Parareal Acceleration of Matrix Multiplication

Toshiya Takami and Akira Nishida
Kyushu University, Japan

Time-evolution problems are widely solved in scientific simulations described by discretized differential equations.

Parallel technique is usually applied through domain decomposition in the space direction, where quantity on the surface of each domain must be shared with its neighbors.

On the other hand, efficient parallelism by the time-domain decomposition seems difficult because of its severe dependency on the previous state.

Contents

- Introduction: Time-domain Decomposition
  - What is Parareal?
  - Parareal-in-Time Algorithm as a Perturbation
- Application: Series by Matrix-Vector Multiplications
  - Convergence Property
  - Speed-up Ratio and Efficiency
- Discussion: Applicability to Other Linear Calculations
- Conclusion

Time-domain Parallelism

- Time-evolution is usually defined by strictly dependent relations, which is difficult to be parallelized

\[ x_{k+1} = \mathcal{F}_k(x_k) \]

- “Parareal-in-Time” is one of the time-domain methods that can be used in spite of such strict dependency.


![Domain Decomposition in Time Direction](image)
What is “Parareal”?

- “Its main goal concerns real time problems, hence the proposed terminology of ‘parareal’ algorithm.”
- “Parareal is not the first algorithm to propose the solution of evolution problems in a time-parallel fashion”

- Multiple Shooting Methods with Newton’s Iterations
- Space-Time Multigrid Method

Parareal-in-Time for Scientific Applications

- Since the first proposal by J-L. Lion, et al., various time-evolution problems have been analyzed.
- PDE with a fluid-structure coupling:
- Molecular Dynamics with Quantum Iteration:
  L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah,
- Quantum Control Problem:
- Symplectic Integrator:

Sequence Defined by Parareal-in-Time

- Instead of the sequence \( \{x_k\} \) defined by \( x_{k+1} = F_k(x_k) \), consider an approximated one \( \{x^{(r)}_k\} \) by an iterative relation,
  \[
  x^{(r+1)}_{k+1} = G_k(x^{(r+1)}_k) + F_k(x^{(r)}_k) - G_k(x^{(r)}_k)
  \]
  where \( G_k(\cdot) \) is a coarse evolution that approximates the original one \( F_k(\cdot) \), and \( r \) is the order of the approximation.
- The exact operators \( F_k(\cdot) \) can be calculated in parallel.
- In the limit \( r \to \infty \), it has been shown that the approximate sequence converges to the original one: \( \{x^{(r)}_k\} \to \{x_k\} \) which is called the Parareal-in-Time method.

Parareal-in-Time as a Perturbation (1)

- While convergence of the Parareal-in-Time method is already shown as the multiple-shooting method, a simple explanation is given here using a linear problem defined by bounded operators:
  consider a sequence of vectors \( \{x_k\} \) defined by \( x_{k+1} = F_k x_k \)
  \[
  = [G_k + (F_k - G_k)] x_k
  \]
  where we assume \( G_k \) and \( F_k - G_k \) are bounded.
- Convergence of the sequence above can be explained as a perturbation expansion and the spectral radius of operators.
**Parareal-in-Time as a Perturbation (2)**

- For linear problems, the operator $\mathcal{F}_j$ is separated into a coarse operator $\mathcal{G}_j$ and its correction $\mathcal{F}_j - \mathcal{G}_j$,
  
  \[ x_k = [\mathcal{G}_k + (\mathcal{F}_k - \mathcal{G}_k)] x_{k-1} = \prod_{j=0}^{k-1} [\mathcal{G}_j + (\mathcal{F}_j - \mathcal{G}_j)] x_0 \]

- If we introduce $\{x_k^{(r)}\}$ as r-th order approximation of $\{x_k\}$
  
  \[ x_{k+1}^{(r+1)} = [\mathcal{G}_k + (\mathcal{F}_k - \mathcal{G}_k)] x_k^{(r+1)} = \mathcal{G}_k x_{k}^{(r+1)} + (\mathcal{F}_k - \mathcal{G}_k) x_k^{(r)} \approx \mathcal{G}_k x_{k}^{(r+1)} + (\mathcal{F}_k - \mathcal{G}_k) x_k^{(r)} \]

- Thus, the Parareal-in-Time calculations for linear problems can be analyzed as a higher order perturbation.

**How to Calculate the Parareal Sequence**

- The parallel procedure of this calculation is:
  
  - $\mathcal{F}_k - \mathcal{G}_k$ are parallel in $k$, but $\mathcal{G}_k$ are sequential.
  - $k$-direction is first, $r$-direction is sequential.

**Residual Errors from Spectral Radius**

- Errors can be estimated from the rest of expanded terms:
  
  \[ \sum_{j=r+1}^{k} \varepsilon^j \sum (Terms \ with \ F^j)x_0 \]

- Error from the exact sequence is bounded by
  
  \[ \frac{|x_k^{(r)} - x_0|}{\rho(G^r) |x_0|} \leq \sum_{j=r+1}^{k} \binom{k}{j} \left( \frac{\varepsilon \rho(F)}{\rho(G)} \right)^j \]

- By the Stirling’s formula, the magnitude of r-th term is
  
  \[ \left( \frac{k}{r} \right) \left( \frac{\varepsilon \rho(F)}{\rho(G)} \right)^r \left( 2\pi(k-r)^{r} \right)^{1/2} \left( \frac{\varepsilon \rho(F)}{\rho(G)} \right)^{r} \]

**Series defined by Matrix-Vector Multiplication**

- A linear time-evolution defined by matrix multiplications, $x_k = (G + \varepsilon F)x_{k-1} = (G + \varepsilon F)^k x_0$ can be represented as an expanded sum in the order of $\varepsilon$.

