This lecture covers the following topics:

- Taylor series generation of approximate analytic solutions
- The Gaussian integral and its polynomial approximations
- Taylor series-based solution of nonlinear equations
- Padé approximation: Approximation with a rational function
Taylor Series Generation of Approximate Analytic Solutions

In the previous lecture, the Taylor series was introduced. It is based on a theorem that states that any smooth function, \( f(x) \), can be expressed as a polynomial in the neighborhood of a point \( a \). Assuming all derivatives of the function exist at \( a \), the series takes the form (duplicated from Lecture 8):

\[
f(x) = f(a) + \frac{(x-a)}{1!} f^{(1)}(a) + \frac{(x-a)^2}{2!} f^{(2)}(a) + \cdots + \frac{(x-a)^n}{n!} f^{(n)}(a) + \cdots
\]

Let \( f(x) \) represent a complicated analytic function, typically derived from physical principles to describe the input/output relationship for a physical system. In engineering and scientific applications, it would be desirable to come up with a polynomial approximation of \( f(x) \) that is sufficiently accurate in a neighborhood \([\alpha \beta]\) of \( x \) (i.e., \( x \in [\alpha \beta] \)).

Instead of dealing with the complicated function \( f(x) \), for all \( x \), we would accept a polynomial approximation over the finite interval \([\alpha \beta]\) where the system is presumed to operate. Then, we can take advantage of the relative ease of dealing with a polynomial when analyzing the system at hand. These ideas are illustrated in the following sections.
The Gaussian Integral and its Polynomial Approximations

The Gaussian integral is the integral of the function \( f(x) = e^{-x^2} \) over the entire real line. It is named after the German mathematician and physicist Carl Friedrich Gauss. The integral is expressed as,

\[
\int_{-\infty}^{+\infty} e^{-x^2} \, dx = \sqrt{\pi}
\]

or (since \( e^{-x^2} \) is an even function),

\[
2 \int_{0}^{+\infty} e^{-x^2} \, dx = \sqrt{\pi}
\]

This integral has a wide range of applications. The same integral with finite limits is closely related both to the error function and the probability density function (pdf) of the normal distribution. In physics this type of integral appears frequently, for example, in quantum mechanics, to find the probability density of the ground state of the harmonic oscillator.

The following figure shows a plot of the function \( f(x) = e^{-x^2} \). The Gaussian integral represents the area under the curve, for all \( x \).
**Example:** Determining the probability of a dart landing in a specific region of its target.

A dart is launched at a target board by an *expert* dart thrower who is aiming at the center of the target from a fixed distance away. The probability for the dart to land inside a circle of radius $r$ inches (centered at the origin) can be expressed as the integral of the Gaussian function:

$$p(r) = \frac{2}{\sqrt{\pi}} \int_0^r e^{-x^2} \, dx$$

We are interested in solving the above integral for a given finite $r$. Unfortunately, an exact analytic solution does not exist. We can only solve the integral for $r$ approaching infinity (utilizing the Gaussian integral value, $\sqrt{\pi}$). As would be expected, $p(\infty) = \frac{2}{\sqrt{\pi} \cdot 2} = 1$ (the probability of the dart landing at any distance form the origin is obviously 1).

Numerical methods (considered later in this course; also, refer to Matlab function `erf`) can be used to generate the following graphical representation for the probability as a function of $r$. 

![Graphical representation of $p(r)$](image)
We may generate an analytic solution (a formula) for the probability, \( p(r) \), which is accurate for small \( r \), as follows. Employ a polynomial approximation for \( e^{-x^2} \) and then evaluate the integral analytically. The Taylor series approximation of \( e^{-x^2} \) for degrees 2, 4 and 10 (with expansion point \( a \) set to zero) are the polynomials (your turn: verify these results)

\[
\begin{align*}
  f_2(x) &= 1 - x^2 \\
  f_4(x) &= 1 - x^2 + \frac{1}{2} x^4 \\
  f_{10}(x) &= 1 - x^2 + \frac{1}{2} x^4 - \frac{1}{6} x^6 + \frac{1}{24} x^8 - \frac{1}{120} x^{10}
\end{align*}
\]

The following plots depict the approximation quality for each polynomial for \( x \in [0, 1] \) (red and black plots overlap):

