1. Nonlinear. Have ODEs,

$$M\mathbf{u}'(t) = \mathcal{N}(\mathbf{u}, t) \quad \text{in } (0, T], \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{1}$$

where M is a mass matrix and $\mathcal{N}\colon \mathbb{R}^N \times \mathbb{R}_+ \to \mathbb{R}^N$ is a discrete, time-dependent, nonlinear operator. Then, an s-stage IRK scheme approximately propagates the discrete solution by

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \delta t \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{2}$$

with stage vectors satisfying the block system of s nonlinear equations,

$$M\mathbf{k}_i = \mathcal{N}\left(\mathbf{u}_n + \delta t \sum_{j=1}^{s} a_{ij}\mathbf{k}_j, t_n + \delta t c_i\right), \quad i = (1, \dots, s). \tag{3}$$

For a given pair $(\boldsymbol{u}_n, t_n)$, let's define the operator $F_i\colon \mathbb{R}^{sN} \to \mathbb{R}^N$ that encodes the ith stage vector,

$$F_i(\boldsymbol{k}) \coloneqq M\boldsymbol{k}_i - \mathcal{N}\left(\boldsymbol{u}_n + \delta t \sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j, t_n + c_i\delta t\right), \tag{4}$$

with $\boldsymbol{k} = (\boldsymbol{k}_1, \dots, \boldsymbol{k}_s)^\top$ the vector of stages. Then, the nonlinear system we need to solve to advance forwards one time step is

$$\begin{bmatrix} F_1(\boldsymbol{k}) \\ \vdots \\ F_s(\boldsymbol{k}) \end{bmatrix} =: F(\boldsymbol{k}) = 0. \tag{5}$$

But rather than solving this nonlinear system, we make a slight change of variables. Say we make the following simple change of variables:

$$\boldsymbol{w} = (A_0 \otimes I)\boldsymbol{k} \quad \Longleftrightarrow \quad \boldsymbol{k} = (A_0^{-1} \otimes I)\boldsymbol{w} \tag{6}$$

and we first solve the nonlinear problem for $\boldsymbol{w}$, then solve a linear problem to obtain $\boldsymbol{k}$ from $\boldsymbol{w}$ (well, we'd actually obtain $(\boldsymbol{b}_0^\top \otimes I)\boldsymbol{k} = (\boldsymbol{b}_0^\top A_0^{-1} \otimes I)\boldsymbol{w} = (\boldsymbol{d}_0^\top \otimes I)\boldsymbol{w}$ since this is what we really need). Now the nonlinear system becomes

$$F(\boldsymbol{w}) = (A_0^{-1} \otimes M)\boldsymbol{w} - \begin{bmatrix} \mathcal{N}(\boldsymbol{u}_n + \delta t \boldsymbol{w}_1, t_n + c_1\delta t) \\ \vdots \\ \mathcal{N}(\boldsymbol{u}_n + \delta t \boldsymbol{w}_s, t_n + c_s\delta t) \end{bmatrix} = 0. \tag{7}$$

Remark 1 (Nonlinear decoupling). What's interesting here is that the components of $\boldsymbol{w}$ are now only linearly coupled to each other (via the $A_0^{-1} \otimes M$ term), where with the previous formulation the components of $\boldsymbol{k}$ were nonlinearly coupled (every equation involves $\mathcal{N}$ being evaluated with $(\boldsymbol{k}_i)_{i=1}^s$). This may be useful for a Picard iteration or something, since now the s systems can maybe be solved in parallel.

The Jacobian is

$$F'(\boldsymbol{w}) = A_0^{-1} \otimes M - \delta t \begin{bmatrix} \mathcal{N}_1' & & \\ & \ddots & \\ & & \mathcal{N}_s' \end{bmatrix} \tag{8}$$

where

$$\mathcal{N}_i' \equiv \mathcal{N}'\left(\boldsymbol{u}_n + \delta t \boldsymbol{w}_i^0, t_n + c_i \delta t\right). \tag{9}$$

The linear systems we solve during the Newton-like solve take the form

$$\left[R_0 \otimes M - (Q_0^\top \otimes I)(\delta t N')(Q_0 \otimes I)\right](Q_0^\top \otimes I)\boldsymbol{\delta w} = -(Q_0^\top \otimes I)F(\boldsymbol{w}), \tag{10}$$

by using the Schur decomposition of $A_0^{-1} = Q_0 R_0 Q_0^\top$ with block upper triangular R_0. And,

$$N' \approx \begin{bmatrix} \mathcal{N}_1' & & \\ & \ddots & \\ & & \mathcal{N}_s' \end{bmatrix}. \tag{11}$$

