CS510 Project
Robust Newton’s Method

Group 2: Baichuan Huang, Jiawei Wu, Zelong Li

Introduction

This project is about implementing the Robust Newton’s Method and visualize its results, also comparing it with Newton’s Method.

The Robust Newton’s Method is a polynomial root finding algorithm, it is a fixed iteration method. There are some fixed point iteration methods, such as Newton’s method, Halley’s method, Horner’s method and etc. Since Robust Newton’s Method is meant to overcome some critical points of Newton’s Method, we were mainly comparing these two methods.

We were considering a complex polynomial

$$p(z) = a_n z^n + ... + a_1 z + a_0$$

with coefficients are complex values, $z = x + iy,\ i = \sqrt{-1}$ and x and y are real values.

Newton’s Method is defined as follows:

If z_j is an approximation of $p(z) = 0$ and $p'(z_j) \neq 0$, then, the next iteration $z_{j+1} = z_j - \frac{p(z_j)}{p'(z_j)}$.

Robust Newton’s Method is defined as follows:

If z_0 is an approximation of $p(z) = 0$, then, the next iteration $\hat{N}_p(z_0) = z_0 + \frac{c_k}{3}\frac{u_k}{|u_k|}e^{i\theta}$, where

$$\begin{aligned} k &= \min\{j \in \{1, \cdots, n\} : p^{(j)}(z_0) \neq 0\} \\ u_k &= \frac{1}{k!} p(z_0)\overline{p^{(k)}(z_0)} \\ \gamma &= 2 \cdot Re\left(u_k^{k-1}\right) \\ \delta &= -2 \cdot Im(u_k^{k-1}) \\ c_k &= max\{|\gamma|, |\delta|\} \end{aligned}$$

$$\theta = \begin{cases} 0, & if\ c_k = |\gamma|, \gamma < 0 \\ \pi/k, & if\ c_k = |\gamma|, \gamma > 0 \\ \pi/(2k), & if\ c_k = |\delta|, \delta < 0 \\ 3\pi/(2k), & if\ c_k = |\delta|, \delta > 0 \end{cases}$$

$$A = \max_{j \geq 0}\left\{\frac{\left|p^{(j)}(z_0)\right|}{j!}\right\}$$

Moreover, we have a Hybrid Newton’s Method which mainly is Newton’s Method except the Robust version was used to eliminate bad points that occurred in Newton’s Method.

The implementation will be discussed next, and comparisons and visualization will be present as the last part of this report.

Implementation

The code has been tested running under Python 3.6.9. The required packages are as follows:

- numpy == 1.17.4
- sympy == 1.4
- matplotlib == 3.1.2

Numpy package is useful for scientific computing, sympy package helps to calculate the derivative of polynomial, and matplotlib package is a well-known tool for numerical visualization.

The final version of this project has been released on https://github.com/baichuan55555/CS510-Project-1

Example to run the codes: There are three main python files: newton.py, robust.py, hybrid.py. More specifically, run “python newton.py” to show the result of Newton’s Method, run “python robust.py” to show the result of Robust Newton’s Method, and run “python hybrid.py” to show the result of Hybrid Newton’s Method. After running one of these files, one line of input should be added, which is the polynomial (related to variable “x”) to be visualized, e.g., “x**3 - 1”.

Hybrid Newton’s Method and Newton’s Method can be visualized in a very short time, usually several seconds, while Robust Newton’s Method usually takes several minutes. Please be patient and have a cup of coffee or tea until the result appears.

The following figures and tables can be used to show detailed implementation. The first table shows the parameter. The parameter is used the same for each kind of Newton’s Method.

Table 1. Parameters of Newton’s Method

Parameters	Definition	Default
max_iter	Maximum time of iteration accepted	500
tol	Tolerance of error	$1 * 10^{-8}$

The first figure is the flow chart of Newton’s Method and Robust Newton’s Method.

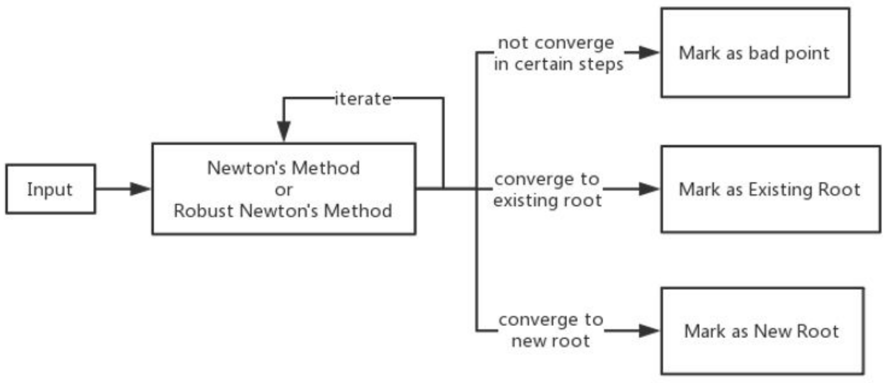

Input
Newton's Method or Robust Newton's Method
iterate
not converge in certain steps
Mark as bad point
converge to existing root
Mark as Existing Root
converge to new root
Mark as New Root

Fig 1. Robust Newton’s Method and Newton’s Method

From the second figure, how to properly combine Newton's Method and Robust Newton's Method in Hybrid Newton's Method has been clearly shown, which takes full advantage of the efficiency of Newton's Method and the robustness of Robust Newton's Method.

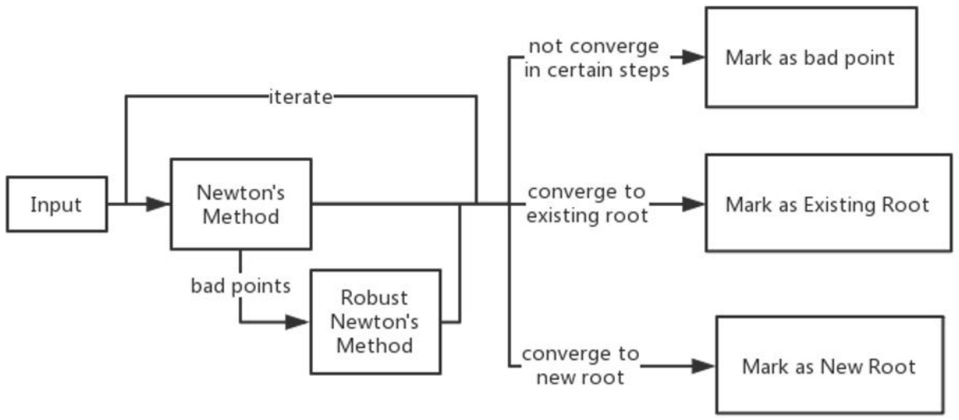
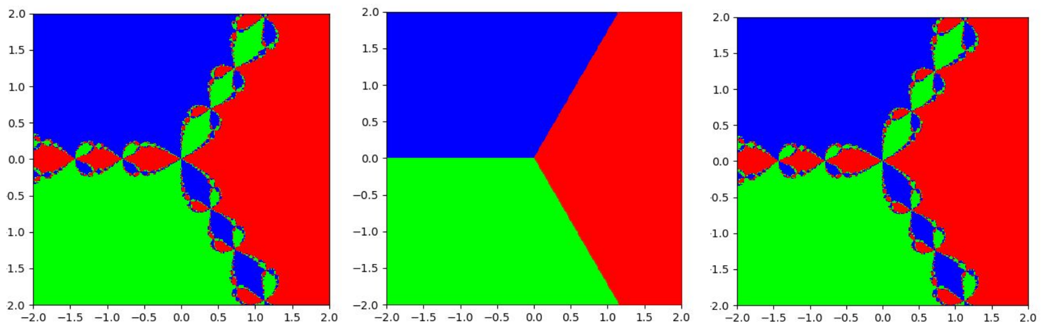


Fig 2. Hybrid Newton's Method

Experiments

In this part, we pick some polynomials and input them into our algorithms. We show the output of each algorithm, do a comparison and give a brief analysis of each image. The parameters of these plots as follows: the interval of the real part is $[-2, 2]$ and the complex part is the same; the resolution of plots is 400×400.

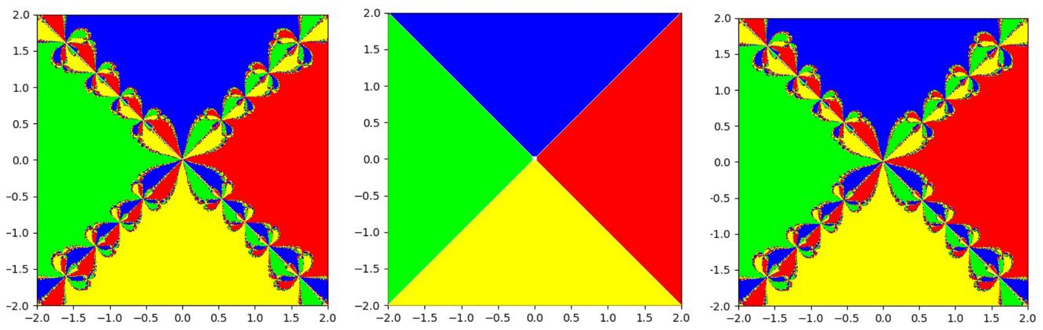
- Polynomial: $P(z) = z^3 - 1$
  - Description: Very simple polynomial with one real roots 1 and two image roots $-\frac{1}{2} \pm \frac{\sqrt{3}}{2} i$, we use this polynomial to test whether our algorithms work as we expected.
  - Results:



Newton’s method, robust Newton’ s method, hybrid Newton’s Method from left to right.

- ○ Observation: We can see from the image that the boundary between basin of attractions is quite complex in Newton’s and hybrid Newton’s method. On the contrast, Robust Newton’s method gives a clear and nice border.
- ○ Analysis: Hybrid Newton’s method has similar performance as Newton’s method as hybrid Newton’s method is based on original Newton’s method and we do not have many bad points in this example. Robust Newton’s method has good performance as we expected.

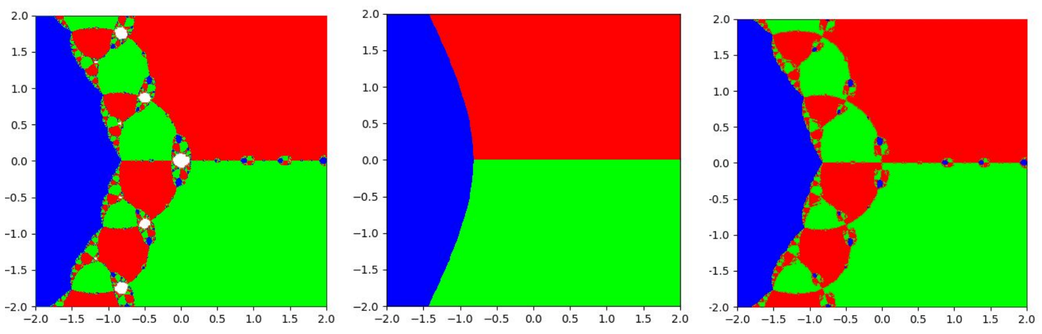
- Polynomial: $P(z) = z^4 - 1$
  - ○ Description: Very simple polynomial with two real roots ±1 and two image roots ±i, we use this polynomial to test whether our algorithms work as we expected.
  - ○ Results:



Newton’s method, robust Newton’ s method, hybrid Newton’s Method from left to right.

  - ○ Observation: We can see from the image that the boundary between the basin of attractions is quite complex in Newton’s and hybrid Newton’s methods. On the contrast, Robust Newton’s method gives a clear and nice border. What is more, if we zoom in, we can see the bad points in each image (represented as white areas)
  - ○ Analysis: Hybrid Newton’s method has similar performance as Newton’s method as hybrid Newton’s method is based on original Newton’s method and we do not have many bad points in this example. Robust Newton’s method has good performance as we expected. The reason why we do not observe bad points in former experiment is that each pixel in our image represents 0.01 in complex plane. The smaller the bad areas are, the less likely that it would be observed in our program.

- Polynomial: $P(z) = z^3 - 2z + 2$
  - ○ Description: We choose this polynomial because when we apply the original Newton’s method and start from 0, it will lead us to 1. However, if we continue from 1, it will lead us back to 0.
  - ○ Results:

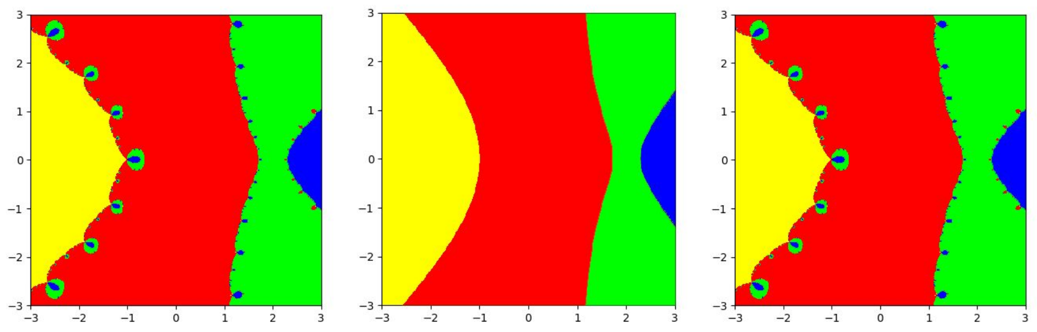


Newton’s method, robust Newton’ s method, hybrid Newton’s Method from left to right.

- ○ Observation: We can see the bad points clearly in the image generated by Newton’s method. Robust newton’s method avoids those bad points and generate “good” image. Hybrid Newton’s method has similar pattern as Newton’s method. However, it successfully avoid those bad points.
- ○ Analysis: We say robust Newton’s method generates “good” image as we are supposed to observe critical points near $\pm\frac{\sqrt{6}}{2}$, which do not show up in this image. It is mainly because the resolution of our image is so low that the algorithm skips the critical points during iterations.

