# Approximation of Exponential Functions by Composite Taylor Polynomials

Candidate Number: **1040706**
Oxford Masters in Mathematical Sciences (OMMS)
Trinity 2020

# Contents

**Acknowledgement**

We are extremely grateful to our supervisor, Professor Yuji Nakatsukasa, for sharing his knowledge, excellent guidance, patience and advice throughout the dissertation. Without his support, none of this would have been possible.

# Introduction

The subject of approximation theory is concerned with devising methods (often called algorithms) for approximating, in an efficient manner, a (real or complex) function $f$ by 'simpler' functions. The efficiency of these algorithms depend upon the accuracy of the method in the computer's floating point arithmetic and the ease with which they can be implemented (i.e. their complexity). Typically, the class of polynomials of the form:

$$p(z) = a_0 + a_1 z + a_2 z^2 + \cdots + a_m z^m$$

where $\{a_k\}$ are constants and $z \in \mathbb{C}$ are used to approximate $f$. One of the major advantages of polynomial approximation is that, if $f$ is *analytic* in a closed domain, then polynomials converge *geometrically* to $f$. Secondly, the derivative and indefinite integral of a polynomial are easy to determine and the result is again a polynomial making them the ideal choice in quadrature and root finding. More elegant approximation theory is covered in the book *'Approximation Theory and Approximation Practice'*, Trefethen [1] and reliable algorithms are implemented in the Chebfun software package [2].

A relatively unexplored question is to approximate $f$ by *composite polynomials*, of the form:

$$p(z) = q_s \circ q_{s-1} \circ \ \dots q_2 \circ q_1(z) = q_s(q_{s-1}(\dots q_2(q_1(z))))$$

where each $q_i$ is a carefully chosen polynomial. In this report, our main focus is to investigate the power and limitations of composite polynomials to approximate *exponential function* both in the scalar and matrix case.

The exponential of a matrix $A \in \mathbb{C}^{n\times n}$ is defined as:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

which was first introduced by Laguerre [3] in 1867.

A common question which people may ask is why are matrix exponential functions so important? This is mainly owed to their key role in the solution of partial differential equations (PDEs) [4, 5] such as heat equation:

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta u \\ u(x,0) = u_0 \end{cases} \quad \text{on } \Omega.$$

The solution has the form $e^{At}v_0$ where $A$ is the matrix discretization of the Laplace operator $\Delta$ and $v_0$ is discretization of $u_0$. PDEs need to be solved in all kinds of discipline from engineering to finance and therefore exponential functions can be found everywhere. Matrix exponential also has applications in other branches of mathematics with no direct link to differential equations such as network centrality analysis [6] and Markov chain analysis [7].

In the last century, various methods have been introduced for computing $\exp(A)$ as described in the classic paper *'Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later'* by Moler, Loan [8] which reached the conclusion that the most effective method was the scaling and squaring method based on the composition of Padé approximants [9]. In this report, we focus our attention solely on the scaling and squaring method but we base it on the *composition of polynomials* instead of Padé rational functions and approximate $\exp(A)$ as:

$$e^A = \left(e^{A/\sigma}\right)^{\sigma} \approx q\Big(q\Big( \dots q\Big(T_m(A/\sigma)\Big)\Big)\Big) = p(A)$$

where $\sigma$ is a positive scaling parameter, $T_m$ is a $m$ degree truncated Taylor series of $\exp(A/\sigma)$ and $q$ is a suitably chosen polynomial. Particular focus is given to the distinction between *polynomial degree* and *degrees of freedom*, because composite polynomials can have a very high degree despite having a small degrees of freedom to describe them. As Trefethen mentioned in [1, chap. 23]: the polynomial approximations are useful not because $p(z)$ is easier to evaluate than $\exp(z)$, but because $p(A)$ is easier to evaluate than $\exp(A)$. Although, during the majority of this report, we will consider the scalar exponential function, we will make useful links with the matrix case.

The structure of the report will be as follows: In Chapter 1, we will define some useful theorems and definitions from complex and numerical analysis and then go on to make an important link between Taylor expansion for scalar and matrix functions. We also touch upon the floating point arithmetic which will help to explain many of the phenomena (like flattening of the error curves) we observe later in the report. In Chapter 2, we introduce the scaling and squaring method, explain why its superior than just truncating a Taylor series and also compute its complexity for a $n \times n$ matrix $A$. The author also carries out some *original* error analysis by:

- computing a set of complex values where our absolute error is in the order of machine precision (see Section 2.6).
- combining the truncation and rounding errors incurred by the algorithm to give a bound on where we expect the relative error curves to flatten out for a fixed value of $m$ and $\sigma$ (see Section 2.7).

The style of this report is slightly different to a traditional one; most numerical experiments are illustrated computationally with the help of the Numpy [10] and Scipy [11] packages in Python which allows the reader to see examples on the fly and have a better understanding of the content. The belief of the author is that one understands a numerical algorithm if one can write a short code for it by handling any exceptions the code might throw and also be able to provide a valid explanation for the output of the code. The code is uploaded on GitHub and the reader can access it via the following link: `https://anonymous.4open.science/r/b10787f0-dbfb-4a8d-91ba-d1a4c6675e8c/`. It is also included in the *zipped folder* provided by the author under the path `code/`.

# Chapter 1

# Useful Definitions and Theorems

## 1.1 Polynomial Space and Degrees of Freedom

In this section, we define some useful terminology which we will use in the rest of the report.

**Definition 1.1.1.** Univariate Polynomial Space $\mathcal{P}_m$
Let $\mathbb{F}$ be an arbitrary field[1]. We define the polynomial space of degree $m$ as:

$$\mathcal{P}_m = \left\{ \sum_{k=0}^{m} \alpha_k z^k \ : \ \alpha_k \in \mathbb{F}, \alpha_m \neq 0 \right\}.$$

**Definition 1.1.2.** Degrees of Freedom (DoF)
The degrees of freedom of a polynomial $p \in \mathcal{P}_m$ is the *minimum* number of coefficients $\alpha_k \in \mathbb{F}$ which have the freedom to vary *independently* from each other and describe $p$ uniquely.

**Note:** The notion of degrees of freedom and degrees of the polynomial are different: degree of a polynomial which we denote by $deg(p)$, is the highest power of polynomial's monomials whereas the DoF is equivalent to the dimensionality of the space $\mathcal{P}_m$. It is important that these two concepts are not confused.

**Example**:

- So $p \in \mathcal{P}_m$ has DoF = $(m+1)$ and $deg(p) = m$.
- Let $p \in \mathcal{P}_m$ and $q \in \mathcal{P}_n$, then DoF of the composition $p \circ q$ is $(m+n+2)$ and $deg(p \circ q) = mn$ whereas $pq$ has DoF of $(m+n+2)$ and $deg(pq) = m+n$.

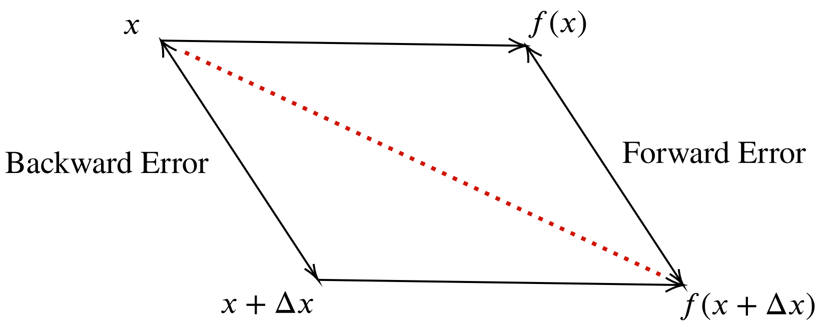
Just by looking at the second example, we can see how powerful the composition of polynomials can be: both $p \circ q$ and $pq$ have the same degrees of freedom, however, by composition, we have increased the degree of polynomial geometrically. Thus, this gives an intuition that by just a few compositions, we can increase the degree of a polynomial significantly while keeping degrees of freedom fairly small.

[1]In this report, we will take our field to be either $\mathbb{R}$ or $\mathbb{C}$

## 1.2 Types of Errors

Let $f : X \to Y$ be a function mapping from vector space $X$ into a vector space $Y$. Suppose, we want to compute $y = f(x)$, where $y$ is the *true* (exact) output, for a given input $x \in X$. But many times, we can only compute *approximation* of $y$, which we express as $\hat{y} = \hat{f}(x)$ [2]. The potential causes for these errors is the *round-off errors* and the *truncation errors*. Then we can measure these errors in two ways:

- **Forward Error**: it is the difference between the approximation $\hat{y}$ and the true value $y$. Key thing to note is that the input $x \in X$ is fixed and then we define:
  – (Absolute) Forward Error: $F_{abs} = \|\hat{y} - y\| = \|\hat{f}(x) - f(x)\|$.
  – (Relative) Forward Error: $F_{rel} = \frac{\|\hat{y}-y\|}{\|y\|} = \frac{\|\hat{f}(x)-f(x)\|}{\|f(x)\|}$.
  where we use a suitable norm[3] $\|\cdot\|$ to measure these errors. This is a natural quantity to measure, but since in practice, we do not know the true $y$, we can only get an upper bound on this error which may not be *tight*.
- **Backward Error**: here we ask the question, what set of data actually solves the problem? I.e what is the smallest $\Delta x$ such that $f(x+\Delta x) = \hat{y}$. So if we define $\hat{x} = x + \Delta x$, then:
  – (Absolute) Backward Error: $E_{abs} = \|\hat{x} - x\| = \|\Delta x\|$.
  – (Relative) Backward Error: $E_{rel} = \frac{\|\hat{x}-x\|}{\|x\|} = \frac{\|\Delta x\|}{\|x\|}$.



Note that if $X = \mathbb{R}$, then $\lfloor -\log_{10} E_{rel} \rfloor$ represents how many *decimal points* do the approximate and exact values agree on and $\lfloor -\log_{10} E_{abs} \rfloor$ represents the same for the *number of digits*.

**Example**: Suppose we want to compute $y = \sqrt{2}$ and $\hat{y} = 1.4$, and if we define the function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ as $f(x) = \sqrt{x}$, then we can let $x = 2$ and $\hat{x} = 1.96$. So the absolute errors:

a) (Absolute) Forward Error: $|\sqrt{2} - 1.4| \approx 0.01421$.
b) (Absolute) Backward Error: $|2 - 1.96| = 0.04$ .

[2]Often, $\hat{f}$ is expressed as an algorithm which is implemented on a computer under floating point arithmetic (see next section).

[3]By equivalence of norms in finite-dimensional vector space, any vector norm suffices. But in practice, usually the 1-norm, Euclidean norm or sup-norm are used.

## 1.3 Floating Point Arithmetic

The results in this section are inspired from the book *'Accuracy and Stability of Numerical Algorithms'*, Higham [12, pg. 35-41] and the UCL course *'Numerical Methods'* [13, pg. 15-16] taught by David Hewett.

When we implement numerical algorithms on a computer, it is recommended to understand how a computer stores and manipulates numbers. Since a computers only has finite memory, we cannot store the uncountable set of real numbers. Therefore, we store floating point numbers which cannot be arbitrarily close to each other. The definition below tries to formalise this:

**Definition 1.3.1.** Let $\mathcal{F} \subset \mathbb{R}$ be a *discrete* subset of $\mathbb{R}$ (since there are gaps between two adjacent numbers). Let $\epsilon_{mach} = 2^{-53} \approx 1 \times 10^{-16}$ be the machine precision[4] of most double precision computers. Then, the function $fl : \mathbb{R} \to \mathcal{F}$ is the mapping from reals to its closest floating point representation. More formally, $\forall x \in \mathbb{R}$, there exists $\varepsilon$ such that[5] $|\varepsilon| \leq \epsilon_{mach}$ and $fl(x) = x(1+\varepsilon)$. The relative error in the floating point representation $fl(x)$ satisfies:

$$|fl(x) - x| \leq \epsilon_{mach}|x|.$$

We say approximation is only correct to machine precision because we cannot hope to do any better than this.

Whenever the numerical algorithm carries out arithmetic operations such as $+, -, \times, \div$, the result returned has to be rounded to the nearest floating point number. This introduced *rounding error* can have serious implications on the accuracy and stability of an algorithm. As a trivial example, consider the operation of subtracting two real numbers that are very close to each other. This kind of operation may lead to a loss of significant digits in numerical computations if one is not careful. In Python this can be illustrated as follows:

```
x = 1e-15
print(((1+x)-1)/x)
>> 1.1102230246251565
```

We expected an output of 1, but due to floating point arithmetic, we see a relative error of 11%. A great deal of care needs to be taken when dealing with arithmetic operations between very large/small numbers (by magnitude).

In relative sense, one might say than an algorithm $\hat{f}$ for a problem $f$ is ***accurate*** if for every $x \in X$:

$$\frac{\|\hat{f}(x) - f(x)\|}{\|f(x)\|} = O(\epsilon_{mach})^{6}.$$

[4]I am aware that Prof. Higham as shown in https://en.wikipedia.org/wiki/Machine_epsilon#Values_for_standard_hardware_floating_point_arithmetics defines $\epsilon_{mach} = 2^{-52}$ but then includes a factor of $1/2$ when defining relative error. So to avoid this extra factor, $\epsilon_{mach}$ is described as above.

[5]to be more precise, it is not true $\forall x \in \mathbb{R}$ but only which lie in the representable range of $\mathcal{F}$, i.e. for double precision, if $0 < 10^{-308} \leq |x| \leq 10^{308}$.

[6]This is "Big O" notation: for functions $f, g$ defined on $\mathbb{R}$, we say $f(x) = O(g(x))$ as $x \to x_0$ if, there exists constants $C, \delta > 0$ such that $|f(x)| \leq C|g(x)|$ for $|x - x_0| < \delta$.

The accuracy and *stability* of an algorithm is discussed in more depth in Trefethen [14, pg. 102].

## 1.4 Useful Results for Matrices

Here we state some important results for matrices which help us in error analysis of matrix exponential function in the Section 2.3.

**Definition 1.4.1.** Subordinate Matrix Norm
The subordinate matrix norm $\|\cdot\|$ is given by:

$$\|A\| = \sup_{\substack{x\in\mathbb{R}^n\setminus\{0\} \\ \|x\|_V=1}} \|Ax\|_V \tag{1.1}$$

where $\|\cdot\|_V$ is a vector norm.

The reason for introducing the matrix norm is because of their *submultiplicative* property:

$$\|AB\| \leq \|A\|\|B\|. \tag{1.2}$$

This helps to simplify our error analysis later on.

