RIDC-DD: a Parallel Space–Time Algorithm

Benjamin Ong [1] Ronald Haynes [2]

[1]Michigan State University, East Lansing, MI

[2]Memorial University of New Foundland, Canada

June 18, 2013

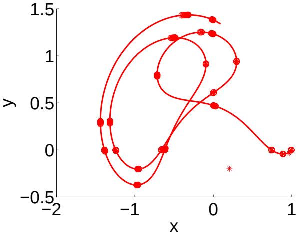
Work kindly supported by the AFOSR, BAA 9550-12-1-0455

Outline:

- Motivating Examples
- RIDC (Revisionist Integral Defect Correction)
- RIDC-DD (RIDC with Domain Decomposition)
- Scaling Studies
- Future Work

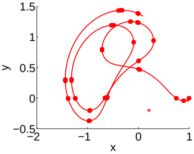
Motivating Example

- A low-order time integrator accumulates error
- E.g. Arenstof 3-body problem (earth, moon, rocket)
  - periodic orbit

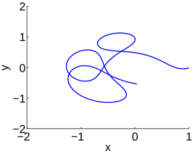


  - Forward Euler integrator, stepsize adaptivity; circles indicate rejected steps

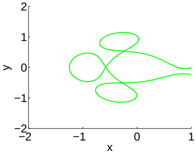
- Idea: While computing a low-order (inaccurate) solution, simultaneously correct the solution (in parallel)



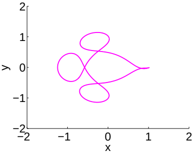
(a) prediction



(b) 1^{st} Correction



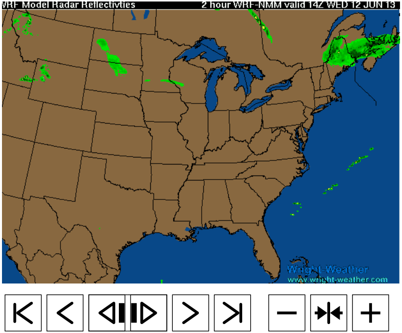
(c) 2^{nd} Correction



(d) 3^{rd} Correction

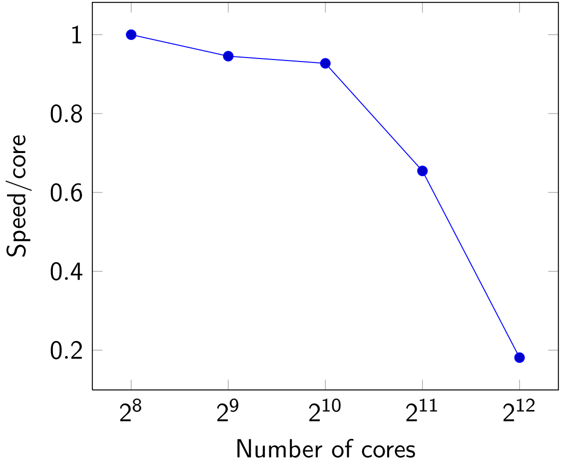
Motivating Example #2

- Many production software encounter strong scaling limits
- e.g. WRF (Weather Research and Forecasting Model) developed by NCAR



Motivating Example #2

- Strong Scaling Study



Big Picture ...

PDE of interest:

$$\begin{aligned} u_t &= \mathcal{L}(t, u), \quad (x, t) \in \Omega \times [0, T] \\ u(t, x) &= g(t, x), \quad x \in \partial\Omega \\ u(t, x) &= u0(x), \quad x \in \Omega \end{aligned}$$

- Serial time integrator, spatially parallel code, scales to N_x processors.
- Idea: Simultaneously solve PDE *and error PDEs*.
- Parallel space–time algorithm scales to $N_x \times N_t$ processors.