From visual inspection, the quadratic provides good approximation for \( 0 < x < 0.25 \). The degree-4 polynomial provides good approximation for \( 0 < x < 0.5 \). And, finally, the degree-10 polynomial provides good approximation for \( 0 < x < 1 \).
Substituting these polynomial approximations in the integral for $p(r)$ and solving, gives the following *analytic approximations* for the probability that the dart lands inside a circle of radius $r$:

$$p_2(r) = \frac{2}{\sqrt{\pi}} \int_0^r (1 - x^2) \, dx = \frac{2}{\sqrt{\pi}} \left( r - \frac{1}{3} r^3 \right)$$

$$p_4(r) = \frac{2}{\sqrt{\pi}} \int_0^r \left( 1 - x^2 + \frac{1}{2} x^4 \right) \, dx = \frac{2}{\sqrt{\pi}} \left( r - \frac{1}{3} r^3 + \frac{1}{10} r^5 \right)$$

$$p_{10}(r) = \frac{2}{\sqrt{\pi}} \left( r - \frac{1}{3} r^3 + \frac{1}{10} r^5 - \frac{1}{42} r^7 + \frac{1}{216} r^9 - \frac{1}{1320} r^{11} \right)$$

The following is a graphical comparison of the analytic approximations of the probability integral:

![Graphical Comparison](image)

**Your turn:** Determine (via experimentation) the proper degree of the polynomial approximation so that the probability $p(r)$ can be “accurately” approximated for $0 < r < 2$. 
Matlab probability evaluations*:

```
EDU>> p2=@(r) (2/sqrt(pi))*(r-(1/3)*r.^3);
EDU>> p4=@(r) (2/sqrt(pi))*(r-(1/3)*r.^3+(1/10)*r.^5);
EDU>> p10=@(r) (2/sqrt(pi))*(r-(1/3)*r.^3+(1/10)*r.^5 ... 
- (1/42)*r.^7+(1/216)*r.^9-(1/1320)*r.^11);
EDU>> x=[0.1 0.2 0.3 0.4 0.5];
EDU>> [p2(x); p4(x); p10(x)]'

ans =
0.112461790320519   0.11246291869687   0.112462916018285
0.222666822306848   0.222702930440195   0.222702589210380
0.328358337624794   0.328632533762398   0.328626759440119
0.427279577940167   0.428435038207273   0.428392354253397
0.517173784918777   0.520699969815950   0.520499863540304
```

The last column corresponds to $p_{10}(r)$ and has the highest accuracy. So, the probability of our expert dart thrower to land a dart within 0.5 inch from the center is, approximately, 0.52.

Therefore, for the above problem, we have established that the polynomial

$$p_{10}(r) = \frac{2}{\sqrt{\pi}} \left( r - \frac{1}{3}r^3 + \frac{1}{10}r^5 - \frac{1}{42}r^7 + \frac{1}{216}r^9 - \frac{1}{1320}r^{11} \right)$$

can be used to determine the probabilities as long as, $0 \leq r < 1.4$.

* If you define an anonymous function in Matlab with an independent variable $r$, you may still evaluate it at another variable regardless of what name you give to that variable. For example, if we define $f=@(r) \cdot \cdot \cdot \cos(r)$ and set $r=[1 \ 2 \ 3]$ and $x=[1 \ 2 \ 3]$, then $f(x)$ and $f(r)$ will return identical results.
In mathematics, the integral

\[ p(r) = \frac{2}{\sqrt{\pi}} \int_0^r e^{-x^2} dx \]

is known as the \textit{error function}. Matlab has a special function to compute it (numerically). The function name is \texttt{erf}. It accepts a scalar or a vector argument. For \( x=[0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5] \), \( \text{erf}(x) \) returns:

```
EDU>> erf(x)
ans =
   0.112462916018285
   0.222702589210478
   0.328626759459127
   0.428392355046668
   0.520499877813047
```

From the above results, the \textit{true error} for our approximation \( p_{10}(0.5) \), is:

```
EDU>> format short
EDU>> erf(0.5)-p10(0.5)
ans =
   1.4273e-08
```

The following figure presents a plot for the function \( f(x) = \text{erf}(x) \), for \( x \in [0 \ 2.5] \).
Taylor Series-Based Solution of Nonlinear Equations

In many engineering and science problems, the need arises to solve for the value(s) of an independent variable, $x$, that satisfies the equation $h(x) = g(x)$. This problem can also be expressed as finding the zero’s (roots) of the function $f(x) = h(x) - g(x) = 0$. Matlab can compute the zero’s (as we saw in an earlier lecture) using the roots command, if $f(x)$ is a polynomial in $x$. However, numerical techniques must be used for general, non-linear $f(x)$; An example would be $f(x) = x^2 \cos(x) - e^x = 0$.