**Lemma 1.4.1.** (Higham [15, pg. 237])
For $A \in \mathbb{C}^{n\times n}$ and for a matrix norm $\|\cdot\|$, we have the inequality:

$$e^{-\|A\|} \leq \|e^A\| \leq e^{\|A\|}.$$

*Proof*:
The upper bound is trivial to show:

$$\|e^A\| = \left\|\sum_{k=0}^{\infty} \frac{A^k}{k!}\right\| \leq \sum_{k=0}^{\infty} \frac{\|A\|^k}{k!} = e^{\|A\|}.$$

To show the lower bound, note that $I = e^A e^{-A}$:

$$\begin{aligned} 1 = \|e^A e^{-A}\| \leq \|e^A\| \cdot \|e^{-A}\| &\leq \|e^A\| e^{\|A\|} \qquad \text{(by 1.2)} \\ \implies e^{-\|A\|} &\leq \|e^A\|. \qquad \square \end{aligned}$$

## 1.5 Useful Results from Complex Analysis

These results are taken from lectures taught at University College London (UCL): “An Introduction to Complex Analysis” [16, pg. 25-52], reproduced here by the courtesy of Michael Singer.

**Definition 1.5.1.** Analytic Function
Let $\Omega \subset \mathbb{C}$ be an open set. We say a function $f : \Omega \to \mathbb{C}$ is *analytic*, if for each point

$z_0 \in \Omega$, there exists some open disc $D(z_0, \delta) = \{z \in \Omega : |z - z_0| < \delta\} \subset \Omega$ such that $\forall z \in D(z_0, \delta)$, $f(z)$ can be expressed as a convergent power series. i.e.

$$f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k.$$

**Definition 1.5.2.** Complex Differentiability and Holomorphic
We say a function $f : \Omega \to \mathbb{C}$ is *complex differentiable* at a point $z_0 \in \Omega$, if there exists some $M$ such that:

$$\varepsilon(h) = f(z_0 + h) - f(z_0) + Mh$$

with the property that $\varepsilon(h) = o(h)$ [7] as $h \to 0$. Then, $M$ is denoted by $f'(z_0)$. We say $f$ is *Holomorphic* in $\Omega$ if it is complex differentiable at every point in $\Omega$ .

A function which is holomorphic on the whole of complex plane is called an *Entire* function.[8]

**Theorem 1.5.1.** Maximum Modulus Principle (Priestley [17, pg. 189])
Suppose $\Omega \subset \mathbb{C}$ is a closed and bounded region. If $f : \Omega \to \mathbb{C}$ is:

- continuous in $\Omega$
- holomorphic on the interior of $\Omega$
- not a constant function

Then, $|f|$ achieves its maximum on the boundary of $\Omega$ i.e.:

$$max\{|f(z)| : z \in \Omega\} = max\{|f(z)| : z \in \partial\Omega\}$$

**Example**: Take a closed disc centered at $z_0 = 0$ of radius $r$ and $f(z) = e^z$. Then, from Theorem 1.5.1, the maximum of exponential function is attained on $|z| = r$ for all $z \in \overline{D}(0, r)$. By the increasing property of exponential function, we know this maximum occurs at $z = r$.

## 1.6 Extending Taylor Series to Function of Matrices

In this section, we create a *link* between Taylor expansion for scalars and Taylor expansion for *function of matrices*. Let us first state some useful theorems and results as covered in the book *'An Introduction to Numerical Analysis'*, Süli and Mayers [18, pg 420-421].

**Theorem 1.6.1.** Taylor's Theorem for scalars (real case)
Let $\Omega \subset \mathbb{R}$ and let $f : \Omega \to \mathbb{R}$ be $m$-times differentiable on $\Omega$ for some $m \in \mathbb{N}$. Then, given $x_0, x \in \Omega$ with $x \neq x_0$:

$$f(x) = \sum_{k=0}^{m} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + R_m(x; x_0)$$

[7]This is "Little o" notation: for functions $f, g$ defined on $\mathbb{R}$, we say $f(x) = o(g(x))$ as $x \to x_0$ if $\lim_{x \to x_0} \frac{|f(x)|}{|g(x)|} = 0$ for $\forall x_0 \in \{\mathbb{R} \cup \pm\infty\}$.

[8]Infact, the exponential function is entire and therefore any $z \in \mathbb{C}$ can be expanded into a convergent Taylor series. More on this in the next section.

where the remainder term satisfies $R_m(x; x_0) = o(|x - x_0|^m)$ as $x \to x_0$. If $f$ is $(m+1)$ - times differentiable on $\Omega$, then we express remainder as:

$$R_m(x; x_0) = \frac{f^{(m+1)}(\xi)}{(m+1)!}(x - x_0)^{m+1}$$

for some $\xi$ between $x$ and $x_0$.

If the derivatives of $f$ are continuous, then we can express the remainder term in an *integral form* by invoking the *Fundamental Theorem of Calculus* [19].

**Remark:** Although, the above theorem only looks at Taylor series in the real case, we can generalise this to a complex case $f : \mathbb{C} \to \mathbb{C}$ provided $f$ is holomorphic in some open set $\Omega \subset \mathbb{C}$. Then, by the theorem given in Rudin [20, pg. 208-213], we say $f$ is analytic in some open disc $D(z_0, \delta) \subset \Omega$ where the coefficients of the convergent Taylor expansion are $\frac{f^{(n)}(z_0)}{n!}$ which are derived by *Cauchy Integral Formula* [17, chap. 13][9].

Now we are ready to make the link by using the theorem in Higham [15, pg. 76].

**Theorem 1.6.2.** Convergence of *Matrix* Taylor series
Suppose $f : \mathbb{C} \to \mathbb{C}$ is analytic around $z_0$:

$$f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k \quad \text{where} \quad a_k = \frac{f^{(k)}(z_0)}{k!}$$

in some open disc $D(z_0, r)$ where $r > 0$ is the *radius of convergence*. If $A$ is a $n \times n$ complex square matrix, then $f(A)$ also has a convergent Taylor expansion given by:

$$f(A) = \sum_{k=0}^{\infty} a_k \left(A - z_0 I\right)^k$$

if, and only if, the *distinct* eigenvalues of $\lambda_i \in \sigma(A)$ satisfies one of the following conditions:

a) $|\lambda_i - z_0| < r$.
b) $|\lambda_i - z_0| = r$ and the series for $f^{(n_i - 1)}(\lambda_i)$ is convergent where $n_i$ is the dimension of the largest Jordan block of $\lambda_i$ in Jordan Canonical Form of $A$.

*Proof*: The proof is given in Higham [15, pg. 76] which we omit here.

From this theorem, we see a clear link between scalar and matrix Taylor expansion; as long as eigenvalues of $A$ are within the radius of convergence (i.e $\lambda \in D(z_0, r)$ for all $\lambda \in \sigma(A)$), then we have a sufficient condition for convergence of matrix Taylor series of $f(A)$.

[9] we use the Cauchy Estimates: $|f^{(k)}(z_0)| \leq \frac{k!M}{\delta^k}$ where $|f(z)| \leq M$ for all $z \in D(z_0, \delta)$ and use this to show that $f(z) = \sum_{k=0}^{\infty} \frac{f^{(k)}(z_0)}{k!}(z - z_0)^k$ is a convergent series where $f^{(k)}(z_0) = \frac{k!}{2\pi i} \oint_\gamma \frac{f(t)}{(t - z_0)^{k+1}} dt$.

**Example**:

1) The scalar exponential function is expressed as $\exp(z) = \sum_{k=0}^{\infty} \frac{z^k}{k!}$ centered around 0. Since $\exp(z)$ is an entire function (since radius of convergence $r = \infty$), this implies it is analytic on $\mathbb{C}$. Then, by Theorem 1.6.2, we can extend it to the matrix case; i.e. for any $A \in \mathbb{C}^{n \times n}$:

$$\exp(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$$

is a convergent series.

2) The logarithmic function $\log(1+z) = \sum_{k=1}^{\infty} (-1)^{(k+1)} \frac{z^k}{k}$ converges if $|z| < 1$. Therefore:

$$\log(I + A) = A - \frac{A^2}{2} + \frac{A^3}{3} - \dots \tag{1.3}$$

is convergent if the *spectral radius* [10] of $A$ is strictly less than 1 ($\rho(A) < 1$).

Note, this theorem assumes that the computations are performed in exact arithmetic with no rounding errors. If there are rounding errors (which always happens in real life computations), then these properties are not guaranteed. We will see in the next chapter, that even when approximating an entire function such as exponential function, we run into trouble due to floating point arithmetic.

[10]Spectral radius of A is defined as the largest eigenvalue of A by magnitude i.e. $\rho(A) = max\{|\lambda| : \lambda \in \sigma(A)\}$.

# Chapter 2

# Approximating the Exponential Function

## 2.1 Why not just use *plain* Taylor Expansion?

In Section 1.6, we showed that we can express matrix exponential for any $A \in \mathbb{C}^{n \times n}$ as a Taylor series by invoking Theorem 1.6.2:

$$\exp(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$$

A natural first idea might be to approximate exponential function by simply *truncating* the Taylor expansion of $\exp(A)$ into $(m + 1)$ terms where $m \in \mathbb{N}$ is the ***truncation parameter***. We approximate

$$\exp(A) \approx T_m(A) = \sum_{k=0}^{m} \frac{A^k}{k!} \tag{2.1}$$

where $T_m \in \mathcal{P}_m$ and we state it as the ***plain Taylor*** approximation of $\exp(A)$. The truncation parameter $m$ is chosen such that adding another term to this series doesn't change the value stored on a computer. In other words, we find a $m$ such that $fl[T_m(A)] = fl[T_{m+1}(A)]$. This might seem like a perfectly valid way of approximating exponential function, but due to floating point arithmetic, we get a pretty poor approximation even in the scalar case as discussed in the book by Moler [21, pg. 10-23]. We demonstrate this by an example:

Let:

$$T_m(z) = \sum_{k=0}^{m} \frac{z^k}{k!}$$

where $m$ is picked using above condition. Choose $z = -40$. The Figure 2.1 is a semi-log plot of coefficients' order $|(-40)^k/k!|$ against the index of the terms, and we use it to compute $m$. Therefore, for a machine precision of $\epsilon_{mach} \approx 1 \times 10^{-16}$, we choose our truncation parameter to be $m = 138$. The Python code shows the extent of the errors:

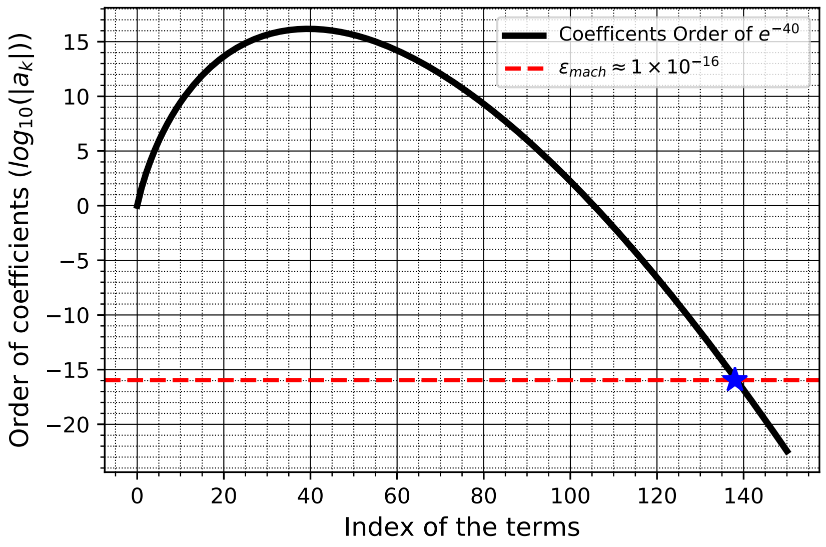


Figure 2.1: Semi-log plot highlighting the order of coefficients of truncated Taylor expansion of $\exp(-40)$ against the index $k$. It helps us to determine the truncation parameter $m$ shown by a blue star.

```
import numpy as np
m = 138; z = -40
#Calculating the coefficients of exp of the form a_k = z^k/k!
coeffs = [z**k/factorial(k) for k in range(0,m+1)]
print(f'Approximation: {sum(coeffs)}')
print(f'Expected Value: {np.exp(z)}')
print(f'Relative Error: {abs(sum(coeffs)-np.exp(z))/abs(np.exp(z))}')
>> Approximation: -3.880859375
>> Expected Value: 4.248354255291589e-18
>> Relative Error: 9.134971195413257e+17
```

Listing 2.1: Highlighting the issues with plain Taylor approximation of $\exp(z)$

The computed value not only has a wrong sign but it is also off by $O(10^{17})$ in relative terms. In particular, if you consider the terms $\frac{z^{39}}{39!}$ and $\frac{z^{40}}{40!}$, they have the values $-1.4816805.. \times 10^{16}$, $1.4816805.. \times 10^{16}$ respectively. I.e they have the same magnitude but different signs so we expect them to cancel each other out. But since the error is larger than the accuracy on a double precision computer, we get *catastrophic cancellations* giving us terrible approximation in floating point arithmetic. Same can be shown for matrix case $A \in \mathbb{C}^{n\times n}$ which is highlighted in the paper [8, pg. 10] where Moler and Loan consider the matrix:

$$A = \begin{bmatrix} -49 & 24 \\ -64 & 31 \end{bmatrix} \tag{2.2}$$

and show that under machine precision of $\epsilon_{mach} \approx 1 \times 10^{-6}$ and truncation parameter of $m = 59$, the approximation and exact value are the following:

$$T_m(A) = \begin{bmatrix} -22.25880 & -1.432766 \\ -61.499931 & -3.474280 \end{bmatrix} \qquad \exp(A) = \begin{bmatrix} -0.735759 & 0.551819 \\ -1.471518 & 1.103638 \end{bmatrix}.$$

This again leads to a poor approximation of $e^A$. Observe that, in approximating exponential function, we encounter two types of errors: ***truncation*** error because we are using a finite number of terms of its Taylor expansion and ***rounding*** error due to limited machine precision. We urge the reader to have a good understanding of these errors and be able to distinguish between them.

In the next section, we will try to show why the plain Taylor approximation is inadequate by doing rounding error analysis.

