval of the time parallelism.

Data-driven coarse propagator
In the following illustration of the data-driven coarse propagator, the forecast memory is set to \( n = 3 \):

1. Serial computation of the first initial guess \( X_{m,1}^{0} = G(T_{m-1}, T_m, X_m) \)
   - value computed on fine grid via time-integration used for \( F \)
   - computed forecast
   - value of forecast at end of sub-interval

2. Parallel computation of the fine approximations \( F(T_{m-1}, T_m, X_m) \)
   - initial value of subinterval computed by forecast
   - value computed on fine grid via \( F \)
   - value of fine approximation \( F \) at end of sub-interval

3. Serial computation of the correction step \( X_{m+1}^{k+1} = F(T_{m-1}, T_m, X_m) + G(T_{m-1}, T_m, X_{m+1}^{k}) - G(T_{m-1}, T_m, X_{m}) \)
   - value computed by fine propagator \( F \)
   - value computed by coarse propagator \( G \) during last forecast
   - value of the new forecast computed by \( G \)
   - corrected initial value of next sub-interval

Numerical experiments

<table>
<thead>
<tr>
<th>Burgers equation</th>
</tr>
</thead>
</table>
| \( \begin{align*}
\frac{\partial X(x,t)}{\partial t} + \frac{1}{2} \frac{\partial X^2(x,t)}{\partial x} &= 0.02e^{x} \\
X(x,0) &= a \quad \forall t > 0 \\
X(x;0) &= 1 \quad \forall x \in [0,100] \\
(a, b) &= [2.5, 3.5] \times [0.02, 0.03] \\
\text{time} t &= \{0.25\}
\end{align*} \) |

The subsequent illustration demonstrates the behavior of the solution during the iterations of the time-parallelism.

To exemplify the convergence of the time parallelism, the figure shown alongside shows a specific degree of freedom during the parallel iterations. Furthermore, the jumps introduced by the time parallelism are visualized.

This illustration shows the jump condition residual used for the method’s convergence over the time parallel iterations. Note that the proposed method converged after the first iteration, while the Parareal method requires 4 of 4 iterations to satisfy the convergence criterion.

The following table illustrates that the proposed method generates speedup compared to the Parareal framework in [4].

<table>
<thead>
<tr>
<th>used CPUs</th>
<th>Dirichlet Source</th>
<th>FOM ROM Parareal</th>
<th>data-driven ROM Parareal</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3.2004 0.0272</td>
<td>0.1439 0.1979</td>
<td>0.1102 0.0668</td>
</tr>
<tr>
<td>4</td>
<td>3.9235 0.0236</td>
<td>0.1329 0.2084</td>
<td>0.1104 0.0670</td>
</tr>
<tr>
<td>4</td>
<td>3.4385 0.0290</td>
<td>0.1314 0.2102</td>
<td>0.1279 0.0768</td>
</tr>
<tr>
<td>8</td>
<td>3.2004 0.0272</td>
<td>0.0744 0.1261</td>
<td>0.0572 0.0843</td>
</tr>
<tr>
<td>8</td>
<td>3.9235 0.0236</td>
<td>0.0924 0.1143</td>
<td>0.0583 0.0850</td>
</tr>
<tr>
<td>8</td>
<td>3.4385 0.0290</td>
<td>0.0755 0.1160</td>
<td>0.0579 0.0840</td>
</tr>
</tbody>
</table>

Simulation time (sec) for different online points and forecast memory \( n \).

Conclusions
With an accurate forecast of the unknown, the proposed data-driven method converges in less iterations of the time parallelism, which leads to computational speedup measured in walltime.

Extensions are in progress:
- local basis update
- global initial forecast
- dominant reduced coordinate prediction for improved stability

References