Adaptive Error Control for RIDC

Raymond J. Spiteri

Numerical Simulation Laboratory,
Department of Computer Science
University of Saskatchewan

SIAM Conference on Parallel Processing 2012
Savannah, GA
17 February, 2012

Orientation

Saskatchewan

Orientation

Saskatchewan
Easy to draw, hard to spell.

Acknowledgements

- My partners in crime

C. Macdonald

B. Ong

- Support from

NSERC CRSNG / People. Discovery. Innovation.

MITACS

IBM

Outline

Deferred Correction

We are interested in solving the initial-value problem

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in (t_0, t_f), \ \mathbf{y}(t) \in \mathbb{R}^m,$$
$$\mathbf{y}(t_0) = \mathbf{y}_0.$$

Deferred Correction

We are interested in solving the initial-value problem

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in (t_0, t_f), \; \mathbf{y}(t) \in \mathbb{R}^m,$$
$$\mathbf{y}(t_0) = \mathbf{y}_0.$$

Deferred Correction

We are interested in solving the initial-value problem

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in (t_0, t_f),\ \mathbf{y}(t) \in \mathbb{R}^m,$$
$$\mathbf{y}(t_0) = \mathbf{y}_0.$$

Deferred Correction

Computational issues:

- $m \sim \mathcal{O}(n_{\text{grid}})$
- stiffness

Deferred Correction

Canonical algorithms:

- forward Euler (explicit)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t_n \, \mathbf{f}(t_n, \mathbf{y}_n)$$

- backward Euler (implicit)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t_n \, \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$$

Deferred Correction

Canonical algorithms:

- forward Euler (explicit)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t_n \, \mathbf{f}(t_n, \mathbf{y}_n)$$

- backward Euler (implicit)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t_n \, \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$$

Deferred Correction

Canonical algorithms:

- forward Euler (explicit)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t_n \mathbf{f}(t_n, \mathbf{y}_n)$$

- backward Euler (implicit)

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t_n \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$$

Deferred Correction

- serial nature
- lots of information exchange

Deferred Correction

- serial nature
- lots of information exchange

How to take advantage of (massive) parallelism?

Deferred Correction

Natural strategies:

- domain decomposition
- fine-grained parallelism

Deferred Correction

Other strategies:

- specialized integrators
- extrapolation
- PARAREAL
- deferred correction

Deferred Correction

Other strategies:

- specialized integrators
- extrapolation
- PARAREAL
- deferred correction

Deferred Correction

de·fer |diˈfər|
verb (**-ferred, -fer·ring**) [trans.]
put off (an action or event) to a later time; postpone.

cor·rec·tion |kəˈrek SH ən|
noun
the action or process of correcting something;
a change that rectifies an error or inaccuracy

Deferred Correction

- first proposed by Fox (1947)
  - "difference-correction methods"
- Pereyra (1960s), Stetter (1970s), Skeel (1980s); Dutt, Greengard, and Rokhlin (2000); Minion et al. (2000s); Christlieb et al. (2010s); Emmett and Minion (2010s)
- can be applied to BVPs, DAEs, and eigenvalue problems

Deferred Correction

General idea:

1: Compute a provisional solution $\mathbf{Y}^{[0]}$.
2: **for** k=0,1,... **do**
3: Estimate the error $\mathbf{E}^{[k]}$.
4: Correct the current solution by adding error estimate:

$$\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} + \mathbf{E}^{[k]}.$$

5: **end for**

Deferred Correction

General idea:

1: Compute a provisional solution $\mathbf{Y}^{[0]}$.
2: **for** k=0,1,... **do**
3: Estimate the error $\mathbf{E}^{[k]}$.
4: Correct the current solution by adding error estimate:

$$\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} + \mathbf{E}^{[k]}.$$

5: **end for**

Deferred Correction

General idea:

1: Compute a provisional solution $\mathbf{Y}^{[0]}$.
2: **for** k=0,1,... **do**
3: Estimate the error $\mathbf{E}^{[k]}$.
4: Correct the current solution by adding error estimate:

$$\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} + \mathbf{E}^{[k]}.$$

5: **end for**

Accuracy of $\mathbf{Y}^{[k+1]}$ can be $\mathcal{O}\left(\sum_{j=0}^{k} p_j\right)$!

Deferred Correction

3: Estimate the error $\mathbf{E}^{[k]}$.