## 2.2 Rounding Error Analysis

As mentioned in Ghufran [22, pg. 111-112], we try to provide some bounds on the rounding error. For simplicity, we only derive for the real case[1] $x \in \mathbb{R}$. Let $fl(x)$ be the floating point representation of $x$ and from Section 1.3, we can express $fl\left(\frac{x^k}{k!}\right) = \frac{x^k}{k!}(1+\varepsilon_k)$ where $|\varepsilon_k| \leq (2k-1)\cdot \epsilon_{mach}$[2]. Since $k \leq m$, we just express:

$$|\varepsilon_k| \leq 2m\epsilon_{mach}.$$

Therefore, considering the absolute error:

$$\begin{aligned}
\left|fl[T_m(x)] - T_m(x)\right| &= \left| fl\left[\sum_{k=0}^{m} \frac{x^k}{k!}\right] - \sum_{k=0}^{m} \frac{x^k}{k!}\right| \\
&\leq \sum_{k=0}^{m} \left| fl\left[\frac{x^k}{k!}\right] - \frac{x^k}{k!}\right| \\
&= \sum_{k=0}^{m} \left|\frac{x^k}{k!}\varepsilon_k\right| \\
&\leq 2m\epsilon_{mach} \sum_{k=0}^{m} \frac{|x|^k}{k!} \\
&\leq 2m\epsilon_{mach} e^{|x|}.
\end{aligned}$$

Therefore, for sufficiently large $m$, the relative rounding error becomes:

$$\begin{aligned}
\frac{\left|fl[T_m(x)] - T_m(x)\right|}{|T_m(x)|} &\leq \frac{2m\epsilon_{mach} e^{|x|}}{|T_m(x)|} \\
&\approx \frac{2m\epsilon_{mach} e^{|x|}}{|e^x|}
\end{aligned}$$

$$\frac{\left|fl[T_m(x)] - T_m(x)\right|}{|T_m(x)|} \leq \begin{cases} 2m\epsilon_{mach} & \text{if } x \geq 0 \\ 2m\epsilon_{mach} e^{2|x|} & \text{if } x < 0. \end{cases} \tag{2.3}$$

[1] When we write just $x$, we assume that we can express $x$ in exact point arithmetic, i.e. we have infinite memory at our disposal.

[2] the multiplicative constant $(2k-1)$ follows from the fact that we do $k-1$ powers of $x$ and divide $x$, $k$ times for each term in the truncated Taylor expansion of $\exp(z)$.

From Equation 2.3, we see that for non-negative values of $x$, the relative error is just a function of $m$ and grows linearly. However, for negative values of $x$, we see that relative error grows exponentially and can be very large as $x \to -\infty$. Therefore, large rounding errors are observed when trying to approximate $e^x$ for large negative numbers.

For a $z \in \mathbb{C}$, the relative rounding error is shown to be:

$$\frac{\left|fl[T_m(z)] - T_m(z)\right|}{|T_m(z)|} \leq 2m\epsilon_{mach}e^{|z|-\mathrm{Re}(z)} \tag{2.4}$$

where $Re(z)$ represents real component of $z$. Suppose, we are considering the rounding error in Equation 2.4 for $z \in \overline{D}(0, R)$ where $R > 0$. Then, by maximum modulus principle 1.5.1, the error is maximised at the boundary, i.e. for $|z| = R$. But by the above inequality and by the increasing property of exponential function, Equation 2.4 is maximised for $z = -R$. This result comes to some use in Section 2.7.

**Remark**: The Section 2.1 and Section 2.2 gives us an indication that even though exponential function is entire, and thus for all $z \in \mathbb{C}$ (including matrices $A \in \mathbb{C}^{n \times n}$), we should be able to compute $\exp(z)$ accurately; ***the difficulty we are observing is not with truncating the Taylor series, but it is with truncating the floating point version of Taylor series i.e*** $fl[T_m(z)]$. We are getting numerical roundoff errors when magnitude of $z$ is large. Equivalently when the entries of $A$ are large or the matrix norm $||A|| >> 1$, causing entries of $A^k$ to be larger than the digits of accuracy. Since, we are stuck with finite machine precision, all we can do is purposely *scale* the problem so that $|z|$ is *closer* to the origin (similarly $||A|| = O(1)$ or eigenvalues of $A$ are shifted to be near origin) and then compute the truncated Taylor series of that; which is re-scaled back up again. This is precisely the intuition behind Scaling and Squaring Method.

## 2.3 Scalar and Squaring Method (SSM)

In the last century, various methods have been introduced for computing $\exp(A)$ as described in the paper by Moler, Loan [8]. In this report, we will just focus on the Squaring and Scaling Method (SSM) based on composing polynomials as described in Bader [23].

The way SSM works is by exploiting the special property of exponential functions:

$$e^A = (e^{A/\sigma})^\sigma$$

where $\sigma \in \mathbb{R}_{\geq 0}$ is typically taken to be power of two, i.e. $\sigma = 2^s$ and $s \in \mathbb{N}$ is known as the ***scaling parameter***. Then, $\exp(A)$ can be computed reliably and accurately by repeated squaring:

$$e^A = \left(e^{A/2^s}\right)^{2^s} = \underbrace{\left(\dots\left(e^{A/2^s}\right)^{2^{\cdot^{\cdot^{\cdot}}}}\right)^2}_{s-times} = \underbrace{q\Big(q\Big(\dots q}_{s-times}\left(e^{A/2^s}\right)\Big)\Big) \tag{2.5}$$

where function $q(M) = M^2$ for a given matrix (same holds for scalar). The $\exp(A/2^s)$ is usually replaced by a polynomial approximation, for example the $m^{th}$-degree Taylor expansion $T_m(A/2^s) \in \mathcal{P}_m$ or as discussed in majority of the literature, the rational approximations typically, the $[k/m]$-order *Padé Approximation*. Although, approximating exponential function with Padé polynomial is a fascinating area which is covered in depth in Higham [24, 25] and Sastre [26], in this report, we solely consider from the perspective of ***composite Taylor polynomials*** i.e.

$$e^A \approx \left[T_m(A/2^s)\right]^{2^s} \tag{2.6}$$

where:

- *Degree* of a composite Taylor polynomial = $2^s m$.
- *Degrees of Freedom* = $(m+1) + 3s$.
- *Complexity* of the SSM = $O\big((m+s)n^3\big)$ which will be derived in the next section.

We choose $s \in \mathbb{N}$ such that the norm of $A$, i.e. $\|A/2^s\| = O(1)$; to be more precise, we take smallest $s$, such that $\|A/2^s\| \leq 1/2$. [3]With this restriction, Equation 2.6 is computed which is one of the most effective algorithms we know. The pseudo-code of the SSM is described below.

**Algorithm 1:** Scaling and Squaring Algorithm with Taylor Approximation

```
1. Initialisation: Pick a truncation parameter $m$ and choose $s$ such that $\|A\|/2^s \leq 1/2$
   (i.e. $s = \lceil \log_2(\|A\|) + 1 \rceil$).
2. Compute the truncated Taylor expansion of $A/2^s$ of degree $m$ such that
   $T_m\left(\frac{A}{2^s}\right) = \sum_{k=0}^{m} \frac{A^k}{(2^{sk})k!}$.
3. Approximate $\exp(A)$ by repeated squaring which is carried out $s$ times.
```

A more detailed algorithm of scaling and squaring method is covered in Higham [15, pg. 246-247] which explains tricks about how to reduce matrix multiplication cost[4] which we won't discuss here.

Let us fix our $s$ as described in the algorithm and choose $m$ as in Section 1.3 such that $fl[T_m(A/2^s)] = fl[T_{m+1}(A/2^s)]$. We will reconsider the same matrix $A$ as described in the Equation 2.2. Here, we write a short code in Python where we use the Euclidean matrix norm for our analysis and take $\epsilon_{mach} = 1 \times 10^{-16}$. I have purposely excluded some technical details to have a concise code:

```python
n = 2; p = 2; tol = 0.5*np.spacing(1) #dimension, 2-norm, machine precision
A = np.array([[-49,24],[-64,31]])  # the same matrix as Eqn 2.2

norm_A = norm(A,ord = p) #calculating 2-norm
s = int(np.ceil(np.log2(norm_A)+1))  # by solving ||A||/2^s <= 1/2

# doing truncated Taylor expansion until above condition is met
B = A/2**s
k = 0
M0 = np.eye(n)  #identity matrix
```

[3]We make this assumption because this is a sufficient condition to carry out error analysis as in Theorem A.0.5.

[4]Although the analysis in *Functions of Matrices* is on Padé rational functions, same procedure can be applied to Taylor polynomials.

```
while norm(matrix_power(B,k),ord = p)/factorial(k) > tol:
    k+=1
    M0 = np.add(M0,matrix_power(B,k)/factorial(k))

M = matrix_power(M0,2**s)  #composite Taylor approximation
exact_M = expm(A)  #exact matrix exponential
error = norm(M-exact_M,ord=p)/(norm(exact_M,ord = p)) #relative error
correct_digits = -int(round((np.log10(error))))-1 #accurate to number of digits

>> The value of squaring size s: 8
>> Taylor Series has degree m = 9

>> Actual e^A using expm function:
    [[-0.73575876  0.5518191 ]
    [-1.4715176   1.10363824]]

>> Computed SSM Estimate :
    [[-0.73575876  0.5518191 ]
    [-1.4715176   1.10363824]]

>> Relative Forward Error: 1.651983806142441e-13
>> Approx correct to: 12 digits
```

Listing 2.2: Computing the matrix exponential of $A$ given in Equation 2.2 by SSM which highlight how powerful it is.

From the output of the code, we see the advantage of SSM over plain Taylor. By just purposely scaling the problem, we get an accuracy of 12 digits.

In Appendix A, we derive the upper bound on the relative backward error; provided $\|A\|/2^s < 1/2$ and get the following:

$$T_m\left(\frac{A}{2^s}\right) = e^{A+E} \quad \text{and} \quad \frac{\|E\|}{\|A\|} \leq 4\left(\frac{\|A\|}{2^s}\right)^m \frac{1}{(m+1)!} \tag{2.7}$$

where $E$ is some *perturbation* matrix. From Equation 2.7, we see that relative backward error increases as the matrix norm $\|A\|$ increases. So inorder to reduce this error, the upper bound tells that we can either:

a) increase the choice of truncation parameter $m$ or,
b) we can use a large enough choice of $s$ so that $\left\|\frac{A}{2^s}\right\|$ is reduced.

This *"Inverse Error Analysis"* helps us to choose the optimal values of $(m, s)$ to reduce the relative forward error. So suppose we choose our error tolerance $\varepsilon$ (it is defined by the user) where $\varepsilon \geq \epsilon_{mach}$, then we can find many pairs of $(m, s)$ which satisfy:

$$\frac{\|E\|}{\|A\|} \leq \varepsilon. \tag{2.8}$$

Since the composite approximation $\left[T_m\left(\frac{A}{2^s}\right)\right]^{2^s}$ requires about $(m+s)n^3$ flops as covered in Section 2.4, it is wise to choose optimum pairs which will minimize the sum $(m+s)$. We can use relative backward error to derive relative forward error using the fact that

$EA = AE$ (see Appendix A). Then, by Equation 2.7 and Theorem 1.4.1:

$$\begin{aligned}\frac{\left\|\left[T_m\left(\frac{A}{2^s}\right)\right]^{2^s} - e^A\right\|}{\|e^A\|} &= \frac{\|e^{A+E} - e^A\|}{\|e^A\|} \\ &\le \frac{\|e^A\|.\|(e^E - I)\|}{\|e^A\|} \\ &= \|e^E - I\| \\ &\le \|E\|e^{\|E\|} \\ &\le \varepsilon\|A\|e^{\varepsilon\|A\|}.\end{aligned}$$

**Relative Forward Error**:

$$\boxed{\frac{\left\|\left[T_m\left(\frac{A}{2^s}\right)\right]^{2^s} - e^A\right\|}{\|e^A\|} \le \varepsilon\|A\|e^{\varepsilon\|A\|}.} \tag{2.9}$$

So if we choose the tolerance $\varepsilon$ to be sufficiently small for a matrix $A \in \mathbb{C}^{n\times n}$ with large entries, then the forward error gives us a meaningful bound to approximate matrix exponential for a given value of $(m, s)$ such that the conditions in Theorem A.0.5 are satisfied. We do a thorough analysis like this for the scalar case in Section 2.7 where we compute its optimal parameters.

## 2.4 Complexity of the SSM

In this section, we will derive the complexity of SSM for matrix case. Before, we do this, we need to understand some terminology: *Flop* is an acronym for a floating point operation, for example $+, -, \times, \div$ etc. It is useful to measure flop when we are comparing different algorithms for the same problem and want to measure their relative speeds. It is said that an algorithm for a problem of size $n$ has *polynomial time* complexity if it requires at most $O(p(n))$ operations to complete its task, where $p \in \mathcal{P}_m$.

Calculating the complexity of SSM for a $n\times n$ matrix $A$, requires us to split the algorithm into 3 parts and calculating their complexity separately:

1. Matrix multiplication $A^2, \dots, A^m$.
2. Dividing the matrix by $2^s$ and then summing up linear combination of $m+1$ terms i.e. $\sum_{k=0}^{m}\left(\frac{A}{2^s}\right)^k \frac{1}{k!}$.
3. Composition of the truncated Taylor polynomial i.e. $q\Big(q\Big(\dots q\Big(T_m(A/2^s)\Big)\Big)\Big)$ where $q(M) = M^2$.

1. Suppose we want to calculate $A^2$ which we will represent by matrix $C$, then we can compute each entry of $C$ by doing the dot product $c_{ij} = \sum_{k=1}^{n} a_{ik}a_{kj}$. Each $c_{ij}$ requires $n$ flops for scalar multiplication and $(n-1)$ for addition. So for computation of $C$,

we need $(2n-1)n^2 = 2n^3 + O(n^2) = O(n^3)$ flops. We still need to calculate the complexity of $A^k$ for $k > 2$, i.e. $A^3, \dots, A^m$. We assume that the *Paterson-Stockmeyer* Method [27, 28] is implemented in SSM which is an optimal algorithm for evaluating polynomial of matrices by storing $A^{k-1}$ and calculating $A^k$ by $AA^{k-1}$. So in order to compute $A^2, \dots, A^m$, we need about $2(m-1)n^3$ operations.

2. We only have to divide the matrix $A$ by $2^s$ once at the beginning as we can just define $B = A/2^s$ and then continue with the matrix product as above. This requires only $n^2$ flops. So now when summing linear combination of $I, A, A^2, \dots, A^m$, this requires overall $\underbrace{2(m-1)n^3}_{\text{from 1.}} + n^2 + (m-1)n^2 = 2(m-1)n^3 + O\left(n^2\right)$ operations.
3. Since the function $q$ is quadratic when we do $s$-compositions, all we are performing are $s$ matrix multiplications. This implies we need $2n^3 s$ operations for the composition alone.

So overall, the scaling and squaring method requires about $2(m-1)n^3 + 2n^3 s = 2(m+s-1)n^3 = O\left((m+s)n^3\right)$ flops.

$$\boxed{\text{Number of flops for the SSM : } O\left((m+s)n^3\right).} \tag{2.10}$$

Therefore, SSM can be approximately computed in polynomial time even though, we carry out $s$ compositions. So a composition Taylor polynomial can have a relatively small degrees of freedom but a geometrically growing degree. This is advantageous because with limited matrix multiplications, we can approximate $\exp(A)$ to a high level of accuracy. See Section 2.7 for more explanation for the scalar case.

