1 Fully implicit Runge-Kutta

Consider the method-of-lines approach to solving PDEs, where we discretize in space and arrive at a system of ODEs in time,

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

where M is a mass matrix, and $\mathcal{N} \in \mathbb{R}^{M \times M}$ a discrete, time-dependent, nonlinear operator depending on t and $\mathbf{u}$. For nonlinear PDEs, $\mathcal{N}$ is linearized using, for example, a Jacobian or a Picard linearization of the underlying PDE. Let us also consider the specific cases of a linear time-dependent PDE, say $\mathcal{L}(t)$, and a linear time-independent PDE, say $\mathcal{S}$. Then consider time propagation using an s-stage Runge-Kutta scheme, characterized by the Butcher tableaux

$$\begin{array}{c|c} \mathbf{c}_0 & A_0 \\ \hline & \mathbf{b}_0^T \end{array},$$

with Runge-Kutta matrix $A_0 = (a_{ij})$, weight vector $\mathbf{b}_0^T = (b_1, \ldots, b_s)^T$, and nodes $\mathbf{c}_0 = (c_0, \ldots, c_s)$.

Runge-Kutta methods update the solution using a sum over stage vectors,

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

$$\mathbf{k}_i =$$

As noted in Will's paper, once linearizing the nonlinear operator (using e.g., Picard or Newton's), solving for the stages $\mathbf{k}$ can be expressed as a block linear system,

$$\left(\begin{bmatrix} M & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M \end{bmatrix} + \delta t \begin{bmatrix} a_{11}\mathcal{L}_1 & \ldots & a_{1s}\mathcal{L}_s \\ \vdots & \ddots & \vdots \\ a_{s1}\mathcal{L}_s & \ldots & a_{ss}\mathcal{L}_s \end{bmatrix} \right) \begin{bmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_s \end{bmatrix}. \tag{2}$$ {eq:k0}

For DIRK methods, A_0 is lower triangular and (2) is easily inverted by inverting the diagonal blocks, $M - \delta t a_{ii}\mathcal{L}_i$ for $i = 1, .., s$. However, SDIRK methods have at most stage-order one, and ESDIRK methods have at most stage-order two. One interesting phenomenon of using RK methods with the method-of-lines approach to solve PDEs is order reduction, where error in spatial boundary conditions for intermediate RK stages limits the global accuracy. The concept is not fully understood, nor are there many practical fixes for PDEs in higher than one dimension. However, for nonlinear PDEs, it is typically the case that the global order of accuracy is limited to $\approx \min\{p, q+1\}$, for integration order p and stage-order q. Obviously if we actually want high-order integration in time, SDIRK and ESDIRK methods are limiting, and fully implicit high-order RK methods are desirable. Unfortunately, solving the fully implicit stage matrix in (2) is often much more difficult when it is not lower triangular. Here we consider new block preconditioning techniques to facilitate this.

The RK stage system can be reformulated as

$$\left(A_0^{-1} \otimes M + \delta t \begin{bmatrix} \mathcal{L}_1 & & \\ & \ddots & \\ & & \mathcal{L}_s \end{bmatrix} \right) (A_0 \otimes I) \begin{bmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_s \end{bmatrix}.$$

For ease of notation, let us scale both sides of the system by a block-diagonal mass matrix and, excusing the slight abuse of notation, let $\mathcal{L}_i \mapsto \delta t M^{-1}\mathcal{L}_i$, $i = 1, .., s$. Note the time step is now included in $\mathcal{L}_i$. Because $\mathcal{L}_i$ is time-dependent, it is possible that δt is also time-dependent. Now let α_{ij} denote the ij-element of A_0^{-1} (assuming A_0 is invertible). Then, solving (2) can be effectively reduced to inverting the operator

$$A_0^{-1} \otimes I + \begin{bmatrix} \mathcal{L}_1 & & \\ & \ddots & \\ & & \mathcal{L}_s \end{bmatrix} = \begin{bmatrix} \alpha_{11}I + \mathcal{L}_1 & \alpha_{12}I & \ldots & \alpha_{1s}I \\ \alpha_{21}I & \alpha_{22}I + \mathcal{L}_2 & & \alpha_{2s}I \\ \ddots & & \ddots & \vdots \\ \alpha_{s1}I & \ldots & \alpha_{s(s-1)}I & \alpha_{ss}I + \mathcal{L}_s \end{bmatrix}. \tag{3}$$ {eq:k1}

Note, there are a number of methods with one explicit stage preceded or followed by several fully implicit and coupled stages. These will be of particular interest. In such cases, A_0 is not invertible, but the explicit stage can be eliminated from the system. The remaining operator can then be reformulated as above, and the inverse that must be applied takes the form of (3) but based on a principle submatrix of A_0.

