Coursework Solutions

1. (a) For the algorithm to return an output, line 5 must be reached, equivalently $m_0 \geq 1$. This happens if and only if there is at least one root inside $\gamma_{\mathcal{R}}$, otherwise $m_0 = 0$.

   (b) We have $f(z_0) = 0$ if and only if z_0 is the only zero of f inside $\gamma_{\mathcal{R}}$.

   (c) The idea is to ensure at least two distinct zeros are inside the contour $\gamma_{\mathcal{R}}$. For example,

   $$f(z) = (\tilde{z} - z)(\tilde{z} + \mathcal{R}/2 - z),$$

   which has zeros as $\tilde{z}$ and $\tilde{z} + \mathcal{R}/2$, would work. Then the algorithm will return the average of the zeros. In the case of this example, $z_0 = \tilde{z} - \mathcal{R}/4$, $m_0 = 2$.

2. (a) First, we parametrise the integral $z = \tilde{z} + Re^{i\theta}$, $dz = iRe^{i\theta}d\theta$, to obtain

   $$\begin{aligned} I[f;\beta,\gamma_{\mathcal{R}}] &= \frac{1}{2\pi i}\int_0^{2\pi} \frac{f'(\tilde{z}+\mathcal{R}e^{i\theta})}{f(\tilde{z}+\mathcal{R}e^{i\theta})}(\tilde{z}+\mathcal{R}e^{i\theta})^{\beta}\mathcal{R}ie^{i\theta}\ d\theta \\ &= \frac{\mathcal{R}}{2\pi}\int_0^{2\pi} \frac{f'(\tilde{z}+\mathcal{R}e^{i\theta})}{f(\tilde{z}+\mathcal{R}e^{i\theta})}(\tilde{z}+\mathcal{R}e^{i\theta})^{\beta}e^{i\theta}\ d\theta \end{aligned} \tag{1}$$

   Now discretise with points $\theta_n = \frac{2\pi n}{N}$ for $n = 1, \ldots, N$, each with weight $2\pi/N$, to obtain

   $$I_N[f;\beta,\gamma_{\mathcal{R}}] = \frac{\mathcal{R}}{2\pi}\sum_{n=1}^{N} \frac{f'(\tilde{z}+\mathcal{R}e^{i\theta_n})}{f(\tilde{z}+\mathcal{R}e^{i\theta_n})}(\tilde{z}+\mathcal{R}e^{i\theta_n})^{\beta}e^{i\theta_n}\frac{2\pi}{N},$$

   which simplifies to the required result.

   (b) The integral $I[f;0,\gamma_{\mathcal{R}}] = m_0$ for some $m_0 \in \mathbb{N}\cup\{0\}$. Thus, for N which

   $$|I[f;0,\gamma_{\mathcal{R}}] - I_N[f;0,\gamma_{\mathcal{R}}]| < 1/2,$$

   it follows that m_0 is the nearest integer to $I_N[f;0,\gamma_{\mathcal{R}}]$.

   (c) We are given that f has no zeros inside of this closed annulus $\Omega_{\mathcal{R}_-,\mathcal{R}_+}$, and that f is analytic. Therefore, there must be a positive distance between the annulus and any zeros of f, so $|f|$ must be bounded below inside $\Omega_{\mathcal{R}_-,\mathcal{R}_+}$, and thus, the integrand must be bounded above:

   $$\max_{z\in\Omega_{\mathcal{R}_-,\mathcal{R}_+}} \left|\frac{f'(z)}{f(z)}z^{\beta}\,(z-\tilde{z})\right| \leq M$$

   for some $M > 0$.

   Next, find the complex values of θ which correspond to the values of $z(\theta)$ which are inside of the annulus $\Omega_{\mathcal{R}_-,\mathcal{R}_+}$, as these correspond to complex θ where the (parametrised) integrand is analytic. We have

   $$\mathcal{R}_- \leq |z(\theta) - \tilde{z}| \leq \mathcal{R}_+,$$

   so

   $$\mathcal{R}_- \leq \mathcal{R}\left|e^{i\theta}\right| \leq \mathcal{R}_+,$$

   now splitting θ into real and imaginary components:

   $$\mathcal{R}_- \leq \mathcal{R}e^{-\mathrm{Im}\theta} \leq \mathcal{R}_+,$$

   dividing by $\mathcal{R}$ and taking logarithms of each side:

   $$\log\frac{\mathcal{R}_-}{\mathcal{R}} \leq -\mathrm{Im}\theta \leq \log\frac{\mathcal{R}_+}{\mathcal{R}}.$$

Noting $\mathcal{R}_- < \mathcal{R} < \mathcal{R}_+$, the logarithm on the left is negative and the logarithm on the right is positive. The estimate follows by applying theorem 2.4 in the notes which tells us

$$|I - I_N| \leq \frac{4\pi M}{e^{aN} - 1}.$$

The result follows immediately, noting the large N behaviour from the denominator, and that M is independent of N.

