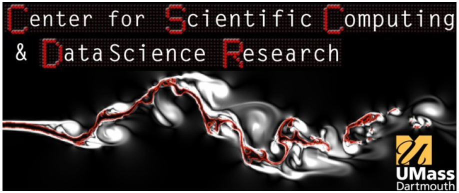
Mixed precision / mixed model time-stepping methods

Sigal Gottlieb
Center for Scientific Computing and Data Science Research
University of Massachusetts Dartmouth



Joint work with Ben Burnett & Zachary Grant (UMassD)

Thanks to the AFOSR and DOE CHaRMNET grants for funding this work

Motivation

We are interested in numerically solving a large system of ordinary differential equation (ODE) of the form

$$\begin{aligned} u_t &= F(u)\ , \qquad t \geq 0 \\ u(t=0) &= u_0 \end{aligned}$$

If the system is *stiff*, then we need to solve it with an implicit solver. Consider the implicit midpoint rule

$$u^{n+1} = u^n + \Delta t F\left(\frac{u^n + u^{n+1}}{2}\right)$$

Motivation

The implicit midpoint rule

$$u^{n+1} = u^n + \Delta t F\left(\frac{u^n + u^{n+1}}{2}\right)$$

can be equivalently written in its Butcher form as:

$$\begin{aligned} y^{(1)} &= u^n + \frac{\Delta t}{2} F\left(y^{(1)}\right) \\ u^{n+1} &= u^n + \Delta t\ F\left(y^{(1)}\right). \end{aligned}$$

Our problem: Implicit solves (needed for the first stage) are expensive.
Our goal: make the implicit solve **cheaper** while retaining the accuracy of the solution.

Simple idea

$$y^{(1)} = u^n + \frac{\Delta t}{2} F_\epsilon \left(y^{(1)} \right)$$
$$u^{n+1} = u^n + \Delta t \; F \left(y^{(1)} \right).$$

We replace F with F_ϵ so the implicit solve is cheap:

- F_ϵ comes from a cheaper model
  - Different physical model
  - Different numerical method for semi-discretizing the PDE
  - Linearization of the operator
- F_ϵ is resulting from a low precision computation
- F_ϵ is resulting from computing the implicit solve with a larger tolerance or low precision

Simple idea

$$y^{(1)} = u^n + \frac{\Delta t}{2} F_\epsilon\left(y^{(1)}\right)$$
$$u^{n+1} = u^n + \Delta t \, F\left(y^{(1)}\right).$$

The first stage will then have an error of order $\Delta t \epsilon$
(*or order ϵ, depending on how this is done and how you define ϵ*)
in this formulation the error is corrected at the second stage so that we have a local error of $\Delta t^2 \epsilon$, (*or $\Delta t \epsilon$, respectively*).
Running this code to final time T_f means that we run it for $N = \frac{T_f}{\Delta t}$ steps so that the error builds up to $O(\Delta t \epsilon)$ (or $O(\epsilon)$) at the final time.

How well does this work?

Let's compare:

1. low precision doing both stages in low precision
2. mixed precision doing the first stage in low precision and the second in high precision
3. high precision doing both stages in high precision

The problem we will solve is a simple nonlinear ODE system (not even stiff):

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_2\,(1 - y_1^2) - y_1 \end{aligned}$$

with initial conditions $y_1(0) = 2$ and $y_2(0) = 0$. We stepped this forward to a final time $T_f = 1.0$.

How well does this work?

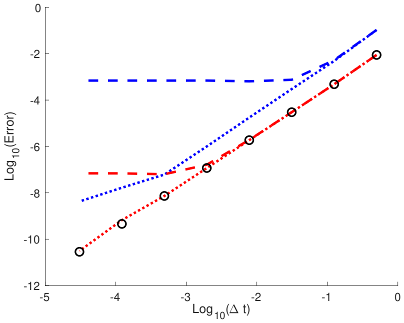
blue is half precision: $\epsilon = O(10^{-4})$

red is single precision: $\epsilon = O(10^{-8})$

dashed line is low precision

dotted line is mixed precision

black dots are high precision



UMass Dartmouth

How efficient is this?

