Math 280: Project 2

(solving polynomial equations)

Overview

In class, we've discussed methods for solving nonlinear equations, especially systems of polynomial equations. In this project, you will find and discuss some polynomial equations "in the wild", implement three methods:

- [one variable] the bisection method (3.1), [already done on HW]
- [multivariable] Newton's method (3.3),
- [one variable] the secant method (3.4),

and then test those methods on your examples and analyze the results. Analysis will include comparison to a reference implementation of the homotopy method [one variable], and integration with a reference implementation using Gröbner bases for dimension reduction [multivariable].

A template will be provided, and projects will be coordinated through CoCalc.

Purpose Statement

1. Gain hands-on experience implementing root-finding techniques on a computer.
   (a) Knowledge: understand the technique both conceptually, as a mathematical object/operation, and concretely, as a computer program.
   (b) Skills: programming, translation from abstract theory and concepts to a concrete implementation.
2. Learn to identify real-world problems which are amenable to solution by these methods.
   (a) Knowledge: what questions are asked in other disciplines that we might be able to answer?
   (b) Knowledge: when do our techniques fail? when do they succeed?
   (c) Skills: assess problems arising outside of mathematics for their amenability to solution by the methods you implement.
   (d) Skills: communication, explaining problems from other fields at a level your peers can appreciate and understand.
3. Practice reflecting on quantitative and qualitative aspects of problems in the sciences, in order to to anticipate their behavior w/r/t numerical methods. Then, assess and enrich your understanding by comparing your expectations to your results and the results of standard numerical tools.
   (a) Knowledge: more detailed quantitative and qualitative aspects of the problems you've selected.
   (b) Knowledge: typical challenges for the techniques we've implemented.
   (c) Knowledge: familiarity with using standard tools and libraries.
   (d) Skills: hypothesize about mathematical phenomena.

(e) Skills: distinguish suitable/unsuitable/less suitable techniques for a given application.

(f) Skills: analyze data (program output) for patterns,

(g) Skills: organizing a report, mechanics of writing and grammar, using LaTeX to typeset mathematical documents.

Task

1. Proposal/Examples [$<$ 1 page]. Find examples of the following kinds on which to test your software:

   - Two univariate polynomial equations, and one system of polynomial equations appearing in physics, chemistry, &c. A good place to start is your Application Log.
   - One non-polynomials to test your implementation of Newton's method or the secant method.
   - For each univariate polynomial, a degeneration and a choice of connecting them to the original.

   Write a brief summary of your examples, which should briefly explain the science underlying the problem in general terms, and specifically state the polynomial(s) you will study. This can be taken directly from your Application Log, expanded if necessary.

   Choose at least one example which you expect to cause problems for some of our numerical methods (e.g. due to singularities). It is usually possible to adjust the physical parameters of your chosen problems to achieve this; you do not need to find physically realistic or achievable parameters if doing so is difficult. These make for a more interesting report!

   Submit your proposal by November 8th, 10pm.

2. Pre-port [1-2 pages]. As we saw in lecture and the readings, these methods are not perfect. With those weaknesses in mind, think about whether you expect these examples to work well with each method? Why or why not?

   The examples where you expect something to go wrong are among the most interesting, because they help us understand the limits of these techniques; historically, they often motivated the development of more refined methods.

   Discuss your expectations for each polynomial, and an explanation for your predictions, especially with respect to the methods. The methods you'll implement all have error bounds discussed in class, which would merit some discussion here (and in the post-port). Your predictions do not need to be correct!

   Submit the pre-port by November 11th, 10pm.

3. Implementation. Using SageMath, write a small program which implements the following root-finding methods discussed in class:

   - [one variable] the bisection method (3.1), [already done on HW]
   - [multivariable] Newton's method (3.3),
   - [one variable] the secant method (3.4),

Your program should take as input a polynomial or standard function, initial guess/endpoints as appropriate, and output (approximations to) its roots. Since these methods have precise error bounds, you might consider building them into your code, even if it means asking the user to provide the constants.

For simplicity, you can take input in the form of clearly labeled modifiable parameters in the source.

Implementations should be finished by November 15th, 10pm, and we will have a lab day(s) on [TBD!].

4. Testing. Apply your implementations to your examples. In your final report, be sure to record the inputs and outputs together, for reproducibility.

   You should also use the provided implementation of homotopy continuation (univariate) for comparison, and the provided Gröbner basis tool for reducing (multivariable) systems of equations to single-variable equations, to which you can apply your univariate methods.

5. Post-port [1+ pages]. Analyze your results. Compare the results to each other, to the standard library, and to your expectations.

   - For each example, compare the results of each method.
   - For each method, compare the results of each example.

   Things to think about while working on this: what about the examples make them more suitable for certain methods, and what about the methods make them more suitable for certain example? Where there are large differences, even expected ones, try to explain them. Since we have fairly precise error bounds for most the three methods you'll implement, it would be good to comment on them here.

   Due by November 19th, 10pm.

6. Report. Reflect on the process and summarize your results in a brief 4-5 page report using the provided template. Upload your source code separately. The report is a compilation-of and expansion-to the preceding parts of the task.

   In particular, it should consist of:

   (a) (< 1 pages, Proposal) Background on the examples you chose.

   (b) ($1 - 2$ pages, Pre-port) Summary of your expectations for those examples. Be sure to make clear *why* you predict certain behavior.

   (c) (1 or 2 pages, Testing/Implementation) Summary of results, including tables and numerical data, plots or charts, and other figures/information you deem relevant.

   (d) (2 pages, Post-port) Analysis of the results.

   Submit a draft by November 19th, 10pm, and the final version by November 22nd, 10pm.

Criteria

Several of the task items above are due before the main report. Turn them in on time for full credit and so that I am able to provide you feedback! Excellent drafts will require very little work to get them to their final form, and the final report is, in part, a compilation of these components, so it is in your best interest to polish them as much as possible along the way.

- **(10%)** Proposal. Provides examples (3%) with background (3%), and explains them clearly (4%). Your target audience is a peer not already familiar with the field and examples you've selected.
- **(20%)** Pre-port. States expectations for each examples (5%), and provides clear mathematical-heuristic explanations for the behavior (15%).
- **(10%)** Implementation. Accurate implementation (8%), readable source augmented by clarifying comments (2%).
- **(5%)** Testing. Summarized clearly in a table.
- **(35%)** Post-port. Analysis of the results. Quality of the comparison: not just *what* differences appear, but *why* you think they happened. Be sure to keep in mind and reference your pre-port, both where your expectations did and did not match reality, and try to explain discrepancies.
- **(10%)** Report. Combines all the described components (2%), and shows evidence of reflection on the process (6%). Incorporates draft feedback (2%).
- **(10%)** Writing quality (grammar, style).

There are not necessarily "right" or "wrong" analyses: your objective (did/did not work) and numerical results are largely unambiguous, but I want to read ***your own, original, thoughts*** about what you think happened. You just need to reflect, thoughtfully and critically, on your observations and make an effort to explain how they arose, then *clearly communicate those explanations, with justification,* to your reader. This is challenging.