2. A simplified Newton algorithm. We take

$$N' = \begin{bmatrix} \mathcal{N}_s' & & \\ & \ddots & \\ & & \mathcal{N}_s' \end{bmatrix} = I \otimes \mathcal{N}_s', \tag{12}$$

such that the matrix

$$\left[R_0 \otimes M - (Q_0^\top \otimes I)(\delta t N')(Q_0 \otimes I)\right] = \left[R_0 \otimes M - \delta t N'\right]$$

is block upper triangular, and can be inverted via backward substitution. During this solve, we have to invert the diagonal blocks, which take the form

$$1 \times 1 \text{ systems:} \quad \zeta_j M - \delta t \mathcal{N}_s', \tag{13}$$

and

$$2 \times 2 \text{ systems:} \quad \begin{bmatrix} \eta_j M - \delta t \mathcal{N}_s' & \phi M \\ -\frac{\beta_j^2}{\phi} M & \eta_j M - \delta t \mathcal{N}_s' \end{bmatrix} \tag{14}$$

This is done with preconditioned GMRES, where for the 1×1 system, a single AMG iteration is applied to $\zeta_j M - \delta t \mathcal{N}_s'$. And for the 2×2 systems the preconditioner is

$$\begin{bmatrix} \eta_j M - \delta t \mathcal{N}_s' & 0 \\ -\frac{\beta_j^2}{\phi} M & \eta_j M - \delta t \mathcal{N}_s' \end{bmatrix}^{-1}, \tag{15}$$

with the action of $(\eta_j M - \delta t \mathcal{N}_s')^{-1}$ approximated via a single AMG iteration.

We have some flexibility in terms of updating $\mathcal{N}_s'$. Some options:

1. Update at the start of every Newton solve (see Figure 1, left panel)

68 2. Update every Newton iteration (See Figure 1, right panel).

```
   --out-directory data1/IRK110_dt4_d2_ex1
   --save-sol-data 1
Time-step 1 of 32 (t=0-->t=0.0625)
Newton iteration  0 : ||r|| = 148.44
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 16
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 13
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 29
Newton iteration  1 : ||r|| = 2.89649, ||r||/||r_0|| = 0.0195129
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 16
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 12
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 30
Newton iteration  2 : ||r|| = 0.0947727, ||r||/||r_0|| = 0.000638457
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 13
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 11
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 28
Newton iteration  3 : ||r|| = 0.00287618, ||r||/||r_0|| = 1.9376e-05
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 10
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 9
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 24
Newton iteration  4 : ||r|| = 8.26524e-05, ||r||/||r_0|| = 5.56806e-07
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 8
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 7
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 20
Newton iteration  5 : ||r|| = 2.26462e-06, ||r||/||r_0|| = 1.52561e-08
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 6
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 6
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 16
Newton iteration  6 : ||r|| = 5.94478e-08, ||r||/||r_0|| = 4.00483e-10
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 4
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 4
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 12
Newton iteration  7 : ||r|| = 1.50257e-09, ||r||/||r_0|| = 1.01224e-11
```

```
   --out-directory data1/IRK110_dt4_d2_ex1
   --save-sol-data 1
Time-step 1 of 32 (t=0-->t=0.0625)
Newton iteration  0 : ||r|| = 148.44
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 16
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 13
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 29
Newton iteration  1 : ||r|| = 2.89649, ||r||/||r_0|| = 0.0195129
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 15
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 12
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 30
Newton iteration  2 : ||r|| = 0.0120248, ||r||/||r_0|| = 8.10075e-05
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 13
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 10
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 26
Newton iteration  3 : ||r|| = 0.000139385, ||r||/||r_0|| = 9.38994e-07
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 10
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 8
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 21
Newton iteration  4 : ||r|| = 1.98368e-06, ||r||/||r_0|| = 1.33635e-08
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 8
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 6
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 17
Newton iteration  5 : ||r|| = 3.06303e-08, ||r||/||r_0|| = 2.06348e-10
  ---Backward solve---
    Block solve 1 of 3: 2x2 block  -->  GMRES: Number of iterations: 5
    Block solve 2 of 3: 1x1 block  -->  GMRES: Number of iterations: 4
    Block solve 3 of 3: 2x2 block  -->  GMRES: Number of iterations: 12
Newton iteration  6 : ||r|| = 4.83776e-10, ||r||/||r_0|| = 3.25906e-12
```