Let’s look at the accuracy (top) and the machine time (bottom) for this approach:

dt	64/64	64/32	64/16
0.01	4.078E-06 0.0005	4.08E-06 0.00029	1.4E-03 0.00011
0.001	4.078E-08 0.016	2.68E-07 0.002	1.6E-04 0.00078
0.0001	4.077E-10 0.026	3.34E-09 0.041	1.58E-05 0.007
1E-05	4.219E-12 0.25	1.53E-09 0.22	1.32E-06 0.081
1E-06	7.375E-12 2.37	1.53E-07 0.75	6.79E-07 0.62

dt	128/128	128/64
0.01	4.078E-06 0.005	4.078E-06 0.00068
0.001	4.078E-08 0.065	4.078E-08 0.0058
0.0001	4.078E-10 0.29	4.078E-10 0.093
1E-05	4.078E-12 2.36	4.078E-12 0.39
1E-06	4.078E-14 15.66	**4.078E-14** 4.17

The results look promising, but it is not great when Δt is small compared to ϵ . . . these methods still need something else.

UMass Dartmouth

Explicit Corrections

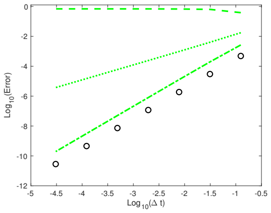
The mixed precision method uses the second stage as a correction of sorts. What if we add in another explicit correction when Δt is small?

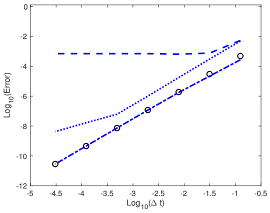
$$\begin{aligned} y^{(1)}_{[0]} &= u^n + \frac{1}{2}\Delta t F^{\epsilon}(y^{(1)}_{[0]}) \\ y^{(1)}_{[1]} &= u^n + \frac{1}{2}\Delta t F(y^{(1)}_{[0]}) \\ u^{n+1} &= u^n + \Delta t F\left(y^{(1)}_{[1]})\right). \end{aligned}$$

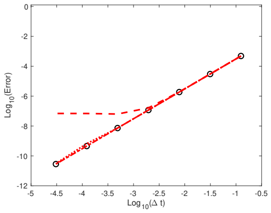
The first stage will then have an error of order $\Delta t \epsilon$, and this is corrected once at the second stage and once at the third stage so that we have a local error of $\Delta t^3 \epsilon$.
Running this code to final time T_f, the error reduces by $O(\Delta t)$ at the final time.

Explicit Corrections

Left: Zero precision $\epsilon = O(1)$, Middle: half precision $\epsilon = O(10^{-4})$, Right: single precision $\epsilon = O(10^{-8})$.







Dashed: low precision; Dotted: mixed precision; Dot-dash: mixed precision with correction.

Explicit Corrections

Double precision with one explicit correction (Julia code)

dt	64/64	64/32	64/16	
0.01	4.078E-06 0.0005	4.078E-06 0.00026	2.12E-05 0.00013	error runtime
0.001	4.078E-08 0.016	4.051E-08 0.002	3.15E-07 0.0014	error runtime
0.0001	4.077E-10 0.026	4.074E-10 0.059	3.04E-09 0.017	error runtime
1E-05	4.21E-12 0.25	4.21E-12 0.18	2.99E-11 0.089	error runtime
1E-06	7.37E-12 2.37	7.46E-12 0.92	7.45E-12 0.83	error runtime

The corrections help us get the accuracy we want less runtime.

Explicit Corrections

Compare no corrections (left) with one correction (right) for 64/16:

Δt	No corrections	One correction
0.01	1.4E-03 0.00011	2.12E-05 0.00013
0.001	1.6E-04 0.00078	3.15E-07 0.0014
0.0001	1.58E-05 0.007	3.04E-09 0.017
1E-05	1.32E-06 0.081	2.99E-11 0.089
1E-06	6.79E-07 0.62	7.45E-12 0.83

Notice that these improve by a factor of Δt, as predicted by the theory.

Quad precision with two explicit corrections (Julia code)

dt	128/128	128/64	128/32	128/16
0.01	4.078E-06 0.005	4.078E-06 0.0008	4.078E-06 0.0006	4.30E-06 0.0003
0.001	4.078E-08 0.065	4.078E-08 0.0074	4.078E-08 0.014	4.11E-08 0.0028
0.0001	4.078E-10 0.29	4.078E-10 0.086	4.078E-10 0.053	4.08E-10 0.067
1E-05	4.078E-12 2.36	4.078E-12 0.55	4.078E-12 0.37	4.078-12 0.24
1E-06	4.078E-14 15.66	4.078E-14 5.12	4.078E-14 2.37	4.078E-14 2.49

For quad precision, we need more corrections to see the accuracy we want. The cost of the corrections is small.