4: Correct the current solution by adding error estimate:

$$\mathbf{Y}^{[k+1]} = \mathbf{Y}^{[k]} + \mathbf{E}^{[k]}.$$

Pipeline!

Deferred Correction

Main classes of deferred correction methods:

- classical deferred correction
- spectral deferred correction
- integral deferred correction

All equivalent mathematically; devil in details.

Classical Deferred Correction

Consider

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in (t_0, t_f),$$
$$\mathbf{y}(t_0) = \mathbf{y}_0.$$

Define the error function

$$\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{y}(t) - \mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t)) + \mathbf{Y}^{[k]}(t)) - \frac{d}{dt}\mathbf{Y}^{[k]}(t),$$
$$\mathbf{e}(t_0, \mathbf{Y}^{[k]}(t_0)) = \mathbf{0}.$$

Classical Deferred Correction

Consider

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad t \in (t_0, t_f),$$
$$\mathbf{y}(t_0) = \mathbf{y}_0.$$

Define the error function

$$\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{y}(t) - \mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t)) + \mathbf{Y}^{[k]}(t)) - \frac{d}{dt}\mathbf{Y}^{[k]}(t),$$
$$\mathbf{e}(t_0, \mathbf{Y}^{[k]}(t_0)) = \mathbf{0}.$$

Spectral Deferred Correction

Rewrite the exact solution in integral form

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(t', \mathbf{y}(t'))\, dt'.$$

Define the *residual* function

$$\mathbf{r}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(t', \mathbf{Y}^{[k]}(t'))\, dt' - \mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{\Delta f_Y}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t))) + \frac{d}{dt}\mathbf{r}(t, \mathbf{Y}^{[k]}(t)).$$

Spectral Deferred Correction

Rewrite the exact solution in integral form

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(t', \mathbf{y}(t'))\, dt'.$$

Define the *residual* function

$$\mathbf{r}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(t', \mathbf{Y}^{[k]}(t'))\, dt' - \mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) = \boldsymbol{\Delta}\mathbf{f}_{\mathbf{Y}}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t))) + \frac{d}{dt}\mathbf{r}(t, \mathbf{Y}^{[k]}(t)).$$

Integral Deferred Correction

Define the *defect* function

$$\delta(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}(t, \mathbf{Y}^{[k]}(t)) - \frac{d}{dt}\mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\left[\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) - \int_{t_0}^{t} \delta(t', \mathbf{Y}^{[k]}(t'))\, dt'\right] = \mathbf{\Delta f_Y}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t))).$$

Note:

$$\delta(t, \mathbf{Y}^{[k]}(t)) = \frac{d}{dt}\mathbf{r}(t, \mathbf{Y}^{[k]}(t)).$$

Integral Deferred Correction

Define the *defect* function

$$\delta(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}(t, \mathbf{Y}^{[k]}(t)) - \frac{d}{dt}\mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\left[\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) - \int_{t_0}^{t} \delta(t', \mathbf{Y}^{[k]}(t'))\, dt'\right] = \mathbf{\Delta f}_{\mathbf{Y}}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t))).$$

Note:

$$\delta(t, \mathbf{Y}^{[k]}(t)) = \frac{d}{dt}\mathbf{r}(t, \mathbf{Y}^{[k]}(t)).$$

Integral Deferred Correction

Define the *defect* function

$$\delta(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}(t, \mathbf{Y}^{[k]}(t)) - \frac{d}{dt}\mathbf{Y}^{[k]}(t).$$

Then

$$\frac{d}{dt}\left[\mathbf{e}(t, \mathbf{Y}^{[k]}(t)) - \mathbf{r}(t, \mathbf{Y}^{[k]}(t))\right] = \mathbf{\Delta f_Y}(t, \mathbf{e}(t, \mathbf{Y}^{[k]}(t))).$$

Note:

$$\delta(t, \mathbf{Y}^{[k]}(t)) = \frac{d}{dt}\mathbf{r}(t, \mathbf{Y}^{[k]}(t)).$$

Revisionist Integral Deferred Correction

- Classically, sweep level by level.
- “Revisionist”: pipeline subsequent levels.

Error and Stepsize Control

Error control:

- Easy to imagine p-refinement.
- Consider h-refinement.
- hp-refinement: future work.