## 2.5 Analysis of the SSM using Contour Plots

Suppose matrix $A \in \mathbb{C}^{n\times n}$ is *diagonalizable*, then its eigenvalue decomposition can be written as:

$$A = X\text{diag}(\lambda_i)X^{-1} \qquad \lambda_i \in \sigma(A), \quad i = 1, \dots, n.$$

Then we can express:

$$e^A = X\text{diag}(e^{\lambda_i})X^{-1}$$

and its composite Taylor approximation as shown in Equation 2.6:

$$Y = X\text{diag}\left(\left[T_m(\lambda_i/2^s)\right]^{2^s}\right)X^{-1} \qquad T_m \in \mathcal{P}_m.$$

Then the absolute error under the Euclidean norm is:

$$\begin{aligned}\|e^A - Y\|_2 &\leq \|X\|_2\|X^{-1}\|_2 \left\|\text{diag}\left(e^{\lambda_i} - \left[T_m\left(\lambda_i/2^s\right)\right]^{2^s}\right)\right\|_2 \\ &\leq \kappa(X) \max_{i=1:n} \left|e^{\lambda_i} - \left[T_m\left(\lambda_i/2^s\right)\right]^{2^s}\right|\end{aligned}$$

$$\|e^A - Y\|_2 \leq \kappa(X) \max_{z\in\mathcal{D}} \left|e^z - \left[T_m\left(z/2^s\right)\right]^{2^s}\right| \tag{2.11}$$

where $\mathcal{D}$ is the region in the complex plane that contains all the eigenvalues of $A$ $\left(\lambda_i \in \mathcal{D}\right)$ and $\kappa(X) = \|X\|_2\|X^{-1}\|_2$ is known as the *condition number* of the eigenvector matrix $X$. We will assume that $X$ is well-conditioned and $\kappa(X)$ is sufficiently small so that it doesn't deteriorate the bound in Equation 2.11. When matrix $A$ is normal[5], $X$ becomes unitary and the term $\max_{i=1:n}\left|e^{\lambda_i} - \left[T_m\left(\lambda_i/2^s\right)\right]^{2^s}\right|$ represents the upper bound on the absolute error.

Let us focus on the analysis of $\max_{i=1:n}\left|e^{\lambda_i} - \left[T_m\left(\lambda_i/2^s\right)\right]^{2^s}\right|$ which we will examine by looking at $\max_{z\in\mathcal{D}}\left|e^{z} - \left[T_m\left(z/2^s\right)\right]^{2^s}\right|$. As long as this error is small in the region $\mathcal{D}$, then we say $Y$ gives a sufficiently good approximation of $e^A$. Same argument can be given for a general matrix $A$ by decomposing it into its *Jordan Blocks* as shown in Fair, Luke [29] which we omit here for simplicity.

We visualize this error by semi-log contour plots where we fix the degrees of the two polynomials ($2^s m$) for fair comparison and take two choices of $s$:

1. $s = 0$ and $m = 160$ for the plain Taylor case (Figure 2.2).
2. $s = 3$ and $m = 20$ for the composite Taylor case (Figure 2.3).

We set the region $\mathcal{D} = \left\{z \in \mathbb{C} \,:\, Re(z), Im(z) \in [-70, 70]\right\}$ and measure both absolute and relative errors:

$$\left|e^{z} - \left[T_m(z/2^s)\right]^{2^s}\right| \qquad\qquad \frac{\left|e^{z} - \left[T_m(z/2^s)\right]^{2^s}\right|}{|e^z|}.$$

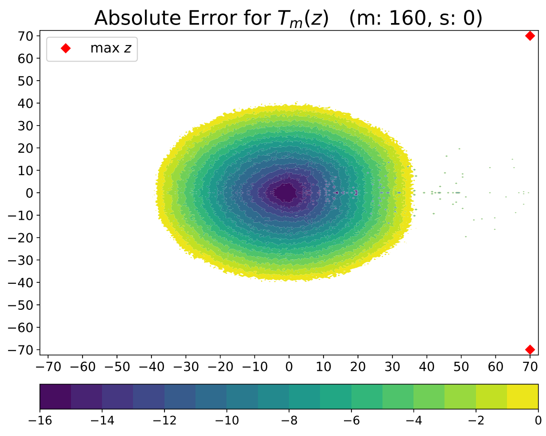
The red diamond in the graphs highlight the point(s) where respective errors are maximised. As we are using the Taylor approximation centered at $z = 0$, the error is zero there (theoretically) and grows as $|z|$ increases. Note, the white region in all four plots indicate the errors which are outside the range of the colours bars and too large to do any analysis on.

In absolute terms, the Figure 2.2a and 2.3a highlight the power of SSM; the region in plain Taylor where we have any digits of accuracy is the small blue region around zero whereas for composite polynomials, it stretches in the left complex plane. Thus, we can infer that plain Taylor does not do a good job globally in absolute terms. However in relative terms (i.e. Figure 2.2b), plain Taylor is better approximant than composite polynomial in positive real axes which is partially due to the fact that in plain Taylor, we are using a very large truncation parameter $m = 160$ leading to a better approximation especially when rounding errors are lower for these values of $z$ (as shown in the inequality 2.4).

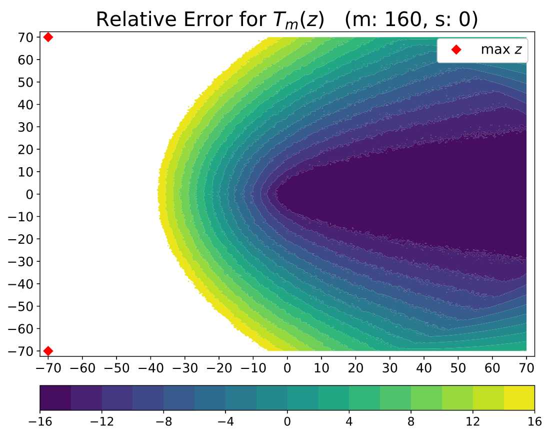
*Absolute Error in Composite Taylor*

Looking at Figure 2.3a, we realise that the plot is highly non-symmetric about the imaginary axis; the absolute error is larger for $z \in \mathcal{D}$ with $Re(z) >> 0$ and small for $Re(z) << 0$. Infact, it is maximized for $z = 70 \pm 70i$ i.e. maximum real value in our region $\mathcal{D}$. The

[5]We say a matrix $A$ is normal, if $A^*A = AA^*$ where $A^* = (\overline{a_{ij}})$ is the conjugate transpose. It has a special property that eigenvector matrix $X$ of $A$ is unitary, i.e. $A = X diag(\lambda_i)X^*$ and $X^*X = I$.

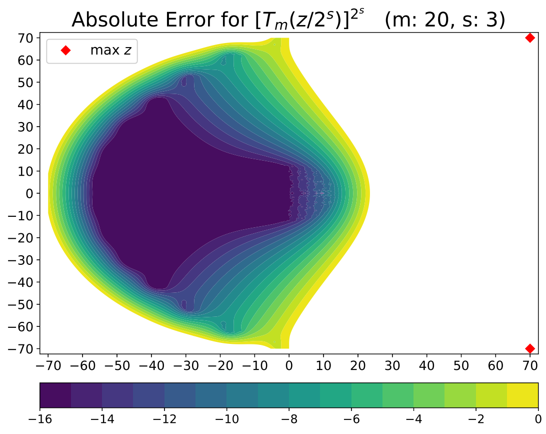


(a) Absolute Error

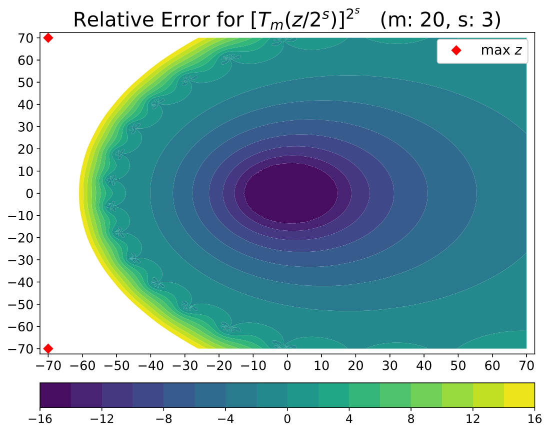


(b) Relative Error

Figure 2.2: Errors for *plain Taylor* $T_m(z)$: $m = 160, s = 0$



(a) Absolute Error



(b) Relative Error

Figure 2.3: Errors for *composite Taylor* $[T_m(z/2^s)]^{2^s}$: $m = 20, s = 3$

region in which error is roughly of the order $O(10^{-16})$ stretches up to $Re(z) \approx -60$. This result is expected and we justify this by doing similar analysis to the paper by Güttel, Nakatsukasa [30, pg 6]. Since $|e^z| = e^{Re(z)}$, then as $Re(z) \to -\infty$, we see exponential function decaying to zero. So we say $|e^z| \approx O(10^{-16})$[6] if, $Re(z) \leq \ln(10^{-16}) \approx -36.8$. Then, for $z \in \mathcal{D}$ with $Re(z) \leq -36.8$, we can approximate absolute error by:

$$|T_m(z/2^s)^{2^s} - e^z| \leq |T_m(z/2^s)^{2^s}| + |e^z| \approx |T_m(z/2^s)^{2^s}|.$$

For example, take $s = 3$ and $m = 20$. Then:

$$\begin{aligned} |T_m(z/8)^8 - e^z| \leq \left|T_m(z/8)^8\right| &\approx O(10^{-16}) \\ \left|T_m(z/8)\right| &\approx O(10^{-2}) \\ \left|\sum_{k=0}^{20} \frac{z^k}{8^k k!}\right| &\approx O(10^{-2}) \\ \iff \frac{|z|^{21}}{8^{21}21!} &\approx O(10^{-2}) \qquad \text{(see Theorem 2.6.2)} \end{aligned}$$

$$|z| \approx (10^{-2} \times 8^{21} \times 21!)^{1/21} \approx 55.8. \tag{2.12}$$

So as per the above example, the composite polynomial with $m = 20$ and $s = 3$ is a good approximation of $e^z$, if $z \in \{w \in \mathcal{D} : |w| \approx 55.8 \text{ and } Re(w) < -36.8\}$ which represents a segment of a circle. This can be seen in Figure 2.3a in the dark blue region and agrees with the above computation. This analysis will be made more rigorous for general value of $m, s$ in the Section 2.6.

*Relative Error in Composite Taylor*

Focusing our attention on relative error in Figure 2.3b, we observe a more symmetric plot. The left complex plane is still poorly approximated for large negative $Re(z)$ justified by the relative error being maximised at $z = -70 \pm 70i$. This is because in relative error, you are dividing through by a very small number in magnitude, which leads to the overall error being magnified. For large positive $Re(z)$, we see the opposite effect, where the large denominator helps to bring the relative error down. Therefore, relative error to approximate $\exp(z)$ captures misinformation and perhaps is not the most reliable way to measure error (atleast for exponential functions).

So the two plots in Figure 2.3 illustrate that composing polynomials is a good approximant for different values of $z \in \mathcal{D}$ depending on whether the error is being measured in absolute or relative terms.

As discussed in the Chebfun website [31] in the context of matrix exponential, we measure the relative forward error by $\|e^A - Y\|_2 / \|e^A\|_2$. Suppose again that $A$ is a normal matrix, then the error simply becomes:

$$\frac{\max_{i=1:n} \left|e^{\lambda_i} - \left[T_m\left(\lambda_i/2^s\right)\right]^{2^s}\right|}{e^{\max_i Re(\lambda_i)}} \leq \frac{\max_{z \in D} \left|e^z - \left[T_m\left(z/2^s\right)\right]^{2^s}\right|}{e^{\max_i Re(\lambda_i)}}. \tag{2.13}$$

[6]In other words it is negligible in floating point arithmetic.

The Equation 2.13 tells us that in matrix exponential case, absolute error is the appropriate metric to measure error as the denominator is a constant and numerator varies with value of $z \in \mathcal{D}$. Then, we can say that $Y$ is a good approximation of $e^A$ provided the eigenvalues of $A$ lies in the blue region of Figure 2.3a.

## 2.6 Best Approximation Region

*All the analysis carried out in this section is the author's own research.*

In this section, we generalise the best approximation region of $e^z$ by composite polynomials in *absolute* sense for arbitrary $m, s \in \mathbb{N} \cup \{0\}$ and machine precision $\epsilon_{mach}$.

Before diving straight in, we need to prove some useful results.

**Lemma 2.6.1.** Let $z \in \mathbb{C}$ and $T_m \in \mathcal{P}_m$ be the plain truncated Taylor expansion of $e^z$ centered around 0. Then we have the following identity:

$$\begin{cases} e^z - 1 = z\int_0^1 e^{tz}\, dt & \text{if } m = 0 \\ e^z - T_m(z) = z\int_0^1 e^{tz} - T_{m-1}(tz)\, dt & \text{if } m \geq 1. \end{cases}$$

*Proof*:
Note: $z\int_0^1 e^{tz}\, dt = \left[e^{tz}\right]_0^1 = e^z - 1$ and

$$\begin{aligned} z\int_0^1 T_{m-1}(tz)\, dt &= z\sum_{k=0}^{m-1} \frac{z^k}{k!}\int_0^1 t^k\, dt \\ &= \sum_{k=0}^{m-1} \frac{z^{k+1}}{(k+1)!} \\ &= \sum_{k=1}^{m} \frac{z^k}{k!} \qquad \text{(re-indexing)} \\ &= T_m(z) - 1. \end{aligned}$$

Therefore, combining the above two expressions gives:

$$z\int_0^1 e^{tz} - T_{m-1}(tz)\, dt = e^z - 1 - [T_m(z) - 1] = e^z - T_m(z). \qquad \square$$

The next theorem is author's own which the author has proved in [32].

**Theorem 2.6.2.** Let $z \in \mathbb{C}$ and $T_m(z) = \sum_{k=0}^{m} \frac{z^k}{k!}$. Then, the absolute truncation error is given by:

$$\left|e^z - T_m(z)\right| \leq \frac{|z|^{m+1}}{(m+1)!} \max\{1, e^{Re(z)}\}. \tag{2.14}$$

In the real case, this is simply the remainder term of the Taylor expansion of $exp(x)$ to $(m+1)$ terms. However in the complex case, it gets a bit more tricky and the proof is not available in any of the textbooks author went through. This is why we decided to include it here:

*Proof by Induction*:
1) Consider the case when $m = 0$, then using the result in 2.6.1, we have:

$$e^z - 1 = z\int_0^1 e^{tz}\,dt.$$

Then:

$$|e^z - 1| \leq |z|\int_0^1 |e^{tz}|dt = |z|\int_0^1 e^{tRe(z)}dt \leq |z|\cdot\max\{1, e^{Re(z)}\}\int_0^1 dt = |z|\cdot\max\{1, e^{Re(z)}\}.$$

We used the fact that $\exp(z)$ is an increasing function and $e^{Re(z)} < 1$ when $Re(z) < 0$.