Quad/half (128/16) precision with $c = 0, 1, 2$ corrections

Δt	no corrections	one correction	two corrections
0.01	1.43E-03 0.0002	2.12E-05 0.0002	4.30E-06 0.0003
0.001	1.69E-04 0.0015	3.16E-07 0.0019	4.11E-08 0.0028
0.0001	1.59E-05 0.015	3.04E-09 0.072	4.08E-10 0.067
1E-05	1.34E-06 0.11	3.07E-11 0.17	4.078-12 0.24
1E-06	6.80E-07 1.28	2.75E-13 1.86	4.078E-14 2.49

Notice that at the first correction these improve by a factor of Δt, as predicted by the theory. However, as the underlying order of the method is $O(\Delta t^2)$, we don't observe the additional factor of Δt in the second correction: the scheme error dominates.

SDIRK with explicit corrections

The third order singly diagonally implicit Runge–Kutta method

$$\begin{aligned}
y^{(1)}_{[0]} &= u^n + \gamma \Delta t F^{\epsilon}(y^{(1)}_{[0]}) \\
y^{(1)}_{[k]} &= u^n + \gamma \Delta t F(y^{(1)}_{[k-1]}) \quad \text{for} \;\; k = 1, ..., m-1 \\
y^{(2)}_{[0]} &= u^n + (1-2\gamma)\Delta t F(y^{(1)}_{[m-1]}) + \gamma \Delta t F(y^{(2)}_{[0]}) \\
y^{(2)}_{[k]} &= u^n + (1-2\gamma)\Delta t F(y^{(1)}_{[m-1]}) + \gamma \Delta t F(y^{(2)}_{[k-1]}) \quad \text{for} \;\; k = 1, ..., m-1 \\
u^{n+1} &= u^n + \frac{1}{2}\Delta t F(y^{(1)}_{[m-1]})) + \frac{1}{2}\Delta t F(y^{(2)}_{[m-1]})).
\end{aligned}$$

with $\gamma = \frac{\sqrt{3}+3}{6}$. If $m=1$ we have the uncorrected method, which should have global $O(\Delta t^3) + O(\epsilon \Delta t)$. Each correction will raise the perturbation order by one, so that we obtain a global error of the form $O(\Delta t^3) + O(\epsilon \Delta t^m)$.

SDIRK with no corrections

Working with the same van der Pol problem.

Δt	128/128	128/64	128/32	128/16
0.01	2.13E-07	2.13E-07	2.33E-07	0.00018
0.001	2.20E-10	2.20E-10	3.39E-08	0.00035
0.0001	2.21E-13	2.21E-13	8.75E-09	1.79E-05
E-05	2.21E-16	2.24E-16	1.04E-09	1.36E-06
E-06	2.21E-19	6.04E-19	1.80E-08	6.80E-07

SDIRK with one explicit correction

After one correction the perturbation error is damped by a factor of Δt, unless the method error dominates:

Δt	128/128	128/64	128/32	128/16
0.01	2.13E-07	2.13E-07	2.13E-07	2.94E-06
0.001	2.20E-10	2.20E-10	1.43E-10	9.92E-07
0.0001	2.21E-13	2.21E-13	1.59E-12	5.99E-09
1E-05	2.21E-16	2.21E-16	1.70E-14	5.46E-11
1E-06	2.21E-19	2.21E-19	4.06E-14	4.91E-13

SDIRK with two explicit corrections

After two corrections the perturbation error is damped by a factor of Δt^2, unless the method error dominates:

Δt	128/128	128/64	128/32	128/16
0.01	2.13E-07	2.13E-07	2.13E-07	2.58E-07
0.001	2.20E-10	2.20E-10	2.20E-10	1.94E-09
0.0001	2.21E-13	2.21E-13	2.21E-13	1.32E-12
1E-05	2.21E-16	2.21E-16	2.21E-16	1.19E-15
1E-06	2.21E-19	2.21E-19	2.90E-19	1.14E-18

SDIRK with three explicit corrections

After three corrections the perturbation error is damped by a factor of Δt^3 and we observe only the method error:

dt	128/128	128/64	128/32	128/16
0.01	2.13E-07	2.131E-07	2.13E-07	2.13E-07
0.001	2.20E-10	2.20E-10	2.20E-10	2.17E-10
0.0001	2.21E-13	2.21E-13	2.21E-13	2.21E-13
1E-05	2.21E-16	2.21E-16	2.21E-16	2.21E-16
1E-06	2.21E-19	2.21E-19	2.21E-19	2.21E-19

UMass | Dartmouth

The theory behind this approach . . . and extensions

The theory behind this approach was developed by Zachary Grant (JSC, January 2022):
Consider a Runge–Kutta method written in the Butcher form

$$y^{(i)} = u^n + \Delta t \sum_{j=1}^{s} A_{ij} F(y^{(j)})$$

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j F(y^{(j)}).$$

In our case, we will be computing both F and F^ϵ, so we write it in the form of an additive Runge–Kutta method:

$$y^{(i)} = u^n + \Delta t \sum_{j=1}^{s} \hat{A}_{ij} F(y^{(j)}) + \Delta t \sum_{j=1}^{s} A^\epsilon_{ij} F^\epsilon(y^{(j)}).$$

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} \hat{b}_j F(y^{(j)}) + \Delta t \sum_{j=1}^{s} b^\epsilon_j F^\epsilon(y^{(j)}).$$

The theory behind this approach . . . and extensions

It makes sense to look at F^ϵ as a perturbation of F, so we say $F(u) = F^\epsilon(u) + \epsilon\tau(u)$ and write the additive Runge–Kutta method as

$$y^{(i)} = u^n + \Delta t \sum_{j=1}^{s} A_{ij} F(y^{(j)}) + \epsilon \Delta t \sum_{j=1}^{s} A^\epsilon_{ij} \tau(y^{(j)}).$$

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j F(y^{(j)}) + \epsilon \Delta t \sum_{j=1}^{s} b^\epsilon_j \tau(y^{(j)}).$$

The case where τ is a smooth and negligible perturbation was considered by Ketcheson. Grant considered cases where τ was smooth or nonsmooth, or even non-deterministic.

The theory behind this approach . . . and extensions

$$
\begin{aligned}
u^{n+1} = & \underbrace{u^n + \Delta t \tilde{b} e F(u^n) + \Delta t^2 \tilde{b}\tilde{c} F_y(u^n) F(u^n)}_{scheme} \\
& + \underbrace{\epsilon \Delta t \left(b^\epsilon e \tau(u^n) + \Delta t \left(b^\epsilon \tilde{c} \tau_y(u^n) F(u^n) + \tilde{b} c^\epsilon F_y(u^n) \tau(u^n) + \epsilon b^\epsilon c^\epsilon \tau_y(u^n) \tau(u^n) \right) \right)}_{perturbation} \\
& + O(\Delta t^3).
\end{aligned}
$$

This shows two sources of error: those of the scheme and those of the perturbation: we expect the error to have order comprised of these two:

$$O(\Delta t^p) + O(\epsilon \Delta t^m).$$

UMass | Dartmouth

The theory behind this approach . . . over-simplified!

Consider the case where we have a diagonally implicit method, and only the implicit part is handled in the cheap way (so $b^\epsilon_j = 0$):

$$
\begin{aligned}
y^{(i)} &= u^n + \Delta t \sum_{j=1}^{i} A_{ij} F(y^{(j)}) + \epsilon \Delta t A^\epsilon_{ii} \tau(y^{(i)}). \\
u^{n+1} &= u^n + \Delta t \sum_{j=1}^{s} b_j F(y^{(j)}).
\end{aligned}
$$

Let's look at the order conditions, derived through Taylor expansions:

The theory behind this approach . . . over-simplified!

Let's look at the order conditions, derived through Taylor expansions, up to third order:

$$y_i = u^n + \Delta t \sum_{j=1}^{i} a_{ij} F(y_j) + \Delta t \, \epsilon a^{\epsilon}_{ii} \tau(y_i) \tag{1}$$

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{s} b_i F(y_i) \tag{2}$$

We can expand these methods as

$$\begin{aligned} u^{n+1} &= u^n + \Delta t \sum_{i=1}^{s} b_i F(y_i) \\ &= u^n + \Delta t \sum_{i=1}^{s} b_i F\left(u^n + \Delta t \sum_{k=1}^{i} a_{ik} F(y_k) + \Delta t \, \epsilon a^{\epsilon}_{ii} \tau(y_i) \right) \end{aligned}$$

The theory behind this approach . . . over-simplified!

$$
\begin{aligned}
u^{n+1} \;&=\; u^n + \Delta t \sum_{i=1}^{s} b_i F(u^n) + \epsilon \Delta t^2 F'(u^n) \sum_{i=1}^{s} b_i a^{\epsilon}_{ii} \tau(y_i) \\
&\quad + \Delta t^2 \sum_{i=1}^{s} b_i F'(u^n) \left(\sum_{k=1}^{i} a_{ik} F(y_k) \right)
\end{aligned}
$$

If we want to eliminate the $\epsilon \Delta t^2$ term, we need

$$b_1 a^{\epsilon}_{11} \tau(y_1) + b_2 a^{\epsilon}_{22} \tau(y_2) + ... + b_s a^{\epsilon}_{ss} \tau(y_s) = 0$$

and cannot assume that τ is smooth so we require

$$|b_i| \, |a^{\epsilon}_{ii}| = 0 \quad \forall i.$$

UMass | Dartmouth

The theory behind this approach . . . over-simplified!

Now we have

$$
\begin{aligned}
u^{n+1} &= u^n + \Delta t \sum_{i=1}^{s} b_i F(u^n) + \Delta t^2 \sum_{i=1}^{s} b_i F'(u^n) \left(\sum_{k=1}^{i} a_{ik} F(y_k) \right) \\
&= u^n + \Delta t \sum_{i=1}^{s} b_i F(u^n) \\
&+ \Delta t^2 \sum_{i=1}^{s} b_i F'(u^n) \left(\sum_{k=1}^{i} a_{ik} F \left(u^n + \Delta t \sum_{j=1}^{k} a_{kj} F(y_j) + \Delta t \, \epsilon a^{\epsilon}_{kk} \tau(y_k) \right) \right) \\
&= u^n + \Delta t \sum_{i=1}^{s} b_i F(u^n) + \Delta t^2 \sum_{i=1}^{s} \sum_{k=1}^{i} b_i a_{ik} F'(u^n) F(u^n) \\
&+ \Delta t^3 \sum_{i=1}^{s} b_i \sum_{k=1}^{i} a_{ik} \sum_{j=1}^{k} a_{kj} F'(u^n) F'(u^n) \\
&\quad + \epsilon \Delta t^3 \sum_{i=1}^{s} b_i F'(u^n) F'(u^n) \left(\sum_{k=1}^{i} a_{ik} a^{\epsilon}_{kk} \tau(y_k) \right)
\end{aligned}
$$

The theory behind this approach . . . over-simplified!

If we want to eliminate the $\epsilon \Delta t^3$ term, we need

$$\sum_{i=1}^{s} \sum_{k=1}^{i} b_i a_{ik} a^{\epsilon}_{kk} \tau(y_k) = 0,$$

- If τ is smooth (e.g. linearization) we can say that we require

$$\sum_{i=1}^{s} \sum_{k=1}^{i} b_i a_{ik} a^{\epsilon}_{kk} = 0.$$

- If τ is non-smooth but deterministic (e.g. rounding) we require

$$\left| \sum_{i=j}^{s} b_i a_{ij} \right| \left| a^{\epsilon}_{jj} \right| = 0 \quad \text{for} \;\; j = 1, ..., s.$$

- If τ is stochastic (e.g. stochastic rounding) we require

$$\sum_{i=1}^{s} \sum_{k=1}^{i} |b_i| \, |a_{ik}| \, |a^{\epsilon}_{kk}| = 0.$$

The theory behind this approach . . . and extensions

- For details on the order conditions, see Z. J. Grant (JSC 2022) paper
- Using this theory we can formally define the order of mixed precision / mixed model implementations of known Runge–Kutta methods with and without corrections.
- This theory also allowed the development of some novel methods.
- This work suggested the use of this approach for mixed precision applications and for mixed model applications.

Efficiency: mixed precision

We are also interested in bigger problems than a small van der Pol system!