In this section, a solution method based on the Taylor series approximation of $f(x)$ is described which also takes advantage of Matlab’s roots function. The algorithm described next is capable of finding real and complex solutions for both real- and complex-valued-coefficient equations.
Consider the problem of finding the roots for the nonlinear equation \( \cos(x) = x \). The solutions could be real and/or complex valued. An estimate of the real roots can be obtained, graphically, as the points of intersection of the two functions \( \cos(x) \) and \( x \) as shown below.

From the graph, the single real root is between 0 and 1. Rewriting the equation as \( f(x) = \cos(x) - x = 0 \) and plotting, we see that the solution is now the value of \( x \) at which makes \( f(x) \) intercepts the \( x \)-axis [i.e., \( f(x) = 0 \)].
We may now employ the Taylor series expansion of the function $\cos(x) - x$ (at the expansion point $a = 0, 0.5$ or $1$, since the root is between $0$ and $1$) and approximate it as a polynomial, say, of degree $19$. The `sym2poly` command can be used to convert the symbolic polynomial to its coefficient vector:

```
EDU>> syms x
EDU>> f=@(x) cos(x)-x;
EDU>> p_taylor=sym2poly(taylor(f,x,0,'order',20));
```

Next, the roots of the polynomial are computed using Matlab’s `roots` command:

```
EDU>> pr=roots(p_taylor)
pr =
  8.703221677284496 + 9.710623120297463i
  8.703221677284496 - 9.710623120297463i
 -8.700166516849322 + 9.709698546217204i
 -8.700166516849322 - 9.709698546217204i
 -8.983357482444774 + 4.926084530699272i
 -8.983357482444774 - 4.926084530699272i
 -9.314203593114499 + 0.0000000000000000i
 -8.628954860978876 + 2.602064056119360i
 -8.628954860978876 - 2.602064056119360i
  9.161943729337086 + 4.943909353406690i
  9.161943729337086 - 4.943909353406690i
  9.353045203305225 + 1.578594012359980i
  9.353045203305225 - 1.578594012359980i
  5.868713178865733 + 2.544884684589450i
  5.868713178865733 - 2.544884684589450i
 -2.486885698569956 + 1.809361340433423i
 -2.486885698569956 - 1.809361340433423i
  0.739085133215160 + 0.0000000000000000i
```

**Your turn:** The Taylor polynomial in the above example is supposed to have degree $19$, and therefore should have had $19$ roots. Why did we only get $18$ roots?
Note that out of the 18 roots, only two are real. We next need to substitute the above roots in \( f(x) \) in order to verify the valid zero’s:

\[
\begin{align*}
\text{EDU} & \gg f(pr) \\
\text{ans} & = \\
& 1.0e+03 * \\
& -6.199576937481191 - 5.456596775705179i \\
& -6.199576937481191 + 5.456596775705179i \\
& -6.159797700639278 + 5.451013937993496i \\
& -6.159797700639278 - 5.451013937993496i \\
& -0.053333080115924 + 0.024516391836313i \\
& -0.053333080115924 - 0.024516391836313i \\
& 0.008320310712212 + 0.000000000000000i \\
& 0.003883023437433 + 0.002190910947133i \\
& 0.003883023437433 - 0.002190910947133i \\
& -0.076914817053919 - 0.023171553833235i \\
& -0.076914817053919 + 0.023171553833235i \\
& -0.011873745775934 - 0.001744938326774i \\
& -0.011873745775934 + 0.001744938326774i \\
& -0.000001343927869 + 0.000004909174472i \\
& -0.000001343927869 - 0.000004909174472i \\
& 0.00000000000002340 + 0.000000000000014i \\
& 0.00000000000002340 - 0.000000000000014i \\
& 0.0000000000000000 + 0.000000000000000i 
\end{align*}
\]

Only the roots of the approximating polynomial that lead to a magnitude of \( f(x) \) close to zero (say, having a magnitude < 10^{-10}) are actual zero’s of \( f(x) \). From the above list, we can say that the last three roots (two complex conjugates and one real) are valid zero’s of \( f(x) \). Therefore, the zero’s are (approximately) at

\[
0.7390851 \\
-2.4868857 \pm 1.8093613 i
\]
`solve_poly` is a function that implements the above hybrid symbolic/numeric method for solving equations:

```
function z = solve_poly(f,x0,TOL)
% Function "solve_poly" employs Taylor series approx.
% function f(x), in the neighborhood of x0. For example,
% to solve cos(x)= x^2, form f(x)=cos(x)- x^2. The function
% f is passed as an anonymous function. e.g.,
% f=@(x) cos(x)-x.^2 (dot operations must be used).
% Inputs are f, x0 and TOL. x0="initial guess" (from plot of f(x))
% TOL affects solution accuracy (default value = 10^-10);
% If no solution(s) is returned, try to increase TOL (say, 10^-6).
% Call can also be made like this: solve_poly(@(x) cos(x)-x.^2,x0,TOL)
% Also works for complex f(x): e.g., solve_poly(@(x) exp(x)-i,0)

    format long
    sym x;
    if nargin==2, TOL = 10^-10; end
    p_taylor=sym2poly(taylor(f,x,x0,'order',20));
    pr=roots(p_taylor);
    m=length(pr);
    values=f(pr);
    z=[];
    s=1;
    for k=1:m
        if abs(values(k))<TOL
            z(s)= pr(k);
            s=s+1;
        end
    end
    disp('solution(s):')
    z=z';
end
```

Solution for \( \cos(x) - x = 0 \) with initial guess (point of expansion) \( x_0 = 0 \) and default tolerance, \( \text{TOL} = 10^{-10} \):

```
>> f=@(x) cos(x)-x;
>> solve_poly(f,0)
solution(s):
ans =
     0.739085133215160
```
Solution with initial guess $x_0 = -1$ and default tolerance, $TOL = 10^{-10}$:

```
EDU>> solve_poly(f,-1)
solution(s):
ans =
  Column 1
-2.486885698906638 + 1.809361341295994i
  Column 2
-2.486885698906638 - 1.809361341295994i
  Column 3
  0.739085133215151 + 0.000000000000000i
```

Calling the function with $x_0 = 0.5$ and $TOL = 10^{-14}$:

```
EDU>> solve_poly(f,0.5, 10^-14)
solution(s):
ans =
  0.739085133215161
```

Solutions employing Matlab’s built-in nonlinear equation solver: **fzero**

```
EDU>> help fzero
fzero  Single-variable nonlinear zero finding.
    X = fzero(FUN,X0) tries to find a zero of the function FUN near X0,
    if X0 is a scalar. It first finds an interval containing X0 where
    the function values of the interval endpoints differ in sign, then
    searches that interval for a zero. FUN is a function handle. FUN
    accepts real scalar input X and returns a real scalar function value F,
    evaluated at X. The value X returned by fzero is near a point where
    FUN changes sign (if FUN is continuous), or NaN if the search fails.
```

```
EDU>> f=@(x) cos(x)-x;
EDU>> fzero(f,0)
ans =
  0.739085133215161
EDU>> fzero(f,-1)
ans =
  0.739085133215161
```
Example. Solve the nonlinear equation $f(x) = \tanh(x) - x^3$ for zeros near $x = 0.5$, employing `solve_poly` and `fzero`.