1.1 Two stages

Consider the simple case of two stages. Then we need to invert the block linear system $\mathcal{M}_2\mathbf{s} = \mathbf{r}$,

$$\begin{bmatrix} \alpha_{11}I + \mathcal{L}_1 & \alpha_{12}I \\ \alpha_{21}I & \alpha_{22}I + \mathcal{L}_2 \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \tag{4}$$ {eq:Mnt}

In the context of 2×2 block operators, the key to inverting the matrix is inverting one of the Schur complements. Define the matrix polynomials

$$\begin{aligned} P_{\mathcal{L}} &:= (\alpha_{11}I + \mathcal{L}_1)(\alpha_{22}I + \mathcal{L}_2) - \alpha_{12}\alpha_{21}I, \\ Q_{\mathcal{L}} &:= (\alpha_{22}I + \mathcal{L}_2)(\alpha_{11}I + \mathcal{L}_1) - \alpha_{12}\alpha_{21}I, \end{aligned}$$

and consider both Schur complements,

$$\begin{aligned} S_{22} &= \alpha_{22}I + \mathcal{L}_2 - \alpha_{12}\alpha_{21}(\alpha_{11}I + \mathcal{L})^{-1} \\ &= \left[(\alpha_{22}I + \mathcal{L}_2)(\alpha_{11}I + \mathcal{L}_1) - \alpha_{12}\alpha_{21}I\right](\alpha_{11}I + \mathcal{L}_1)^{-1} \\ &= Q_{\mathcal{L}}(\alpha_{11}I + \mathcal{L}_1) \\ &= (\alpha_{11}I + \mathcal{L}_1)P_{\mathcal{L}} \\ S_{11} &= P_{\mathcal{L}}(\alpha_{22}I + \mathcal{L}_2) \\ &= (\alpha_{22}I + \mathcal{L}_2)Q_{\mathcal{L}}. \end{aligned}$$

Note that

$$\begin{aligned} P_{\mathcal{L}} &= (\alpha_{22}I + \mathcal{L}_2)Q_{\mathcal{L}}(\alpha_{22}I + \mathcal{L}_2)^{-1} \\ &= (\alpha_{11}I + \mathcal{L}_1)^{-1}Q_{\mathcal{L}}(\alpha_{11}I + \mathcal{L}_1). \end{aligned}$$

Now, we can write $\mathcal{M}_2^{-1}$ in terms of the Schur complements in the following form(s):

$$\begin{aligned} \begin{bmatrix} \alpha_{11}I + \mathcal{L}_1 & \alpha_{12}I \\ \alpha_{21}I & \alpha_{22}I + \mathcal{L}_2 \end{bmatrix}^{-1} &= \begin{bmatrix} (\alpha_{22}I + \mathcal{L}_2)P_{\mathcal{L}}^{-1} & -\alpha_{12}Q_{\mathcal{L}}^{-1} \\ -\alpha_{21}P_{\mathcal{L}}^{-1} & (\alpha_{11}I + \mathcal{L}_1)Q_{\mathcal{L}}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \alpha_{22}I + \mathcal{L}_2 & -\alpha_{12}I \\ -\alpha_{21}I & \alpha_{11}I + \mathcal{L}_1 \end{bmatrix} \begin{bmatrix} P_{\mathcal{L}}^{-1} & \mathbf{0} \\ \mathbf{0} & Q_{\mathcal{L}}^{-1} \end{bmatrix}. \end{aligned} \tag{5}$$ {eq:Minv}

Note, this more or less exactly takes the form of a scalar 2×2 matrix inverse, replacing the $1/det$ with the right scaling by the polynomial inverses..

1.2 The time-independent case

Now suppose $\mathcal{L}_1 = \mathcal{L}_2$. Then then $P_{\mathcal{L}} = Q_{\mathcal{L}}$ and $S_{11} = S_{22}$. Then, $\mathcal{M}_2^{-1}$ (5) can be applied through two applications of $P_{\mathcal{L}}^{-1}$, along with some additional mat-vecs and vector addition. Note that for $\mathcal{L}_1 = \mathcal{L}_2$, $P_{\mathcal{L}}$ is a quadratic polynomial in $\mathcal{L}$, which can be solved in two steps using a polynomial preconditioning based on the roots of the polynomial $P_2(x) := x^2 + (\alpha_{11} + \alpha_{22})x + (\alpha_{11}\alpha_{22} - \alpha_{12}\alpha_{21})$. Moreover, the roots of $P_2(x)$ are exactly the eigenvalues of $-A_0^{-1}$, which is minus one over the eigenvalues of A_0. Denote these eigenvalues $\{\lambda_0, \lambda_1\}$, where

$$\lambda_1, \lambda_2 = -\frac{1}{2}\left(\alpha_{11} + \alpha_{22} \pm \sqrt{(\alpha_{11} + \alpha_{22})^2 - 4(\alpha_{11}\alpha_{22} - \alpha_{12}\alpha_{21})}\right).$$

Then a two-step fixed point iteration solves the problem $P_2(\mathcal{L})\mathbf{s} = \mathbf{g}$ with zero initial guess via

$$\begin{aligned} \mathbf{s}_1 &= \lambda_1\mathcal{L}^{-1}\mathbf{g}, \\ \mathbf{s}_2 &= \mathbf{s}_1 + \lambda_2\mathcal{L}^{-1}(\mathbf{g} - P_2(\mathcal{L})\mathbf{s}_1). \end{aligned}$$

Plugging $\mathbf{s}_1$ into $\mathbf{s}_2$ and pulling $\mathbf{g}$ out the right-hand side yields the closed form

$$\begin{aligned} P_2(\mathcal{L})^{-1} &= (\lambda_1 + \lambda_2)\mathcal{L}^{-1} - \lambda_1\lambda_2\mathcal{L}^{-1}P_2(\mathcal{L})\mathcal{L}^{-1} \\ &= -\lambda_1\lambda_2(\alpha_{11}\alpha_{22} - \alpha_{12}\alpha_{21})\mathcal{L}^{-2} + [\lambda_1 + \lambda_2 - \lambda_1\lambda_2(\alpha_{11} + \alpha_{22})]\mathcal{L}^{-1} - \lambda_1\lambda_2 \\ &= c_a\mathcal{L}^{-2} + c_b\mathcal{L}^{-1} + c_c, \end{aligned}$$

$$
\begin{aligned}
c_a &:= -(\alpha_{12}\alpha_{21} - \alpha_{11}\alpha_{22})^2, \\
c_b &:= -(\alpha_{11} + \alpha_{22})(1 - \alpha_{12}\alpha_{21} + \alpha_{11}\alpha_{22}), \\
c_c &:= \alpha_{12}\alpha_{21} - \alpha_{11}\alpha_{22}.
\end{aligned}
$$

It is not clear if this closed form is useful, but we derive it just in case..

A few thoughts:

- I think this is a robust and detemrinistic algorithm. However, for time-dependent problems, it is possible $\mathcal{L}^{-1}$ is not well-posed (for example, a cycle in a non-diffusive advection that only makes sense with an identity perturbation representing time). Probably we want to go back and precondition $P_2(\mathcal{L})$ as a polynomial preconditioning of $(\alpha_{11}I + \mathcal{L})$. In principle I think there might be such a preconditioning, because S_{22} could also be written as a polynomial of $(\alpha_{11}I + \mathcal{L})$? This needs to be looked at closer, and would probably be the last important piece. If we could do that, we could also use the same solver for each step of the algorithm, rather than needing a different one for different steps..
- I'd hoped to see some kind of closed form for a larger polynomial preconditioning that we are performing on $\mathcal{M}$. It is still not apparent, but would be helpful for generalizing to the multistage setting..
- What if we get complex eigenvalues? This would work in theory, but might be suboptimal in practice...

1.3 Time-independent 3×3

Try to construct the inverse of the 3×3 operator in the time-independent setting by thinking of the matrix as an operator over a commutative ring of invertible matrices $\{C_1 I, C_2 \mathcal{L}\}$. Then, try to extend this to the time-dependent setting..

For nonsingular scalar 3×3 matrix A, the inverse is given by

$$
A^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^{-1} = \frac{1}{\det(A)} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}, \tag{6}
$$

{eq:3inv}

with elements defined by

$$
\begin{aligned}
A_{11} &= (a_{22}a_{33} - a_{23}a_{32}), & A_{12} &= -(a_{12}a_{33} - a_{13}a_{32}), & A_{13} &= (a_{12}a_{23} - a_{13}a_{22}), \\
A_{21} &= -(a_{21}a_{33} - a_{23}a_{31}), & A_{22} &= (a_{11}a_{33} - a_{13}a_{31}), & A_{23} &= -(a_{11}a_{23} - a_{13}a_{21}), \\
A_{31} &= (a_{21}a_{32} - a_{22}a_{31}), & A_{32} &= -(a_{11}a_{32} - a_{12}a_{31}), & A_{33} &= (a_{11}a_{22} - a_{12}a_{21}).
\end{aligned}
$$

Note, the rule of Sarrus yields the determinant as $\det(A) = a_{11}A_{11} + a_{12}A_{21} + a_{13}A_{31}$.