We are looking to solve the viscous Burgers' Equation

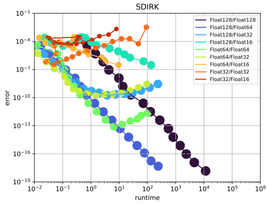
$$u_t + \left(\frac{1}{2}u^2\right)_x = \beta u_{xx}$$

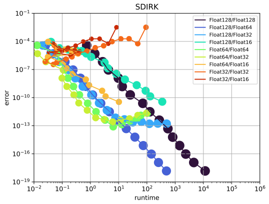
where $\beta = 0.01$
sine wave initial conditions
and zero boundary conditions.

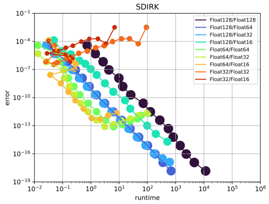
We discretize in space using N_x points, and then solve the ODE system that results, using all these methods.

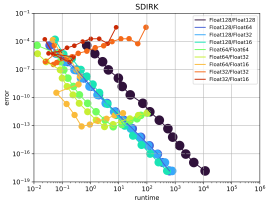
The following are the results of these simulations.

SDIRK with $N_x = 50$



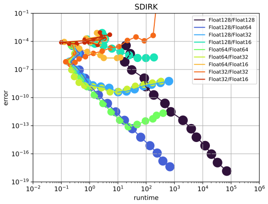


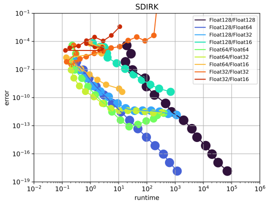


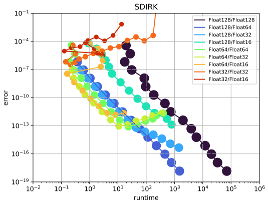


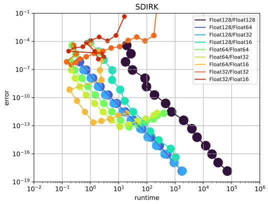
Julia code results: The SDIRK method for the viscous Burgers' equation with $N_x = 50$. No corrections (top left), one correction (top right), two corrections (bottom left), and three corrections (bottom right).

SDIRK with $N_x = 100$



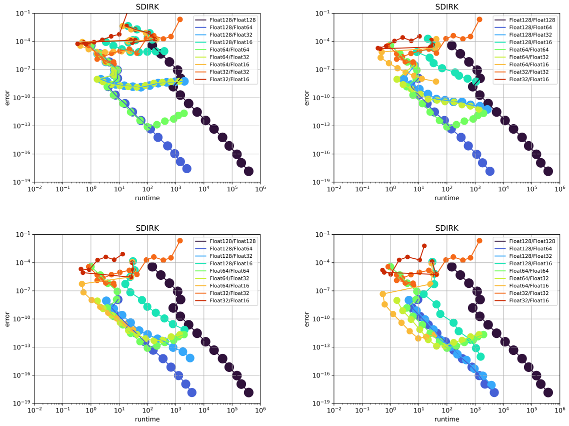






Julia code results: The SDIRK method for the viscous Burgers' equation with $N_x = 100$. No corrections (top left), one correction (top right), two corrections (bottom left), and three corrections (bottom right).

SDIRK with $N_x = 200$



Julia code results: The SDIRK method for the viscous Burgers' equation with $N_x = 200$. No corrections (top left), one correction (top right), two corrections (bottom left), and three corrections (bottom right).

SDIRK with 3 corrections

Error level $\approx 10^{-10}$			
spatial points	64/64	64/32	64/16
$N_x = 50$	0.41	0.240	0.062
$N_x = 100$	1.47	0.74	0.18
$N_x = 200$	6.32	5.4	1.23

For $N_x = 50$ the double/half method with three corrections provides a 4x speedup over the double/single and a 6.6x speedup over the double precision code.

For $N_x = 100$ the double/half method with three corrections providing a speedup factor of 2x over the double/single method, and a speedup of over 8x over the double precision.

For $N_x = 200$ we see a speedup of over 5x over the double precision code.

SDIRK with corrections at error levels $\approx 10^{-18}$

SDIRK with 2 corrections for error level $\approx 10^{-18}$				
spatial points	128/128	128/64	128/32	128/16
$N_x = 50$	11,524	686	–	–
$N_x = 100$	70,449	1,517	–	–
$N_x = 200$	387,140	3,768	–	–
SDIRK with 3 corrections for error level $\approx 10^{-18}$				
$N_x = 50$	11,524	630	778	515
$N_x = 100$	70,449	1,760	1,831	–
$N_x = 200$	387,140	4,760	3,765	–

For $N_x = 50$, the quad/half with three corrections provide a speedup of over 22x over the quad precision code.

