

Exponential Integrators for PDEs arising in meteorology

September 11, 2024

Harvey Sutton

Contents

1	Introduction	2
2	Krylov Subspace Methods	3
2.1	Alogrithms	3
2.2	Numerical Results	4

1 Introduction

In many simulations, it is common to encounter ODEs of the form

$$\dot{u}(t) = Au(t) + R(u) \quad (1)$$

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

Often ODEs also occur when solving PDEs numerically using the method of lines. This arises from discretising the PDE in space yielding an ODE with respect to time, with the form shown in (1).

Solving these ODEs numerically works by progressively stepping through time such as with the explicit forward Euler method, which is given as:

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

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:

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

using a simple explicit approach leads to

$$\approx e^{A\tau}(u(t) + N(u(t)))$$

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

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 Krylov subspace), computing the matrix exponential and then transforming back to the original space. As the matrix exponential will be computed in the lower dimension Krylov 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 Algorithms

The algorithms take 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 get $Av \approx VHV^T v = VH||v||e_1$. 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 $e^A v$ we get

$$\begin{aligned} e^A v &= \left(\sum_{i=0}^{\infty} \frac{VH^iV^T}{i} \right) v \\ &= \sum_{i=0}^{\infty} \frac{VH^iV^T v}{i} \\ &= \sum_{i=0}^{\infty} \frac{VH^i||v||e_1}{i} \\ &= V \left(\sum_{i=0}^{\infty} \frac{H^i}{i} \right) ||v||e_1 \\ &= Ve^H ||v||e_1 \end{aligned}$$

Here we give the algorithms for the 2 Krylov subspace methods we will be comparing. We begin with the Arnoldi algorithm.

Algorithm 1 Arnoldi [2]

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

Here V is given by v_1, \dots, v_m and H is give by h_1, \dots, h_m .
The algorithm below is the Lanczos algorithm and it requires a symmetric matrix. [3]

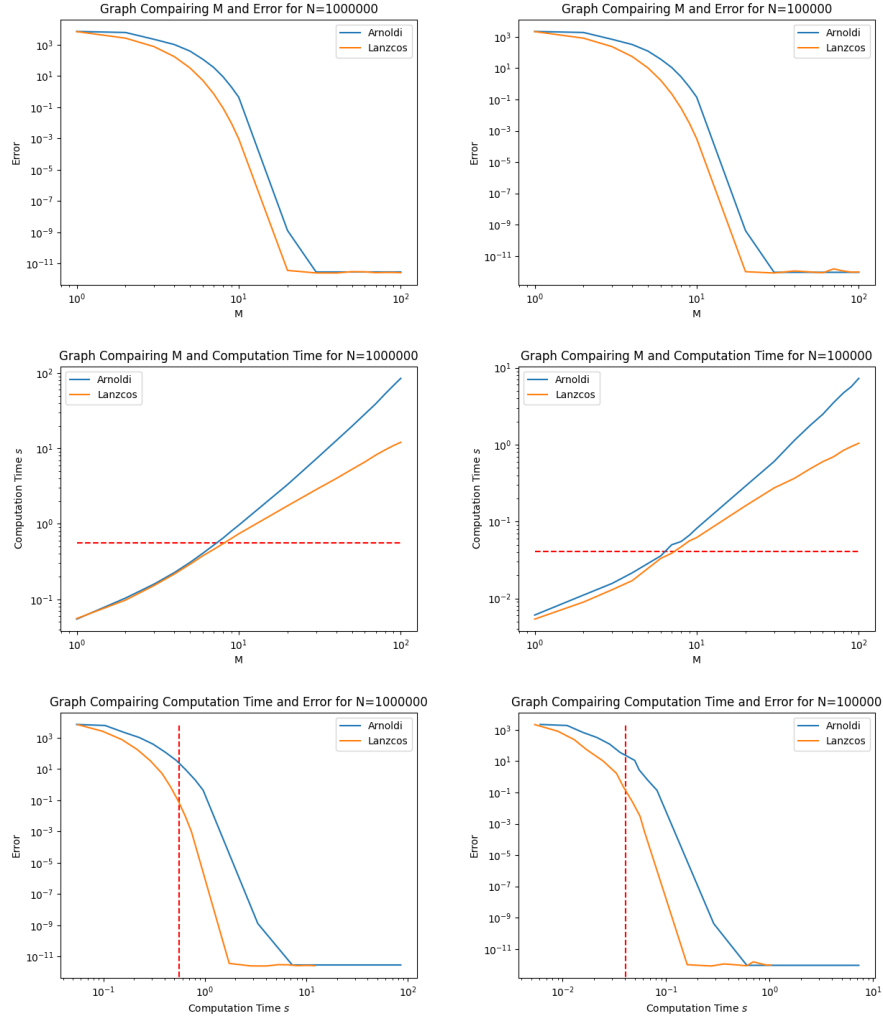
Algorithm 2 Lanczos [4]

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

Here V is given by v_1, \dots, v_m and H is tridiagonal with the leading diagonal being $\alpha_1, \dots, \alpha_m$ and the upper and lower diagonals being β_2, \dots, β_m .

2.2 Numerical Results

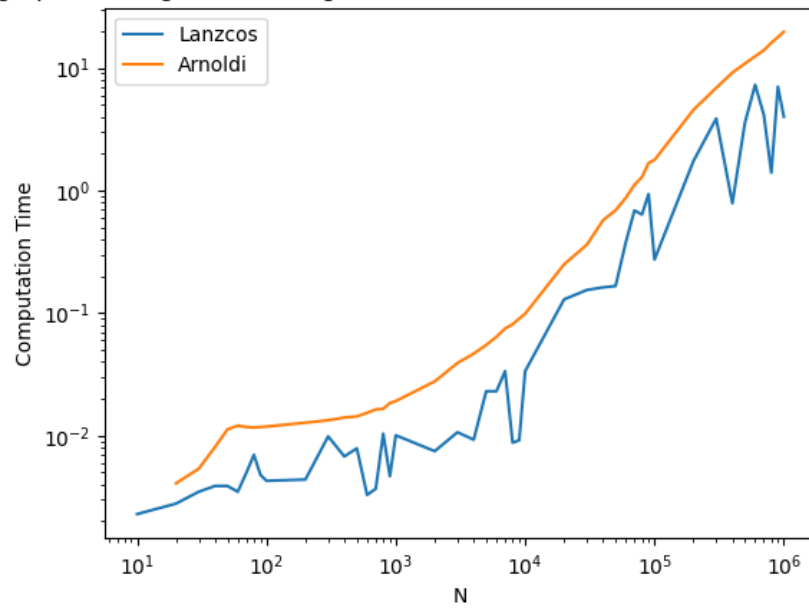
Here we run numerical experiments, computing $e^A v$ for a tridiagonal matrix A of sizes 100,000 and 1,000,000, where A is a triagonal matrix with $2n^2$ along the diagonal and $-n^2$ along the upper and lower diagonals. We will compare the methods with respect to computation time, error and size of the Kyrlov subspace.



These results show that the Lanczos 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.

graph Showing the time to get below an error of $1e-10$ for different matrix si



Again we continue to see that the Lanczos algorithm outperforms the Arnoldi 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.