**Practical 2:**

**Objective**: Round-off and Truncation Errors.

In [computational mathematics](https://en.wikipedia.org/wiki/Computational_mathematics), an iterative method is a mathematical procedure that generates a sequence of improving approximate solutions for a class of problems. A specific implementation of an iterative method, including the [termination](https://en.wikipedia.org/wiki/Algorithm#Termination) criteria, is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) of the iterative method. An iterative method is called convergent if the corresponding sequence converges for given initial approximations

In class we have discussed two different ways that errors can be found in a numerical approximation:

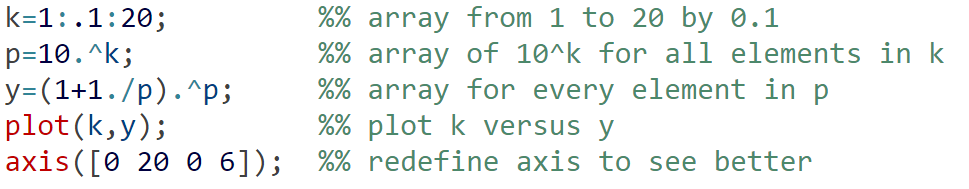
1. In Truncation Error, inaccuracies are introduced by truncation of an infinite process, e.g. a series expansion or infinite decimal or underflow.
2. In Round-Off Error, inaccuracies are introduced by rounding errors in a calculation that accumulate over time.

**Example 1**: The number π = 3.14159265358979... If we use the approximation π ≈ 3.14, what is the absolute error? (Express your answer using chopping to a decimal normalized floating-point representation with 5 significant digits.)

**Answer**: The absolute error is

**Example 2 (Limits to Infinity)**: Let's consider a well-known fact from calculus.

Clearly, we cannot take an infinite limit, so we see what happens for large values of .  Here is a short little program to analyze this limit.



**Sample Output:**

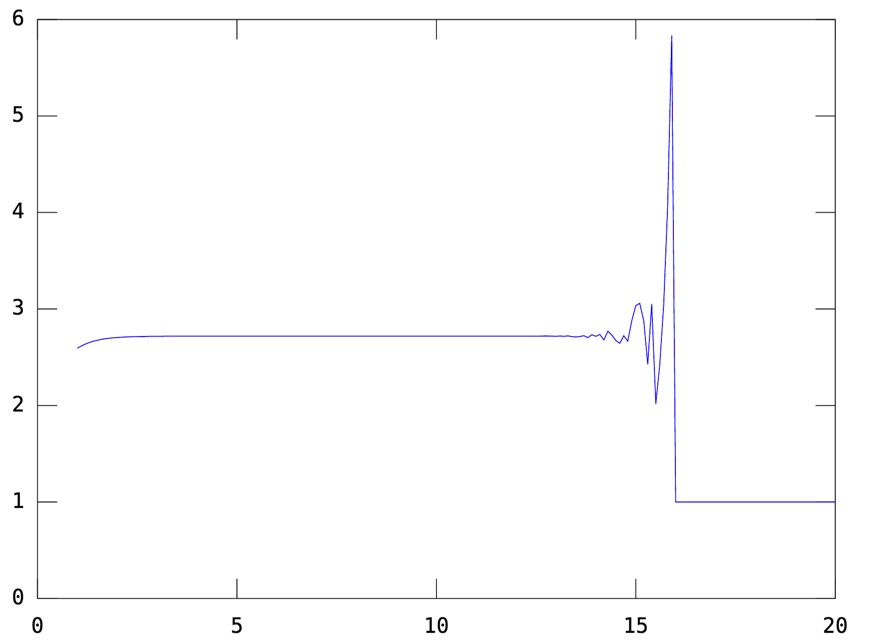


Figure 1: Infinite limit for exponential

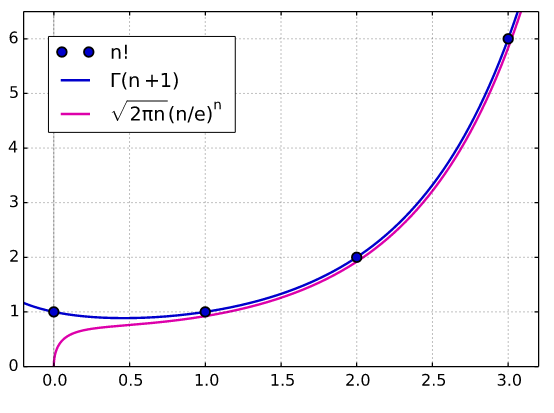
Upon close analysis, it looks like the values plotted are converging rapidly to , but then something weird starts to happen at around .  Then, we get wild fluctuation until or so, then a flat line.