For $N_x = 100$ the quad/double with two corrections provides the most efficient result, with a speedup of 40x over the quad precision code.

For $N_x = 200$ the quad/single with three corrections and the quad/double with two corrections are similar, with a speedup of **two orders of magnitude** over the quad precision code.

Mixed precision and mixed method computations

We will use this approach for mixed precision and mixed method computations.

We need to understand the stability implications of the explicit corrections:

- Corrections are needed when Δt is small
- Corrections are stable when Δt is small

Mixed precision and mixed method computations

We will use this approach for mixed precision and mixed method computations.

We need to understand the stability implications of the corrections:

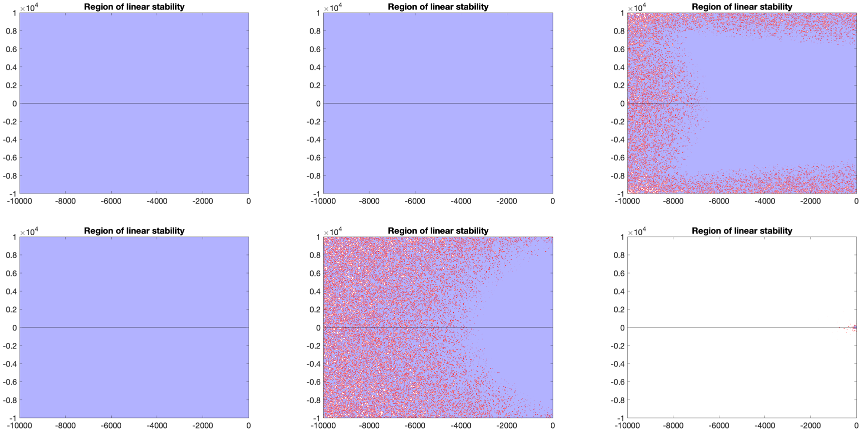
- Corrections are needed when Δt is small

  Δt is small compared to ϵ
- Corrections are stable when Δt is small

  Δt is inversely proportional to the stiffness of the problem

Stability: mixed precision

Regions of linear stability for the SDIRK with no corrections (left), one correction (middle), and two corrections (right), for $\tilde{\epsilon} = \frac{\epsilon}{\lambda} = 10^{-8}$ (top) and 10^{-4} (bottom).



Stability

A brief look at the different notions of ”small” Δt allows us to conclude:

Corrections needed for accuracy

Accuracy requires that we use corrections when $dt^p << \epsilon \Delta t^m$, where p is the order of the method, and m is the order from the perturbation analysis.

For SDIRK, we need corrections when $\Delta t^3 << \epsilon \Delta t$, this occurs when $\Delta t << \sqrt{\epsilon}$.

Corrections are stable

Linear stability analysis allows us to conclude that stability occurs when

$$\Delta t << \frac{1}{\lambda}$$

Stability

Corrections needed

$$\Delta t << \epsilon^{m/p}$$

Corrections stable

$$\Delta t << \frac{1}{\lambda}$$

If ϵ is very small, or the designed method is accurate to perturbations, i.e. $\frac{1}{\lambda} >> \epsilon^{m/p}$, then when we need to correct the corrections will be stable
But if the problem is very stiff: $\frac{1}{\lambda} << \epsilon^{m/p}$ then we need corrections when Δt is moderate, but we only have stability for these corrections when Δt is much smaller!

Stability

The relationship between F and F^ϵ is important to the understanding of stability. This includes the *magnitude* of $F - F^\epsilon$ and also the *relationship* between F and F^ϵ.

Let's say that z_n comes from an evolution of the problem with a fully implicit expensive method (i.e we only use F and not F^ϵ to evolve z) and y_n comes from an evolution of the problem where the diagonal implicit term is computed with the cheap F^ϵ instead. Under conditions on A and b and if F is *contractive* we have

$$(y_{n+1} - z_{n+1})^2 \leq (y_n - z_n)^2 + 2\epsilon\Delta t \sum_{i=1}^{s} b_i a_{ii} \left(F(y_i) - F(z_i), \tau(y_i)\right)$$

So ϵ matters, and the behavior of τ matters, and the relationship between F and F^ϵ matters too.

