
3a)
In the case of sufficiently large numbers for x the computation of this function will fail because of the additon of 1 to x. When using double precision numbers for example, the significand isn't big enough to display the result of the addition if the number is equal or above 10^{15} because double precision numbers only represent to an accuracy of 15 digits. This causes a rounding error which in turn causes the function to essentially ignore the addition of the 1 itself and simply returns 0.

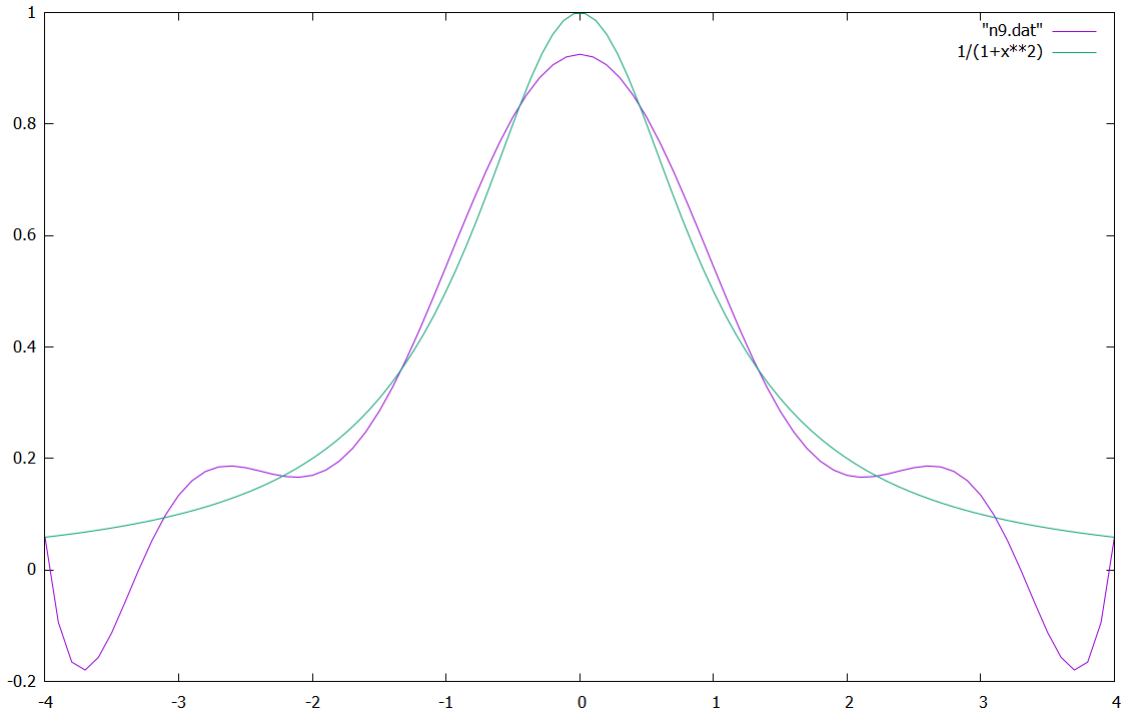
3b)
This issue can be fixed by either using a number format that doesn't exceed the usage range of the function itself or more generally by using a series to expand the function, that doesn't require us to add a fixed number. For example we can use the Puiseux series to expand our function. For simplicities sake and because it is enough for this example we'll only be using the first term of the series (see problem3.c):

$$
\begin{aligned}
f_{series}(x) &= \frac{3\sqrt[4]{\frac{1}{x}}}{4} + O((\frac{1}{x})^1) \\
f_{actual}(10^{20}) &= 7.4999999999999999999062500... \cdot 10^{-6} \\
f_{series}(10^{20}) &= 7.5000000000000001 \cdot 10^{-6}
\end{aligned}
$$

The actual value was obtained using WolframAlpha and comparing both reveals that even only using the first term of our approximation gives us as much accuracy as possible using doubles.

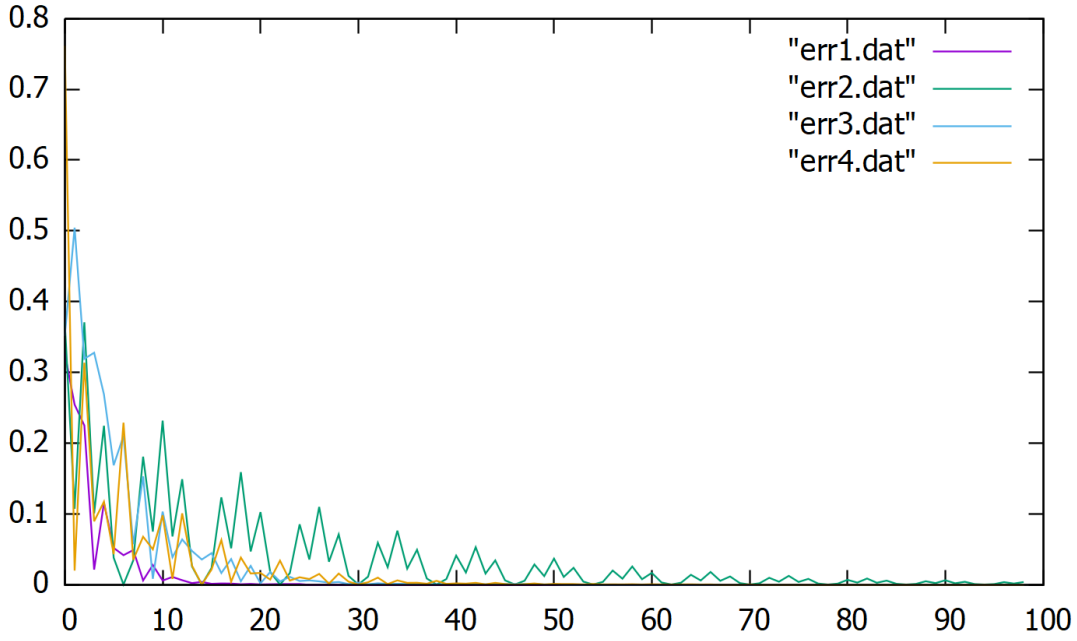
4a)
The provided code/program simply outputs the dataset for 4b (called n9.dat) as well as a couple of datasets for 4c (called err*.dat). Functionality for the seperate parts of the code/functions can be found in the comments.

4b)



4c)
Using different x_c and x I wasn't able to find a conclusive solution to this problem as most values I choose simply result in the error converging towards 0 anyways. Here are a couple of examples:

err1.dat:	$x_c = 4$	$x = 0.75$
err2.dat:	$x_c = 4$	$x = 3$
err3.dat:	$x_c = 8$	$x = 0.75$
err4.dat:	$x_c = 8$	$x = 3$



Error as a function of n