In our case, consider

$$
\mathcal{M}_3 := \begin{bmatrix} (\alpha_{11}I + \mathcal{L}) & \alpha_{12}I & \alpha_{13}I \\ \alpha_{21}I & (\alpha_{22}I + \mathcal{L}) & \alpha_{23}I \\ \alpha_{31}I & \alpha_{32}I & (\alpha_{33}I + \mathcal{L}) \end{bmatrix}. \tag{7}
$$

{eq:Mnt}

Define $\mathcal{N}_3$ as a block 3×3 matrix with entries of A^{-1} as in (6), excluding the $1/\det(A)$. Plugging in, we have entries of $\mathcal{N}_3$ given by

$$
\begin{aligned}
A_{11} &= (\alpha_{22}I + \mathcal{L})(\alpha_{33}I + \mathcal{L}) - \alpha_{23}\alpha_{32}I, \\
A_{12} &= -\alpha_{12}(\alpha_{33}I + \mathcal{L}) + \alpha_{13}\alpha_{32}I, \\
A_{13} &= \alpha_{12}\alpha_{23}I - \alpha_{13}(\alpha_{22}I + \mathcal{L}), \\
A_{21} &= -\alpha_{21}(\alpha_{33}I + \mathcal{L}) + \alpha_{23}\alpha_{31}I, \\
A_{22} &= (\alpha_{11}I + \mathcal{L})(\alpha_{33}I + \mathcal{L}) - \alpha_{13}\alpha_{31}I, \\
A_{23} &= -\alpha_{23}(\alpha_{11}I + \mathcal{L}) + \alpha_{13}\alpha_{21}I, \\
A_{31} &= \alpha_{21}\alpha_{32}I - \alpha_{31}(\alpha_{22}I + \mathcal{L}), \\
A_{32} &= -\alpha_{32}(\alpha_{11}I + \mathcal{L}) + \alpha_{12}\alpha_{31}I, \\
A_{33} &= (\alpha_{11}I + \mathcal{L})(\alpha_{22}I + \mathcal{L}) - \alpha_{12}\alpha_{21}I.
\end{aligned}
$$

Working through the details, it is striaghtforward to confirm that $\mathcal{N}_3\mathcal{M}_3$ is a block-diagonal matrix, with diagonal blocks given by the (block) determinant of $\mathcal{M}_3$,

$$D = (\alpha_{11}I + \mathcal{L})(\alpha_{22}I + \mathcal{L})(\alpha_{33}I + \mathcal{L}) - \alpha_{23}\alpha_{32}(\alpha_{11}I + \mathcal{L}) - \\ \alpha_{13}\alpha_{31}(\alpha_{22}I + \mathcal{L}) - \alpha_{12}\alpha_{21}(\alpha_{33}I + \mathcal{L}) + (\alpha_{13}\alpha_{32}\alpha_{21} + \alpha_{12}\alpha_{23}\alpha_{31})I.$$

Similar to the 2×2 case (albeit algebraically more complicated), this is a cubic polynomial in $\mathcal{L}$. By computing the roots of this polynomial, we can construct error propagation of a three-stage fixed-point iteration that produces the exact inverse of $\mathcal{D}$.

A few more thoughts:

- Working through the algebra, the cancellation does not fully happen if $\mathcal{L}_i$ is time-dependent. However, a lot of it does. There will be some off-diagonal terms that take the form, for example,

  $$(\alpha_{11}I + \mathcal{L}_1)(\alpha_{22}I + \mathcal{L}_2) - (\alpha_{22}I + \mathcal{L}_2)(\alpha_{11}I + \mathcal{L}_1) = \mathcal{L}_1\mathcal{L}_2 - \mathcal{L}_2\mathcal{L}_1.$$

  This provides a nice theoretical tool to analyze what is going on. It is possible these are often quite small. Moreover, we may be able to choose an ordering where these terms only occur on, say, the stritly lower triangular part, in which case a block-triangular preconditioning would also be exact.