FIG. 1. *Simplified Newton convergence and that of its linear solver; first time step of finest resolution Gauss(10) solve from Figure 2.* ***Left:*** *Jacobian fixed throughout Newton solve.* ***Right:*** *Updating approximate Jacobian each Newton iteration. Notice convergence on the right is slightly faster.*

2.1. Numerical results. Numerically approximate the solution of the 2D nonlinear advection-diffusion problem,

$$u_t + 0.85u_x^2 + 1.0u_y^2 = 0.3u_{xx} + 0.25u_{yy} + s(x,y,t), \quad (x,y,t) \in (-1,1)^2 \times (0,2) \tag{16}$$

Some remarks:

- Periodic spatial boundaries. All tests use 4 processors.
- A pth-order IRK scheme is coupled with pth-order central finite differences in space (or p+1st-order if p is odd, as for Radau IIA and some SDIRK schemes).
- Set $\delta t = 2\delta x$
- Tests in Figure 2 indicate that the code is implemented and working properly since theoretically predicated convergence rates are obtained in all cases.
- Solver settings:
  - Newton:
    1. Approximate Jacobian updated at beginning of each time step
    2. `abs tol=rel tol=1e-10`
  - GMRES: `abs tol=rel tol=1e-13`

  These settings don't really make much sense because they lead to significant over-solving most of the time... E.g., really should be setting Newton tolerance proportionate to δt^p and GMRES tolerances adaptively set within Newton iteration to reflect the size of the nonlinear residual and the convergence rate of Newton.
- In terms of work done per accuracy, it seems like Gauss schemes are best, and Radau IIA schemes come in a close second, and that Lobatto IIIC schemes aren't even close. See Figure 3. But, I'm not quite sure how much to take away from this plot because tolerances are not set up very well...
- 4th-order SDIRK schemes:
  - A-stable: 3-stages, the diagonal entries on the upper triangular Schur matrix are $R_{ii} \approx 1.07$ (we solve linear systems like $R_{ii}I - \delta t\mathcal{N}'$)
  - L-stable: 5-stages, the diagonal entries on the upper triangular Schur matrix are $R(i,i)_{ii} \approx 4$. So the linear systems here have a larger identity perturbation c.f. the A-stable scheme, which is why they are easier to solve (they require fewer AMG iterations).

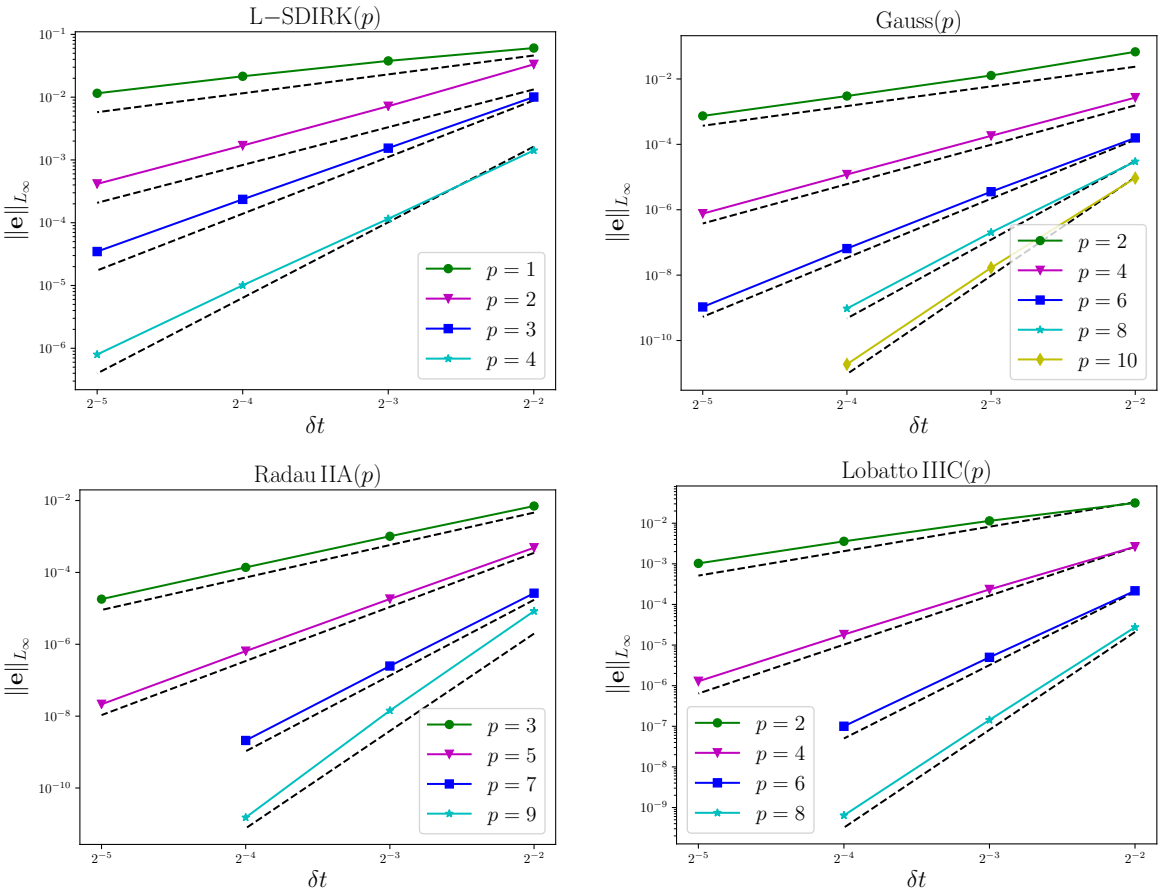

FIG. 2. *L_∞ errors measured at $t \approx 2$. Theoretically predicted convergence rates of $\mathcal{O}(p)$ are shown as dashed black lines. $\mathcal{O}(p)$ IRK schemes are paired with $\mathcal{O}(p)$ spatial discretizations (or $\mathcal{O}(p+1)$ if p is odd), and $\delta t = 2\delta x$.*