Error and Stepsize Control

Classical algorithms for error estimation based on *h*-refinement:

- Step doubling
- Embedded formulas

Error and Stepsize Control

Classical algorithms for error estimation based on *h*-refinement:

- Step doubling
- Embedded formulas

Error and Stepsize Control

Step doubling:
If

$$\mathbf{e}_{n+1/2} = \mathbf{y}(t_n + \Delta t_n) - \mathbf{y}_{n+1} = C(\Delta t)^{p+1} + \mathcal{O}(\Delta t^{p+2}),$$

then

$$\mathbf{e}_{n+1} = \mathbf{y}(t_n + 2\Delta t_n) - \mathbf{y}_{n+1} = 2C(\Delta t)^{p+1} + \mathcal{O}(\Delta t^{p+2}),$$
$$\hat{\mathbf{e}}_{n+1} = \mathbf{y}(t_n + 2\Delta t_n) - \hat{\mathbf{y}}_{n+1} = \hat{C}(2\Delta t)^{p+1} + \mathcal{O}(\Delta t^{p+2}),$$

and hence

$$\mathbf{e}_{n+1} = \frac{\mathbf{y}_{n+1} - \hat{\mathbf{y}}_{n+1}}{2^p - 1} + \mathcal{O}(\Delta t^{p+2}).$$

Error and Stepsize Control

Classical algorithm for stepsize control:
Given user tolerances $\boldsymbol{\tau}_{\text{abs}}$, $\boldsymbol{\tau}_{\text{rel}}$, define

$$\boldsymbol{\tau}_{n+1} = \boldsymbol{\tau}_{\text{rel}} \max(|\mathbf{y}_n|, |\mathbf{y}_{n+1}|) + \boldsymbol{\tau}_{\text{abs}},$$

and

$$\epsilon_{n+1} = \left|\left| \frac{\mathbf{e}}{\boldsymbol{\tau}} \right|\right|_{\text{RMS}}.$$

Then

$$\Delta t_{\text{opt}} = \Delta t_n \left(\frac{1}{\epsilon_{n+1}} \right)^{\frac{1}{p+1}}$$

and

$$\Delta t_{n+1} = \alpha \min(a_{\max} \Delta t_n, \max(\Delta t_{\text{opt}}, a_{\min} \Delta t_n))$$

Error and Stepsize Control

Ideas for error and stepsize control within RIDC:

- on prediction level only
- on all levels

Test Problem 1 (Auzinger)

$$\dot{y}_1 = -y_2 + y_1(1 - y_1^2 - y_2^2),$$
$$\dot{y}_2 = y_1 + 3y_2(1 - y_1^2 - y_2^2),$$

$$y(0) = (1, 0)^T, \quad t \in [0, 10].$$

Exact solution:

$$y(t) = (\cos t, \sin t)^T.$$

Test Problem 2 (Arenstdorf)

$$\ddot{y}_1 = y_1 + 2\dot{y}_2 - \mu' \frac{y_1 + \mu}{D_1} - \mu \frac{y_1 - \mu'}{D_2},$$
$$\ddot{y}_2 = y_2 - 2\dot{y}_1 - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2},$$

$$D_1 = \left((y_1 + \mu)^2 + y_2^2\right)^{3/2}, \quad D_2 = \left((y_1 - \mu')^2 + y_2^2\right)^{3/2},$$

$$\mu = 0.012277471, \quad \mu' = 1 - \mu.$$

$$y_1(0) = 0.994, \quad \dot{y}_1(0) = 0, \quad y_2(0) = 0,$$

$$\dot{y}_2(0) = -2.00158510637908252240537862224.$$

Solution has period $t_{end} = 17.0652165601579625588917206249.$

Order Verification

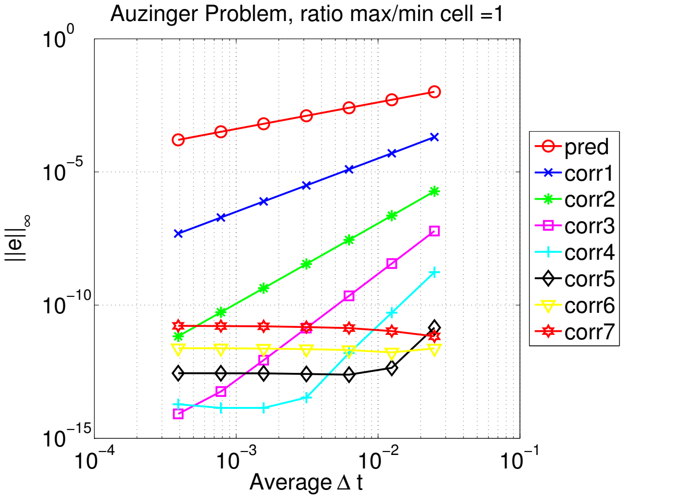
Randomize the step size according to

$$t_{n+1}^{[k]} = t_n^{[k]} + \Delta t_n^{[k]}, \quad k = 0, 1, \ldots, K, \quad n = 0, 1, \ldots,$$