**Example 3 (Stirling’s Approximation)**: When we compute the error in a computational process, we often use two different "metrics" for how accurate the calculation is:

**Absolute Error** = calculated value - true value   
**Relative Erro**r = absolute error / true value

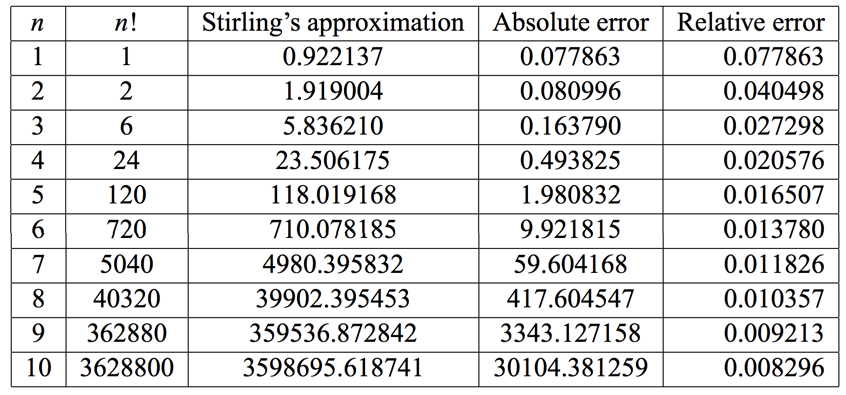
As an example of these two types of errors, we will consider Stirling's Approximation to

In mathematics, **Stirling's approximation** (or **Stirling's formula**) is an approximation for factorials. It is a very powerful approximation, leading to accurate results even for small values of *n*



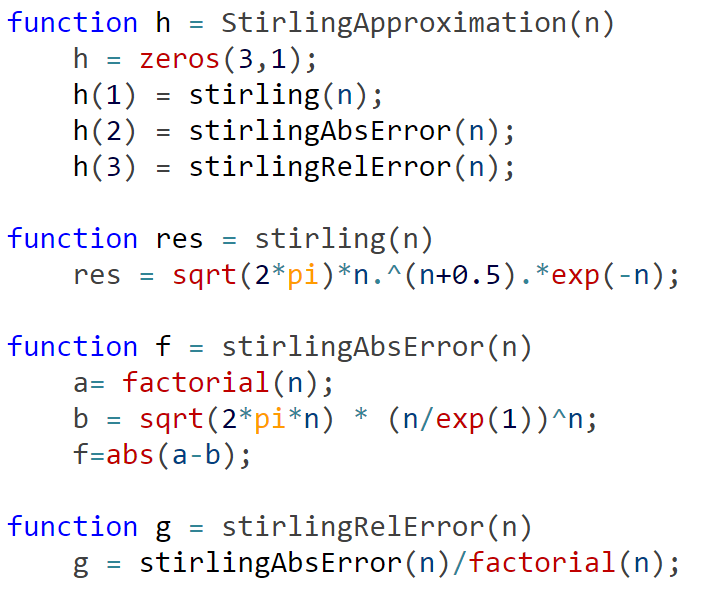
**Figure 2. Comparison of Stirling’s approximation with the factorial**

**Sample Output:**

****

According to the table, the absolute error increases, and the relative error decreases with the increase in .