Comparison between RIDC and Parareal

Parareal

- Large-scale parallelism
- Iterative approach
- Sensitive to choice of coarse/fine integrator
- Symplectic variant for Hamiltonian systems

RIDC

- Small-scale parallelism
- Direct approach
- obvious choices for integrators for physical/error PDEs
- natural way to leverage heterogeneous architectures

Error PDE

- Let u be the exact (unknown) solution to $u_t = \mathcal{L}(t, u)$
- Let η be the approximate solution
- Error: $e(t, x) = u(t, x) - \eta(t, x)$.
- Residual, $r(t, x) = \eta_t(t, x) - \mathcal{L}(t, \eta)$
- Associated error PDE:

$$\begin{aligned} e_t &= u_t - \eta_t \\ &= \mathcal{L}(t, u) - (r + \mathcal{L}(t, \eta)) \\ &= \mathcal{L}(t, \eta + e) - (r + \mathcal{L}(t, \eta)) \end{aligned}$$

- For stability:

$$\left(e + \int_0^t r(\tau, x)\, d\tau\right)_t = \mathcal{L}(t, \eta + e) - \mathcal{L}(t, \eta)$$
$$e(0, x) = 0, \quad x \in \Omega, \qquad e(t, x) = 0, \quad x \in \partial\Omega$$

Error PDE

To discretize, define $q(t,x) = e + \int_0^t r(\tau,x)\,d\tau$.

$$
\begin{aligned}
q_t &= \mathcal{L}\left(t, \eta + q - \int_0^t r(\tau,x)\,d\tau\right) - \mathcal{L}(t,\eta) \\
&= f(t,q)
\end{aligned}
$$

s-stage single-step method:

$$
\frac{q^{n+1} - q^n}{\Delta t} = \sum_{i=1}^{s} b_i K_i
$$

where

$$
K_i = f\left(t^n + c_i \Delta t, q^n + \Delta t \sum_{j=1}^{s} a_{ij} K_j\right)
$$

Concrete Example

$\mathcal{L}(t,u) = u_{xx}$, first-order Backward Euler. After simplification:

$$\eta^{n+1} + e^{n+1} = \eta^n + e^n + \Delta t(\eta_{xx}^{n+1} + e_{xx}^{n+1}) - \Delta t(\eta_{xx}^{n+1}) + \int_{t^n}^{t^{n+1}} \eta_{xx}(\tau, x)\, d\tau$$

- idea of correcting an approximation can be bootstrapped (i.e. find a correction to a corrected solution)
- Adopt notation: $\eta^{[p]}$ (approximate solution), $\eta^{[p]} + e^{[p]} = \eta^{[p+1]}$ (corrected solution)

$$\eta^{[p+1],n+1} = \eta^{[p+1],n} + \Delta t \eta_{xx}^{[p+1],n+1} - \Delta t \eta_{xx}^{[p],n+1} + \int_{t^n}^{t^{n+1}} \eta_{xx}^{[p]}(\tau, x)\, d\tau$$

Backward Euler discretization of original PDE

$$\eta^{[0],n+1} - \Delta t \eta_{xx}^{[0],n+1} = \eta^{[0],n}$$

Backward Euler discretization of error PDEs

$$\eta^{[p+1],n+1} - \Delta t \eta_{xx}^{[p+1],n+1} = \eta^{[p+1],n} - \Delta t \eta_{xx}^{[p],n+1} + \int_{t^n}^{t^{n+1}} \eta_{xx}^{[p]}(\tau, x)\, d\tau$$

- lag necessary (to run in parallel)
- approximate integral sufficiently accurately using quadrature

Memory Footprint / Marching RIDC4

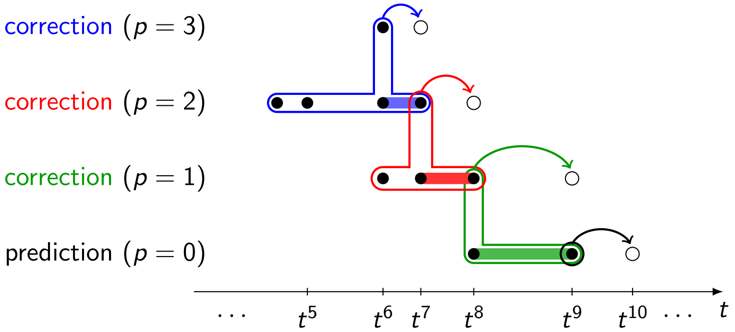


Figure: (i) stencils vary on each level, (ii) white circles are simultaneously computed, (iii) dark circles are stored memory footprint

Convergence (in time)

Theorem

Let $u(t)$, the solution to $u'(t,x) = \mathcal{L}(t,u)$, have sufficient regularity (in time), and suppose that the time domain is discretized into uniformly spaced time intervals. If an $(r_0)^{th}$ order RK method is applied to solve $u'(t) = \mathcal{L}(t,u)$, and $(r_1, r_2, \ldots, r_m)^{th}$ order RK methods are used to solve the corresponding m error PDEs, then the cumulative order (in time) of the RIDC method is $\sum_{j=0}^{m} r_j$.