Matlab has two other build-in functions for zero finding: `fsolve` and `solve`. `fsolve` is based on numerical algorithms and it allows the solution of systems of multiple nonlinear equations. `solve` is part of the Symbolic Toolbox and can solve for exact multiple zeros for a given system of nonlinear equations. We will look more closely at these two functions in a future lecture. For now, here are examples of using such functions to solve the (polynomial) equation $f(x) = x^3 - 2x^2 + 1 = 0$, 

```
EDU>> f=@(x)tanh(x)-x.^3;
EDU>> solve_poly(f,0.5)
solution(s):
ans =
     0.893394573289747
     -0.000000000006610
EDU>> fzero(f,0.5)
ans =
     0.893394573289828
```
The command `double` converts the symbolic result to double-precision, floating-point numeric representation. Of course, since the nonlinear equation in this example is just a polynomial, we could have calculated the roots as,
Padé Approximation: Approximation with a Rational Function

Padé approximation involves approximating a differentiable function \( f(x) \) using a rational function of the form

\[
\tilde{f}(x) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n}{1 + b_1 x + b_2 x^2 + \cdots + b_m x^m}
\]

The coefficients in this rational function are determined by solving a set of \( m + n + 1 \) linear equations. There are different ways of setting up these equations. In this section, we will generate the equations in a fashion similar to that of the Taylor/Maclaurin approximation: We require that \( \tilde{f}(x) \) and its first \( N \) derivatives agree with \( f(x) \) at \( x = 0 \). Since we need \( m + n + 1 \) equations, we set \( N = m + n \). The system of \( N + 1 \) equations would then be written as

\[
\begin{align*}
\tilde{f}(0) &= f(0) \\
\tilde{f}^{(1)}(0) &= f^{(1)}(0) \\
\tilde{f}^{(2)}(0) &= f^{(2)}(0) \\
& \quad \vdots \\
\tilde{f}^{(N)}(0) &= f^{(N)}(0)
\end{align*}
\]

This requires the computation of all derivatives of \( \tilde{f}(x) \) and \( f(x) \) up to and including the \( N \)th derivative.

The advantage of the rational function over polynomial (power series) approximation is that it allows for singularities (points at which \( f(x) \) shoots asymptotically to \( \pm \infty \)). Those singularities can be represented in a rational function as the roots of the denominator polynomial. Rational functions can also have horizontal asymptotes (polynomials can’t). Empirical studies demonstrated that the approximation error is significantly improved using a rational
function over a polynomial (both having the same number of coefficients). Those studies also showed that the order $n$ of the numerator polynomial of $\tilde{f}(x)$ is best set same as or one greater than the order $m$ of the denominator.

**Example.** Determine the Padé approximation for $f(x) = e^x$ assuming $m = n = 1$. Compare the result to the Maclaurin series approximation of order $m + n = 2$ (i.e., $e^x \cong 1 + x + \frac{x^2}{2}$) over the interval $[-1\ 1]$.

The approximating Padé rational function is

$$\tilde{f}(x) = \frac{a_0 + a_1 x}{1 + b_1 x}$$

In order to satisfy the first condition $\tilde{f}(0) = f(0)$, we solve the following equation and obtain $a_0 = 1$

$$\left.\frac{a_0 + a_1 x}{1 + b_1 x}\right|_{x=0} = a_0 = e^0 = 1$$

The first two derivatives of the rational function are

$$\tilde{f}^{(1)}(x) = \frac{d}{dx} \left(1 + \frac{a_1 x}{1 + b_1 x}\right) = \frac{a_1 - b_1}{(1 + b_1 x)^2}$$

$$\tilde{f}^{(2)}(x) = \frac{d^2}{dx^2} \left(1 + \frac{a_1 x}{1 + b_1 x}\right) = \frac{(b_1 - a_1)(2b_1)(1 + b_1 x)}{(1 + b_1 x)^4}$$

which upon substituting in the equations,

$$\tilde{f}^{(1)}(0) = f^{(1)}(0) = \left.\frac{d}{dx} e^x\right|_{x=0} = e^0 = 1$$

$$\tilde{f}^{(2)}(0) = f^{(2)}(0) = \left.\frac{d^2}{dx^2} e^x\right|_{x=0} = e^0 = 1$$
leads to the two equations
\[ a_1 - b_1 = 1 \]
\[ 2b_1(b_1 - a_1) = 1 \]
whose solution is \( a_1 = \frac{1}{2} \) and \( b_1 = -\frac{1}{2} \). Therefore, the Padé approximation is
\[ \tilde{f}(x) = \frac{1 + \frac{1}{2}x}{1 - \frac{1}{2}x} \]
The plots for \( e^x, 1 + x + \frac{x^2}{2} \) and \( \tilde{f}(x) \) are compared in the figure below for \( x \in [-1, 1] \). The superiority of the Padé approximation (over most of the domain) is obvious. For example, the absolute true error at \( x = 0.5 \) is 1.44\% for the power series approximation and 1.09\% for Padé.

