KSP-based exponential time integration

An application of Krylov subspace projection

Nicholas Christensen
School of Computing and Data Science
University of Illinois Urbana-Champaign

March 4, 2025

KSP application: exponential time integration

By the end of this segment you will:

- Know what a matrix exponential is and its relevance to solving differential equations
- Know how to calculate the matrix exponential using eigendecomposition
- Know how to approximate the matrix exponential using Krylov Subspace Projection
- Know the cost and accuracy of KSP-based exponential time integration
- Know how to implement KSP-based exponential time integration
- Know some of the benefits and drawbacks of KSP-based exponential time-integration

The matrix exponential

If $\mathbf{A}$ is an $n \times n$ then the matrix exponential

$$e^{c\mathbf{A}} = \mathbf{I} + c\mathbf{A} + \frac{1}{2!}(c\mathbf{A})^2 + \frac{1}{3!}(c\mathbf{A})^3 \ldots$$

Relevance to solving linear differential equations:

$$\frac{\mathrm{d}\underline{y}}{\mathrm{d}t} = \mathbf{A}\underline{y} \rightarrow \underline{y}(t) = e^{t\mathbf{A}}\underline{y}(0)$$

or

$$\underline{y}_{j+1} = e^{\Delta t\mathbf{A}}\underline{y}_j \text{ where } y_j = y(j\Delta t)$$

Evaluating the matrix exponential

- Due to catastrophic cancellation, computing the matrix exponential using series expansion is generally a bad idea for more than a handful of terms.
- Many methods of numerically time advancing differential equations use only the first few terms

$$\text{Euler Forward: } \underline{y}_{j+1} = \underline{y}_j + \Delta t \mathbf{A} y_j \rightarrow \underline{y}_{j+1} = (\mathbf{I} + \Delta t \mathbf{A}) \underline{y}_j \tag{1}$$

$$\text{Euler Backward: } \underline{y}_{j+1} = \underline{y}_j + \Delta t \mathbf{A} \underline{y}_{j+1} \rightarrow \underline{y}_j = (\mathbf{I} - \Delta t \mathbf{A}) \underline{y}_{j+1} \tag{2}$$

$$\rightarrow \underline{y}_{j+1} = (\mathbf{I} - \Delta t \mathbf{A})^{-1} \underline{y}_j \tag{3}$$

- If $\mathbf{A}$ is diagonalizable, can use the eigendecomposition $\mathbf{A} = \mathbf{S}^{-1} \mathbf{\Lambda} \mathbf{S}$

Matrix exponential with eigendecomposition

$$\frac{\mathrm{d}\underline{y}}{\mathrm{d}t} = \mathbf{A}\underline{y} \tag{4}$$

$$\rightarrow \frac{\mathrm{d}\underline{y}}{\mathrm{d}t} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}\underline{y} \tag{5}$$

$$\rightarrow \frac{\mathrm{d}(\mathbf{S}^{-1}\underline{y})}{\mathrm{d}t} = \mathbf{\Lambda}\mathbf{S}^{-1}\underline{y} \tag{6}$$

$$\rightarrow \frac{\mathrm{d}\underline{w}}{\mathrm{d}t} = \mathbf{\Lambda}\underline{w} \text{ with } \underline{w} = \mathbf{S}^{-1}\underline{y} \tag{7}$$

$$\rightarrow \underline{w}(t) = e^{t\mathbf{\Lambda}}\underline{w}(0) \tag{8}$$

$$\rightarrow \underline{y}(t) = \mathbf{S}e^{t\mathbf{\Lambda}}\mathbf{S}^{-1}\underline{y}(0) \tag{9}$$

$$\rightarrow \underline{y}_{j+1} = \mathbf{S}e^{\Delta t\mathbf{\Lambda}}\mathbf{S}^{-1}\underline{y}_j \tag{10}$$

Matrix exponential with eigendecomposition

We still have to compute the matrix exponential of $e^{\Delta t \mathbf{\Lambda}}$. Is this a problem?

Cost of matrix exponentiation with eigendecomposition

- Problem cost of computing all eigenvectors and eigenvalues scales as $O(n^3)$
- 10^{12} operations if $\mathbf{A}$ is 10000×10000
- Q: How can we reduce the cost?