Convergence (in time)

Interestingly:

Theorem

Let $u(t)$, the solution to $u'(t,x) = \mathcal{L}(t,u)$, have sufficient regularity (in time), and suppose that the time domain is discretized into non-uniformly spaced time intervals. If an $(r_0)^{th}$ order RK method is applied to solve $u'(t) = \mathcal{L}(t,u)$, and $(r_1, r_2, \ldots, r_m)^{th}$ order RK methods are used to solve the corresponding m error PDEs, then the cumulative order (in time) of the RIDC method is at least $(r_0 + m)$.

- This has to do with the "smoothness" of the time discretization

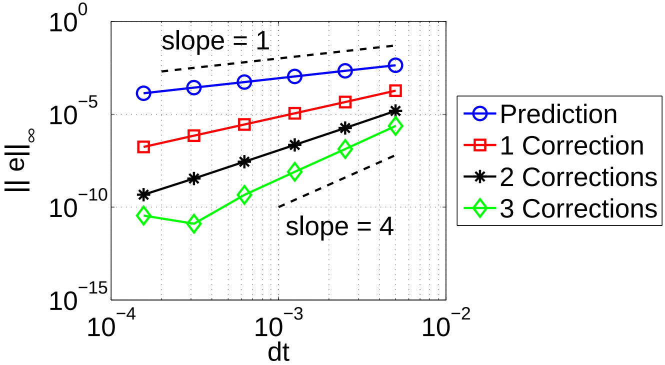


Figure: Uniform spacing, Euler integrators

Comparison with RK Integrators

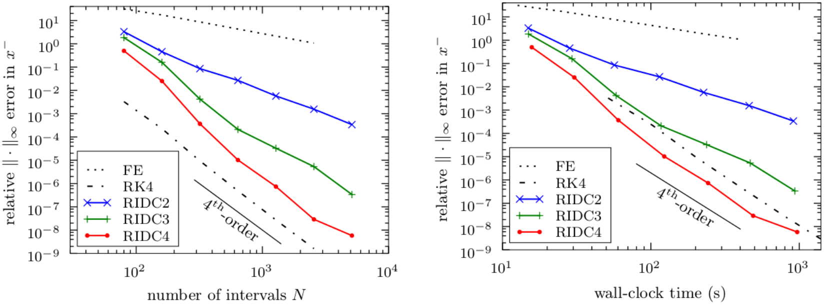


Figure: n-body simulation, 400 particles

Comparison with RK Integrators

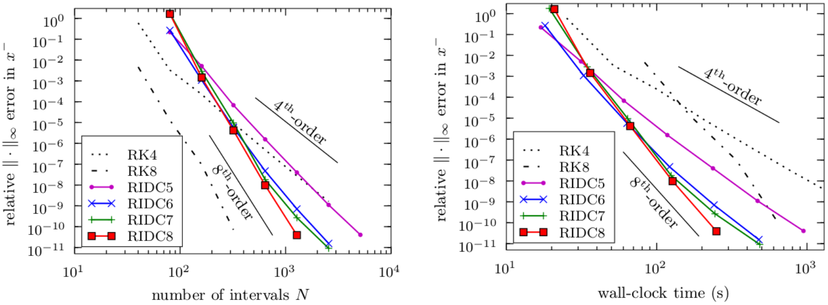


Figure: n-body simulation, 400 particles

Comparison with IMEX Integrators

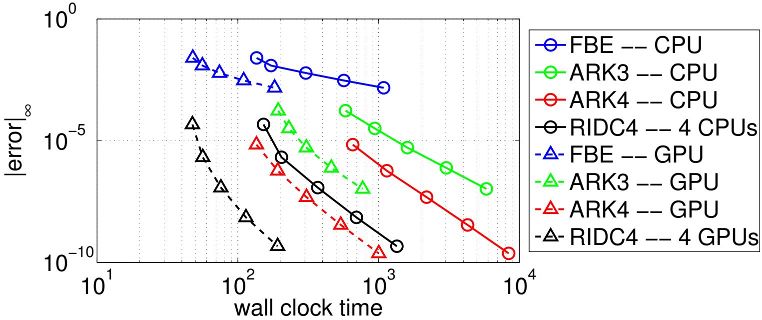


Figure: Burgers' equation using multiple cores and multiple GPUs

Recall ...

