CRISTIAN B. JETOMO
MATH 175 - H4L

First Long Exam (Computational Part)

In this problem, we are to model the propagation of sound in a two-layered ocean of constant depth in which we need to find the solutions of a transcendental function. For this, we consider an ocean environment of finite depth with two layers ($i = 1, 2$) having speed of sounds $c_1 = 1515$ m/s, $c_2 = 1500$ m/s and densities $\rho_1 = 1020$ kg/m^3 and $\rho_2 = 1029$ kg/m^3, respectively. For each layer, the depth is $h := h_1 = h_2 = 50$ m. An acoustic source in Layer 1 is placed which produces sound with frequency $f = \dfrac{c_1}{2\pi}$. The wave produced by the source in layer ($i = 1, 2$) will have different wave numbers given by

$$k = \begin{cases} k_1 &= \frac{2\pi f}{c_1}, \quad \text{in Layer 1} \\ k_2 &= \frac{2\pi f}{c_2}, \quad \text{in Layer 2} \end{cases} \tag{1}$$

Using these parameters, we solve the eigenvalues of the associated separated wave partial differential equations. These eigenvalues coincide with the solutions λ of the transcendental equation

$$\rho_1\gamma_2 \sin(\gamma_1 h) \sin(\gamma_2 h) = \rho_2\gamma_1 \cos(\gamma_1 h) \cos(\gamma_2 h) \tag{2}$$

where $\gamma_i = \sqrt{k_i^2 - \lambda}$ for $i = 1, 2$. For this problem, we reconstruct Equation 2 as a root-finding problem for the function g given by

$$g(\lambda) = \rho_1\gamma_2 \sin(\gamma_1 h) \sin(\gamma_2 h) - \rho_2\gamma_1 \cos(\gamma_1 h) \cos(\gamma_2 h) \tag{3}$$

We apply the **modifed Regula Falsi Method Anderson Bjorck variant** to find the second largest nonzero real root λ of Equation 3. We use the relative jump-based stopping criterion for our iterations with error tolerance $\epsilon = 10^{-6}$ and maximum iteration $N_{\max} = 100$.

To start our root-finding process, we analyze first the graph of the function g.

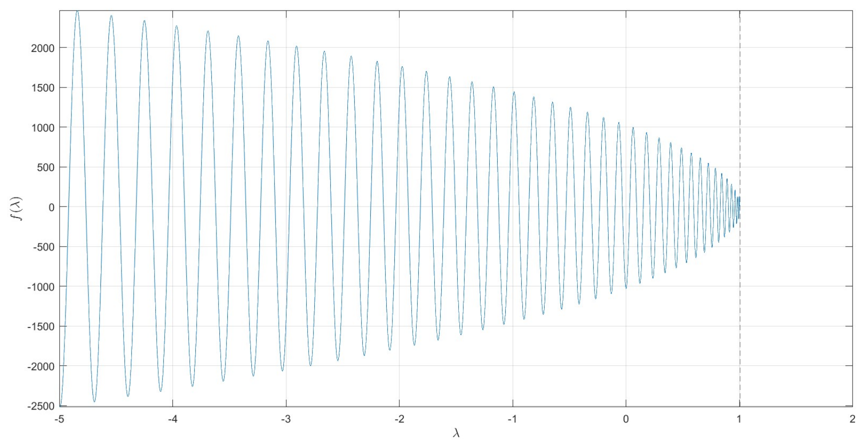

Figure 1: Plot of function $g(\lambda)$ over interval $[-5, 2]$.