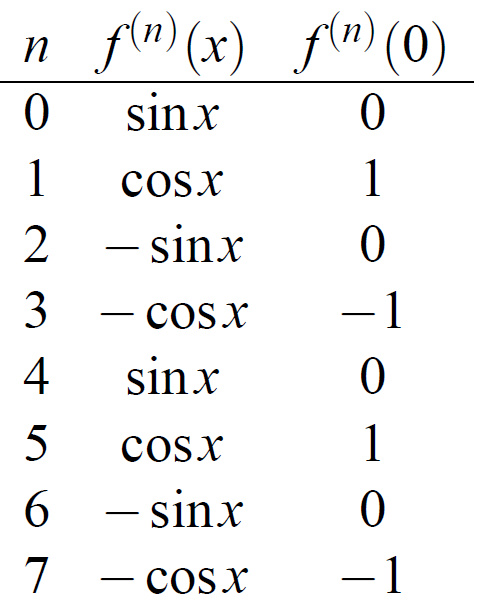
**Sample Code:**

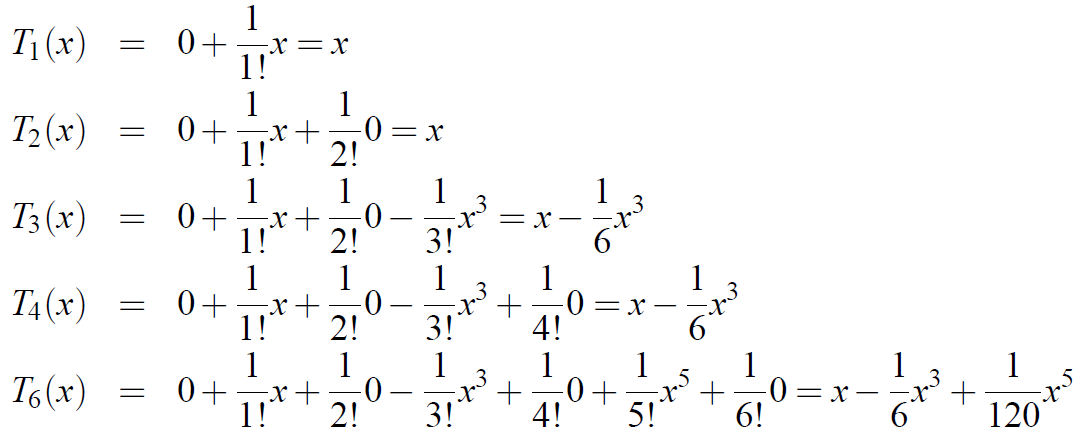


**Example 4: Taylor Polynomials with Error Term**

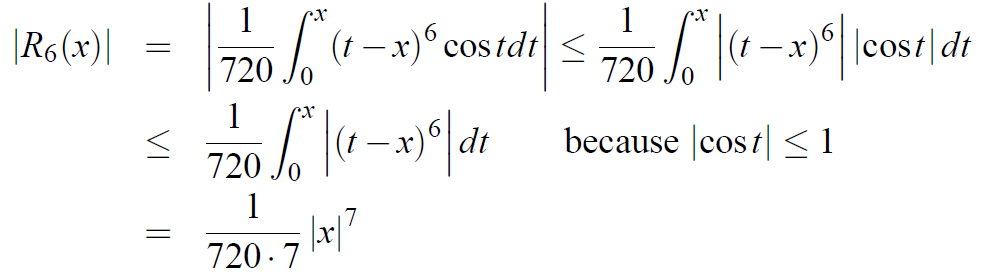
1. The way that computers evaluate the more complex functions is to approximate them by polynomials. There are many other applications where it is useful to have a polynomial approximation to a function. Generally, polynomials are just easier to use.
2. In this section we will show you how to obtain a polynomial approximation of a function. The approximation will include the error term—extremely important since we must know that our approximation is a sufficiently good approximation—how good depends on our application.

Consider the function *f* (*x*) = sin *x* on the interval [−,]. Compute the first, third, fourth and sixth order Taylor polynomial approximations of *f.* Plot *f* and the four polynomials on the same graph.