● Polynomial: $(x+2)(x-1.5)(x-2.0)(x-2.5)$

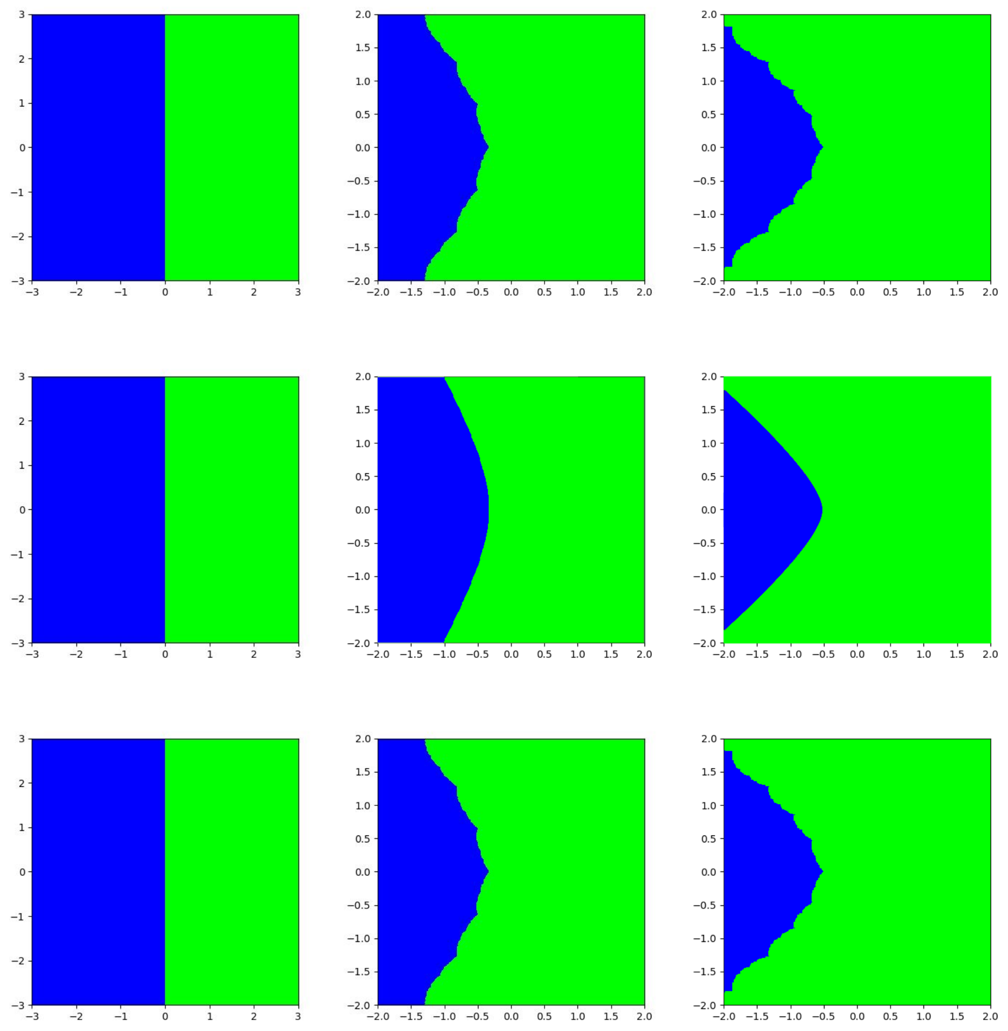
- ○ Description: In this experiment, we put all the roots on a straight line. Notice that the last three roots are closer to each other than the first one.
- ○ Results:



Newton’s method, robust Newton’ s method, hybrid Newton’s Method from left to right.

- ○ Observation: Except for the basic conclusion that same as former experiments(clear border in robust Newton’s, the same pattern between Newton’s and hybrid Newton’s), we find out that the green pattern is obviously narrower than the red one.
- ○ Analysis: The green pattern is narrower as the roots are denser.

- Polynomial: $(x+1)(x-1)^k$
  - Description: In the last experiment, we want to see how will multiply roots affect the performance of the algorithms. We choose k=1, 2 and 3.
  - Results:



K = 1, 2, 3 from up to down, Newton's method, robust Newton' s method, hybrid Newton's Method from left to right.

  - Observation: The green pattern gowns larger as we increase k from 1 to 3.
  - Analysis: The basin of attraction is larger for multiple roots. As it grows from double roots to triple roots, the basin of attraction grows larger.

Reference:

- Bahman Kalantari "Robust Newton Method for Polynomials"