3. Below is an implementation of Q3 in the Julia language, but the students have been told that any language is acceptable. Some marking guidance:

   - Five marks should be given for each correct plot.
   - Machine precision should be achieved at a similar value of N to my implementation.
   - The students should implement their own trapezium rule.
   - For part (3biii), the students should use existing code for the Riemann-Zeta function, but they should compute the derivatives using Cauchy's integral formula.
   - Students should round the order of the zero m_0 to the nearest integer, as instructed.
   - The absolute error should be on the vertical axis, plotted on a logarithmic scale.
   - Axes should be correctly labelled.
   - The students should observe exponential convergence. Assuming they have plotted the error on a logarithmic scale (as instructed), exponential convergence is indicated by a straight line.
   - Hence, if all three plots are produced and similar to mine, there should be no need to look at the student's code.
   - If the students are unable to produce any plots, marks should be given for 'correctness' of the code. Broadly speaking, their code should be doing the same thing as mine. They were told to comment their code, and the comments should indicate their intentions, it should not be necessary to read their code in detail.
   - A maximum of ten marks should be given for clearly commented code, where the student had clearly tried to implement an algorithm equivalent to mine, but they missed a bug (or a few), and their code did not produce any expected results.

Q3_sol_v2

February 3, 2025

[3]:
```
using Plots, SpecialFunctions, LaTeXStrings
```

1 Solution to Question 3

Question three on the *Applied Complex Analysis* Coursework is a coding question. I have implemented an answer in Julia, but the students are free to use any language they choose. They were told to comment their code, this should help determine what their intentions were. Obviously there is more than one way to implement this, for example, the student may implement a separate trapezium rule, rather than absorb it into their approximation as I have.

First, code up the functional $I_N[f, \beta, \gamma_{\mathcal{R}}]$

[6]:
```
function I_N(f::Function, df::Function, N::Integer, z::Number, R::Real, β::
↪Integer)
    # change of variables
    z(θ) = z + R*exp(im*θ)

    # trap rule
    θ_nodes = (1:N) * 2π/N

    # Return the summation derived in the coursework
    return R/(N) * sum(df.(z.(θ_nodes)) ./ f.(z.(θ_nodes)) .* (z.(θ_nodes)).^β .
↪* exp.(im*θ_nodes))

end
```

[6]:
```
I_N (generic function with 1 method)
```

Now I define my version of Algorithm 1:

[8]:
```
function my_root_finder(f::Function, df::Function, N::Integer, z::Number, R::
↪Real)
    # determine order of root
    m_0 = round(Int64, I_N(f, df, N, z, R, 0))
    if m_0  1
        # approximation to root
        z_0 = I_N(f, df, N, z, R, 1) / m_0
        return z_0, m_0
```