Backward Euler discretization of original PDE

$$\eta^{[0],n+1} - \Delta t \eta_{xx}^{[0],n+1} = \eta^{[0],n}$$
$$\mathcal{H}[\eta^{[0],n+1}] = f_0(t,x)$$

Backward Euler discretization of error PDEs

$$\eta^{[p+1],n+1} - \Delta t \eta_{xx}^{[p+1],n+1} = \eta^{[p+1],n} - \Delta t \eta_{xx}^{[p],n+1} + \int_{t^n}^{t^{n+1}} \eta_{xx}^{[p]}(\tau, x)\, d\tau$$
$$\mathcal{H}[\eta^{[p],n+1}] = f_p(t,x)$$

- f_p are known, assuming appropriate lag,
- $\mathcal{H} = (1 - \Delta t\, \partial_{xx})$

Combining Time and Spatial Parallelism

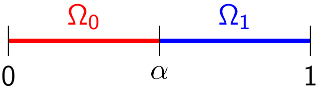
Suffices to consider

$$(1-\alpha\Delta)u = f, \quad x \in [0,1]$$
$$u(0) = 0, \quad u(1) = 0$$

Domain Decomposition:

- split domain into several sub-domains.
- solve coupled system (via transmission conditions) of PDEs

$$
\begin{aligned}
(1-\alpha\Delta)u = f, &\quad x \in [0,1] \\
u(0) = 0, &\quad u(1) = 0
\end{aligned}
$$



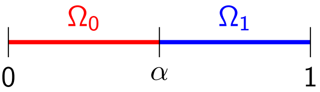
$$
\begin{aligned}
&x \in \Omega_0 \\
(1-\alpha\Delta)(u_0) &= f, \\
u_0(0) &= 0 \\
\mathcal{T}[u_0]\,|_\alpha &= \mathcal{T}[u_1]\,|_\alpha
\end{aligned}
$$

$$
\begin{aligned}
&x \in \Omega_1 \\
(1-\alpha\Delta)(u_1) &= f \\
\mathcal{T}[u_1]\,|_\alpha &= \mathcal{T}[u_0]\,|_\alpha \\
u_1(1) &= 0
\end{aligned}
$$

Optimized transmission condition

Suffices to consider

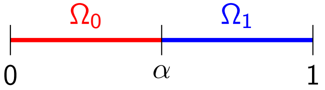
$$(1-\alpha\Delta)u = f, \quad x \in [0,1]$$
$$u(0)=0, \quad u(1)=0$$



$$x \in \Omega_0$$
$$(1-\alpha\Delta)(u_0) = f,$$
$$u_0(0) = 0$$
$$\left.\frac{\partial u_0}{\partial x}\right|_{\alpha} + \gamma u_0(\alpha) = \left.\frac{\partial u_1}{\partial x}\right|_{\alpha} + \gamma u_1(\alpha)$$

$$x \in \Omega_1$$
$$(1-\alpha\Delta)(u_1) = f$$
$$\left.\frac{\partial u_1}{\partial x}\right|_{\alpha} - \gamma u_1(\alpha) = \left.\frac{\partial u_0}{\partial x}\right|_{\alpha} - \gamma u_0(\alpha)$$
$$u_1(1) = 0$$

