MATH60006/70006 Applied Complex Analysis — Dr Andrew Gibbs

Coursework

In this coursework, you will use the *argument principle* to implement and analyse a root-finding algorithm. For f analytic in a simply connected domain $D \subset \mathbb{C}$, we define

$$I[f;\beta,\gamma] := \frac{1}{2\pi i}\oint_\gamma \frac{f'(z)}{f(z)} z^\beta dz, \quad \text{for } \beta \in \mathbb{N}\cup\{0\}, \tag{1}$$

where γ is a closed anti-clockwise-oriented contour in D, without any loops. Below, Algorithm 1 uses equation (1) to search for a root of f inside of a circular subset of D.

Algorithm 1 Basic root-finder

1: **function** BasicRootFinder($f, \tilde{z}, \mathcal{R}$)

2: $$\gamma_\mathcal{R} \leftarrow \{\tilde{z} + \mathcal{R}e^{i\theta} : \theta \in [0, 2\pi]\}$$

3: $$m_0 \leftarrow I[f; 0, \gamma_\mathcal{R}]$$

4:     **if** $m_0 \geq 1$ **then**

5: $$z_0 \leftarrow I[f; 1, \gamma_\mathcal{R}]/m_0$$

6:         **return** z_0, m_0

7:     **end if**

8: **end function**

Throughout this project, you should assume the following:

Assumption 1 *For all contour integrals over $\gamma_\mathcal{R}$ (as defined in Algorithm 1), $\mathcal{R}$ is chosen such that $\gamma_\mathcal{R} \subset D$ and $f(z) \neq 0$ for $z \in \gamma_\mathcal{R}$.*

1. This question is about understanding when Algorithm 1 works, and when it fails. For this question, you should assume that all integrals in lines 3 and 5 are caclulated in exact arithmetic.

   (a) State further assumptions about the roots of f and the contour $\gamma_\mathcal{R}$, such that Algorithm 1 returns an output.

   [1 mark]

   (b) State further assumptions about the roots of f and the contour $\gamma_\mathcal{R}$, such that the output
   $$z_0, m_0 = \texttt{BasicRootFinder}(f, \tilde{z}, \mathcal{R}),$$
   satisfies $f(z_0) = 0$.

   [2 marks]

   (c) Given $\tilde{z}$ and $\mathcal{R}$, construct an f such that Algorithm 1 returns
   $$z_0, m_0 = \texttt{BasicRootFinder}(f, \tilde{z}, \mathcal{R}),$$
   where
   $$f(z_0) \neq 0.$$

[2 marks]

2. Typically in practice, we are unable to compute the integrals on lines 3 and 5 of Algorithm 1 in exact arithmetic. This question is about the design and analysis of an approximation to the general case, as defined in (1).

   (a) For $\gamma_{\mathcal{R}}$ defined as in line 2 of Algorithm 1, show that an N-point trapezium rule approximation to (1) is given by

$$\begin{aligned} I[f;\beta,\gamma_{\mathcal{R}}] &\approx I_N[f;\beta,\gamma_{\mathcal{R}}] \\ &:= \frac{\mathcal{R}}{N}\sum_{n=1}^{N}\frac{f'(\tilde{z}+\mathcal{R}e^{i\theta_n})}{f(\tilde{z}+\mathcal{R}e^{i\theta_n})}(\tilde{z}+\mathcal{R}e^{i\theta_n})^{\beta}e^{i\theta_n}, \end{aligned} \tag{2}$$

   where $\theta_n := 2\pi n/N$ for $n = 1,\dots,N$.

[2 marks]

   (b) Assuming convergence

$$I_N[f;0,\gamma_{\mathcal{R}}] \to I[f;0,\gamma_{\mathcal{R}}], \quad \text{as } N \to \infty,$$

   explain why

$$I[f;0,\gamma_{\mathcal{R}}] = \text{round}\left(\text{Re}\{I_N[f;0,\gamma_{\mathcal{R}}]\}\right),$$

   for sufficiently large N, where

$$\text{round}(x) : \mathbb{R} \to \mathbb{Z}$$

   rounds to the nearest integer.

[2 marks]

   (c) Suppose now that f has no zeros in the closed annulus

$$\Omega_{\mathcal{R}_-,\mathcal{R}_+} := \{z \in \mathbb{C} : \mathcal{R}_- \le |\tilde{z} - z| \le \mathcal{R}_+\},$$

   for $0 < R_- < \mathcal{R} < R_+$. By referring to an appropriate theorem in the course notes, show that

$$|I_N[f;\beta,\gamma_{\mathcal{R}}] - I[f;\beta,\gamma_{\mathcal{R}}]| = O(e^{-aN}), \quad \text{as } N \to \infty,$$

   where

$$a = \max\left\{\log\frac{R_+}{\mathcal{R}}, \log\frac{\mathcal{R}}{R_-}\right\}.$$

[6 marks]

3. In this question, you will implement and test a modified version of Algorithm 1.

   (a) Some guidance:

   - Use a programming language of your choice. You should choose a language which supports complex arithmetic. My advice would be to use Matlab/Octave, Python, or Julia. You will need to be able to compute complex values of the Riemann-Zeta function, further information is given in (3(b)iii).
   - The integrals in lines 3 and 6 of Algorithm 1 should be approximated using the trapezoidal approximation (defined in equation (2) above); therefore your algorithm should contain an additional input N. You should also pass f' as a separate input. With this in mind, your function inputs should look similar to that in Algorithm 2 below.
   - You should manually code the trapezium rule yourself.
   - You should improve the accuracy for large N by incorporating knowledge from question (2b), adding a further modification to line 3.
   - Comment your code clearly.
   - Label your axes in your plots.
   - In each problem below, you should achieve machine precision accuracy, typically stagnating between 10^{-14} and 10^{-16} error, for larger values of N.
   - Noting the result of question (2c), you should observe exponential convergence for your method. Using a logarithmic scale for the absolute error on the vertical axis, this will look (roughly) like a straight line.

Algorithm 2 Your root-finder

function YOURROOTFINDER($f, f', \tilde{z}, \mathcal{R}, N$)

$\quad\vdots$

end function

   (b) Plot the absolute error (on a logarithmic scale, on the vertical axis) of your algorithm, against N (standard scale, on the horizontal axis), for the following experiments:

   i. Approximate the *golden ratio* $z_0 = (1+\sqrt{5})/2$, which is the positive root of the equation $f(z) = z^2 - z - 1$. Use $\mathcal{R} = 1$ with an initial guess of $\tilde{z} = 1$, for $N = 1, 2, \ldots, 80$

   ii. Approximate $z_0 = \pi$ using $f(z) = (1 + e^{iz})/e^{iz}$, for which π is a simple root (you do not need to prove this). Use $\mathcal{R} = 1$, with an initial guess of $\tilde{z} = 3$, for $N = 1, 2, \ldots, 25$.

   iii. Approximate the first non-trivial zero

$$z_0 = \frac{1}{2} + i14.13472514173469\ldots$$

   of the *Riemann-Zeta function* $f(z) = \zeta(z)$. Use initial guess $\tilde{z} = 1/2 + 15i$, $\mathcal{R} = 1$, for $N = 10, 20, \ldots, 400$.

   You should use pre-existing software to compute ζ, **do not attempt to code this yourself**. In Matlab/Octave this is simply `zeta`, in Python this can be done using recent versions of the SciPy library, and in Julia this can be done using the SpecialFunctions package. However, to compute $\zeta'(z)$, **you must use Cauchy's integral formula** over a circular contour of radius $1/10$ centred at z, approximated with an N-point trapezium rule (the same N as for your argument principle approximation, in the 'outer' integral).

[15 marks for full question]