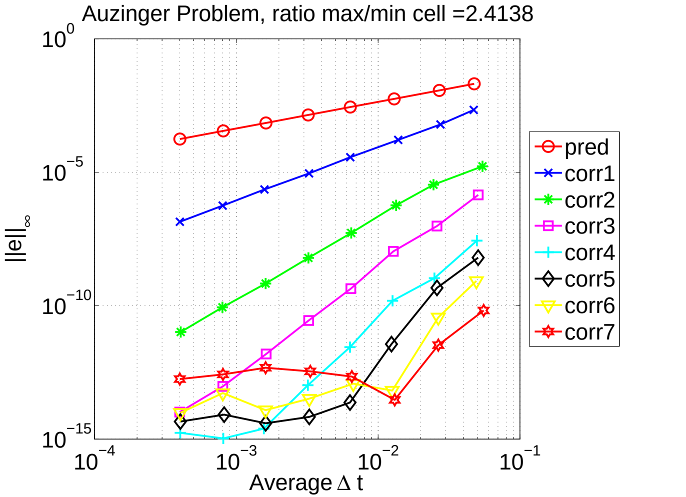
where $\Delta t_n^{[k]}$ is randomly selected from

$$\left[\frac{1}{\omega}\Delta t_{n-1}^{[k]}, \omega \Delta t_{n-1}^{[k]}\right], \quad \omega \geq 1.$$

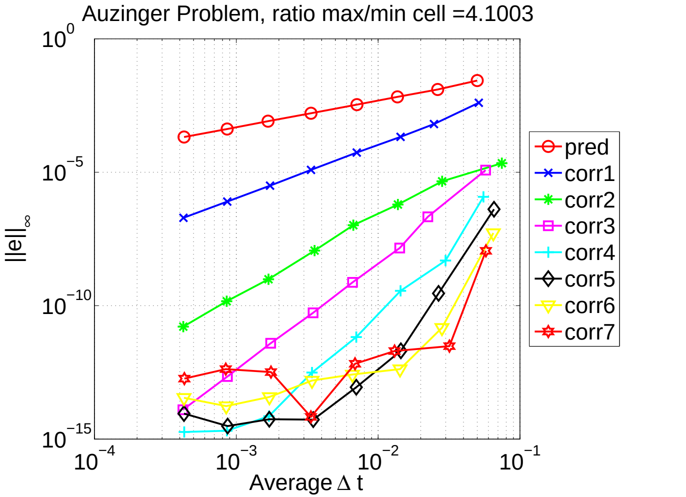
Results


Auzinger Problem, ratio max/min cell =1
10^{0}
10^{-5}
10^{-10}
10^{-15}
$||e||_\infty$
10^{-4}
10^{-3}
10^{-2}
10^{-1}
Average Δ t
pred
corr1
corr2
corr3
corr4
corr5
corr6
corr7

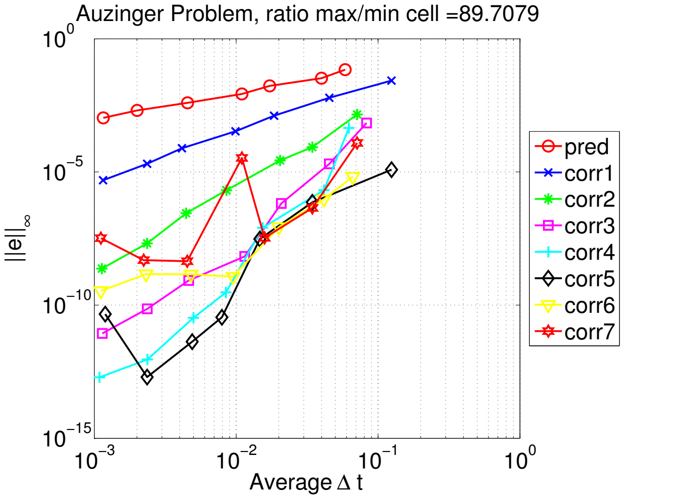
Results


Auzinger Problem, ratio max/min cell =2.4138
10^{0}
10^{-5}
10^{-10}
10^{-15}
$\|e\|_\infty$
10^{-4}
10^{-3}
10^{-2}
10^{-1}
Average Δ t
pred
corr1
corr2
corr3
corr4
corr5
corr6
corr7