This is a fair comparison because we are comparing two approximation functions that require about the same computation time to evaluate; It took approximately 0.05 msec for one evaluation of each function using Matlab running on one particular laptop (try it on your laptop employing the tic/toc Matlab instructions).
One disadvantage of the approach used in the above formulation to determine the coefficients of the rational function is the fact that the resulting equations are nonlinear algebraic equations. One way to avoid this is a formulation that results in a set of linear equations which can be easily solved (refer to the numerical solution methods of Lectures 15 and 16). In the following, we assume that the Maclaurin polynomial approximation \( \hat{f}(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_N x^N \) is available. Recall that the \( c_i \) coefficient is proportional to the \( i \)th derivative of \( f(x) \), \( f^{(i)}(0) \). We will force \( \hat{f}(x) \) and its first \( N \) derivatives to be identical to \( f(x) \) and its first \( N \) derivatives, respectively, at \( x = 0 \).

So, we begin by computing the difference \( \hat{f}(x) - \tilde{f}(x) \),

\[
c_0 + c_1 x + c_2 x^2 + \cdots + c_N x^N - \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n}{1 + b_1 x + b_2 x^2 + \cdots + b_m x^m} = \]

\[
\frac{(c_0 + c_1 x + c_2 x^2 + \cdots + c_N x^N)(1 + b_1 x + b_2 x^2 + \cdots + b_m x^m) - (a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n)}{1 + b_1 x + b_2 x^2 + \cdots + b_m x^m}
\]

Here, we assume that \( N \) is selected to be equal to \( m + n \). In order to have \( \tilde{f}(0) = f(0) \), we need \( c_0 - a_0 = 0 \). So, \( a_0 = c_0 \). Now, in order for the error \( \hat{f}(x) - \tilde{f}(x) \) to be minimized, the coefficients of the powers of \( x \) up to and including \( x^N \) in the numerator must all be zero. This gives rise to \( N \) linear equations for the \( a_i \)'s and \( b_j \)'s, \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \). Here are the first three equations (can you detect a pattern that you can program?)

\[
b_1 c_0 + c_1 - a_1 = 0
\]

\[
b_2 c_0 + b_1 c_1 + c_2 - a_2 = 0
\]

\[
b_3 c_0 + b_2 c_1 + b_1 c_2 + c_3 - a_3 = 0
\]

\[
\cdots
\]

\[
\cdots
\]
Note that, in each equation, the sum of the subscripts on the factors of each product is the same, and is equal to the exponent of the $x$-term in the numerator. These $N$ equations involve $N$ unknowns and are linear. (Recall that $a_0 = c_0$ for all approximations and should be included among these equations).

**Example.** Redo the last example employing the above formulation.

Start by writing (we set $N = m + n = 2$)

$$\hat{f}(x) - \tilde{f}(x) = \frac{\left(1 + x + \frac{x^2}{2}\right)(1 + b_1 x) - (a_0 + a_1 x)}{1 + b_1 x}$$

$$= \frac{(1 - a_0) + (b_1 - a_1 + 1)x + (b_1 + \frac{1}{2})x^2 + \frac{b_1}{2}x^3}{1 + b_1 x}$$

Setting the numerator coefficients of the terms in $x$ (up to and including the quadratic term) to zero, we obtain the system of three linear equations,

1. $1 - a_0 = 0$
2. $b_1 - a_1 + 1 = 0$
3. $b_1 + \frac{1}{2} = 0$

These equations can be solved numerically. But they are so simple that we can solve them by hand to get, $a_0 = 1$, $b_1 = -\frac{1}{2}$ and $a_1 = \frac{1}{2}$, which is the same solution we obtained in the last example:

$$\tilde{f}(x) = \frac{1 + \frac{1}{2}x}{1 - \frac{1}{2}x}$$
Your turn: Determine (by hand calculations) the Padé approximation \( \tilde{f}(x) \) for \( f(x) = \tan^{-1}(x) \). Assume \( n = 5 \) and \( m = 4 \). Compare (in table format) the true absolute error (in percent) for \( \tilde{f}(x) \) and for the Maclaurin series approximation of order \( m + n = 9 \),

\[
\tilde{f}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9}
\]

for the following values \( x \in \{0.2, 0.4, 0.6, 0.8, 1\} \). Also, generate a plot comparing \( f(x), \tilde{f}(x) \) and \( \hat{f}(x) \) for \( x \in [-1.2, 1.2] \).