Schwarz Iteration

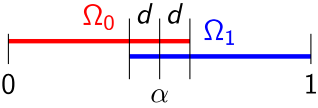


For $k = 1, 2, \ldots,$ solve

$$
\begin{aligned}
&x \in \Omega_0 \\
&(1 - \alpha\Delta)({u_0}^k) = f, \\
&{u_0}^k(0) = 0 \\
&\left.\frac{\partial {u_0}^k}{\partial x}\right|_\alpha + \gamma {u_0}^k(\alpha) = \\
&\left.\frac{\partial {u_1}^{k-1}}{\partial x}\right|_\alpha + \gamma {u_1}^{k-1}(\alpha)
\end{aligned}
$$

$$
\begin{aligned}
&x \in \Omega_1 \\
&(1 - \alpha\Delta)({u_1}^k) = f \\
&\left.\frac{\partial {u_1}^k}{\partial x}\right|_\alpha - \gamma {u_1}^k(\alpha) = \\
&\left.\frac{\partial {u_0}^{k-1}}{\partial x}\right|_\alpha - \gamma {u_0}^{k-1}(\alpha) \\
&{u_1}^k(1) = 0
\end{aligned}
$$

- overlap required
- rate of convergence increases as overlap increases



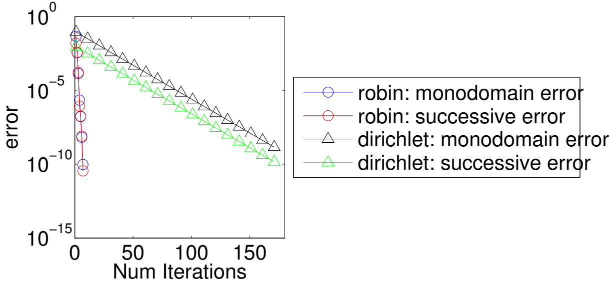
For $k = 1, 2, \ldots,$ solve

$$\begin{aligned} x &\in \Omega_0 \\ (1 - \alpha\Delta)(u_0^k) &= f, \\ u_0^k(0) &= 0 \\ u_0^k(\alpha + d) &= u_1^{k-1}(\alpha + d) \end{aligned}$$

$$\begin{aligned} x &\in \Omega_0 \\ (1 - \alpha\Delta)(u_1^k) &= f, \\ u_1^k(1) &= 0 \\ u_1^k(\alpha - d) &= u_0^{k-1}(\alpha - d) \end{aligned}$$

Schwarz Convergence

- linear heat equation in $\mathbb{R}^1$
- sixteen subdomains



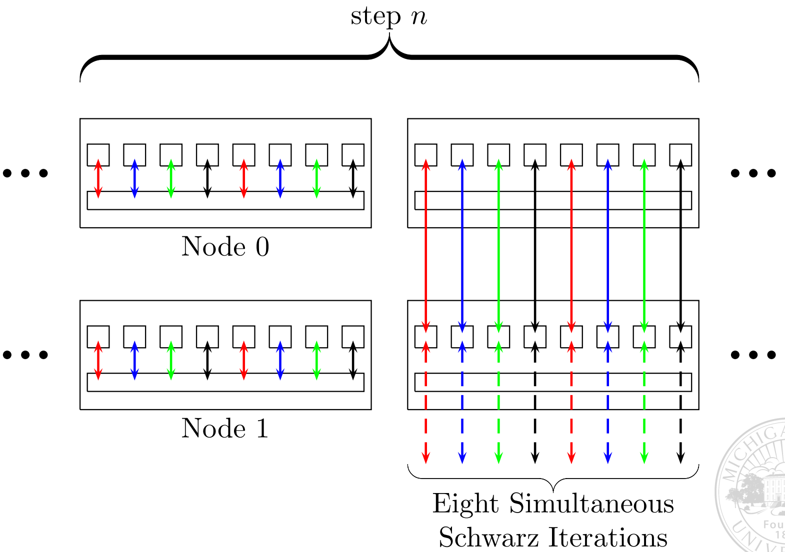
Linear heat equation in $\mathbb{R}^2$

- 10×10 non-overlapping domains
- Optimized transmission conditions
- Eighth order RIDC method
- Transmission coefficients found recursively.

Implementation:

- Nodes: 2-socket, 8-core Sandy Bridge processors, FDR IB
- each socket handles one subdomain
  - communicates with other sockets via MPI (for dd)
  - communicates via OpenMP threads within socket (for RIDC)
- time scaling: vary number of threads on each socket

Hybrid MPI–OpenMP Implementation



Efficiency

- RIDC design: minimize memory footprint
- startup complicated before marching in a pipe
- parallel efficiency = # RIDC steps / N_t
- where # RIDC steps = startup steps + march-in-pipe

Efficiency

- RIDC design: minimize memory footprint
- startup complicated before marching in a pipe
- parallel efficiency = # RIDC steps / N_t
- where # RIDC steps = startup steps + march-in-pipe

prediction	0	1	2	3	4	$\cdots$	N_t
	t_0	t_1	t_2	t_3	t_4	$\cdots$	t_{N_t}

Figure: RIDC1

Efficiency

- RIDC design: minimize memory footprint
- startup complicated before marching in a pipe
- parallel efficiency = # RIDC steps / N_t
- where # RIDC steps = startup steps + march-in-pipe

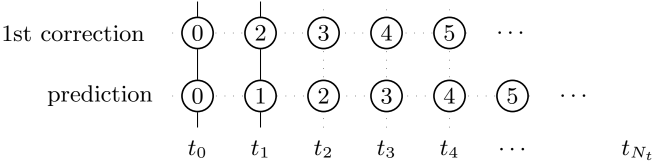


Figure: RIDC2

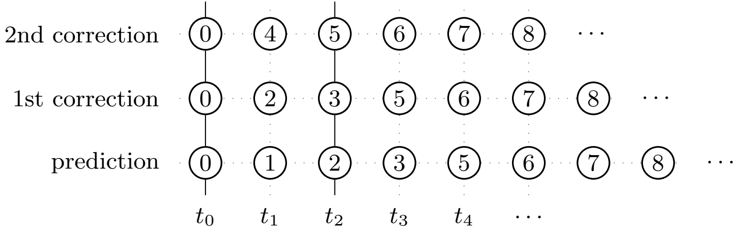


Figure: RIDC3

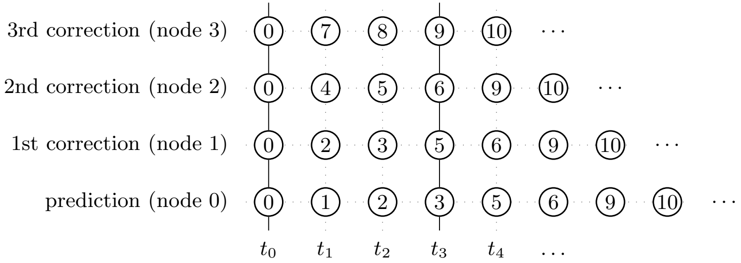


Figure: RIDC4

Theoretical Efficiency

- Assume no overhead communication cost
- N_t time steps total
- $p \cdot N_t$ processors available for RIDCp

scheme	efficiency
RIDC1	$\frac{N}{N}$
RIDC2	$\frac{N}{N+1}$
RIDC3	$\frac{N}{N+3}$
RIDC4	$\frac{N}{N+6}$
RIDC5	$\frac{N}{N+10}$

Theoretical Efficiency

- Assume no overhead communication cost
- N_t time steps total
- $p \cdot N_t$ processors available for RIDCp

scheme	efficiency
RIDC1	$\frac{N}{N}$
RIDC2	$\frac{N}{N+1}$
RIDC3	$\frac{N}{N+3}$
RIDC4	$\frac{N}{N+6}$
RIDC5	$\frac{N}{N+10}$

Theoretical Efficiency:

$$\frac{N}{N + \frac{p(p-1)}{2}}$$

$N_x \times N_y \times N_t$	# cores	walltime	speedup	efficiency
$10 \times 10 \times 1$	100	12 minutes	$1.0\times$	1.0
$10 \times 10 \times 2$	200	6 minutes	$2.0\times$	0.9795
$10 \times 10 \times 4$	400	3.2 minutes	$3.7\times$	0.9299
$10 \times 10 \times 8$	800	1.8 minutes	$6.5\times$	0.8143

Future Work / Collaborators

Future Work:

- Publishing software
- Fault-Tolerant RIDC-DD
- Multi-Level RIDC-DD
- contributing RIDC to a community software
  (e.g. WRF, a regional weather forecasting model)

Collaborators:

- DD: Ronald Haynes (Memorial University of Newfoundland)
- RIDC: Colin Macdonald (Oxford), Ray Spiteri (U. Sask)
- Software: Kyle Ladd (MSU)
- Fault Tolerance: Andrew Christlieb (MSU), Scott High (MSU)
- WRF: Yang Zhang + team (NC State)