Results


Auzinger Problem, ratio max/min cell =4.1003
10^{0}
10^{-5}
10^{-10}
10^{-15}
$||e||_\infty$
10^{-4}
10^{-3}
10^{-2}
10^{-1}
Average Δ t
pred
corr1
corr2
corr3
corr4
corr5
corr6
corr7

Results

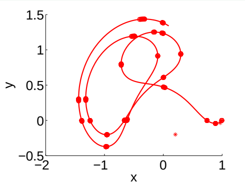

Auzinger Problem, ratio max/min cell =89.7079
10^0
10^{-5}
10^{-10}
10^{-15}
$||e||_\infty$
10^{-3}
10^{-2}
10^{-1}
10^0
Average Δ t
pred
corr1
corr2
corr3
corr4
corr5
corr6
corr7

Error and Stepsize Control on Prediction Level Only

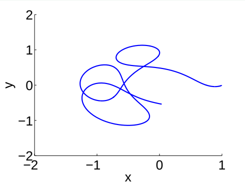
First idea:

- Perform standard error and stepsize control on prediction level.
- Use same stepsizes on remaining levels.

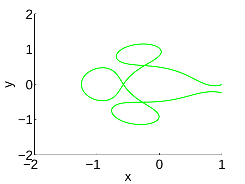
Results (Orbit Problem)



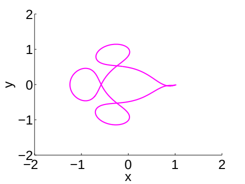
(a) Level 0



(b) Level 1



(c) Level 2



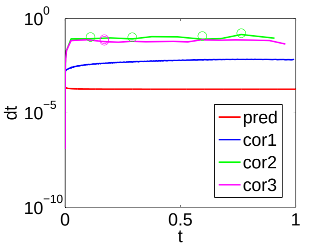
(d) Level 3

Error and Stepsize Control on All Levels

Better idea:

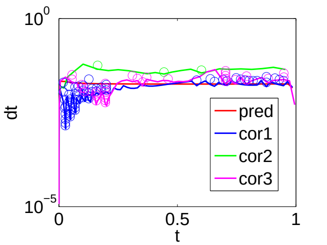
- Perform standard error and stepsize control on all levels.
- Tolerance settings:
  - Same at every level.
  - Loose at first, gradually tighten.
- How to sensibly set tolerances: future work.

Results (Auzinger Problem)



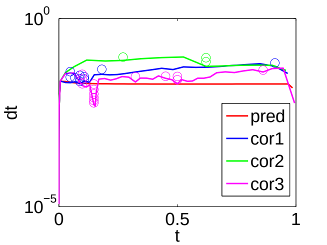
ℓ	rtol	atol	error	naccept	nreject
0	10^{-8}	10^{-10}	2.028e-05	5480	0
1	10^{-8}	10^{-10}	8.824e-07	197	0
2	10^{-8}	10^{-10}	1.917e-08	19	4
3	10^{-8}	10^{-10}	6.386e-07	25	2

Results (Auzinger Problem)



p	rtol	atol	error	naccept	nreject
0	1e-04	1e-06	2.031e-03	59	0
1	1e-06	1e-08	7.002e-05	81	61
2	1e-08	1e-10	1.412e-07	30	3
3	1e-10	1e-12	9.847e-08	60	33

Results (Auzinger Problem)



p	rtol	atol	error	naccept	nreject
0	1e-04	1e-06	2.031e-03	59	0
1	1e-05	1e-07	1.853e-04	31	10
2	1e-07	1e-09	1.505e-06	21	3
3	1e-09	1e-11	9.473e-07	44	14

Conclusions

- Deferred correction is a well-studied idea
- RIDC is a framework that allows for parallelization
- Error and stepsize control can be implemented in RIDC