As observed, the function is defined on the interval $(-\infty, 1]$. The figure also shows that the graph of g is fluctuating rapidly across the λ-axis which shows that it has infinitely many solutions with the largest located near $\lambda = 1$. The reason for this is that since the frequency $f = \frac{c_1}{2\pi}$, then from Equation 1, we have

$$k = \begin{cases} k_1 = 1 \\ k_2 = \frac{c_1}{c_2} = \frac{1515}{1500} = 1.01 \end{cases}$$

Consequently, $\gamma_1 = \sqrt{1-\lambda}$ and $\gamma_2 = \sqrt{1.0201 - \lambda}$. Since our goal is to find a nonzero *real* root λ, then

$$\begin{aligned} 1 - \lambda &\geq 0 \\ 1 &\geq \lambda \\ 0 \leq &\lambda \leq 1 \end{aligned} \qquad\qquad \begin{aligned} 1.0201 - \lambda &\geq 0 \\ 1.0201 &\geq \lambda \\ 0 \leq &\lambda \leq 1.0201 \end{aligned}$$

This implies that if $\lambda \leq 1$ and $\lambda \leq 1.0201$, then an upper bound for λ is 1, hence the graph we observed in Figure 1. We try to take a closer look at the graph of the function g.

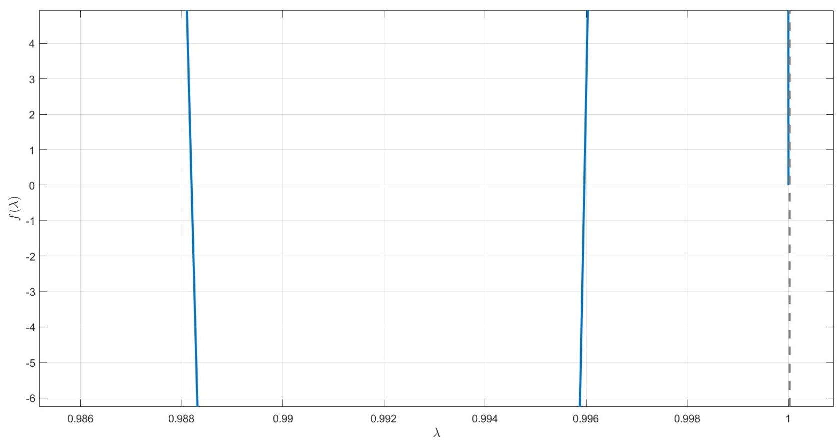

Figure 2: Plot of $g(\lambda)$ over interval $[0.986, 1]$.

We see from Figure 2 that the function has a real root near $\lambda = 0.996$ and $\lambda = 1$ since g passed the λ-axis at these points. To check if 1 is a root, we evaluated $g(1)$ and obtained from MATLAB that

```
The value of g evaluated at 1 is 0.000000.
```

Hence, 1 is a root of g. This implies that the root near $\lambda = 0.996$ is our desired root. So, in our iteration process, we shall use the initial interval $[0.990, 0.998]$ to find the desired second largest root. Implementing the algorithm in MATLAB, we obtained the results below.

```
n       a           b           c           rel jump
0       0.993513    0.998000    0.993513    NaN
1       0.993513    0.996640    0.996640    3.137570e-03
2       0.995493    0.996640    0.995493    1.152269e-03
3       0.995493    0.996041    0.996041    5.504103e-04
4       0.995953    0.996041    0.995953    8.871886e-05
5       0.995953    0.995963    0.995963    1.001103e-05
6       0.995962    0.995963    0.995962    1.840443e-07
Root is 0.9959624485 after 6 iterations.
Note that the function value at the estimate is -3.133681E-05.
```

Figure 3: Result of Implementing Modified Regula Falsi Method (Anderson-Bjorck algorithm) in MATLAB on function g.

As observed, the iteration process converges to an estimated root using the initial interval $[0.990, 0.998]$. This indicates that this obtained estimate is our desired root. That is, it is an estimate to the second largest nonzero real root λ of g. The estimated root obtained is $\lambda^* = 0.9959624485$ with $g(\lambda^*) = -3.133681 \times 10^{-5}$. This implies that our estimate is a good approximation to λ. The iteration process also enjoyed fast convergence since a good estimate is already obtained after only 6 iterations. This suggests that Regula Falsi with Anderson-Bjorck algorithm is a good method to solve the given problem.