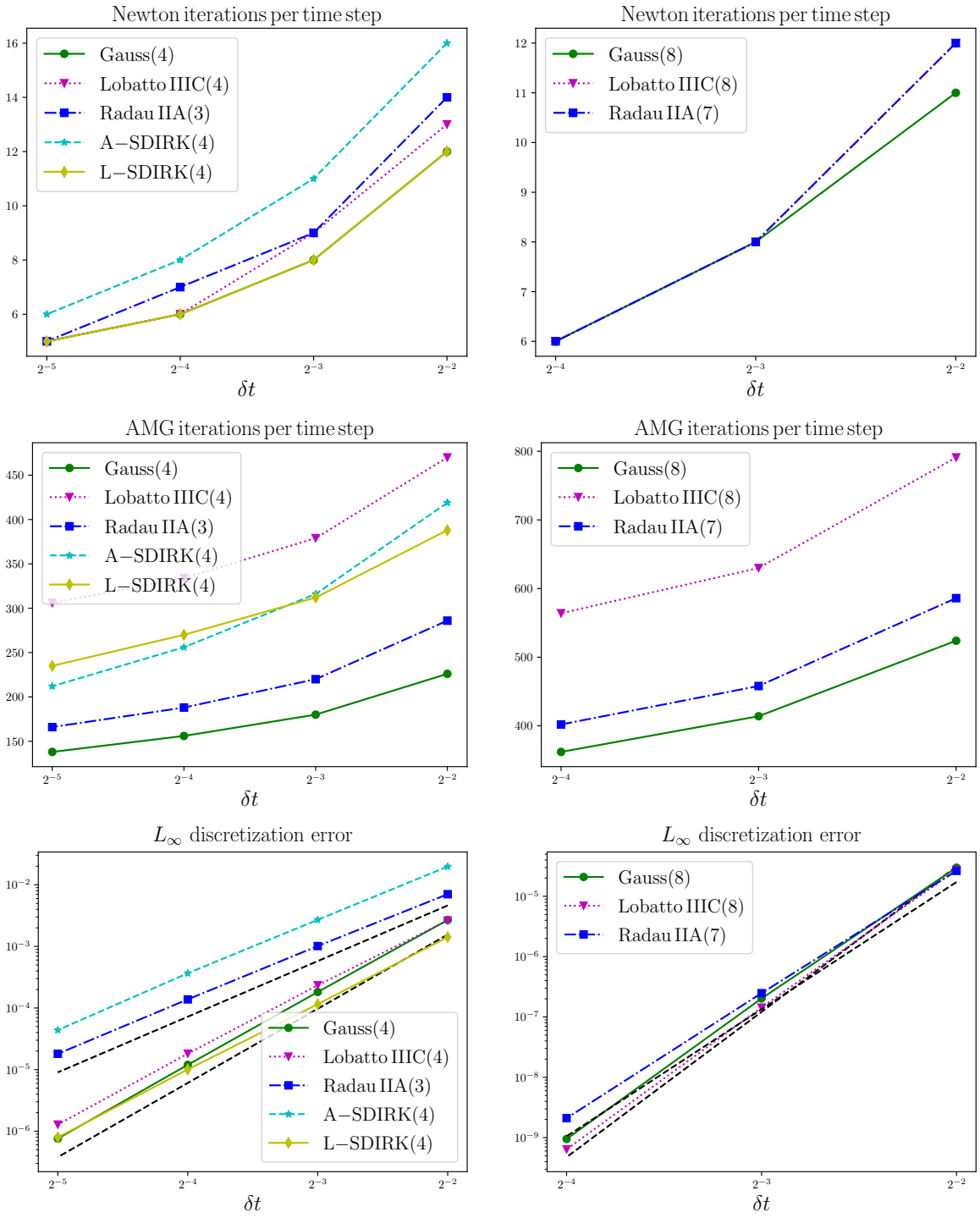

FIG. 3. *Average iterations per time step (data associated with convergence plots in Figure 2).* ***Left:*** *4th(ish)-order schemes.* ***Right:*** *8th(ish)-order schemes.* ***Top:*** *Newton iterations.* ***Middle:*** *AMG iterations.* ***Bottom:*** *Discretization errors. The systems that AMG is applied to have dimension* $n = (16^2, 32^2, 64^2, 128^2)$ *for* $\delta t = (2^{-2}, 2^{-3}, 2^{-4}, 2^{-5})$. *Not really sure why so many Newton iterations decrease with resolution... See also Figure 4.. In terms of work done per accuracy, it seems Gauss < Radau IIA ≪ Lobatto IIIC. Note also for this particular problem, 4th-order Gauss does significantly better than both L-stable and A-stable 4th-order SDIRK schemes. Notice that the error constant for A-SDIRK(4) is much larger, and convergence appears only ≈3rd-order (Figure 4 shows that eventually it does reach 4th-order however).*

