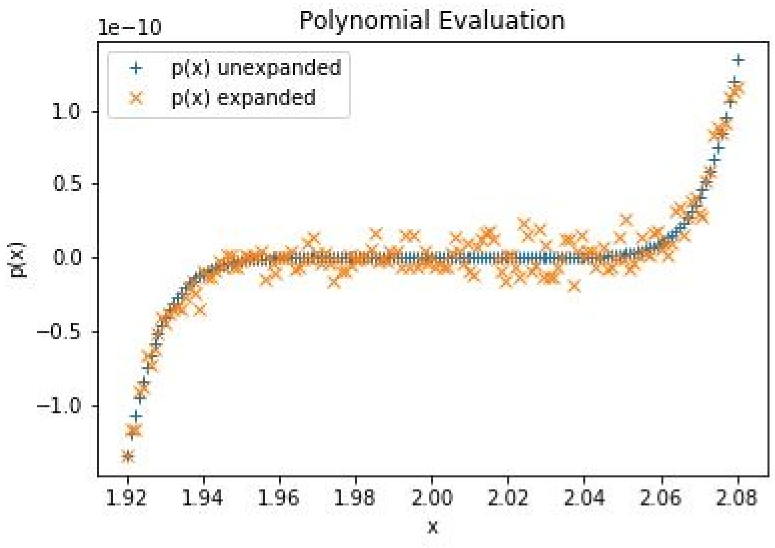
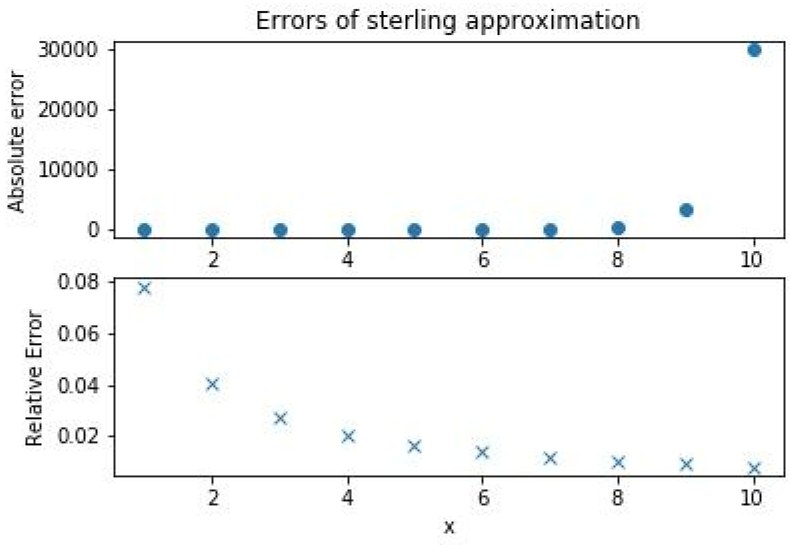
Scientific Computing: Homework 1

Harsh Bandhey 2017234

Problem 1: Errors in polynomial evaluation



Problem 2: Errors in Stirling’s approximation

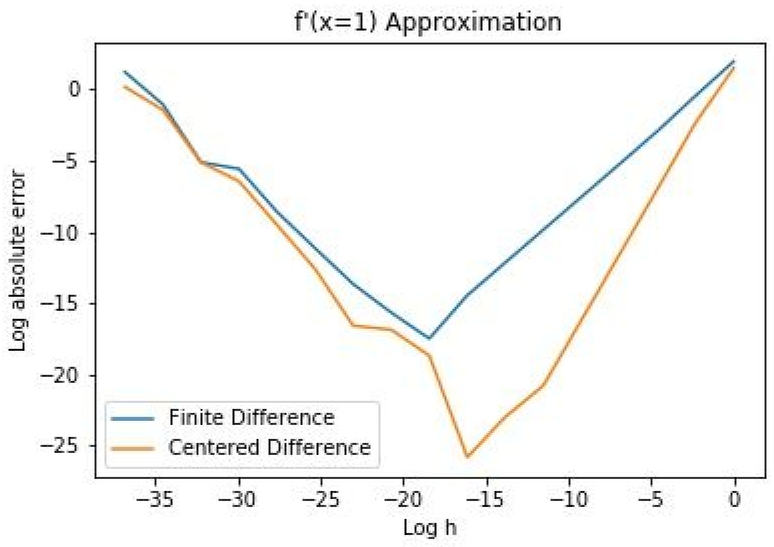


Problem 3: Analysis of Computational Errors

(a) For the polynomial evaluation, both from the graph and understanding of floating-point operations I would choose the unexpanded form. As we can see on the graph the expanded form is very much off the true values throughout the graph, this closely corresponds with the fact that that the expanded for has more floating-point operations per calculation, both rounding and machine error and this gets added upon each floating-point operation in a calculation of the

(b) As seen in the graphs, for Stirling's approximation, the absolute error increases and the relative error decreases with the increase in *n*.

Problem 4: Finite differences and ε_M



We examine the graph for Finite Difference Approximation and there is a minimum log absolute error, thus there exists a minimum absolute error. Through in-code analysis, we find that it is 2.5541353476654 65e-08 on our runtime. This occurred at $h=10^{-8}$. In numpy, the ε_M i.e. the precision value of np.float64 is 10^{-16}. This corresponds very well with the correlation that $h \approx \sqrt{\varepsilon_M}$.

We also examine the same for Centered Difference Approximation. In this case, also there is a minimum log absolute error, thus there exists a minimum absolute error. In this case, the in-code analysis gave a slightly smaller minimum error of 6.2239102760486276e-12 occurring at $h = 10^{-7}$ which is slightly bigger than the value of h for which the minimum occurred using the Finite-Difference formula. Though this is more than 10^{-8} we expect the relation $h \approx \sqrt{\varepsilon_M}$ to be the same. The machine error of the Centered Difference Approximation soon overtakes the truncation error.