Reducing the cost with Krylov subspace projection

- Approximate $\underline{y}_{j+1}$ in the rank $k \ll n$ Krylov subspace of matrix powers spanning $\mathbf{K}_k = \left[\underline{y}_j \;\; \underline{\Delta} t\mathbf{A}\underline{y}_j \;\; (\Delta t\mathbf{A})^2\underline{y}_j \;\; (\Delta t\mathbf{A})^3\underline{y}_j \;\; \ldots \;\; (\Delta t\mathbf{A})^{k-1}\underline{y}_j\right]$
- Let $\tilde{\mathbf{A}}_k = \mathbf{Q}_k^T\mathbf{A}\mathbf{Q}_k$ where the matrix $\mathbf{Q}_k$ with the same span as $\mathbf{K}_k$ comes from Arnoldi iteration
    - Eigenvalues and eigenvectors of $\tilde{\mathbf{A}}_k$ (Ritz values/Ritz vectors) approximate the largest eigenvalues and eigenvectors of $\mathbf{A}$

Reducing the cost with Krylov subspace projection

$$
\begin{aligned}
\underline{y}(t) &= \mathbf{S}e^{t\boldsymbol{\Lambda}}\mathbf{S}^{-1}\underline{y}(0) && (11)\\
&\rightarrow \underline{y}(t) \approx \mathbf{Q}_k\tilde{\mathbf{S}}e^{t\tilde{\boldsymbol{\Lambda}}}\tilde{\mathbf{S}}^{-1}\mathbf{Q}_k^T\underline{y}(0) && (12)\\
&\rightarrow \underline{y}_{j+1} \approx \mathbf{Q}_k\tilde{\mathbf{S}}e^{\Delta t\tilde{\boldsymbol{\Lambda}}}\tilde{\mathbf{S}}^{-1}\mathbf{Q}_k^T\underline{y}_j && (13)
\end{aligned}
$$

where $\tilde{\mathbf{S}} = \tilde{\mathbf{S}}(\underline{y}_j)$ and $\tilde{\boldsymbol{\Lambda}} = \tilde{\boldsymbol{\Lambda}}(\underline{y}_j)$ are the time-step dependent eigenvalues and eigenvectors of $\tilde{\mathbf{A}}_k = \tilde{\mathbf{A}}_k(\underline{y}_j)$

Activity: cost comparisons

On your own, determine the costs of the following methods. Then compare your answers with a partner.

Euler Forward approach: $\underline{y}_{j+1} = y_j + \Delta t \mathbf{A} y_j$

Eigendecomposition approach: $\underline{y}_{j+1} = \mathbf{S} e^{\Delta t \mathbf{\Lambda}} \mathbf{S}^{-1} \underline{y}_j$

KSP approach: $\underline{y}_{j+1} \approx \mathbf{Q}_k \tilde{\mathbf{S}} e^{\Delta t \tilde{\mathbf{\Lambda}}} \tilde{\mathbf{S}}^{-1} \mathbf{Q}_k^T \underline{y}_j$

Convergence

The exponential integrator approximates

$$e^{\Delta t \mathbf{A}} \underline{y}_j = (\mathbf{I} + \Delta t \mathbf{A} + \frac{1}{2!}(\Delta t \mathbf{A})^2 + \frac{1}{3!}(\Delta t \mathbf{A})^3 \ldots) \underline{y}_j$$

in the span of

$$\left[\underline{y}_j \;\; \underline{\Delta} t \mathbf{A} \underline{y}_j \;\; (\Delta t \mathbf{A})^2 \underline{y}_j \;\; (\Delta t \mathbf{A})^3 \underline{y}_j \;\; \ldots \;\; (\Delta t \mathbf{A})^{k-1} \underline{y}_j \right]$$

- In infinite precision, the first $k-1$ terms of the series are represented exactly.
- Leading order error term: $c(\Delta t)^k \rightarrow O((\Delta t)^k)$ error for a single time step
- For all time steps $n_{ts} = T_f / \Delta t \rightarrow \frac{cT_f}{\Delta t}(\Delta t)^k \rightarrow O((\Delta t)^{k-1})$
- Often achieved in practice, but total error also depends on the error $\mathbf{Q}_k$ and the accuracy of the Ritz values and Ritz vectors
- This is only the temporal error. The spatial error depends on the spatial operator

Motivation for use

- Easy to formulate high-order time stepper without lengthy derivation
- High-order exponential time stepper is stable for relatively large time steps
  - Compared to typical explicit methods like Euler Forward, can take larger time steps $\rightarrow$ Fewer time steps
  - Compared to typical implicit methods like Euler Backward, does not require solving an $n \times n$ system of equations
- Drawbacks
  - Can be more expensive per time step than typical explicit methods
  - Requires storing basis for KSP

Hands on

- Sign into Colab with your Illinois account and save notebook to Drive

Colab link: https://tiny.cc/rp1c001