MAT 508 Programming Project

Diego G. A. Avalos Galvez

Winter 2018

Abstract

Let S be a real skew-symmetric matrix. In this project, we compare Matlab's `expm` function with Diego Avalos's thesis formula for e^S, where S is of size 5, by randomly generating matrices S such that the entries of S satisfy $|S_{ij}| \leq 5$. The goal of this paper is to demonstrate cases where `expm` fails, but Avalos's closed formula gives an appropriate value for e^S. To accomplish this, we test `expm(10^k*S)` for integer values of k ($0 \leq k \leq 20$), and observe that the Matlab output goes to zero, `Inf`, or `NaN`, whereas Avalos's formula gives a more valid value or e^S. Furthermore, we generate skew-symmetric matrices with small and large entry values and compare our to methods against the determinant of e^S (=1).

Introduction

In Diego Avalos's Master's thesis *Exponentials of Real Skew-Symmetric Matrices in Terms of Their Eigenvalues*, he computes formulas for the exponential of any real-valued skew-symmetric matrix up to size 9 in terms of their eigenvalues and powers of the matrix. Particularly, in this paper we will study the effectiveness of Avalos's formulas against Matlab's `expm` function. We will look at real skew-symmetric matrices $S = -S^T$ of size five with four different purely imaginary eigenvalues $\pm\theta_1 i$, $\pm\theta_2 i$ ($\theta_2 > \theta_1 > 0$), and zero. We will rely on the Matlab function `charpoly` to compute the coefficients of the characteristic polynomial of S, which we denote by $p(X) = \det(XI_5 - S)$, because computing these coefficients does not cost anything, as they are simply sums and products of the entries of S. For instance, since $p(X)$ has the form

$$p(X) = X^5 + aX^3 + bX$$

then we can use the quadratic formula to obtain

$$\theta_1 = \sqrt{\frac{a - \sqrt{a^2 - 4b}}{2}}$$

and

$$\theta_2 = \sqrt{\frac{a + \sqrt{a^2 - 4b}}{2}}.$$

Observe that $\theta_2 > \theta_1 > 0$ because $a^2 > 4b$ is ensured by the fact that S has five different eigenvalues. Hence, to get e^S we can use Avalos's formula for the Case 3.5.2, namely

$$\begin{aligned} e^S =& I_5 + \frac{\theta_2^3 \sin\theta_1 - \theta_1^3 \sin\theta_2}{\theta_1\theta_2(\theta_2^2 - \theta_1^2)} S + \frac{\theta_2^4(1-\cos\theta_1) - \theta_1^4(1-\cos\theta_2)}{\theta_1^2\theta_2^2(\theta_2^2 - \theta_1^2)} S^2 \\ &+ \frac{\theta_2 \sin\theta_1 - \theta_1 \sin\theta_2}{\theta_1\theta_2(\theta_2^2 - \theta_1^2)} S^3 + \frac{\theta_2^2(1-\cos\theta_1) - \theta_1^2(1-\cos\theta_2)}{\theta_1^2\theta_2^2(\theta_2^2 - \theta_1^2)} S^4. \end{aligned} \tag{*}$$

Using the identities $\theta_1\theta_2 = \sqrt{b}$ and $\theta_2^2 - \theta_1^2 = \sqrt{a^2-4b}$ we get the more computationally friendly formula

$$\begin{aligned} e^S =& I_5 + \frac{\theta_2^3 \sin\theta_1 - \theta_1^3 \sin\theta_2}{\sqrt{a^2b-4b^2}} S + \frac{\theta_2^4(1-\cos\theta_1) - \theta_1^4(1-\cos\theta_2)}{b\sqrt{a^2-4b}} S^2 \\ &+ \frac{\theta_2 \sin\theta_1 - \theta_1 \sin\theta_2}{\sqrt{a^2b-4b^2}} S^3 + \frac{\theta_2^2(1-\cos\theta_1) - \theta_1^2(1-\cos\theta_2)}{b\sqrt{a^2-4b}} S^4. \end{aligned} \tag{1}$$

In this paper, we will randomly generate real skew-symmetric matrices of size five S on Matlab whose entries S_{ij} are constrained to integer values such that $|S_{ij}| \leq 5$, and use Formula (1) to compute e^S and compare it to `expm(S)`. Furthermore, since we are seeking to demonstrate that our closed form formula for e^S performs better than `expm`, we will also compare our formula with the exponential $e^{10^k S}$, where k takes on integer values from zero to 20.
The beauty of Formula (*) is that if instead of e^S, we desire to compute $e^{10^k S}$, then we can use the fact that if λ is an eigenvalue of S, then $r\lambda$ is an eigenvalue of rS for any scalar r and simplify the coefficients of S. Therefore, if we use Formula (*) to compute $e^{10^k S}$ and cancel the expressions 10^k from each numerator and denominator of each power of $10^k S$, we obtain

$$\begin{aligned} e^{10^k S} =& I_5 + \frac{\theta_2^3 \sin 10^k\theta_1 - \theta_1^3 \sin 10^k\theta_2}{\sqrt{a^2b-4b^2}} S + \frac{\theta_2^4(1-\cos 10^k\theta_1) - \theta_1^4(1-\cos 10^k\theta_2)}{b\sqrt{a^2-4b}} S^2 \\ &+ \frac{\theta_2 \sin 10^k\theta_1 - \theta_1 \sin 10^k\theta_2}{\sqrt{a^2b-4b^2}} S^3 + \frac{\theta_2^2(1-\cos 10^k\theta_1) - \theta_1^2(1-\cos 10^k\theta_2)}{b\sqrt{a^2-4b}} S^4. \end{aligned} \tag{2}$$

We code Formula (2) into Matlab to compute the exponentials of $10^k S$ and compare them against `expm(10^k*S)`. Observe that we are relying on Matlab's values of sines and cosines of very large values (e.g., $\cos 10^{20}$).

Discussion and Results

Let S be a randomly generated real skew-symmetric matrix with coefficients $|S_{ij}| \leq 5$. We denote the matrix exponential $e^{10^k S}$ computed with Formula (2) by M, and the matrix exponential of $10^k S$ computed with Matlab's `expm` by $\hat{M}$. We will be evaluating the relative error $||M - \hat{M}||$ (the norm is Frobenius) as we increase the values of k from 0 to 20. To make the observations easier, we plot our results for $\frac{1}{100}\log ||M - \hat{M}||$ in Figure 1. We see that both methods of computing $e^{10^k S}$ almost perfectly coincide up to $k = 15$, but after that value the relative error grows immensely.

However, the relative error between the computed exponentials using both methods only informs us about the closeness of the results, but does not tell us about the correctness of the outcomes. We will now test our results in a different way using one fact from Lie Algebras, namely, for any $S \in \mathfrak{so}(n)$, $e^S \in SO(n)$. In other words, given any real skew-symmetric matrix S, its matrix exponential e^S is an orthogonal matrix such that $\det e^S = 1$. For instance, we will run three tests by randomly generating one hundred skew-symmetric five-by-five matrices and computing the absolute error of their determinants by using both Matlab's `expm` and Formula (1). Figure 2 illustrates our first test, where we generate matrices such that $|S_{ij}| \leq 10$, so the entries of S are very closed together, and observe that the errors for `expm` and Formula (1) are equally close to zero, except for some Formula (1) values. Figure 3 shows our second test, where we make $|S_{ij}| \leq 100$, which makes the entries of S more spread out and larger. Here, the errors are much more stable and better for Formula (1) than for `expm`.

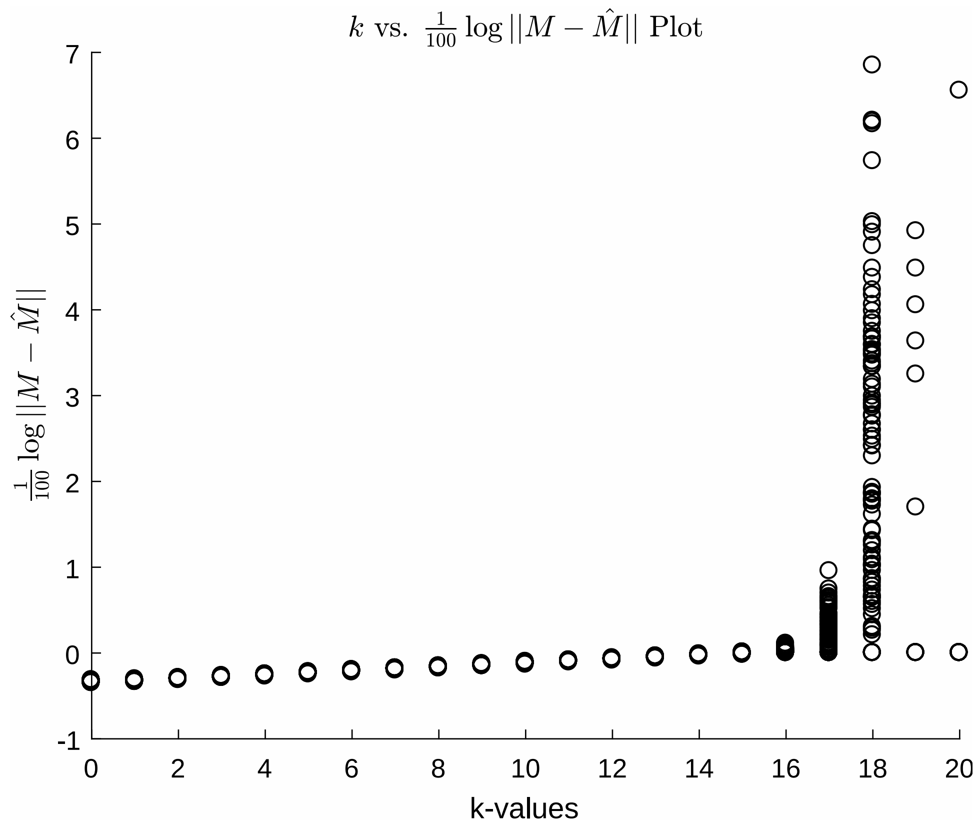

Figure 1: Relative Error $||M - \hat{M}||$ Analysis

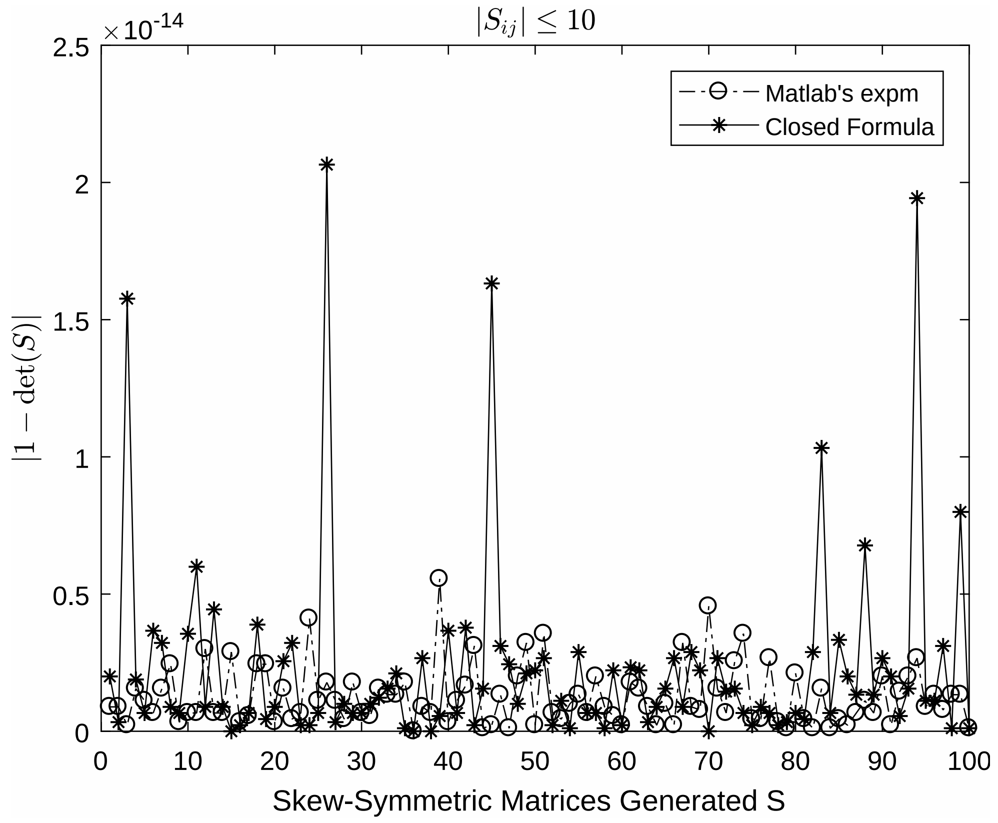

Figure 2: Absolute Determinant Error for $|S_{ij}| \leq 10$

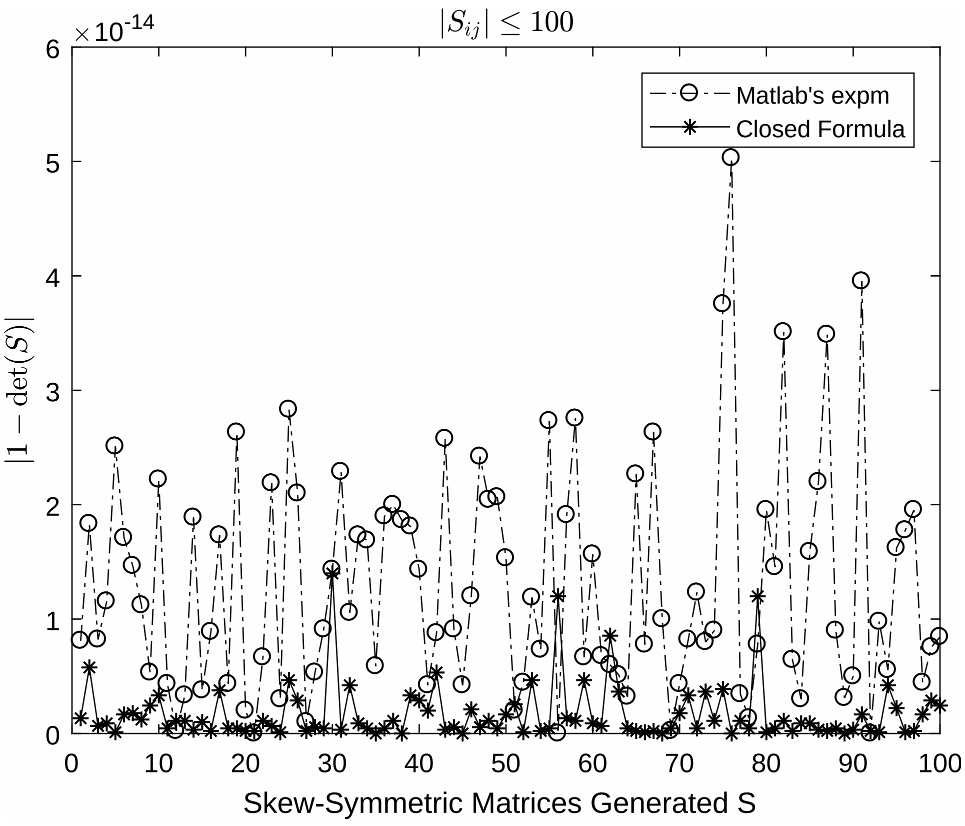

Figure 3: Absolute Determinant Error for $|S_{ij}| \leq 100$

Finally, in our second test, we make $|S_{ij}| \leq 1000$ even larger and spread out, and Figure 4 demonstrates that the method which uses Formula (1) performs much better.

In conclusion, our implemented Formula (1) and `expm` perform very similarly when the entries of the skew-symmetric matrix are small and close. On the other hand, if we have a skew-symmetric matrix with large and very spread out (uniformly random) entry values, Formula (1) gives a more stable and more accurate result than `expm`.

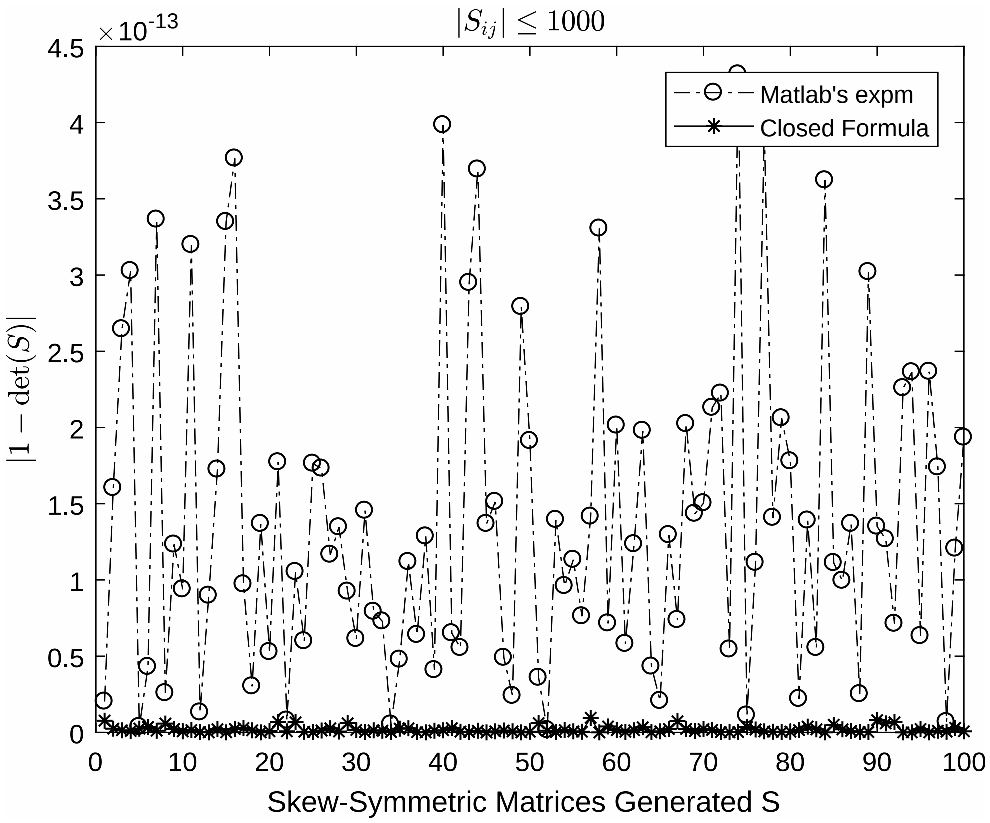

Figure 4: Absolute Determinant Error for $|S_{ij}| \leq 1000$

Source Code

```
function [MyExponential, MatlabExponential, rel_error] = expskew5(r,k)
% r is a 1x10 vector, the terms correspond to the entries above the
% diagonal of the skew-symmetric matrix S. Use k=0 to get the value e^S.
format long
S = [ 0     r(1) r(2)  r(3)   r(4);...
     -r(1)   0   r(5)  r(6)   r(7);...
     -r(2) -r(5)  0    r(8)   r(9);...
     -r(3) -r(6) -r(8) 0      r(10);...
     -r(4) -r(7) -r(9) -r(10)  0    ];
k=10^k;
MatlabExponential = expm(k*S);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Computing my exponential
A = charpoly(S); a = A(3); b = A(5);
t1 = sqrt((a-sqrt(a^2-4*b))/2);
t2 = sqrt((a+sqrt(a^2-4*b))/2);

c1 = (t2^3*sin(k*t1)-t1^3*sin(k*t2))/(t1*t2*(t2^2-t1^2));
c2 = (t2^4*(1-cos(k*t1))-t1^4*(1-cos(k*t2)))/(t1^2*t2^2*(t2^2-t1^2));
c3 = (t2*sin(k*t1)-t1*sin(k*t2))/(t1*t2*(t2^2-t1^2));
c4 = (t2^2*(1-cos(k*t1))-t1^2*(1-cos(k*t2)))/(t1^2*t2^2*(t2^2-t1^2));

MyExponential = eye(5) + c1*S + c2*S^2 + c3*S^3 + c4*S^4;
E = MyExponential-MatlabExponential;
rel_error = norm(E);
```