Exponential Integrators for PDEs arising in meteorology

August 31, 2024

Harvey Sutton

Contents

1 Introduction

It is common within applied mathematics to encounter ODEs of the form

$$\dot{u}(t) = Au(t) + N(u) \tag{1}$$

Where A is a large sparse matrix and N denotes some nonlinear term. As a result, finding accurate approximate solutions to these equations is of major importance.

These ODEs also occure when solving PDEs numerically using the method of lines. This arrives from discretising the PDE in space yielding an ODE with respect to time, with the form shown in (1).
Soving these ODEs numerically works by progressively stepping through time such as with the explicit forward Euler method, which is given as:

$$u(t+h) = u(t) + h\dot{u}(t) = u(t) + h(Au(t) + N(u(t)))$$

for a given step size h. Another method (and the one we will be focusing on here) employs the use of the exponential integrator, which writes the solution to problem (1) over a given time step as:

$$u(t+h) = e^{Ah}u(t) + \int_0^h e^{A(h-z)}N(u(t+z))dz$$

$$\text{where } e^A = \sum_{i=0}^{\inf} \frac{A^n}{n!}$$

An important requirement when using this method is to compute approximations of $e^A v$ efficiently and accurately for the given sparse matrix A and vector v. One approach would be first to compute e^A and apply the resulting matrix to the vector v. However, even for sparse A, e^A will likely be dense. For large A this will result in very high memory usage, which may not be practical when approximating solutions to (**??**). One solution to reduce memory usage is to compute $e^A v$ directly without computing e^A as an intermediary step.
Various methods have been proposed for this such as by Al-Mohy, Awad H. and Higham, Nicholas J[1] which is currently used in the Scipy package as `scipy.sparse.linalg.expm_multiply`. Another class of methods known as Krylov subspace methods have gained traction recently[3]. These work by projecting the problem onto a space with fewer dimensions (the kyrlov subspace), computing the matrix exponential and then transforming back to the original space. As the matrix exponential will be computed in the lower dimension kyrlov subspace the computation cost of this step will not be significant.

2 Krylov Subspace Methods

In this section, we will look into existing krylov subspace methods and then compare how these methods perform for approximating $e^A v$.

2.1 Alogrithms

Here we give the algorithms for the 2 krylov subspace methods we will be comparing. Each of these algorithms takes a matrix $A \in \mathbb{R}^{n \times n}$, a vector $v \in \mathbb{R}^n$ and an integer m that determines the number of dimensions of the krylov subspace used. From these algorithms we will get a matrix $H \in \mathbb{R}^{m \times m}$ and another matrix $V \in \mathbb{R}^{n \times m}$ such that $A \approx VHV^T$ and $VV^T = I$. From here we can get $Av \approx VHV^Tv = VH||v||e_1$.

Algorithm 1 Arnoldi [2]

procedure ARNOLDI($A, \hat{v}_1, m$)
  $v_0 = 0$
  for $j = 1, 2, ..., m$ **do**
    for $i = 1, 2, ..., j$ **do**
      $h_{ij} \leftarrow v_i^T A v_i$
    $\theta_j \leftarrow Av_j - \sum_{i=1}^{j} h_{ij} v_i$
    $h_{j+1,j} \leftarrow ||\theta_j||$
    $v_{j+1} \leftarrow \theta_j / h_{j+1,j}$

Here V is given by $v_1, ..., v_m$ and H is give by $h_1, ..., h_m$.

Algorithm 2 Lanczos [4]

procedure LANCZOS(A symetric, $\hat{v}_1, m$)
  $v_0 = 0$
  for $i \in \{1, 2, ..., m\}$ **do**
    $\beta_i \leftarrow ||\hat{v}_i||$
    $v_i \leftarrow \hat{v}_i / ||\hat{v}_i||$
    $\alpha_i = v_i^T A v_i$
    $\hat{v}_{i+1} = Av_i - \alpha_i v_i - \beta_i v_{i-1}$

Here V is given by $v_1, ..., v_m$ and H is tridiagonal with the leading diagonal being $\alpha_1, ..., \alpha_m$ and the upper and lower diagonals being $\beta_2, ..., \beta_m$.

2.2 Relevance To Matrix Exponentials

By using $A \approx VHV^T$ and the fact that $V^TV = I$ we can apply this to e^A as follows:

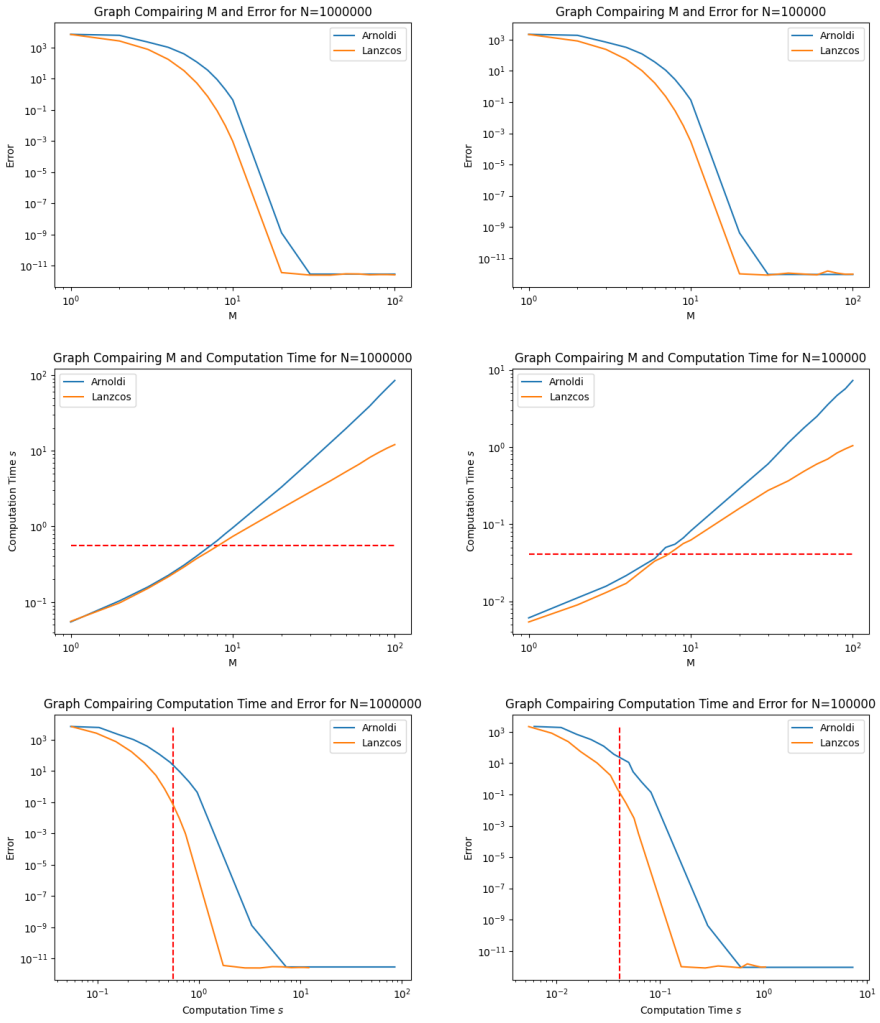
$$\begin{aligned} e^A &= \sum_{i=0}^{\infty} \frac{A^i}{i} \\ &= \sum_{i=0}^{\infty} \frac{(VHV^T)^i}{i} \\ &= \sum_{i=0}^{\infty} \frac{VH^iV^T}{i} \end{aligned}$$

and then when computing e^Av we get

$$\begin{aligned} e^Av &= (\sum_{i=0}^{\infty} \frac{VH^iV^T}{i})v \\ &= \sum_{i=0}^{\infty} \frac{VH^iV^Tv}{i} \\ &= \sum_{i=0}^{\infty} \frac{VH^ie_1}{i} \\ &= V(\sum_{i=0}^{\infty} \frac{H^i}{i})e_1 \\ &= Ve^He_1 \end{aligned}$$

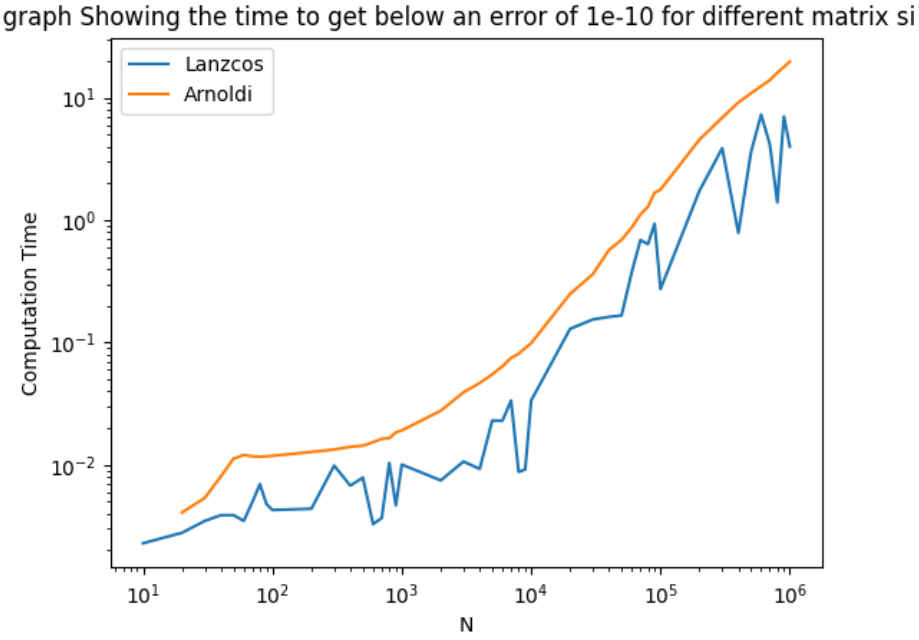
2.3 Numerical Results

Here we run numerical experiments, computing e^Av for a tridiagonal matrix A of sizes $100,000$ and $1,000,000$. We will compare the methods with respect to computation time, error and size of the Kyrlov subspace.



These results show that the Lanzcos algorithm outperforms the Arnoldi algorithm in both convergence rate and computational time.

Below we compare the time required for the different algorithms to get below an error of 10^{10} when computing $e^A v$ for various matrix sizes.



Again we continue to see that the Lanzcos algorithm outperforms the Arnodli algorithm.

References

[1] Awad H. Al-Mohy and Nicholas J. Higham. Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM Journal on Scientific Computing*, 33(2):488–511, January 2011.

[2] Shitao Fan. An introduction to krylov subspace methods. November 2018.

[3] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, January 2003.

[4] I. U. OJALVO and M. NEWMAN. Vibration modes of large structures by an automatic matrix-reductionmethod. *AIAA Journal*, 8(7):1234–1239, July 1970.