Assignment I: First Steps and 2D Plots

Due 9/02/2014

1. Plot the functions

\[ j_0(x) = \frac{\sin x}{x} \]
\[ j_1(x) = \frac{\sin x}{x^2} - \frac{\cos x}{x} \]

for \( x \in [0, 10] \). Write a code to create your data input file for \textit{xmgrace} and take special care for \( x \to 0 \).

2. Complex Numbers

A complex number \( z \) is defined in terms of its real and imaginary parts as

\[ z = x + iy = r \ e^{i\phi}, \]

where \( r = \sqrt{x^2 + y^2} \) and \( \phi = \tan^{-1}\left(\frac{y}{x}\right) \).

(a) Write a program that gets the computer to print a table of the form

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>x</th>
<th>y</th>
<th>( \sqrt{z} )</th>
<th>( \ln z )</th>
<th>atan(y/x)</th>
<th>atan2(y,x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4\pi</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>15\pi/4</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>4\pi</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Here \( \phi \) will increase with uniform steps, and the other columns are to be the computer’s output. A start of the code is given as \textit{cnumbers.f90}. Look up the relevant Fortran intrinsic functions in the SunStudio 12 Fortran Library Reference under Fortran 95 Intrinsic Functions (see under references on the class URL). You may choose \( r = 1 \) for the magnitude, but should try with more than one value.

(b) Make a plot of the output phases obtained with the arctangent functions versus the input phase \( \phi \).

(c) If your plotting program appears to be making some strange jumps, you may need to use more points near a multiple of \( \pi/2 \) and avoid being precisely “at” a multiple of \( \pi/2 \).
(d) If your compiler is not bright enough to automatically use a complex library routine when you feed it a complex number, you may have to look up the particular function name required to evaluate a complex function. See SunStudio 12 Fortran Library.

(e) State clearly where the computer has placed the cuts for the sqrt, ln, atan, and atan2 functions. Compare with the descriptions in the SunStudio 12 Fortran Library.

3. Summing Series

For this exercise we examine the power series for the exponential function

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots$$

(3)

We want to use this series to calculate $e^{-x}$ for $x = 0.1, 1., 10., 100., 1000.$, with an absolute error in each case of less one one part in $10^8$.

Assuming that the error in the summation is approximately the last term summed, the absolute error of one part in $10^8$ is given as

$$\left| \frac{term}{sum} \right| < 10^{-8},$$

(4)

where $term$ is the last term in the series (3), and $sum$ is the accumulated sum.

(a) First write down for yourself a pseudo-code for the given $x$ values. This pseudo-code should be included in your write-up. You may use the provided code `exp-good.f90` as template, however you need to make sure you understand it and explain how it works in your write-up.

Present your results in a table of the form

| $x$ | $imax$ | $sum$ | $\left| \frac{sum-\exp(-x)}{sum} \right|$ |
|-----|--------|-------|--------------------------------|
|     |        |       |                                |

| ... | ... | ... | ... |

where $\exp(-x)$ is calculated with the built-in exponential function.

(b) Modify your code from above to one that calculates the sum in a ‘bad way’, i.e. uses explicit factorials. Then assess the performance of the two different codes:

1. Observe how the good and bad series summations fail for large $x$. In particular notice whether there are underflows or overflows.
2. Produce a table for the ‘bad’ code similar to the one for the ‘good’ code.

3. Use a built-in timing function on your computer to compare the execution times for each method. (use the command `time` with your executable).