Hint: you need to first show that the problem results in a system of 10 by 10 equations in the form,

\[
\begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & c_1 & c_0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & c_2 & c_1 & c_0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & c_3 & c_2 & c_1 & c_0 \\
0 & 0 & 0 & 0 & 0 & -1 & c_4 & c_3 & c_2 & c_1 \\
0 & 0 & 0 & 0 & 0 & 0 & c_5 & c_4 & c_3 & c_2 \\
0 & 0 & 0 & 0 & 0 & 0 & c_6 & c_5 & c_4 & c_3 \\
0 & 0 & 0 & 0 & 0 & 0 & c_7 & c_6 & c_5 & c_4 \\
0 & 0 & 0 & 0 & 0 & 0 & c_8 & c_7 & c_6 & c_5
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5 \\
a_6 \\
a_7 \\
a_8 \\
b_4
\end{bmatrix}
= -
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3 \\
c_4 \\
c_5 \\
c_6 \\
c_7 \\
c_8 \\
c_9
\end{bmatrix}
\begin{bmatrix}
0 \\
1 \\
0 \\
-1/3 \\
0 \\
1/5 \\
0 \\
-1/7 \\
0 \\
1/9
\end{bmatrix}
\]

Also, a system of linear equations, \( Ax = d \), can be solved using the Matlab instruction \( x = \text{inv}(A)*d \).

Note: Matlab has a built-in symbolic toolbox function, \textit{pade}, that can be used to generate the Padé approximation at point \( a \). The function call is \textit{pade}(\textit{f}, \textit{x}, \textit{a}, ['order'], [\textit{n} \ \textit{m}]). Here is the solution for the above example:

\begin{verbatim}
>> syms x;
>> f=@(x) atan(x);
>> P=pade(f,x,0,'Order',[5 4]);
P =
(x*(64*x^4 + 735*x^2 + 945))/(15*(15*x^4 + 70*x^2 + 63))
\end{verbatim}

or, equivalently,
\[ \tan(x) \approx \frac{x + \frac{7}{9}x^3 + \frac{64}{945}x^5}{1 + \frac{10}{9}x^2 + \frac{5}{21}x^4} \]

**Your turn:** Employ the method of this section to establish the Padé approximation (with \( m = n = 4 \)) for \( f(x) = \cos(x) \) as,

\[ \tilde{f}(x) \approx \frac{15,120 - 6,900x^2 + 313x^4}{15,120 + 660x^2 + 13x^4} \]

Verify your answer using Matlab’s built-in function `pade`. Plot the function \( f(x) = \cos(x) \) and its Padé approximation for \( x \in [-5 \, 5] \).

**Your turn:** Write the function, \([a,b] = \text{pade1}(f,n,m,x1,x2)\), that generates the Padé coefficient vectors \( a \) and \( b \) for the rational function approximation, \( \tilde{f}(x) \). Also, it should generate a plot displaying \( f(x) \) and \( \tilde{f}(x) \) for \( x \in [x1 \, x2] \). \( n \) and \( m \) are the order of the numerator and denominator of \( \tilde{f}(x) \), respectively. \( f \) represents the (anonymous function) \( f(x) \) being approximated. Note: Your function `pade1` should generate internally the required Maclaurin polynomial coefficients (Taylor series applied at the origin). You must not use the built-in Matlab function `pade`; i.e., your program is supposed to generate the required system of \((m + n + 1)x(m + n + 1)\) equations and solve it. Test your function for \( f(x) = \tan^{-1}(x) \), with \( n = 5 \) and \( m = 4 \) and \([x1 \, x2]=[-3 \, 3] \).