2) Assume that Equation 2.14 holds for all truncation parameter up to and including $m-1$:

$$\left|e^z - T_{m-1}(z)\right| \leq \frac{|z|^m}{m!}\max\{1, e^{Re(z)}\}.$$

Let us now consider $|e^z - T_m(z)|$:

$$\begin{aligned}|e^z - T_m(z)| &\leq |z|\int_0^1 \left|e^{tz} - T_{m-1}(tz)\right| dt\\ &\leq \frac{|z|^{m+1}}{m!}\max\{1, e^{Re(z)}\}\int_0^1 t^m\,dt\\ &= \frac{|z|^{m+1}}{(m+1)!}\max\{1, e^{Re(z)}\}. \qquad \square\end{aligned}$$

So now focusing on the original problem, in Section 2.5, we showed that $|e^z| = O(10^{-16}) \iff Re(z) < -36.8$. So, now generalising it to any machine precision, $|e^z| = O(\epsilon_{mach}) \iff Re(z) < \ln(\epsilon_{mach}) < 0$ provided $\epsilon_{mach} < 1$ which is always the case in practice since otherwise we will have no digits of accuracy. So, if we restrict our choice of $z \in \mathbb{C}$ such that $Re(z) < \ln(\epsilon_{mach})$, then $\max\{1, e^{Re(z)}\} = 1$. Then, the truncation error for a point $z/2^s$, where $s$ is the scaling parameter becomes:

$$\left|e^{z/2^s} - T_m(z/2^s)\right| \leq \frac{|z|^{m+1}}{2^{s(m+1)}(m+1)!}.$$

Since we choose $z \in \mathbb{C}$ such that $Re(z) < \ln(\epsilon_{mach})$, then in the floating point arithmetic, our absolute error becomes:

$$\begin{aligned}\left|e^z - \left[T_m(z/2^s)\right]^{2^s}\right| &\leq |e^z| + \left|T_m(z/2^s)\right|^{2^s}\\ &\approx \left|T_m(z/2^s)\right|^{2^s}\\ &\leq \left[\frac{|z|^{m+1}}{2^{s(m+1)}(m+1)!}\right]^{2^s}.\end{aligned}$$

So in order to have the absolute error to be of order $O(\epsilon_{mach})$, it suffices to find $z \in \mathbb{C}$ such that:

$$\left[\frac{|z|^{m+1}}{2^{s(m+1)}(m+1)!}\right]^{2^s} \leq \epsilon_{mach} \quad \text{and} \quad Re(z) < \ln(\epsilon_{mach})$$

$$\iff$$

$$|z| \leq \left[\epsilon_{mach}^{2^{-s}} \cdot 2^{s(m+1)}(m+1)!\right]^{\frac{1}{m+1}} = r_{s,m} \quad \text{and} \quad Re(z) < \ln(\epsilon_{mach}). \tag{2.15}$$

So the region in $\mathbb{C}$ where we have the approximation of $O(\epsilon_{mach})$ by composite polynomials in absolute term is:

$$\boxed{\mathcal{R} = \left\{z \in \mathbb{C} \, : \, |z| \leq r_{s,m} \quad \text{and} \quad Re(z) < \ln(\epsilon_{mach})\right\}.} \tag{2.16}$$

where the region $\mathcal{R}$ represents a circular segment in the complex plane. Therefore in the matrix case as shown in Equation 2.13, we say that composite Taylor approximates $\exp(A)$ well for an arbitrary choice of $m$, $s$ and $\epsilon_{mach}$, if $A$ is well-conditioned and its the eigenvalues lie in $\mathcal{R}$ (i.e. $\sigma(A) \subset \mathcal{R}$).

**Remark**: The region $\mathcal{R}$ stores the information of two types of errors (in absolute sense) which we observe when we approximate $e^z$ by the composite Taylor polynomial:

1. Truncation error: since we are approximating $e^z$ by only $2^s m$ degree polynomial.
2. Rounding error: since we only have finite memory, we can only represent a value to $\lfloor -\log_{10}(\epsilon_{mach}) \rfloor$ digits of accuracy.

So the result in this section states that if one is given a truncation parameter $m$, scaling parameter $s$ and machine precision $\epsilon_{mach}$, then one can find a set of values of $z \in \mathbb{C}$ i.e. the region $\mathcal{R}$, such that absolute error is in the order of machine precision.

This theory fits well with what we observe in practice. Let us again look at some contour plots from Section 2.5. We take our machine precision to be $\epsilon_{mach} \approx 1 \times 10^{-16}$ and take $m \in \{20, 30\}$ and $s \in \{3, 4, 5\}$. The red circular segment in Figure 2.4, 2.5, 2.6, 2.7 represent the region $\mathcal{R}$ calculated above. We can see that as $m, s$ increases, the dark blue region (which highlight absolute error of $O(10^{-16})$) gets more and more covered by the region $\mathcal{R}$. For example, in Figure 2.4, $\mathcal{R}$ only covers around 25% of blue region whereas, it is nearly 90% for Figure 2.5 when $s$ varies from 3 to 5.

**Note**: This theory only works if we have chosen sufficiently large $s$ and $m$ such that $|\ln(\epsilon_{mach})| > r_{s,m}$. This is because, if this is not true, then $\mathcal{R}$ is an empty set and Equation 2.15 does not give us any useful information about the region for $O(\epsilon_{mach})$ approximation. For example, if we take $\epsilon_{mach} \approx 1 \times 10^{-16}$, $s = 2$ and $m = 20$, then $r_{2,20} \approx 22.42 < \left|\ln(10^{-16})\right| = 36.8$ leading to $\mathcal{R} = \emptyset$.

These contour plots display the theoretically computed region $\mathcal{R}$ (red region) for different values of $m, s$ in absolute terms where the error is of the order of machine precision.

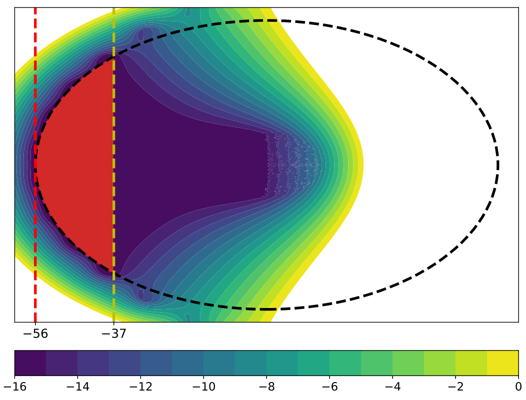


Figure 2.4: $m = 20$ and $s = 3$

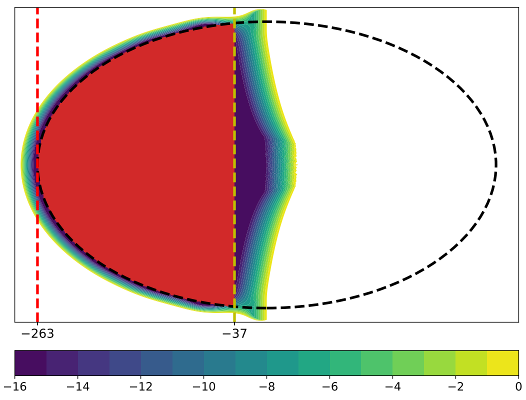


Figure 2.5: $m = 20$ and $s = 5$

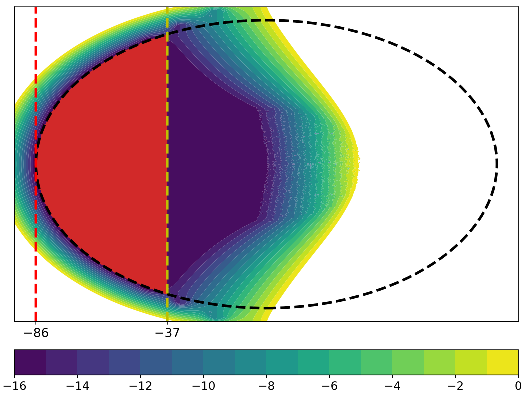


Figure 2.6: $m = 30$ and $s = 3$

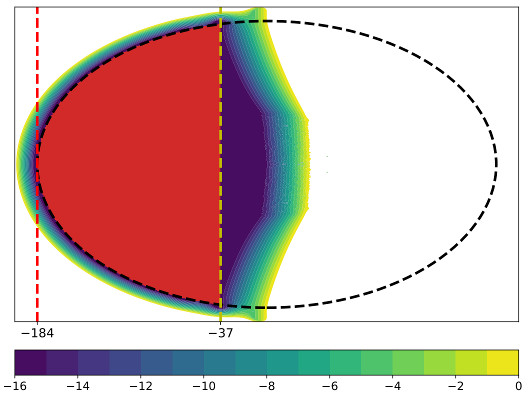


Figure 2.7: $m = 30$ and $s = 4$

## 2.7 Error Analysis for different values of $m$ and $s$

*All the analysis carried out in this section is the author's own research.*

In this section, we consider how the error (both in absolute and relative terms) between $\exp(z)$ and $\left[T_m(z/2^s)\right]^{2^s}$ varies with respect to degree and degrees of freedom (DoF)[7] of composite Taylor polynomials. We later compute the optimal choice of truncation and scaling parameters $m, s$ to reach a user-defined accuracy. We first construct an experi-

[7] to recap the definition of degree and degrees of freedom, see Section 1.1.

ment, where we fix the scaling parameter $s$. The algorithm is described below:

```
Algorithm 2: Experimentation
1. Initialisation: Pick a radius $R > 0$; pick $d, s \in \mathbb{N}$ and finally choose the
   maximum value of truncation parameter $m_{max}$;
2. Generate $d$ equispaced complex numbers $z \in \mathbb{C}$ such that $|z| = R$ and call this
   set $\mathcal{T}$. $\big($this set is defined as $\mathcal{T} = \{z_i \in \mathbb{C} : |z_i| = R\,,\ i = 1, \dots, d\}\big)$;
For $m$ in $1, \dots, m_{max}$;
    3. Calculate the composite Taylor $\left[T_m(z/2^s)\right]^{2^s} = \left[\sum_{k=0}^{m} \frac{z^k}{(2^{sk})k!}\right]^{2^s}$ for $z \in \mathcal{T}$;
    4. Calculate the maximum error in absolute or relative terms for each $m$;
       $M_m = \max\left\{\left|e^z - \left[T_m(z/2^s)\right]^{2^s}\right| : z \in \mathcal{T}\right\}$   OR
       $M_m = \max\left\{\frac{\left|e^z - \left[T_m(z/2^s)\right]^{2^s}\right|}{|e^z|} : z \in \mathcal{T}\right\}$;
5. Store all the maximum errors in $\mathcal{M} = \{M_m : m = 1, \dots, m_{max}\}$;
6. Plot the graph of $\log_{10} \mathcal{M} = \{\log_{10} M_m : m = 1, \dots, m_{max}\}$ against the degree of
   the composite polynomial $(2^s m)$ OR against the degrees of freedom $(m + 1 + 3s)$
   for each $m = 1, \dots, m_{max}$;
```

Note, by the maximum modulus principle 1.5.1, the error is maximised on the boundary of the closed disc $\overline{D}(0, R)$; this is why we have only considered looking at equispaced points on the circle of radius $R$. Therefore, this algorithm is actually equivalent to looking for the maximum point $z$ in $\overline{D}(0, R)$ (which happens to lie on the boundary).

We visualize the observations by graphs where we vary our scaling parameter $s \in \{0, \dots, 4\}$. We fix our radius $R = 20$ and take $d = 1000$. Here, we provide the numerical evidence to answer the following questions:

1. How does the degree of the composite Taylor approximant affect the maximum error? Here, we vary our truncation parameter $m \in \{1, \dots, 20\}$.
2. How does the DoF of composite Taylor approximant affect the maximum error? Here, we vary $m \in \{1, \dots, 55\}$.

We measure this maximum error over $\mathcal{T}$, in both absolute (Figure 2.8a, 2.9a) and relative (Figure 2.9a, 2.9b) terms.

### 2.7.1 Absolute Error Analysis

In Figure 2.8a, we see that plain Taylor ($s = 0$) requires the lowest $deg(T_m)$ to achieve a given absolute error which is highlighted by the steepness of the error curve. This is because, degree of plain Taylor is just $m$ and as value of $m$ increases, Taylor expansion converges to $e^z$. Since for $s > 0$, the composite Taylor degree is $2^s\tilde{m}$, we have a smaller value of $\tilde{m}$ to achieve the same level of accuracy[8].

[8]Here, $m = 2^s\tilde{m}$ and $m, \tilde{m}$ are different truncation parameters with respect to plain and composite Taylor approximants.

However, when we compare with respect to DoF in Figure 2.9a, we see the importance of SSM. As the value of $s$ increases, we require a lower DoF to get the same accuracy. Plain Taylor is more expensive because to achieve a certain tolerance, a higher value of $m$ is needed. So, if we consider the matrix exponential of $A \in \mathbb{C}^{n\times n}$, this results in higher matrix multiplications than in SSM. From Section 2.4, we know the complexity of the plain and composite Taylor for matrix exponential function is:

$$O\Big(mn^3\Big) \qquad\qquad O\Big((\tilde{m}+s)n^3\Big)$$

respectively. In plain Taylor, complexity is just a function of $m$ whereas for composite, it is a function of both $\tilde{m}$ and $s$. Letting $\tilde{m}$ and $s$ to vary means we can the reduce complexity of composite Taylor significantly and still get excellent approximation. The Figure 2.9a verifies this by looking at $s = 4$ curve which is the steepest in comparison to others.

For absolute error, we observe the maximum error at $z = R$ for all $z \in \mathcal{T}$ because exponential function is an increasing function and grows exponentially for $Re(z) > 0$. For large $m$, we start to see both set of lines flattening out in Figure 2.8a, 2.9a around $10^{-7}$. This is because, the order of $\exp(20) \approx O(10^8)$, which means the absolute error is in the order of $10^{-7}$ as we have about 15 digits of accuracy in double precision $\Big(\lfloor 16 - \log_{10}(e^{20})\rfloor\Big)$.

### 2.7.2 Relative Error Analysis

We expect the maximum relative error to be at $z = -R$ for all $z \in \mathcal{T} \subset \overline{D}(0,R)$. This is because, for $Re(z) < 0$, the exponential function is less than 1 and exponentially decays to zero as $R$ increases, leading to a negligible denominator. This causes the relative error to blow-up.

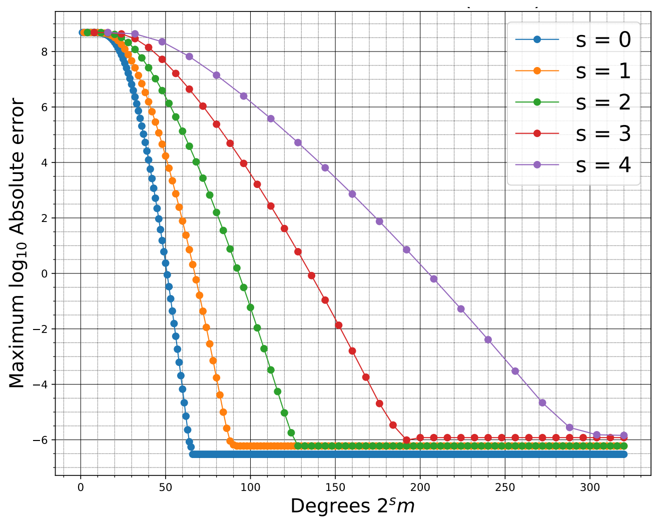
#### Relative error at the peak

Here, we compute the value of $m$ where we get a peak in the Figure 2.8b and 2.9b for varying values of $s$:

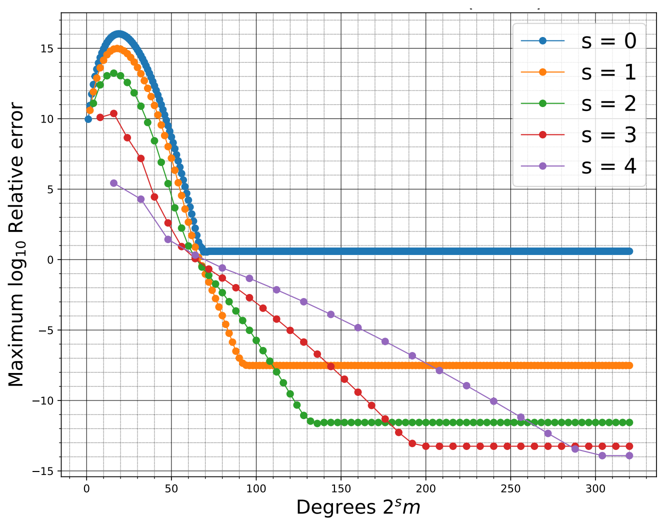
- We take our $z = -R$ because we already know relative error is maximised here.
- We show in Appendix B that approximating $e^z$ by $\big(T_m(z/2^s)\big)^{2^s}$ gives you the largest error for each $s \in \{0,\ldots,4\}$ when truncation parameter is taken to be $m = \left[\!\left[\frac{|z|}{2^s}\right]\!\right] = \left[\!\left[\frac{R}{2^s}\right]\!\right]$. Here the notation $[\![x]\!]$ represents the closest integer to $x \in \mathbb{R}$.
- Since in Figure 2.8b, the plot is with respect to degrees, we must multiply $m$ by $2^s$, giving us maximum relative error to be just at $[\![R]\!]$ degrees.
- Similarly, the Figure 2.9b is with respect to DoF, therefore, the peak is at $\big([\![R/2^s]\!] + 1\big) + 3s$ DoF.
- Therefore, relative error for this choice of $m$ (and $z$) is:

$$O\left(\frac{(-R)^{[\![R/2^s]\!]}}{\big([\![R/2^s]\!]\big)!}\Big/ e^{-R}\right) \tag{2.17}$$

  for each $s$.

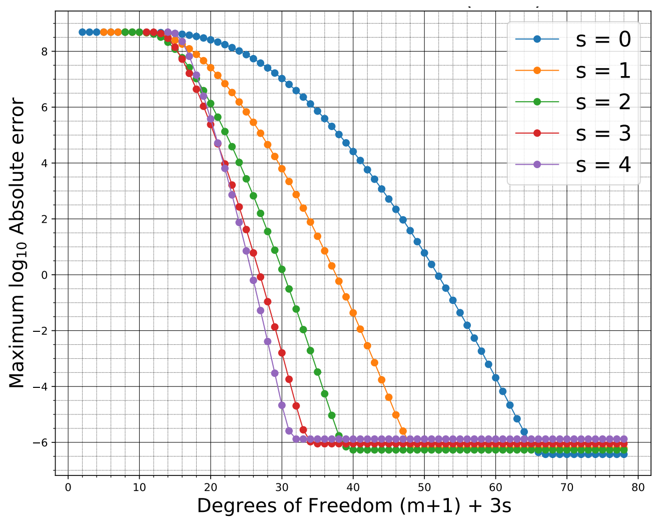


(a) Absolute Error

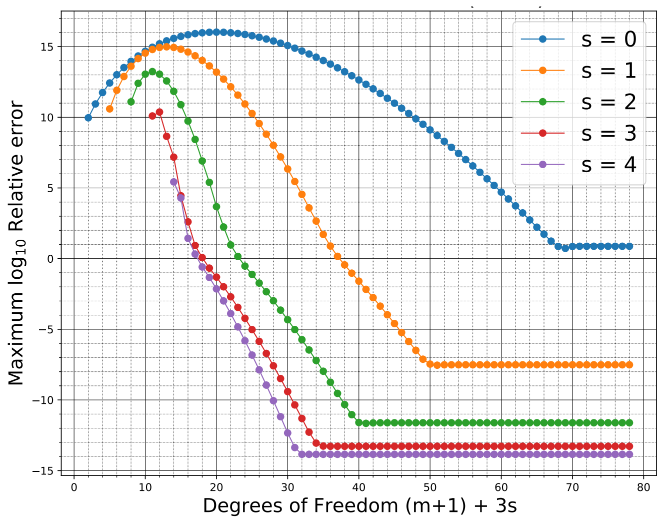


(b) Relative Error

Figure 2.8: Error with respect to degrees of $[T_m(z/2^s)]^{2^s}$ in $\mathcal{T}$ for different values of $s$ and $R = 20$.



(a) Absolute Error



(b) Relative Error

Figure 2.9: Error with respect to Degrees of Freedom of $[T_m(z/2^s)]^{2^s}$ in $\mathcal{T}$ for different values of $s$ and $R = 20$.

For example, if $s = 2$ (and $z = -20$), we get $m = 5$ and the numerator in Equation 2.17 to be :

$$\frac{(-20)^5}{5!} = O(10^4)$$

and the denominator to be $e^{-20} = O\left(10^{-9}\right)$. So overall, we have the peak at $O(10^{13})$ for degree 20 and DoF 12 composite polynomial in the Figure 2.8b and 2.9b respectively. This is verified by the green line in both of these plots. Note, this analysis is purely related to the truncation parameter $m$ and its affect on (relative) truncation error for varying values of $s$.

## Stagnation of the error lines

Another interesting feature of the graphs in 2.8b and 2.9b is the flattening of the error lines (i.e. relative error becoming stagnant after a sufficiently large $m$) for each scaling parameter $s$. To explain this for an arbitrary $s$, we must first consider the derivation of a truncation error bound.

From the Theorem 2.6.2, we know that for $z \in \mathbb{C}$ where $Re(z) < 0$ (which holds because we take our $z = -R$ anyway), we have:

$$\left|e^{z/2^s} - T_m(z/2^s)\right| \leq \varepsilon \quad \text{where} \quad \varepsilon = \frac{|z|^{m+1}}{2^{s(m+1)}(m+1)!}.$$

This implies there exists a $\delta_m \in \mathbb{R}$ such that $T_m(z/2^s) - e^{z/2^s} = \delta_m$ where $|\delta_m| \leq \varepsilon$. Therefore:

$$\begin{aligned}
\left|\left[T_m(z/2^s)\right]^{2^s} - e^z\right| &= \left|\left(e^{z/2^s} + \delta_m\right)^{2^s} - e^z\right| \\
&= \left|e^z + 2^s\left(e^{z/2^s}\right)^{2^s-1}\delta_m + O(\delta_m^2) - e^z\right| \\
&\leq 2^s|\delta_m|\left(e^{z/2^s}\right)^{2^s-1} + O(|\delta_m|^2) \\
&\approx 2^s|\delta_m|\left(e^{z/2^s}\right)^{2^s-1} \qquad (2.18) \\
&\leq \frac{|z|^{m+1}}{2^{sm}(m+1)!}e^z \times e^{-z/2^s}.
\end{aligned}$$

Here, we have made the assumption that we have sufficiently large $m$ when approximating $e^{z/2^s}$ by plain Taylor $T_m(z/2^s)$ such that $O(|\delta_m|^2)$ is negligible. Since relative error is maximised at $z = -R$, then using the above error bound:

$$\frac{\left|e^z - \left[T_m(z/2^s)\right]^{2^s}\right|}{|e^z|} \leq \frac{\left|e^{-R} - \left[T_m(-R/2^s)\right]^{2^s}\right|}{|e^{-R}|} \leq \frac{R^{m+1}}{2^{sm}(m+1)!}e^{R/2^s} \qquad \forall z \in \mathcal{T}. \qquad (2.19)$$

The Equation 2.19 gives a purely theoretical error bound in relative sense for approximating $e^z$ by composite Taylor polynomial. This bound shows that in exact arithmetic, if we fix $s$ and let $m$ increase, then the composite Taylor (and even plain Taylor for $s = 0$) should converge to $\exp(z)$ and relative truncation error should tend to zero. However, when performing the numerics, for a sufficiently large $m$, the error lines stagnate for all $s$. This is solely linked to the limited machine precision available to us; the plots in

Displaying the numerical and theoretical relative error for approximating $e^{-20}$ with composite Taylor for varying $s$ and $m$.

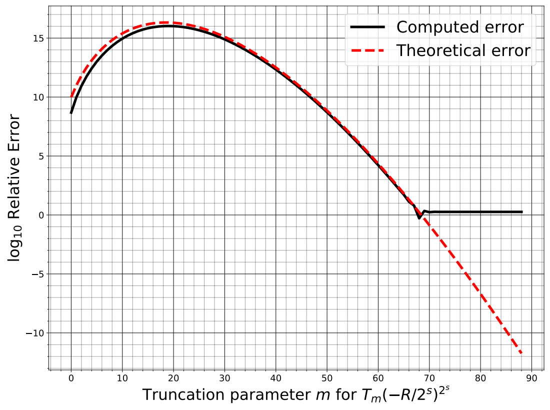


Figure 2.10: $s = 0$

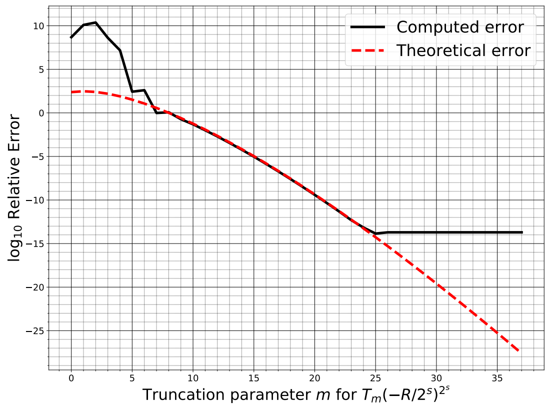


Figure 2.11: $s = 3$

Figure 2.10 and 2.11 compare the *computed relative error* and *theoretical relative error* for $z = -R = -20$ in Equation 2.19 for $s = 0$ and $s = 3$ respectively. Note, the error bound in Equation 2.19 only holds for large values of $m$, therefore Figure 2.11 does not contradict the bound[9].

**Side Remark**: As seen from the semi-log plots, the error curve gets steeper as $m$ increases instead of remaining linear, until rounding errors cut off the progress. This is because, Taylor approximant converges faster than *geometric* for an *entire* function as discussed in Trefethen [1, chap. 8]. Another *fascinating result* which we do not discuss anymore than this.

Moving on, we now try to compute:

1. The smallest value of truncation parameter $m$, where the error lines in Figure 2.8b and 2.9b start to stagnate for a fixed $s$ at $z = -R$.
2. The relative error at which the error lines stagnate for a given $s$.

*Computing truncation parameter $\overline{m}$*

We are aware that the error $T_m(-R/2^s) - e^{-R/2^s} = \delta_m$ is maximised at $m = [\![R/2^s]\!]$ as highlighted in the Appendix B. Therefore, for a given $s \in \mathbb{N} \cup \{0\}$:

$$|\delta_m| \leq \left| \frac{(-R/2^s)^{[\![R/2^s]\!]}}{([\![R/2^s]\!])!} \right| \qquad \forall m.$$

We also know from Theorem 2.6.2 that $\left|e^{-R/2^s} - T_m(-R/2^s)\right| \leq \frac{R^{m+1}}{2^{s(m+1)}(m+1)!}$ for a given $m$ and $s$. Since, in the floating point arithmetic, we only have about 15 digits of accuracy,

[9]for small $m$, the Equation 2.19 does not apply as $O(|\delta_m|^2)$ may not be negligible and higher order terms are required.

we are looking for the *smallest* $m$ which we will denote by $\overline{m}$ such that for a fixed $s$:

$$\underbrace{\left|\log_{10}\left(\left|\frac{(-R/2^s)^{[\![R/2^s]\!]}}{([\![R/2^s]\!])!}\right|\right)\right|}_{\text{Order of the error at the peak}} + \left|\log_{10}\left(\frac{R^{\overline{m}+1}}{2^{s(\overline{m}+1)}(\overline{m}+1)!}\right)\right| \approx \left|\log_{10}\left(\epsilon_{mach}\right)\right|$$

where $\log_{10}$ calculates the exponent of the order of the error (i.e. $a$ if error is $O(10^a)$). This equation simplifies to finding $\overline{m}$ such that:

$$\frac{R^{\overline{m}+1}}{2^{s(\overline{m}+1)}(\overline{m}+1)!} \approx \left|\frac{(-R/2^s)^{[\![R/2^s]\!]}}{([\![R/2^s]\!])!}\right| \cdot \epsilon_{mach}. \tag{2.20}$$

The value of $\overline{m}$ from Equation 2.20 can be computed easily using Python by running a while loop (as shown in the Listing 2.3).

*Computing the stagnating relative error*

Once the truncation parameter $\overline{m}$ is computed, we can easily compute the relative error. We will calculate this in two separate (but similar) ways:

- From Equation 2.19, the stagnating relative error is

$$\frac{R^{\overline{m}+1}}{2^{s\overline{m}}(\overline{m}+1)!}e^{R/2^s}$$

  an upper bound which is tight for $z = -R$.
- Note, for this specific $\overline{m}$ calculated in the previous section:

$$|\delta_{\overline{m}}| = \left|e^{-R/2^s} - T_{\overline{m}}(-R/2^s)\right| \leq \frac{R^{\overline{m}+1}}{2^{s(\overline{m}+1)}(\overline{m}+1)!} \approx \left|\frac{(-R/2^s)^{[\![R/2^s]\!]}}{([\![R/2^s]\!])!}\right| \cdot \epsilon_{mach} \tag{2.21}$$

  using Theorem 2.6.2 and Equation 2.20. Therefore, using this bound, we can calculate the relative error by indirectly. So using the inequality in Equation 2.18:

$$\frac{\left|e^{-R} - \left[T_{\overline{m}}(-R/2^s)\right]^{2^s}\right|}{|e^{-R}|} \leq 2^s \left|\delta_{\overline{m}}\right| e^{R/2^s} \leq 2^s \left|\frac{(-R/2^s)^{[\![R/2^s]\!]}}{([\![R/2^s]\!])!}\right| e^{R/2^s} \cdot \epsilon_{mach}.$$

  Therefore, the second method for computing stagnant error is more efficient since we do not need to physically compute $\overline{m}$ and user may not even be interested in computing it.

***Summary***

Therefore, in this subsection for a fixed $s \in \mathbb{N} \cup \{0\}$ we have calculated:

1. The smallest truncation parameter $\overline{m}$ at which error lines start to flatten by solving the equation:

$$\frac{R^{\overline{m}+1}}{2^{s(\overline{m}+1)}(\overline{m}+1)!} \approx \left| \frac{(-R/2^s)^{[\![R/2^s]\!]}}{\left([\![R/2^s]\!]\right)!} \right| \cdot \epsilon_{mach}.$$

   From this, we can calculate the degree and DoF of the composite polynomial $\left(2^s\overline{m} \text{ and } (\overline{m}+1+3s) \text{ respectively}\right)$.
2. The relative error at which the lines flatten for each $s$ which is approximately:

$$2^s \left| \frac{(-R/2^s)^{[\![R/2^s]\!]}}{\left([\![R/2^s]\!]\right)!} \right| e^{R/2^s} \cdot \epsilon_{mach}.$$

The Python code gives the order of relative error at $z = -R$ when curves for each $s$ in Figure 2.8b and 2.9b starts to flatten out by making use of the above analysis.

```python
R = 20  #radius of all the points we are considering
S = np.arange(0,5)  #s in {0,...,4}
e_mach = 0.5*np.spacing(1)   #machine precision 1x10^{-16}

truncation_para = []  #list to store m
Degrees = []   #list to store degree of poly (2^s*m)
DoF = []       #list to store DoF ((m+1)+3s)
Flattening_error = []  #list to store error where flattening begins
for s in S:
    '''Calculating the truncation parameter m where we see line flattening'''
    max_m = int(R/2**s)   #value of m where we have the largest error
    delta = np.exp(-R/2**s)-((-R/2**s)**max_m)/factorial(max_m))*e_mach   #
    as shown in the above equation
    m = 0
    error = R**(m+1)/(power(2,s*(m+1))*factorial(m+1)) #theoretical error for a
    given m
    while error > delta:
        m+=1
        error = R**(m+1)/(power(2,s*(m+1))*factorial(m+1))
    truncation_para.append(m)  #the m_bar is stored in this list

    '''Here we use above equation to calculate the rel error where the flatten
    takes place'''
    flatting_val = np.log10(2**s*np.abs(((-R/2**s)**max_m)/factorial(max_m))*np.
    exp(R/2**s)*e_mach)
    Flattening_error.append(flatting_val)
    '''Calculating the degree and DoF for using the m and s value computed'''
    Degrees.append(2**s*m)
    DoF.append((m+1)+3*s)
print(truncation_para, Degrees, DoF, Flattening_error)
```

Listing 2.3: Here we compute the truncation parameter $\overline{m}$, the relative error for $m \geq \overline{m}$, degree and DoF of composite polynomial where lines start to flatten out for each $s$.

The output of the code in Listing 2.3 for different values of $s$ is displayed in the Table 2.1. We also visualize these results in Figure 2.12 where we see our theoretical values aligning perfectly with the numerical results we observe inz Figure 2.8b. Similar plot can be made to emphasise the stagnation error point with respect to DoF in 2.9b for different values of $s$.

Furthermore, this analysis gives us another reason why the SSM is superior than plain Taylor; suppose we are measuring error in relative sense, than due to limited machine precision, we cannot get any better approximation even if we take a truncation error $m > \bar{m}$ [10]. For plain Taylor, if we are trying to approximate $e^{-20}$, then we have no digits of accuracy[11] regardless of how large we take $m$ to be as shown in Figure 2.12. Therefore, taking $s > 0$ actually helps in decreasing the error so we have atleast few digits of accuracy. So if a user wants to approximate $\exp(z)$ to a certain accuracy, i.e. $tol \geq \epsilon_{mach}$, then we can find the *smallest* scaling parameter $\bar{s}$ and truncation parameter $\bar{m}$, such that the user gets atleast this accuracy with the lowest number of floating point operations required (lowest complexity). We can write this more formally as follows:

For all $z \in \overline{D}(0, R)$ and user-defined accuracy $tol \geq \epsilon_{mach}$, there exists $\bar{s} = \bar{s}(tol)$ and $\bar{m} = \bar{m}(\bar{s})$ such that $\forall s \geq \bar{s}$ and $\forall m \geq \bar{m}$:

$$\frac{\left| e^z - \left[T_m(z/2^s)\right]^{2^s} \right|}{|e^z|} \leq tol.$$

Note, this observation is especially useful for $z \in \overline{D}(0, R)$ where $Re(z) < 0$ because as we have mentioned countable times, this is when relative error of approximation $e^z$ is the largest. This definition gives a *sufficient* condition; it is possible that approximation of $e^z$ to $tol$ accuracy is reached even when $m < \bar{m}$. This is because the way we have carried out the analysis in this section is to consider the maximum relative error in the closed disc $\overline{D}(0, R)$ which is maximised at $z = -R$ and use that bound for all $z \in \overline{D}(0, R)$ (by maximum modulus principle). Therefore, this error bound also works for these $z$ but is not as *tight*.

Table 2.1: Displays the information about where we expect (theoretically) the error curve flattening out for different values of $s$ as seen in Figure 2.8b and 2.9b.

| $s$ | $\bar{m}$ | Degree ($2^s\bar{m}$) | DoF ($\bar{m}+1+3s$) | Stagnation $\log_{10}$ Error |
|---|---|---|---|---|
| 0 | 68 | 68 | 69 | 0.365775 |
| 1 | 47 | 94 | 51 | -7.870378 |
| 2 | 33 | 132 | 40 | -11.765389 |
| 3 | 25 | 200 | 35 | -13.470914 |
| 4 | 19 | 304 | 32 | -14.110692 |

**Example**: If we take $tol = 10^{-10}$ and choose $z = -16 - 12i$, then $|z| = 20 = R$. So using the Table 2.1, lowest scaling parameter where we get atleast $tol$ accuracy is $\bar{s} = 2$ and the corresponding $\bar{m} = 33$. The code in Listing 2.4 tries to compute the optimum $\bar{m}$ for this particular $z$ such that relative error is atleast $O(10^{-10})$. The code shows we had

[10] theoretically we would expect the error to keep on decreasing, as seen by the red lines in Figure 2.10 and Figure 2.11.

[11] since at $s = 0$, relative error is O(1), we actually have error larger than the value of $e^{-20}$, giving us terrible approximation.

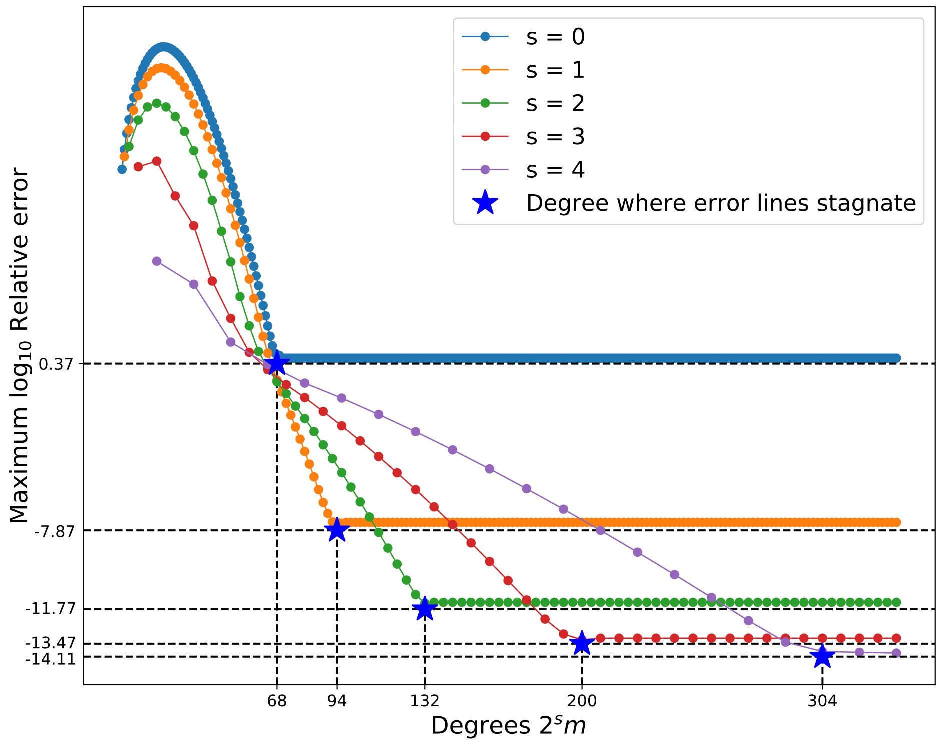


Figure 2.12: This figure verifies that the theoretical values computed in Table 2.1 matches with what we discover numerically.

a better candidate for lowest truncation parameter i.e. $\overline{m} = 31$. But as discussed in the above paragraph, taking $\overline{m} = 33$ suffices and it is good enough for the user's purpose (even though it might be slightly more expensive).

```
z = -16-12j
'''s_bar and m_bar from the table 2.1'''
s_bar = 2
m_bar = 33
for m in range(m_bar-3,m_bar+4):#log10 theo rel error in the neighbourhood of
    m_bar
  rel_error=abs(np.exp(z)-(approx_taylor_exp(z/2**s_bar,m))**(2**s_bar))/abs(np.
    exp(z)) #relative error for each m
  print(f'm = {m}\t{np.log10(rel_error)}')
>> m = 30 -9.960227032445225
>> m = 31 -10.773311796253571  #we reach our tol=10^{-10} here
>> m = 32 -11.591233560374302
>> m = 33 -12.37352349159408    #this is the m we theoretically computed
>> m = 34 -12.373051299536062
>> m = 35 -12.373051299536062
>> m = 36 -12.358146811022987
>> m = 37 -12.358146811022987
```

Listing 2.4: Testing whether the $\bar{s}$ and $\overline{m}$ computed above agrees with all $z \in \overline{D}(0, R)$ for a given accuracy *tol*.

Finding these $\bar{s}$ and $\overline{m}$ is especially beneficial in matrix exponential case as doing floating point arithmetic on a large square matrix can be costly. Furthermore, for a higher accuracy, we actually reduce the DoF of the composite polynomial as seen in the Table 2.1 which leads to the reduction in complexity of the algorithm. *This is remarkable - we have higher accuracy and lower complexity when we apply the scaling and squaring method.*

# Concluding Remarks

In this report, we demonstrated some of the attractive features of the scaling and squaring method based on composing polynomials. We observed that:

- it is a highly efficient way of generating a high degree polynomial and,
- by purposely scaling the problem, we can potentially get a higher accuracy in floating point arithmetic for different $z \in \mathbb{C}$.

Throughout this method, we encountered two types of errors: a) the truncation error because we used a finite degree composite polynomial and b) the numerical roundoff error due to only 15 digits of accuracy on a computer. In Section 2.6, we theoretically found a region $\mathcal{R}$ in the complex plane where the composite Taylor approximation of scalar exponential function was of the order of machine precision for arbitrary values of truncation and scaling parameters. This was extended in Section 2.7, where we saw how composing polynomials lead to better accuracy and lower degrees of freedom, something which is very beneficial in the computation of matrix exponential function. We also saw that in a closed disc of radius $R$ centred at 0, the maximum relative error occurs at $z = -R$. We went on to find the optimal values of $m$ and $s$ where the relative error is minimised for a user defined accuracy, before the rounding errors cut off the progress.

It cannot be stressed enough that mathematically, an algorithm may seem remarkable perhaps due to its extraordinary convergence rate, but it may fail catastrophically if the rounding errors due to floating point arithmetic are not taken into account. This is something every numerical analyst battles with and the author was no exception.

In our future endeavours, we would like to make the error bounds in Section 2.7 tight for every $z \in \mathbb{C}$ so that we do not just base the bound on the maximum error in a disc of radius $R = |z|$. We would also like to do a rigorous comparison between using Taylor or Padé approximants in the scaling and squaring method and observe how much accuracy we get for given degrees of freedom of the composite polynomial and rational approximant respectively. We have included the contour plots making this comparison (although informally) in the zipped folder with the path `plots/pade_comparison` and urge the reader to have a brief look.

# References

[1] Lloyd N Trefethen. *Approximation theory and Approximation Practice*. SIAM, 2019.

[2] Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun guide, 2014.

[3] Edmond Nicolas Laguerre. Le calcul des systemes linéaires, extrait d'une lettre adresséa m. hermite. *Oeuvres de Laguerre, Ch. Hermite, H. Poincaré, and E. Rouché, editors*, 1:221–267, 1867.

[4] Robert Alexander Frazer, William Jolly Duncan, Arthur Roderich Collar, et al. *Elementary matrices and some applications to dynamics and differential equations*, volume 1963. Cambridge University Press, 1938.

[5] Lloyd N Trefethen, Asgeir Birkisson, and Tobin A Driscoll. *Exploring ODEs*, volume 157. SIAM, 2017.

[6] Ernesto Estrada and Juan A Rodriguez-Velazquez. *Subgraph centrality in complex networks*. *Physical Review E*, 71(5):056103, 2005.

[7] Roger B Sidje and William J Stewart. *A numerical study of large sparse matrix exponential arising in Markov chains*. *Computational statistics & data analysis*, 29(3):345–368, 1999.

[8] 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, 2003.

[9] Robert C Ward. *Numerical computation of the matrix exponential with accuracy estimate*. *SIAM Journal on Numerical Analysis*, 14(4):600–610, 1977.

[10] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[11] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: = Fundamental algorithms for scientific computing in Python. *Nature methods*, pages 1–12, 2020.

[12] Nicholas J Higham. *Accuracy and Stability of Numerical Algorithms*, volume 80. SIAM, 2002.

[13] David Hewett. Lecture notes on *'MATH0033: Numerical Methods'*. `https://moodle-snapshot.ucl.ac.uk/18-19/pluginfile.php/808981/mod_resource/content/25/lecture_notes_2018.pdf`, October 2018.