Remarks: *I give my utmost acknowledgement to Ms. Alyssa Gabrielle Q. Salazar and Ms. Deanne Erica S. Valderama for their insights which helped me to answer this problem.*

APPENDIX A

Screenshot of MATLAB program used for the Modified Regula Falsi Method with Anderson-Bjorck Algorithm to solve the desired root for g.

```
clc
clear

% defines symbolic variable
syms L  % lambda

%% Initializing global variables

% densities and speed of sound
p1 = 1020;
p2 = 1029;
c1 = 1515;
c2 = 1500;

% depth
h = 50;

% frequency
f = c1/(2*pi);

% wave number
k1 = (2*pi*f)/c1;
k2 = (2*pi*f)/c2;

% gamma
S1 = sqrt(k1^2 - L);
S2 = sqrt(k2^2 - L);

% defined function
func = p1*S2*sin(S1*h)*sin(S2*h) - p2*S1*cos(S1*h)*cos(S2*h);
```

```
% for halting criterion
max_iteration = 100;
ErrorTol = 10^(-6);

% updating table
table = zeros(max_iteration, 5);

%% Evaluation at lambda = 1
f_1 = double(subs(func, 1));
fprintf('The value of g evaluated at 1 is %0.6f. \n', f_1)

%% initializing initial endpoints [a, b]
a = 0.99;
b = 0.998;

%% Plotting Given Function

figure
fplot(func)
xlim([-5, 2])
xlabel('$\lambda$', 'Interpreter','latex', 'FontSize', 12)
ylabel('$f(\lambda)$', 'Interpreter','latex', 'FontSize', 12)
grid
```

```

% display table title
fprintf('%s \t\t %s \t\t\t %s \t\t\t %s \t\t %s \n', 'n', 'a', 'b', 'c', 'rel jump')

% iteration counter
iter_count = 0;

while iter_count <= max_iteration

    % estimate
    if iter_count == 0  % first iteration

        % function values at current endpoints
        f_a = double(subs(func, a));
        f_b = double(subs(func, b));

        % current estimate
        c = (a*f_b - b*f_a)/(f_b - f_a);

        % function value at current estimate
        f_c = double(subs(func, c));

        % updating new interval
        if f_c == 0
            break
        elseif f_a*f_c < 0
            root_inLeftInt = true;
            b = c;
        else
            root_inLeftInt = false;
            a = c;
        end
```

```
    else

        % constant multiplier
        if root_inLeftInt == true
            if (1 - (f_c/f_b)) > 0
                const = 1 - (f_c/f_b);
            else
                const = 1/2;
            end
        else
            if (1 - (f_c/f_a)) > 0
                const = 1 - (f_c/f_a);
            else
                const = 1/2;
            end
        end

        % function values at current endpoints
        f_a = double(subs(func, a));
        f_b = double(subs(func, b));

        % saving previous estimate (for rel jump)
        prev_c = c;

        % current estimate
        if root_inLeftInt == true
            c = (a*f_b - b*(const)*f_a)/(f_b - (const)*f_a);
        else
            c = (a*(const)*f_b - b*f_a)/((const)*f_b - f_a);
        end
```

```
    % function value at current estimate
    f_c = double(subs(func, c));

    % updating new interval
    if f_c == 0
        break
    elseif f_a*f_c < 0
        root_inLeftInt = true;
        b = c;
    else
        root_inLeftInt = false;
        a = c;
    end
end

% relative jump
if iter_count == 0
    rel_jump = nan;
else
    rel_jump = abs(c-prev_c)/abs(c);
end

% update table and print iteration's result
table(iter_count+1, :) = [iter_count, a, b, c, rel_jump];
fprintf('%d \t %0.6f \t %0.6f \t %0.6f \t %0.6e \n', iter_count, a, b, c, rel_jump)


% stopping criterion
if rel_jump < ErrorTol
    break
end

    iter_count = iter_count + 1;
end

%% Display Results

fprintf('Root is %.10f after %i iterations. \n', c, iter_count)
fprintf('Note that the function value at the estimate is %.6E. \n', f_c)
```