Stability: mixed model

If F^{ϵ} is a linearization of F done well, then we expect

$$F - F^{\epsilon} = O(\Delta t^2)$$

.

This means we expect to see a proper decay of the growth of the error that depends on Δt.

We also expect, somehow, that the stiffness of F is captured by F^{ϵ}, so that we have a stable behavior of the errors.

What is we have a mixed model where this is not the case?

UMass | Dartmouth

Stability: mixed model

Solve the heat equation

$$u_t = u_{xx}, \quad \text{on} \;\; x \in [0, 2\pi]$$

with periodic boundary conditions and initial condition $u(x,0) = \frac{1+\sin(x)}{2}$.

$$\begin{aligned} y_1 &= u^n + \frac{1}{2}\Delta t D_c \, y_1 \\ u^{n+1} &= u^n + \Delta t D_s \, y_1 \end{aligned}$$

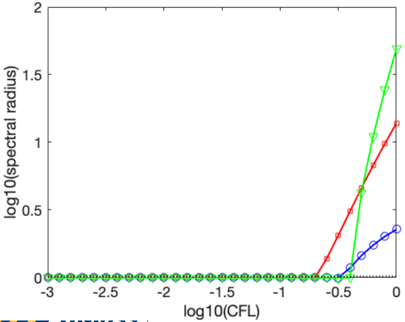
For the implicit solution, use a second order centered difference scheme D_c.
For the explicit stage, use a high order spectral methods D_s.
A one-step evolution with q corrections, becomes $u^{n+1} = P u^n$ where

$$P = \left(I + \Delta t D_s \left(I + \frac{1}{2}\Delta t D_s \right)^q \left(I - \frac{1}{2}\Delta t D_c \right)^{-1} \right).$$

Stability: mixed model

To analyze stability, look at the eigenvalues: the spectral radius of the matrix

$$P = \left(I + \Delta t D_s \left(I + \frac{1}{2} \Delta t D_s \right)^q \left(I - \frac{1}{2} \Delta t D_c \right)^{-1} \right).$$



no corrections (circles)
one correction (squares)
two corrections (triangles)
The implicit solve on D_c is not damping out the explicit step with D_s, so this process is only stable for small CFL numbers.

UMass | Dartmouth

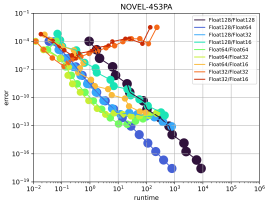
Conclusions

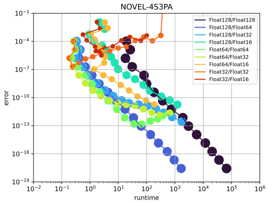
- Mixed precision Runge-Kutta approach may provide significant savings in runtime and allow the efficiency of single and half precision that would otherwise be unstable
- Mixed precision Runge-Kutta methods display the expected perturbation order (this is true of mixed model as well)
- For the mixed precision case, explicit corrections are needed and are beneficial for sufficiently small time-step
- Explicit corrections improve the perturbation order but stability may suffer if step-size is not small enough
- We are investigating other approaches for stable corrections: implicit-inspired corrections at the cost of an explicit correction using a Broyden type approach works very well numerically – we are now doing a formal analysis to understand this.
- We are analyzing the nonlinear stability properties for the mixed model and mixed precision perturbations

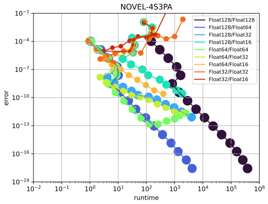
UMass | Dartmouth

Thank you!

NovelA method







Julia code results: The NovelA for the viscous Burgers' equation with $N_x = 50$ (left), $N_x = 100$ (middle), and $N_x = 200$ (right).

NovelA method

.

Major efficiency gains are seen for the quad/double code:

NovelA for error level $\approx 10^{-18}$			
spatial points	128/128	128/64	savings factor
$N_x = 50$	8,622	800	10.77x
$N_x = 100$	66,404	1662	39.95x
$N_x = 200$	381,190	4155	91.74x

We note that it is also possible to add explicit corrections to the NovelA method when needed, and we expect this to improve the mixed precision errors and efficiency for small Δt.