- The third order approximation:
  
  \[ x_k^{(3)} = [G^k + \varepsilon \sum (Terms \ with \ F) + \varepsilon^2 \sum (Terms \ with \ two \ F's) + \varepsilon^3 \sum (Terms \ with \ three \ F's)] x_0 \]
**Example 1: Real-Symmetric Matrix**

- Consider a sequence of the real vector \( \{x_k\} \) defined by a real-symmetric matrix \( G + \epsilon F \), where \( G \) is diagonal and \( F \) is a random matrix with elements satisfying \( N=1024, \, \epsilon=0.01 \) and in this case, we can set \( \rho(F) \approx \rho(G) \approx 1 \).
- Dashed: error of \((r+1)\)-th term

\[
|\langle F_{ij} \rangle| = \frac{1}{2N} \quad |\langle F_{ij} \rangle| = \frac{1}{4N}
\]

**Example 2: Unitary Matrix**

- Unitary time-evolution is often used in quantum mechanics:
  \[
x_k = \exp \left[ \frac{\Delta t}{i\hbar} H \right] x_{k-1} \approx (I - i\epsilon H)^k x_0
\]
  where \( H \) is a Hermitian matrix satisfying \( |\langle H_{ii} \rangle| = |\langle H_{ij} \rangle| = \frac{1}{4N} \).
- \( N=1024, \, \epsilon=0.01 \) and in this case, we can set \( \rho(H) \approx 1 \).
- Dashed: error of \((r+1)\)-th term

**Implementation by MPI**

- We can implement the Parareal-in-Time iteration on parallel computers by the use of MPI_Send/Recv.
- The original number of calculations, \( K(T_g + T_f) \), is compared with the parareal one, \( KT_g + T_c(P-1) + rKT_g + T_f)/P \).

**Estimation of Speed-up Ratio**

- Speed-up Ratio will be represented in the function,
  \[
  S(r, K, P) = \frac{K(T_g + T_f)}{KT_g + T_c(P-1) + rKT_f + T_g} = \frac{P}{r + TP + \frac{P(P-1)}{K}t}
  \]
  where \( T \equiv T_g/(T_g + T_f) \) and \( t \equiv T_c/(T_g + T_f) \).
- Property of the Speed-up:
  - Max. Efficiency: small \( P \)
  - Max. Speed-up: \( K \gg P \gg 1 \)
Parallel Performance on a Cluster Machine

- The performance of the Parareal method is studied on a cluster with 768 cores (Westmere 12 cores x 64 nodes).
- When we increase the order \( r \) of the parareal iterations.
  - Large matrix: almost linear
  - Small matrix: not linear

(512 parallel execution)

Speed-up Ratio

- The dashed line represents \( S(r, K, P) = \frac{P}{r + TP + \frac{P(P - 1)}{K} t} \)
- Actual speed-up is almost half of the value by \( S(r, K, P) \).
- Efficiency is very low, which is bounded by \( 1/r \).
- \( N=8192 \): linear speed-up
- \( N=1024 \): saturated

Applicability to General Iterative Methods

- What types of problems can be accelerated by the Parareal?
- Since the usual row/column parallelism is effectively used in linear libraries, this algorithm should be applied to:
  - Long Time Series \( K \gg P \gg 1 \)
  - Large Matrices \( \text{cost}(G) \gg \text{cost}(F) \)
- How about other algorithm?

Parareal SOR (preliminary result 1)

- In order to solve a linear equation \( [D + \epsilon(L + U)]x = b \), the SOR (successive over-relaxation) method is used:
  \[ x_{k+1} = (1 - \omega)x_k + \omega(D + \epsilon U)^{-1}[b - \epsilon Lx_k] \]
- The parareal accelerated versions are
  Model 1: \( G \) is defined as an identity operator:
  \[ x^{(r+1)}_k = x^{(r+1)}_k + \omega \left\{ (D + \epsilon U)^{-1}[b - \epsilon Lx^{(r)}_k] - x^{(r)}_k \right\} \]
  Model 2: \( G \) is defined by \( \epsilon = 0 \):
  \[ x^{(r+1)}_k = (1 - \omega)x^{(r+1)}_k + \omega D^{-1}b + \omega \left\{ (D + \epsilon U)^{-1}[b - \epsilon Lx^{(r)}_k] - D^{-1}b \right\} \]
  \[ = (1 - \omega)x^{(r+1)}_k + \omega(D + \epsilon U)^{-1}[b - \epsilon Lx^{(r)}_k] \]
**Parareal SOR (preliminary result 2)**

- The parareal converges to the exact sequence only for $w<1$, while $w$ is usually chosen $1<w<2$ (graphs below: $w=0.2$).
- Applicable only for the case of slow convergence?

![Graphs showing convergence](image)

**Conclusion**

- The Parareal-in-Time method for linear evolution problems was analyzed as a perturbation expansion.
  - Remaining errors were obtained through spectral radius of the coarse and fine transformation.
  - A simple linear transformation (matrix multiplication) was accelerated by the Parareal-in-Time algorithm.
  - Convergence properties were analyzed.
  - The speed-ups were compared to the theoretical estimate.

**Further Studies**

- How to include this algorithm into parallel linear libraries.
  - interfaces, implementations, tuning, ...
- hybrid parallel implementations with the parareal-in-time and the usual row/column parallelisms
- How wide is this algorithm applied to linear calculations such as iterative solvers of linear equations: SOR, CG, etc?
- Application of this method in massively parallel computers such as $K$-computer, Kobe, Japan.

**Thank you for your attention.**