This is an annotation of the `test_convergence.m` MATLAB script from 1094 Session 4. The purpose of this script is to permit numerical experiments on whether or not series of the form

\[ \sum_{n=0}^{\infty} a_n x^n \]

converge. The ratio test says that the series will converge absolutely for values of \( x \) such that

\[ |x| < R = \lim_{n \to \infty} \left| \frac{a_n}{a_{n+1}} \right| . \]

\( R \) is called the “radius of convergence” of the series. We want to test for the value of \( R \) experimentally (by computer experiment, that is!).

1. **Strategy**
   We’ll try a brute force method. For a given value of \( x \) (requested from the user), we calculate the sum of two terms, then three terms, then four terms, and so on up to \( N \) terms (for a fixed value of \( N \)). Then we plot the partial sums against the corresponding maximum \( n \) and see whether it is converging or diverging.

2. **Comments and Help**
   The beginning of a script should always have comment lines explaining what the program is for, how it is used, and its history. Comment lines begin with `%` (everything after a `%` anywhere in the program is a comment).

   % test_convergence checks for the radius of convergence of a
   % Taylor series by plotting the sum for a fixed value of x
   % (supplied by the user) as a function of the number of terms
   % in the sum.

   % Programmer: Dick Furnstahl   furnstahl.1@osu.edu
   %
   % Revision history:
   % 15-Apr-2005 -- original test_convergence.m for Physics 263
   %
   The lines up to the first blank line are printed in response to
   >> help test_convergence
   typed at the command line.

3. **Clearing and Initialization**
   We first set all variables to zero with a `clear` statement to avoid conflicts with previous definitions. Then we request a value of \( x \) to test and a maximum number of terms in the sum (which we denote \( N \)).
clear; % this sets variables to zero, so we start with a clean slate

x = input('enter a test value of x: '); % get a value of x to try
N = 500; % maximum number of terms in the series to sum

You may need to decrease the number of terms in some cases to get a usable plot.

4. Creating Vectors

A vector in MATLAB can be created element by element. The first element of the vector \( nvec \) is \( nvec(1) \), the second element is \( nvec(2) \), and so on. We start the partial sum vector (called \( my\_sum \)) with the first two terms of the series (i.e., \( n = 0 \) plus \( n = 1 \)).

% We’ll create two vectors, nvec and my_sum. The i’th elements of the
% vectors are referenced by nvec(i) and my_sum(i), with i starting at 1.

nvec(1) = 1; % first element of the vector nvec = (1,2,3,...,N)
my_sum(1) = 1 - x; % start with the sum of the x^0 and x^1 terms

5. For Loop

A “for loop” steps through values of a variable; in the example here \( n \) starts at 2 and steps by ones up to \( N \). For each value of \( n \), the statements between the \texttt{for} line and the \texttt{end} statement are evaluated with the current value of \( n \). This does a particular sum here (which you can change!).

for n = 2:N % now add in the rest, i.e., n = 2, 3, 4, 5, ..., N
    nvec(n) = n; % build the nvec vector (1,2,3,4,...,N) entry by entry
    my_sum(n) = my_sum(n-1) + (-1)^n * (x)^n; % add the n’th term to the sum
end

6. Plotting the Results

Finally, we create a plot and add labels.

% now make a plot of the sum vs. the number of terms
plot(nvec,my_sum); % plot(x_vector,y_vector) plots one line
xlabel('n'); % add the x-axis label
ylabel('sum'); % add the y-axis label

7. Running the Script

Run the script from the command line by typing its name (no .m):

>> test_convergence

or by clicking the “run” button in the editor window.