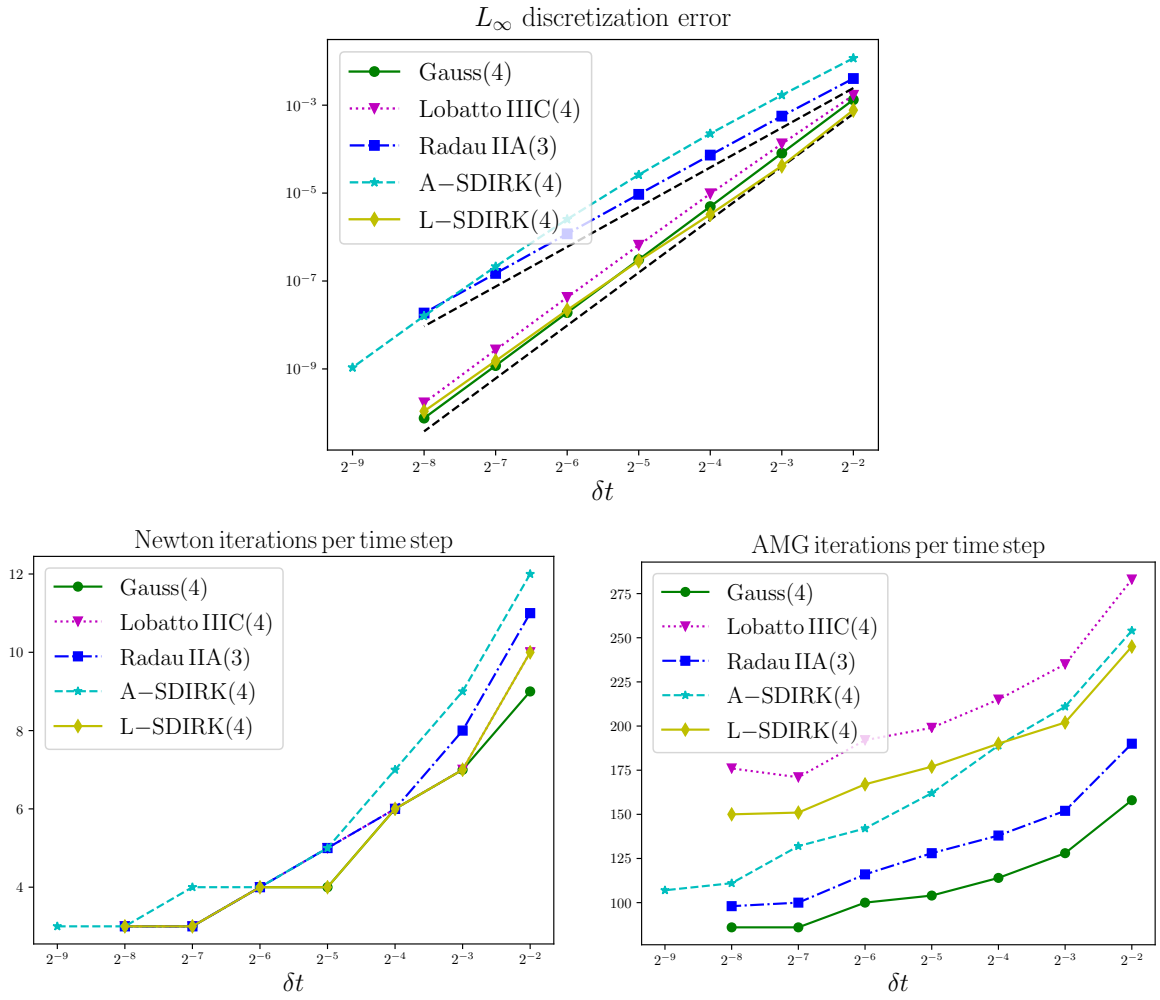

FIG. 4. *Average iterations per time step for 4th(ish)-order discretizations of the* ***1D problem****,* $u_t + 0.85u_x^2 = 0.3u_{xx} + s(x,t)$*. Notice that A-SDIRK(4) does appear to eventually achieve 4th-order convergence, but the error is significantly larger than that of the other 4th-order schemes. Thus, Gauss(4) outperforms A-SDIRK(4) in terms of AMG iterations per accuracy.*

3. A quasi-Newton algorithm. The real Jacobian is

$$F'(\boldsymbol{w}) = A_0^{-1} \otimes M - \delta t \begin{bmatrix} \mathcal{N}_1' & & \\ & \ddots & \\ & & \mathcal{N}_s' \end{bmatrix} \tag{17}$$

where

$$\mathcal{N}_i' \equiv \mathcal{N}'\left(\boldsymbol{u}_n + \delta t \boldsymbol{w}_i^0, t_n + c_i \delta t\right). \tag{18}$$

Before inverting the Jacobian, we scale it by $(Q_0^\top \otimes I)$ from the left and $(Q_0 \otimes I)$ from the right, making use of the Schur decomposition of $A_0^{-1} = Q_0 R_0 Q_0^\top$,

$$\widehat{F}'(\boldsymbol{w}) \coloneqq (Q_0^\top \otimes I) F'(\boldsymbol{w}) (Q_0 \otimes I), \tag{19}$$

$$= R_0 \otimes M - \delta t (Q_0^\top \otimes I) \begin{bmatrix} \mathcal{N}_1' & & \\ & \ddots & \\ & & \mathcal{N}_s' \end{bmatrix} (Q_0 \otimes I). \tag{20}$$