If we consider the expression for *R*6,



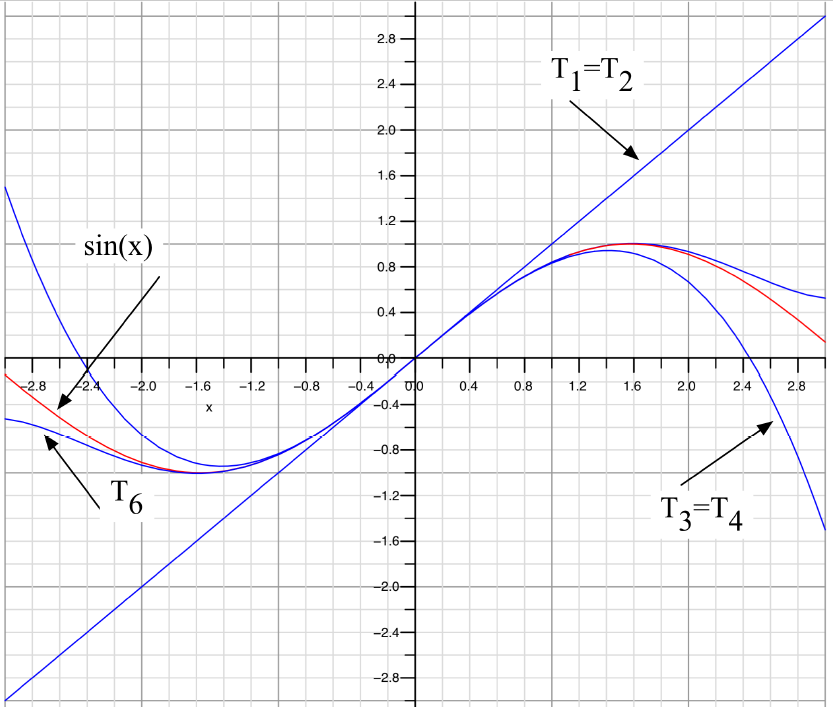


Figure 1: Taylor polynomial approximations for *y* = sin *x*

There are four functions plotted:

*y* = sin *x*,

*y* = *T*1(*x*) (which is identical to *y* = *T*2(*x*))

*y* = *T*3(*x*) (= *T*4(*x*))

*y* = *T*6(*x*).)

* It looks as if *T*1 is a pretty good approximation to *y* = sin *x* near *x* = 0 (which shouldn’t surprise us since *y* = *T*1(*x*) represents the tangent to the curve *y* = sin *x* at *x* = 0. It should be pretty clear that *y* = *T*3(*x*) is a better approximation to *y* = sin *x* that *y* = *T*1(*x*) and that *y* = *T*6(*x*) is better than *y* = *T*3(*x*).
* If you look carefully, it appears that *T*6 is a good approximation well beyond the [−.5, .5] interval analyzed above. You must remember that there are inequalities involved in the analysis above—that means that the results might be better than we computed them to be.

**Example 5: Taylor Polynomials (Series Experiments)**

Let’s use a cosine series to approximate cos(1.6), using just three terms in the Taylor polynomial for f(x) = cos(x) centered at x = pi/2 (x is approx. 1.57). In this problem, f(x) = cos(x) and c = pi/2.

f(0)(x) = cos(x) f(0)(pi/2) = cos(pi/2) = 0

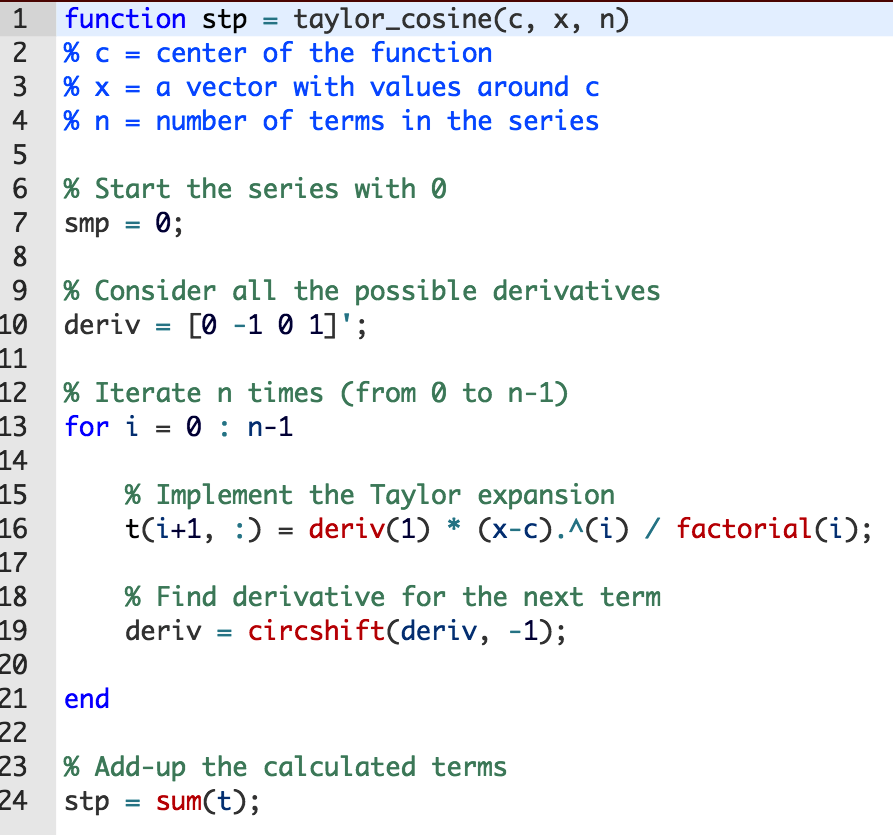
f(1)(x) = -sin(x) f(1)(pi/2) = -sin(pi/2) = -1

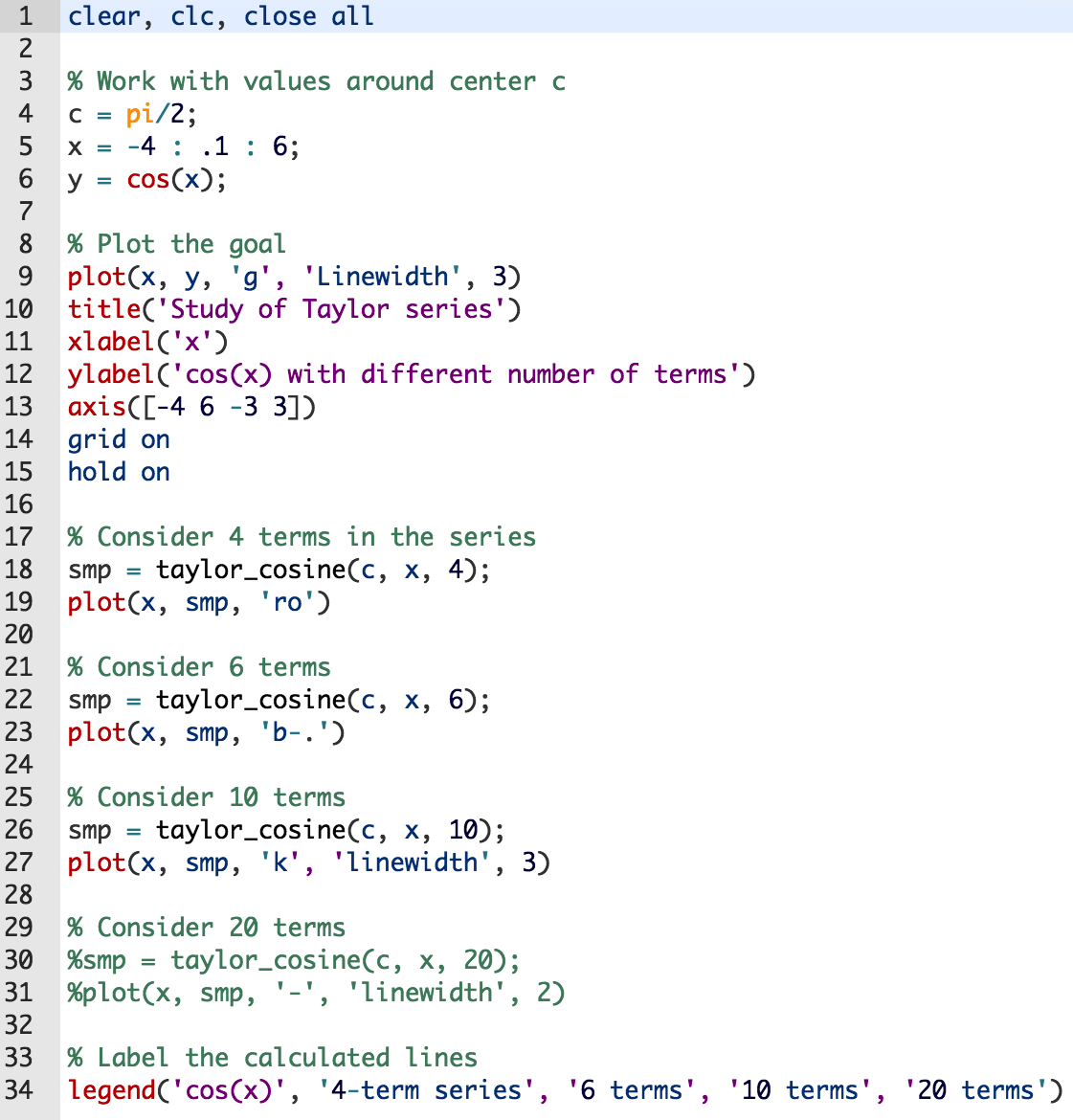
f(2)(x) = -cos(x) f(2)(pi/2) = -cos(pi/2) = 0

1. Import the following files into Octave-Online.

* taylor\_cosine.m
* taylor\_demo.m

1. Compile and run the taylor\_demo.m file.





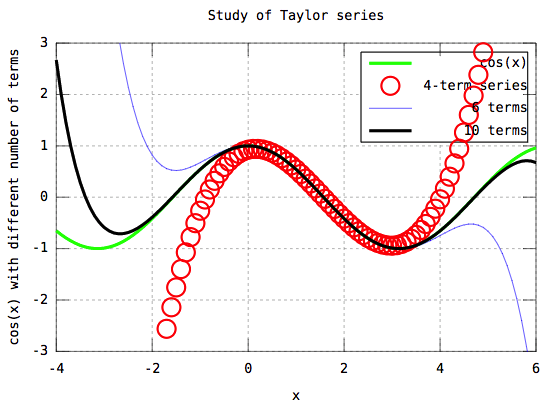
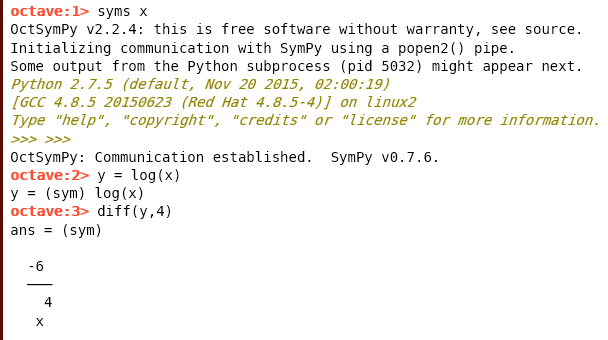


Figure 2: Taylor polynomial approximations for *y* = cos *x*

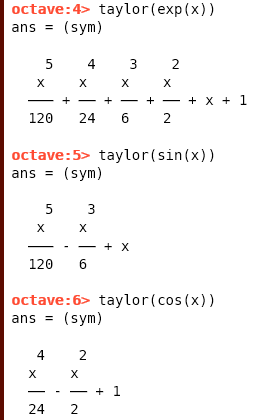
We see that all of the Taylor expansions work well when we are close to pi/2 (x approx. 1.57). More terms approximate better a larger portion of the cosine curve.

**Example 5: Taylor Series in MATLAB**

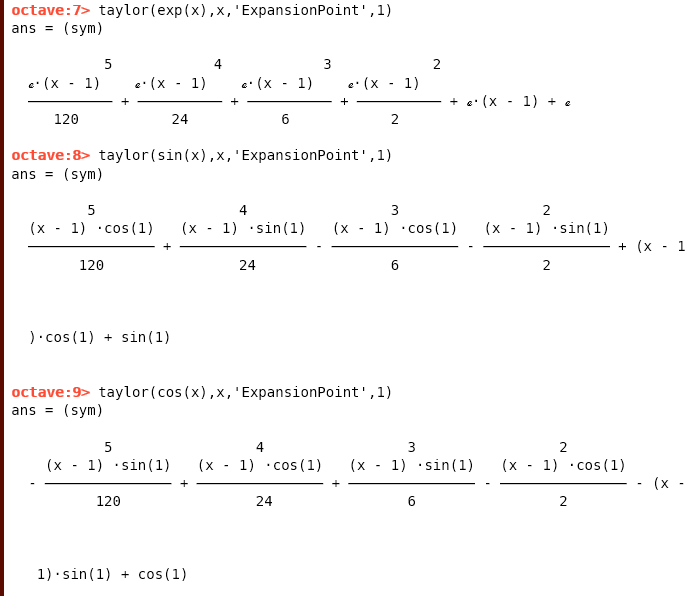
1. Find the first and fourth derivative of ln *x*.



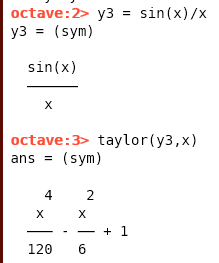
1. Find the Maclaurin series expansions of , sin , and cos



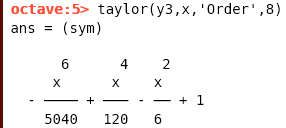
1. Find the Taylor series expansions at *x* = 1 for , sin , and cos .



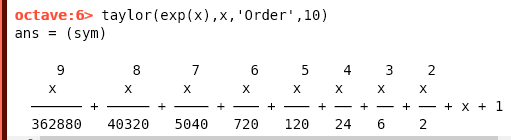
1. Find the Maclaurin series expansion for.



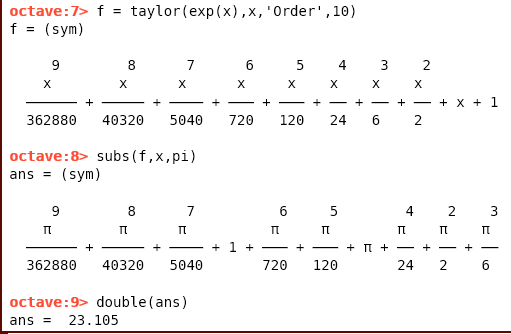
1. Find the Maclaurin series expansion for up to order 8.



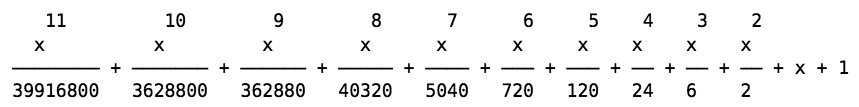
1. Find the Maclaurin series expansion for up to order 10.



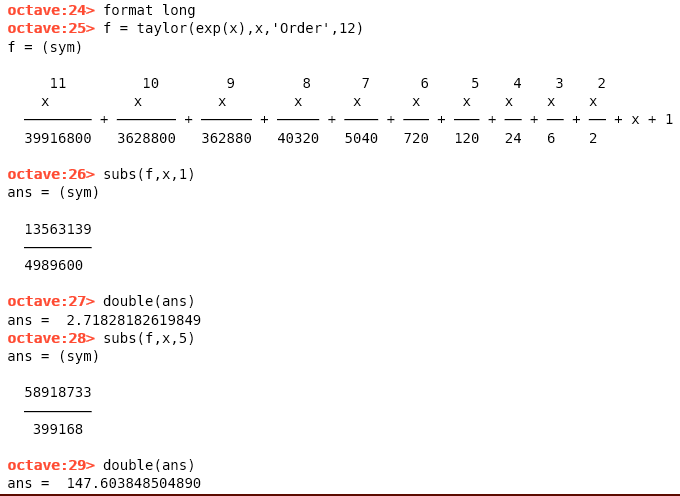
1. Evaluate the expansion in Part(6) with x=pi



1. Evaluate the following Taylor expansion f with x=1 and x=5.



Use chopping to give 8 decimal places.



ANS : 2.71828182,

147.60384850