[14] Lloyd N Trefethen and David Bau III. *Numerical Linear Algebra*. SIAM, 1997.

[15] Nicholas J Higham. *Functions of Matrices: Theory and Computation*, volume 104. SIAM, 2008.

[16] Michael Singer. Lecture notes on *'An introduction to complex analysis'*. `https://moodle-snapshot.ucl.ac.uk/17-18/pluginfile.php/4506347/mod_resource/content/3/analytic.pdf`, February 2018.

[17] Hilary A Priestley. *Introduction to complex analysis*. Oxford University Press, 2003.

[18] Endre Süli and David F Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[19] Garret Sobczyk and Omar León Sánchez. *Fundamental theorem of calculus*. *Advances in Applied Clifford Algebras*, 21(1):221–231, 2011.

[20] Walter Rudin. *Real and complex analysis*. Tata McGraw-hill education, 2006.

[21] George Elmer Forsythe, Michael A Malcolm, and Cleve B Moler. *Computer Methods for Mathematical Computations*, volume 259. Prentice-Hall Englewood Cliffs, NJ, 1977.

[22] Syed Muhammad Ghufran. *The computation of matrix functions in particular, the matrix exponential*. PhD thesis, University of Birmingham, 2010.

[23] Philipp Bader, Sergio Blanes, and Fernando Casas. *Computing the Matrix Exponential with an Optimized Taylor Polynomial Approximation*. *Mathematics*, 7(12):1174, 2019.

[24] Nicholas J Higham. *The scaling and squaring method for the matrix exponential revisited*. *SIAM Review*, 51(4):747–764, 2009.

[25] Nicholas J Higham and Awad H Al-Mohy. *Computing matrix functions*. *Acta Numerica*, 19:159–208, 2010.

[26] Jorge Sastre, J Ibánez, P Ruiz, and Emilio Defez. *Accurate and Efficient Matrix Exponential Computation*. *International Journal of Computer Mathematics*, 91(1):97–112, 2014.

[27] Niv Hoffman, Oded Schwartz, and Sivan Toledo. *Efficient evaluation of matrix polynomials*. In *International Conference on Parallel Processing and Applied Mathematics*, pages 24–35. Springer, 2017.

[28] Jorge Sastre. *Efficient Evaluation of matrix polynomials*. *Linear Algebra and its Applications*, 539:229–250, 2018.

[29] Wyman Fair and Yudell L Luke. *Padé approximations to the operator exponential*. *Numerische Mathematik*, 14(4):379–382, 1970.

[30] Stefan Güttel and Yuji Nakatsukasa. *Scaled and Squared sub-diagonal Padé approximation for the matrix exponential*. *SIAM Journal on Matrix Analysis and Applications*, 37(1):145–170, 2016.

[31] Yuji Nakatsukasa and Stefan Güttel. *Rational approximation to the exponential in a complex region*. `https://www.chebfun.org/examples/approx/ScalingAndSquaring.html`, July 2012.

[32] Candidate 1040706. *Calculating the truncation error for exponential function in complex plane* `https://math.stackexchange.com/q/3568570`. Mathematics Stack Exchange.

[33] Edvin Deadman and Samuel D Relton. *Taylor's theorem for matrix functions with applications to condition number estimation*. *Linear Algebra and its Applications*, 2016.

# Appendix A

# Inverse Analysis for Composite Taylor Polynomials

The analysis in this appendix is very similar to the one carried out in Moler, Loan [8, pg. 29-31] for Padé approximants which the author has altered for the Taylor approximants.

**Theorem A.0.1.** Taylor Theorem: Integral Form
Let $f : \Omega \to \mathbb{R}$ be a function that has $(m+1)$ *continuous* derivatives in a neighbourhood of $x_0 \in \Omega$. Then, $\forall x \in \Omega$ the remainder can be defined as:

$$R_m(x; x_0) = \int_{x_0}^{x} f^{(m+1)}(t) \frac{(x-t)^m}{m!} dt. \tag{A.1}$$

This result is very important when we prove Lemma A.0.4 about the bounds on the truncation error.

The Theorem A.0.2, is from Higham [15, pg. 235] but the proof is provided by the author.

**Theorem A.0.2.** If any two matrices $A, B \in \mathbb{C}^{n\times n}$ *commute* i.e $AB = BA$, then $e^{(A+B)t} = e^{At}e^{Bt}$ for all $t \in \mathbb{C}$.

*Proof*: Since $A, B$ commute, then note the following:

$$Ae^{Bt} = A\sum_{k=0}^{\infty} \frac{B^k t^k}{k!} = \sum_{k=0}^{\infty} \frac{ABB^{k-1}t^k}{k!} = \sum_{k=0}^{\infty} \frac{BAB^{k-1}t^k}{k!} = \cdots = \sum_{k=0}^{\infty} \frac{B^k t^k}{k!} A = e^{Bt}A \tag{A.2}$$

Similar result can be shown for $A$, $B$, $A+B$ and their exponents. Let us now define $g(t) = e^{(A+B)t}e^{-Bt}e^{-At}$. By using the product rule for derivatives on $g(t)$, we get that:

$$\begin{aligned} g'(t) &= (A+B)e^{(A+B)t}e^{-Bt}e^{-At} + e^{(A+B)t}(-B)e^{-Bt}e^{-At} + e^{(A+B)t}e^{-Bt}(-A)e^{-At} \\ &= (A+B)g(t) - Bg(t) - Ag(t) \\ &= \mathbf{0} \end{aligned}$$

where $\mathbf{0}$ is a $n \times n$ zero matrix. Since $g'(t) = \mathbf{0}$ for all $t \in \mathbb{C}$, it follows that $g(t)$ is a constant matrix, i.e. $g(t) = C \in \mathbb{C}^{n \times n}$. Taking $t = 0$, we observe:

$$C = g(0) = e^{(A+B)0} e^{-B0} e^{-A0} = e^{\mathbf{0}} e^{\mathbf{0}} e^{\mathbf{0}} = I \tag{A.3}$$

$$\implies g(t) = I \quad \forall t \tag{A.4}$$

Then if we multiply both sides of A.4 by $e^{At}e^{Bt}$, then we get the desired result. □

From now on we will assume that $\|\cdot\| = \|\cdot\|_2$ represents an Euclidean norm.

**Lemma A.0.3.** : If $\|H\| < 1$, then the natural logarithmic function $\log(I + H)$ exists and $\|\log(I + H)\| \leq \frac{\|H\|}{1-\|H\|}$.

*Proof*: Let the power series of $\log(I + H)$ be given by

$$\log(I + H) = \sum_{k=1}^{\infty} (-1)^{k+1} \left( \frac{H^k}{k} \right).$$

Since, $\|H\| < 1$, then by the example 1.3 in Chapter 1, we know that the function has a convergent power series[1]. Then:

$$\|\log(I + H)\| \leq \sum_{k=1}^{\infty} \frac{\|H\|^k}{k} \leq \sum_{k=1}^{\infty} \|H\|^k = \frac{\|H\|}{1 - \|H\|}$$

since $1/k \leq 1$ for all $k$ and the right most expression is the limit of the geometric series for $\|H\| < 1$. □

**Lemma A.0.4.** : If $\|A\| \leq 1/2$ and $m \geq 1$, then we can express the truncated Taylor polynomial centred at zero, $T_m \in \mathcal{P}_m$ of the exponential function as $T_m(A) = \exp(A + F)$ where $F = \log(I + H)$ and $\|F\| \leq \frac{4\|A\|^{m+1}}{(m+1)!}$.

*Proof*: From the Integral form of the Taylor Theorem A.0.1, we know that the remainder term of $\exp(x)$ can be expressed as:

$$\exp(x) - T_m(x) = \int_0^x \exp(t) \frac{(x-t)^m}{m!} dt = \frac{x^{m+1}}{m!} \int_0^1 e^{vx} (1 - v)^m \, dv$$

where $t = vx$. Then as shown in the paper [33, pg. 4-5], the above also holds for matrix exponential function. Then after a suitable change of variable ($u = 1 - v$), we have the following formula:

$$T_m(A) = e^A - \frac{A^{m+1}}{m!} \int_0^1 e^{(1-u)A} u^m du. \tag{A.5}$$

After multiplying A.5 by $e^{-A}$ on the left we get:

$$e^{-A} T_m(A) = I - \frac{A^{m+1}}{m!} \int_0^1 e^{-uA} u^m du \tag{A.6}$$

[1]Note: by definition $\rho(H) \leq \|H\|$ under any matrix norm. So condition that $\|H\| < 1$ is sufficient for $\rho(H) < 1$ and convergence of $\log(I + H)$ by Theorem 1.6.2.

while realising that $e^{-A}A^{m+1} = A^{m+1}e^{-A}$ as highlighted in Theorem A.0.2.

Define: $H = -\frac{A^{m+1}}{m!}\int_0^1 e^{-uA}u^m du$ and note that by Lemma 1.4.1 and the fact that $u \in (0,1)$:

$$\|e^{-uA}\| \le e^{\|uA\|} \le e^{\frac{|u|}{2}} \le e^{\frac{1}{2}}.$$

Then calculating the upper bound for $\|H\|$:

$$\begin{aligned}\|H\| \le \frac{\|A\|^{m+1}}{m!}\int_0^1 \|e^{-uA}\|.|u^m|du &\le \frac{\|A\|^{m+1}}{m!}e^{\frac{1}{2}}\int_0^1 u^m du \\ &= \frac{\|A\|^{m+1}}{(m+1)!}\underbrace{e^{\frac{1}{2}}}_{<2} \\ &\le \frac{\left(\frac{1}{2}\right)^{m+1}\times 2}{2!} < \frac{1}{2} < 1\end{aligned} \tag{A.7}$$

where the integral is defined since $m \ge 1$. But then by Lemma A.0.3,

$$\|F\| = \|\log(I+H)\| \le \frac{\|H\|}{1-\|H\|}.$$

Since $\|H\| < 1/2$, this implies $\frac{\|H\|}{1-\|H\|} \le 2\|H\|$ from above. Then by using Equation A.7 we have :

$$\|F\| = \|\log(I+H)\| \le 2\|H\| \le \frac{4\|A\|^{m+1}}{(m+1)!}. \tag{A.8}$$

Since $F = \log(I+H) \implies e^F = I + H = e^{-A}T_m(A) \implies e^Ae^F = T_m(A)$. But it is easy to see that $F$ is a function of $A$ and therefore by Theorem A.0.2, it follows that matrix $A$ and $F$ commute:

$$e^Ae^F = e^{A+F} = T_m(A).$$

Thus we have shown that we can express $T_m(A) = e^{A+F}$ and $\|F\| \le \frac{4\|A\|^{m+1}}{(m+1)!}$. $\square$

**Theorem A.0.5.** Let $s \in \mathbb{N} \cup \{0\}$ be such that $\frac{\|A\|}{2^s} \le \frac{1}{2}$, then the Composite Taylor polynomial, $\left[T_m\left(\frac{A}{2^s}\right)\right]^{2^s} = e^{A+E}$ where $E$ can be seen as a perturbation matrix. Secondly the backward error is given as follows:

$$\frac{\|E\|}{\|A\|} \le 4\left(\frac{\|A\|}{2^s}\right)^m \frac{1}{(m+1)!}.$$

*Proof*: From Lemma A.0.4, $T_m\left(\frac{A}{2^s}\right) = e^{\frac{A}{2^s}+F}$ and $\|F\| \le 4\left(\frac{\|A\|}{2^s}\right)^{m+1}\frac{1}{(m+1)!}$. If we choose $E = 2^sF \implies [T_m(A/2^s)]^{2^s} = \left[e^{\frac{A+E}{2^s}}\right]^{2^s} = e^{A+E}$. So we can calculate the relative backward error as follows:

$$
\begin{aligned}
\frac{\|E\|}{\|A\|} = \frac{\|2^s F\|}{\|A\|} &\leq \frac{2^s \cdot 4 \left\| \frac{A}{2^s} \right\|^{m+1} \cdot \frac{1}{(m+1)!}}{\|A\|} \\
&= \frac{2^s \cdot 4}{2^{s(m+1)}} \cdot \|A\|^m \cdot \frac{1}{(m+1)!} \\
&= 4 \left( \frac{\|A\|}{2^s} \right)^m \frac{1}{(m+1)!}. \quad \square
\end{aligned}
$$

# Appendix B

# Term with the largest magnitude in Taylor Expansion of $\exp(z)$

From the scaling and squaring method, we know that for a sufficiently large truncation and scaling parameters $m$ and $s$, we can approximate exponential function by $2^s m$ degree composite Taylor polynomial i.e.:

$$
e^z \approx \left(T_m(z/2^s)\right)^{2^s} = \left[ \sum_{k=0}^{m} \frac{z^k}{(2^{sk})k!} \right]^{2^s}.
$$

The coefficients of the Taylor polynomial $T_m(z/2^s)$ which are $a_k = \frac{z^k}{(2^{sk})k!}$ first grow because of power term in the numerator, but then starts to decay as factorial term in the denominator starts to dominate. We are interested in the term $k$ when the magnitude of $a_k$ is the largest. Consider the ratio of $a_k$ and $a_{k-1}$ term:

$$
\frac{a_k}{a_{k-1}} = \frac{\frac{z^k}{2^{sk}k!}}{\frac{z^{k-1}}{2^{s(k-1)}(k-1)!}} = \frac{z}{2^s k}.
$$

We have a tipping point or in other words, the maximum occurs when

$$
\left| \frac{z}{2^s k} \right| = 1 \iff k = \left[\!\left[ \frac{|z|}{2^s} \right]\!\right].
$$

where $[\![x]\!]$ represents the closest integer to $x \in \mathbb{R}$. If we try to approximate $\exp(z)$ with composite Taylor with degree $m = \left[\!\left[ \frac{|z|}{2^s} \right]\!\right]$, then we will have the worse approximation of exponential function with absolute error of the order:

$$O\left(\frac{z^{[\![|z|/2^s]\!]}}{([\![|z|/2^s]\!])!}\right). \tag{B.1}$$

Note, this is purely a mathematical approximation of the order, assuming we are in exact arithmetic. However, what we may see in practice (due to floating point arithmetic) can be slightly different.

Some might argue that perhaps the worst approximation in absolute terms should be of the order:

$$O\left(\left(\frac{(z/2^s)^{[\![|z|/2^s]\!]}}{([\![|z|/2^s]\!])!}\right)^{2^s}\right)$$

instead of B.1. Infact, they both give a pretty good approximation of the peaks in the plots 2.8b and 2.9b for different values of scaling parameter $s$. It was just that B.1 did a better job for our choice of $R$. Which choice is better (and more stable) is something we would like to research in our future endeavours.