Now we write the second matrix in the form

$$\widehat{P} \coloneqq (Q_0^\top \otimes I) \begin{bmatrix} \mathcal{N}_1' & & \\ & \ddots & \\ & & \mathcal{N}_s' \end{bmatrix} (Q_0 \otimes I) = \begin{bmatrix} \boldsymbol{z}_{1,1}^\top \boldsymbol{\mathcal{N}}' & \cdots & \boldsymbol{z}_{1,s}^\top \boldsymbol{\mathcal{N}}' \\ \vdots & & \vdots \\ \boldsymbol{z}_{s,1}^\top \boldsymbol{\mathcal{N}}' & \cdots & \boldsymbol{z}_{s,s}^\top \boldsymbol{\mathcal{N}}' \end{bmatrix}, \tag{21}$$

where $\boldsymbol{z}_{k,\ell}^\top = \Big((Q_0^\top)_{k,1}(Q_0)_{1,\ell}, \ldots, (Q_0^\top)_{k,s}(Q_0)_{s,\ell} \Big) \in \mathbb{R}^s$ is a row vector, and $\boldsymbol{\mathcal{N}}' = (\mathcal{N}_1', \ldots, \mathcal{N}_s')$ is a column vector of the Jacobians of $\mathcal{N}_i$, meaning that

$$\boldsymbol{z}_{k,\ell}^\top \boldsymbol{\mathcal{N}}' = \sum_{i=1}^{s} (z_{k,\ell})_i \mathcal{N}_i'. \tag{22}$$

Note that by the orthogonality of Q_0,

$$\boldsymbol{z}_{k,\ell}^\top \mathbf{1} = (Q_0^\top Q_0)_{k,\ell} = \begin{cases} 1, & k = \ell, \\ 0, & k \neq \ell, \end{cases} \tag{23}$$

which tends to suggest that the block diagonal entries of P are most important (e.g., think about the case in which the matrices in $\boldsymbol{\mathcal{N}}'$ approximately have the same eigenvectors and eigenvalues).

In the quasi-Newton algorithm, we choose some block upper triangular $\widetilde{P} \approx \widehat{P}$, which has a (block) sparsity pattern contained within that of $R_0 \otimes M$. The point of making it block upper triangular is that we can invert *exactly* the resulting approximate Jacobian via back substitution. So, during the Newton solves we "exactly" invert the approximate Jacobian,

$$\widetilde{F}'(\boldsymbol{w}) \coloneqq R_0 \otimes M - \delta t \tilde{P} \approx R_0 \otimes M - \delta t P =: \widehat{F}'(\boldsymbol{w}). \tag{24}$$

There are a few different options for choosing $\widetilde{P} \approx \widehat{P}$:

0. **(biggest approximation)** Truncate $\widehat{P}$ to be block diagonal and them lump the coefficients of $\boldsymbol{z}_{ii}$ to the largest one so that each diagonal block of $\widetilde{P}$ contains only one matrix from $\boldsymbol{\mathcal{N}}'$.
1. **(medium approximation)** Truncate $\widehat{P}$ to be block diagonal (can include the off-diagonal entries in 2×2 diagonal blocks, or not)
2. **(smallest approximation)** Truncate $\widehat{P}$ to be block upper triangular. This means doing a lot of matrix-vector products (just how much work this is depends on the implementation, I think; can maybe try and exploit the underlying Kronecker-product structure to reduce this), but potentially improving convergence of Newton's method

3.1. Numerical results. Am having trouble coming up with a good test problem... Struggling to find a meaningful nonlinear, diffusion-dominated PDE?

- Current problem from previous example is no good because as mesh is refined, discrete diffusion $\gg$ discrete advection, so problem looks more and more linear (this is why in Figure 3 and 4 the Newton iterations approach 1 as the spatial/temporal mesh is refined).

- Also, for a fixed mesh resolution, as I time step, the number of Newton iterations per time step decreases. Why?! This makes it too hard to make comparisons

So... as an initial test, measure number of iterations for one time step ($t = 0 \to 0.3125$) of

$$u_t + 0.85u_x^2 + 1.0u_y^2 = 0.3u_{xx} + 0.25u_{yy} + s(x, y, t), \tag{25}$$

using $\delta t = 10\delta x$ (this makes problem much harder for Newton, and highlights differences between simple- and quasi-Newton solvers), and $n_x \times n_y = 64 \times 64$ (4096 spatial DOFs). See Table 1.

TABLE 1

Number of Newton and AMG iterations. "Simple" = simplified Newton, "Quasi(0),(1),(2)" = quasi Newton with different approximations $\widetilde{P}$ with increasing cost per Newton iteration. Interpret results with a grain of salt because there are hidden costs for the different Newton methods including matrix-vector products, assembling of Jacobians, assembling of AMG preconditioners.

	Newton iterations/AMG iterations			
Discretization	Simple	Quasi(0)	Quasi(1)	Quasi(2)
ASDIRK(4)+C4	**23**/519	**4**/118	**4**/118	**4**/118
LSDIRK(4)+C4	**19**/570	**4**/169	**4**/169	**4**/169
Gauss(4)+C4	**17**/322	**4**/92	**4**/92	**4**/92
Radau IIA(5)+C4	**19**/646	**9**/330	**9**/329	**6**/249
Gauss(6)+C6	**17**/531	**8**/277	**8**/278	**7**/250
Lob IIIC(6)+C6	**19**/927	**13**/712	**11**/620	**7**/436
Radau IIA(7)+C6	**18**/868	**9**/464	**9**/462	**7**/386
Gauss 8(8)+C8	**17**/794	**8**/434	**8**/434	**6**/364
Lob IIIC 8(8)+C8	**18**/1211	**13**/922	**12**/817	**9**/672
Radau IIA(9)+C8	**17**/1090	**9**/596	**9**/582	**7**/504

- Note SDIRK and Gauss(4) have sparse Q_0, so the different quasi-Newton variants are the same (hence constant Newton/AMG iterations)
- Quasi-Newton converges significantly faster than Simple-Newton

TABLE 2

Number of Newton and AMG iterations. "Simple" = simplified Newton, "Quasi(0),(1),(2)" = quasi Newton with different approximations $\widetilde{P}$ with increasing cost per Newton iteration using larger time step $\delta t = 10 \times h$.

	Newton iterations/AMG iterations($\gamma = \eta$, $\gamma = \eta + \beta^2/\eta$)			
Discretization	Simple	Quasi(0)	Quasi(1)	Quasi(2)
Gauss(4)+C4	**17**/(296,268)	**4**/(84,76)	–	–
Lob IIIC(4)+C4	**20**/(689,573)	**17**/(555,447)	**13**/(453,373)	**10**/(421,363)
Radau IIA(5)+C4	**19**/(604,490)	**9**/(300,246)	**9**/(296,244)	**6**/(244,208)
Gauss 8(8)+C8	**17**/(794,656)	**8**/(414,344)	**8**/(412,342)	**6**/(364,310)
Lob IIIC 8(8)+C8	**18**/(1211,957)	**13**/(880,706)	**12**/(803,645)	**9**/(672,550)
Radau IIA(9)+C8	**17**/(1090,844)	**9**/(589,465)	**9**/(578,458)	**7**/(504,412)

TABLE 3

Number of Newton and AMG iterations. "Simple" = simplified Newton, "Quasi(0),(1),(2)" = quasi Newton with different approximations $\widetilde{P}$ with increasing cost per Newton iteration using smaller time step $\delta t = 1 \times h$.

	Newton iterations/AMG iterations($\gamma = \eta$, $\gamma = \eta + \beta^2/\eta$)			
Discretization	Simple	Quasi(0)	Quasi(1)	Quasi(2)
Gauss(4)+C4	**5**/(104,84)	**3**/(68,54)	–	–
Lob IIIC(4)+C4	**6**/(265,168)	**5**/(232,152)	**5**/(235,151)	**5**/(225,145)
Radau IIA(5)+C4	**5**/(199,139)	**4**/(165,114)	**4**/(165,114)	**4**/(169,116)
Gauss 8(8)+C8	**5**/(254,200)	**4**/(222,176)	**4**/(222,176)	**4**/(226,176)
Lob IIIC 8(8)+C8	**5**/(424,280)	**5**/(403,267)	**5**/(398,264)	**4**/(357,233)
Radau IIA(9)+C8	**5**/(365,261)	**4**/(301,215)	**4**/(301,215)	**4**/(299,219)

Method		
Gauss(8)	β^2/η^2	1.59 / 0.09
	$\gamma = \eta$ –	20 / 11
	$\gamma = \gamma_*$	13 / 11
Radau IIA(9)	β^2/η^2	0 / 3.20 / 0.32
	$\gamma = \eta$	8 / 24 / 13
	$\gamma = \gamma_*$	8 / 14/ 11
Lobatto IIIC(8)	β^2/η^2	0 / 4.88 / 0.38
	$\gamma = \eta$	8 / 29 / 14
	$\gamma = \gamma_*$	8 / 15 / 12

TABLE 4

Number of GMRES iterations for each stage when solving $u_t = \Delta u + s$, with a single time step of $\delta t = h = 1/256$. Different values of γ are used in the preconditoner, as indicated.

3.2. Newer results.