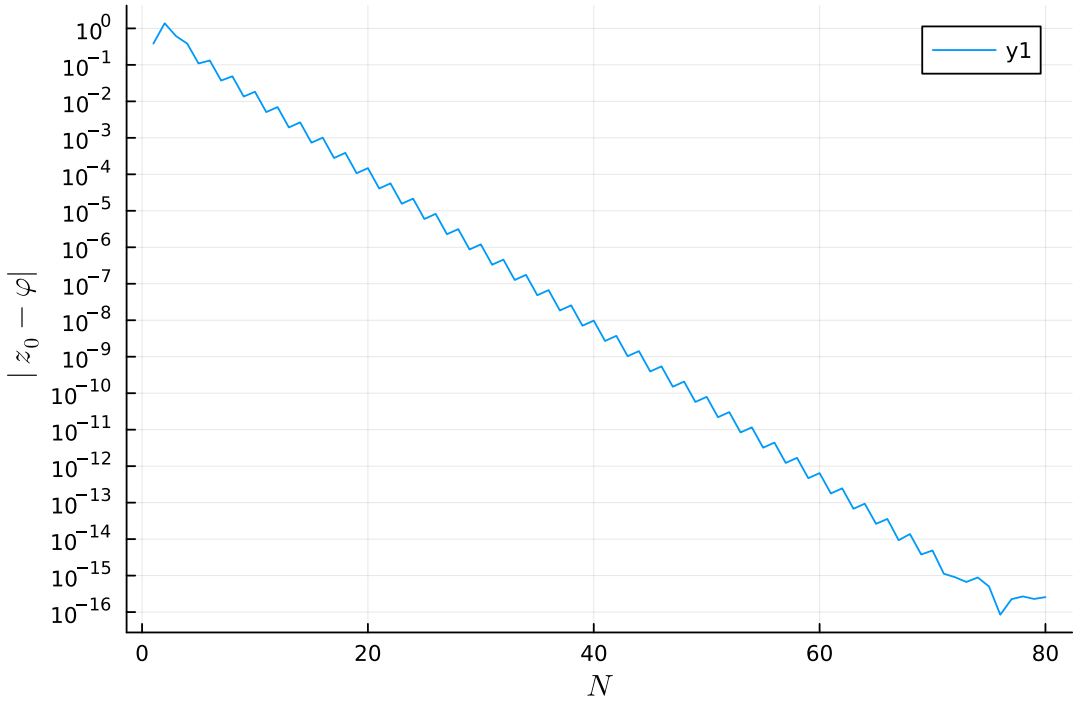
```
    end
end
```

[8]:
```
my_root_finder (generic function with 1 method)
```

1.1 Qb(i)

[10]:
```
R = 1
z = 1
# function and derivative
f(z) = z^2-z-1
df(z) = 2z - 1
#initialise values for approximation and errors
I_N_vals = zeros(ComplexF64, 80)
abs_errs = zeros(ComplexF64, 80)
# the golden ratio, which is a zero of f:
φ = (1+sqrt(5))/2
for N in 1:80
    I_N_vals[N], _ = my_root_finder(f, df, N, z, R)
end
abs_errs = abs.(I_N_vals .- φ)
yticks = 10.0 .^(1:-1:-16)
plot(1:80,abs_errs,y_scale=:log10, xlabel=L"$N$",␣
 ↪ylabel=L"|z_0-\varphi|",yticks=yticks)
```

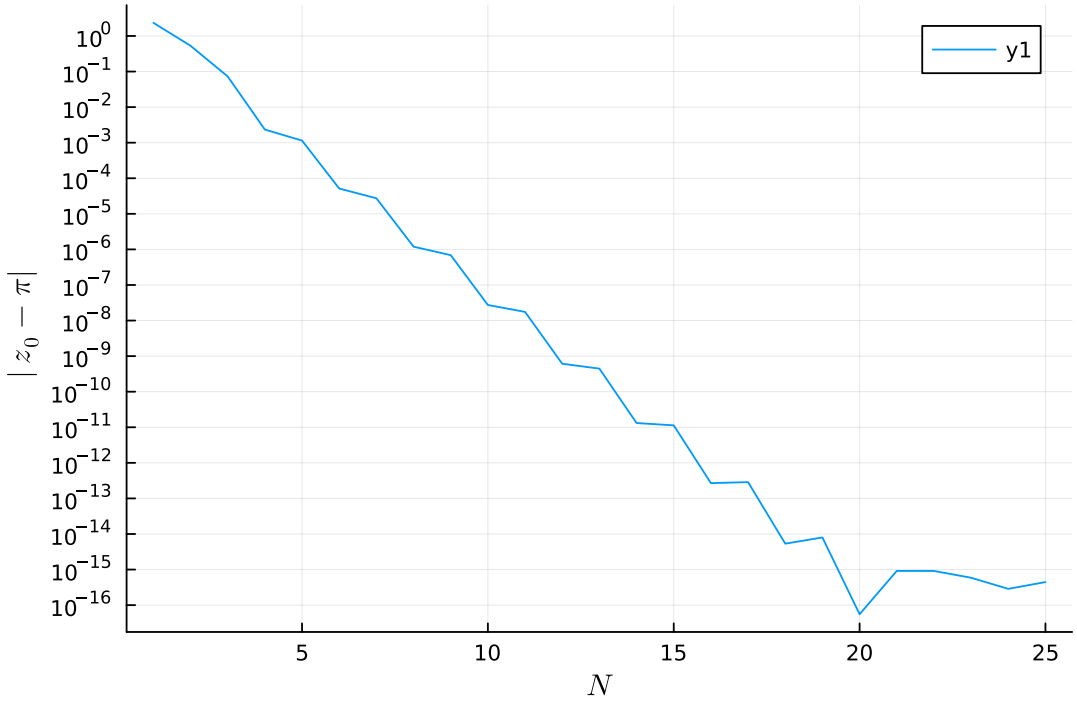
[10]:



1.2 Qb(ii)

[224]:
```
R = 1
z = 3
# function and derivative
f(z) = (1+exp(im*z))/exp(im*z)
df(z) = -im*exp(-im*z)
#initialise values for approximation and errors
I_N_vals = zeros(ComplexF64, 25)
abs_errs = zeros(ComplexF64, 25)
for N in 1:25
    I_N_vals[N], _ = my_root_finder(f, df, N, z, R)
end
abs_errs = abs.(I_N_vals .- π)
plot(1:25,abs_errs,y_scale=:log10, xlabel=L"$N$",␣
↪ylabel=L"|z_0-\pi|",yticks=yticks)
```

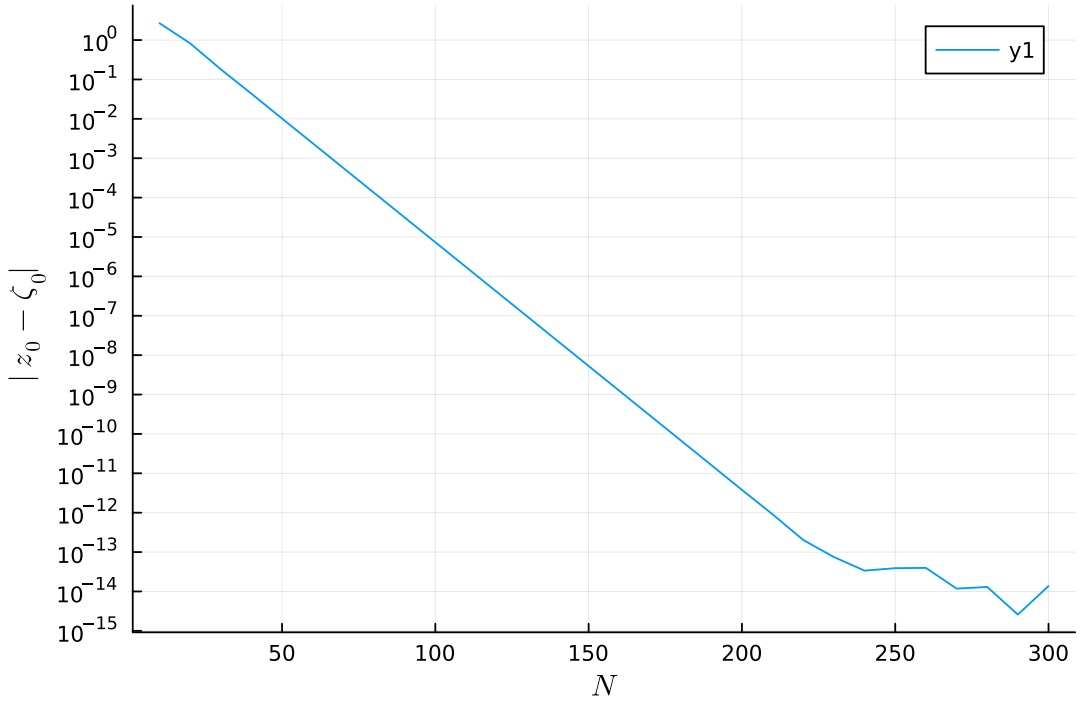
[224]:



1.3 Qb(iii)

[222]:
```
R = 1
z = 0.5+15*im

# zeta function
f(z) = zeta(z)

# derivative of zeta function, via a Cauchy integral
function zeta_diff(ξ, N, R)
    z(θ) = ξ + R*exp(im*θ)
    # trap rule
    θ_nodes = (1:N) * 2π/N
    return 1/N * sum(zeta.(z.(θ_nodes)) ./ (R*exp.(im*θ_nodes)))
end

#initialise values for approximation and errors
Nmax = 300
I_N_vals = zeros(ComplexF64, Int64(Nmax/10))
abs_errs = zeros(ComplexF64, Int64(Nmax/10))

count = 1
for N in 10:10:Nmax
    df(z) = zeta_diff(z, N, 1/10)
    my_root_finder(f, df, N, z, R)
    I_N_vals[count], _ = my_root_finder(f, df, N, z, R)
    count += 1
end

# the zero, accurate to 16 digits
z_0_exact = 1/2 + 14.13472514173469*im
abs_errs = abs.(I_N_vals .- z_0_exact)
plot(10:10:Nmax, abs_errs, y_scale=:log10, xlabel=L"$N$",␣
 ↪ylabel=L"|z_0-\zeta_0|",yticks=yticks)
```

[222]:


y1
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-14}
10^{-15}
$|z_0 - \zeta_0|$
50
100
150
200
250
300
N

[ ]: