In the Name of the God the Compassionate and the Merciful

Associated Prof.: Reza Faraji Dana

Mohammad Reza Arani

810100511

Hw8 - Computational Electro Magnetics

University of Tehran

1401/10/26

Table of Contents

Table of Contents

Problem-1:

Infinitesimal Dipole is located between 2 PEC sheets at:

$$(x', y', z')$$

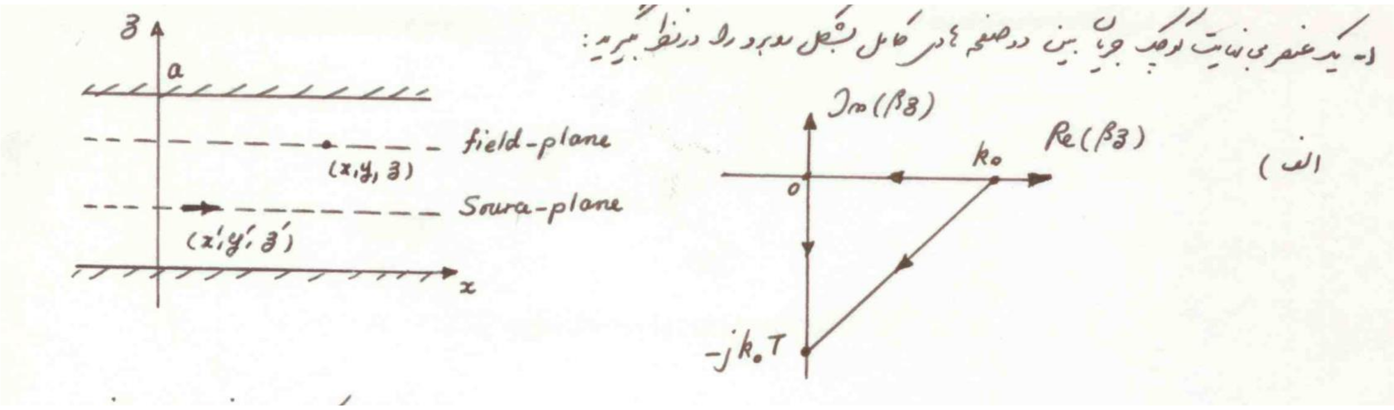


Figure 1

As the problem induces, it is to be solved using complex images method first implemented at waterloo university.

- In spatial domain, the single point source is equal to a plane source in spectral domain.

  <Due to Sommerfeld's identity>
- As "T" value grows larger, our approximation gets valid for Near-field approximation of the problem.
- As "a" value shrinks and get closer to 0, the radiation from the source point will be suppressed by its images and the answer tends to 0.
- Transform from spectral domain to spatial domain is undertaken using Prony's series approximation and then using Weyl Identity augmented with Grover's R_{eff} formula.

Starting the solution of Complex Images Method:

- Consider below given spectral function which is obtained from ODE solution of Wave Equation in spectral domain in this geometry:

$$\tilde{g} = \frac{1}{2j\beta_z}\left(e^{-j\beta_z(z-z')} - e^{-j\beta_z(z+z')} + \frac{e^{-j\beta_z\left(2a+z-z'\right)} - e^{-j\beta\left(2a+z+z'\right)} + e^{-j\beta_z\left(2a-z+z'\right)} - e^{-j\beta_z\left(2a-z-z'\right)}}{1-e^{-j\beta_z(2a)}}\right)$$

By extracting $e^{-j\beta_z 2a}$ from the nominator we can then consider below function to approximate:

$$\tilde{F}(\beta_z) = \frac{e^{-j\beta_z 2a}}{1-e^{-j\beta_z 2a}} \simeq \sum_{n=1}^{N} a_m e^{\beta_z b_m}\ ;\ a_m,\ b_m \in \mathbb{C}$$

This function behaves as:

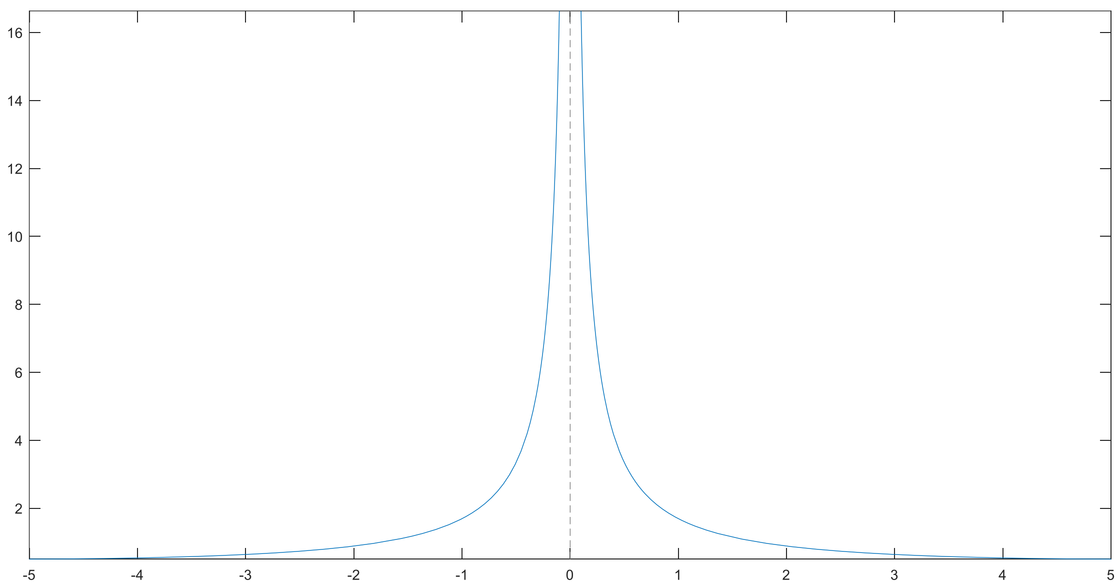


Figure 2

X-Axis is β_z and the Y-Axis is the $\tilde{F}(\beta_z)$

Then we have:

$$\tilde{g} = \frac{1}{2j\beta_z}\left(e^{-j\beta_z(z-z')} - e^{-j\beta_z(z+z')} + e^{-j\beta_z(2a)}\,\frac{e^{-j\beta_z\left(z-z'\right)} - e^{-j\beta\left(z+z'\right)} + e^{-j\beta_z\left(-z+z'\right)} - e^{-j\beta_z\left(-z-z'\right)}}{1-e^{-j\beta_z(2a)}}\right)$$

Or

$$\tilde{g} = \frac{1}{2j\beta_z}\left(e^{-j\beta_z(z-z')} - e^{-j\beta_z(z+z')} + \tilde{F}(\beta_z)\left[e^{-j\beta_z(z-z')} - e^{-j\beta_z(z+z')} + e^{-j\beta_z(-z+z')} - e^{-j\beta_z(-z-z')}\right]\right)$$

Using Weyl Identity

$$g\left(\vec{r}, \vec{r}'\right) = \frac{e^{-jkR_0}}{4\pi R_0} - \frac{e^{-jkR_1}}{4\pi R_1} + \sum_{m=1}^{N} a_m \sum_{l=1}^{3} (-1)^l \frac{e^{-jkR_{ml}}}{4\pi R_{ml}}$$

$$R_{ml} = \sqrt[2]{\rho^2 + (Z_l + jb_m)^2};$$

Where Z_l equals to:

$$Z_l = \begin{cases} l = 0 & |z - z'| \\ l = 1 & z + z' \\ l = 2 & |-z + z'| \\ l = 3 & -z - z' \end{cases}$$

- Applying Grover's method, we obtain:

$$\psi(m, n) = \frac{1}{\Delta\ell_m \Delta\ell_n} \int_{\Delta\ell_m} \int_{\Delta\ell_n} g\left(\vec{r}, \vec{r'}\right) d\ell d\ell' =$$

$$\simeq \frac{e^{-jkR_0^{eff}}}{4\pi R_0^{eff}} - \frac{e^{-jkR_1^{eff}}}{4\pi R_1^{eff}} + \sum_{m=1}^{N} a_m \sum_{l=0}^{3} \frac{e^{-jkR_{ml}^{eff}}}{4\pi R_{ml}^{eff}}$$

In which R^{eff} may be calculated using below equation:

$$R^{eff} = \frac{\Delta\ell_m \Delta\ell_n}{M}$$

And M can be reached using:

$$M = \alpha \sinh^{-1}\left(\frac{\alpha}{d}\right) - \beta \sinh^{-1}\left(\frac{\beta}{d}\right) - \gamma +$$
$$\sinh^{-1}\left(\frac{\gamma}{d}\right) + \delta \sinh^{-1}\left(\frac{\delta}{d}\right) - \sqrt[2]{\alpha^2 + d^2} + \sqrt[2]{\beta^2 + d^2} +$$
$$\sqrt[2]{\gamma^2 + d^2} - \sqrt[2]{\delta^2 + d^2}$$

$$\gamma = \Delta\ell_m + \delta$$

$$\beta = \Delta\ell_n + \delta$$

$$\alpha = \Delta\ell_n + \delta + \Delta\ell_m$$

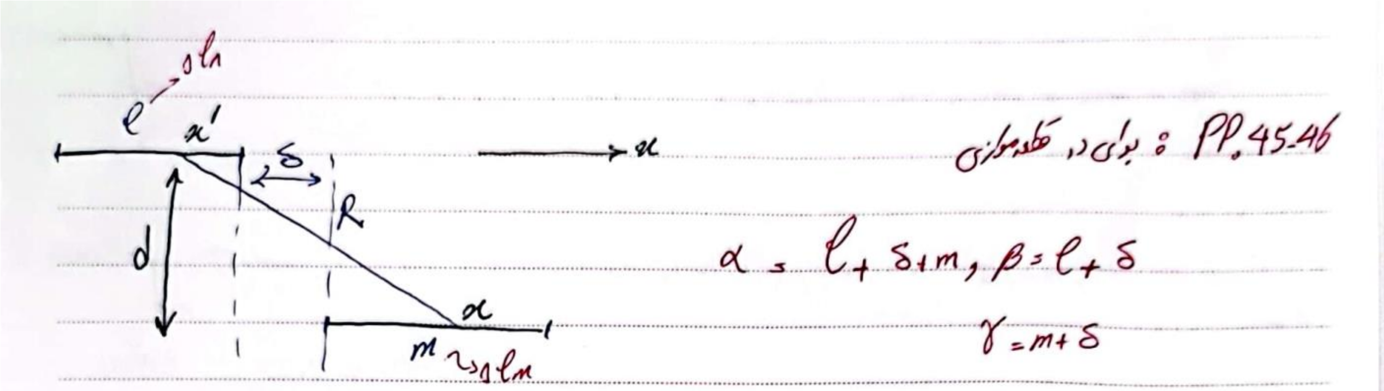


Figure 3

Prony's Method Tests:

Prony's method as utilized above, is implemented in MATLAB as:

- First, Equation system is achieved using formulas obtained in the course.

$$\underbrace{\begin{bmatrix} f_{N-1} & f_{N-2} & \cdots & f_0 \\ f_N & f_{N-1} & \cdots & f_1 \\ \vdots & \vdots & \ddots & \vdots \\ f_{2N-2} & f_{2N-3} & \cdots & f_{N-1} \end{bmatrix}}_{N\times N} \cdot \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}}_{N\times 1} = \underbrace{\begin{bmatrix} -f_N \\ -f_{N+1} \\ \vdots \\ -f_{2N-1} \end{bmatrix}}_{N\times 1}$$

این معادله ضرایب $\{\alpha_1, \ldots, \alpha_N\}$ را خواهد داد:

Figure 4

```
Coeff_Alpha = zeros(M,M);

for p=1:M

    Coeff_Alpha(p,:) = flip(F_B_z(1+p-1:M+p-1)) ;

end
Sampled_Points = -conj(F_B_z(M+1:2*M))';
Alpha          = inv(Coeff_Alpha)*  Sampled_Points   ;
```

- Now by having these Coefficients we have to calculate the roots of the polynomial with this form:

$$\mu^{N} + \alpha_1 \mu^{N-1} + \alpha_2 \cdot \mu^{N-2} + \cdots + \alpha_{N-1} \mu + \alpha_N = 0 \Rightarrow$$

$$(\mu - \mu_1)(\mu - \mu_2) \cdots (\mu - \mu_N) = 0 \Rightarrow$$ ریشه‌ها هستند

Figure 5

Which can be done using MATLAB built-in function “roots”.

```
Mu = roots([1 ; Alpha]);
B_m  = log(Mu);
```

By obtaining the μ values, we have paved the path until calculation of α values.

This can be done by forming another system of equations which is depicted below:

$$\underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mu_1 & \mu_2 & \cdots & \mu_N \\ \mu_1^2 & \mu_2^2 & \cdots & \mu_N^2 \\ \vdots & \vdots & & \vdots \\ \mu_1^{2N-1} & \mu_2^{2N-1} & \cdots & \mu_N^{2N-1} \end{bmatrix}}_{2N \times N} \cdot \underbrace{\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{bmatrix}}_{N \times 1} = \underbrace{\begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(2N-1) \end{bmatrix}}_{2N \times 1} \Rightarrow$$

$$\underline{\underline{A}} \cdot \underline{x} = \underline{b}$$

$$\underbrace{\underline{x}_{LS}}_{\text{Least Square}} = \underbrace{P_{inv}}_{\text{Pseudo Inverse}}(\underline{\underline{A}}) \cdot \underline{b}$$

Figure 6

```
Coeff_A = zeros(M,M);

for p=1:M
    Coeff_A(p,:) = (Mu.^(p-1)).*ones(M,1) ;
end

Sampled_Points_2 = conj(F_B_z(1:M))';
A = inv(Coeff_A) * Sampled_Points_2;
```

Test Case 1:

From the class notes we have:

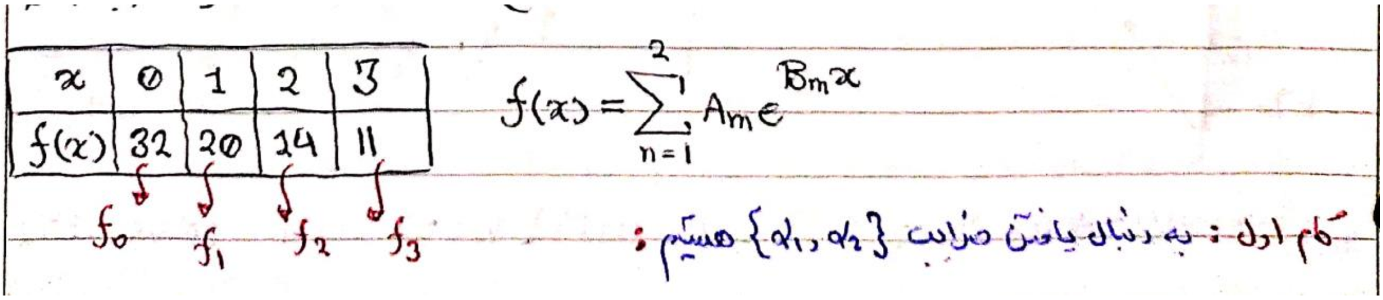


Figure 7

```
>> Out_object.Mu

ans =

   1.000000000000000
   0.500000000000001

>> Out_object.Alpha

ans =

  -1.500000000000001
   0.500000000000001
```

Figure 8

Obtained values are the same as what it should be!

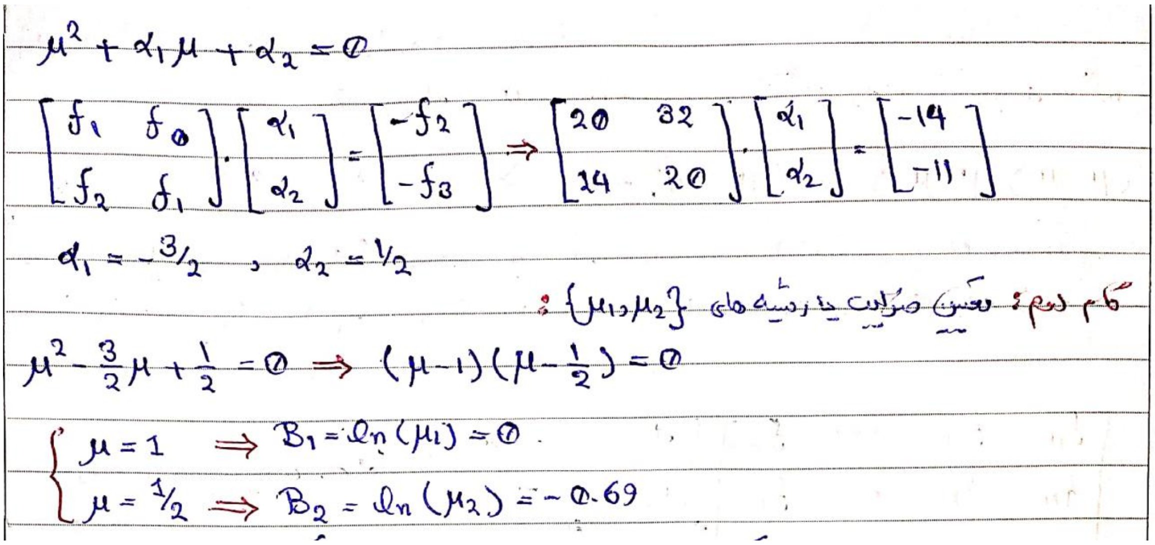


Figure 9

Also, b_m values are the same:

```
>> Out_object.B_m

ans =

                    0
  -0.693147180559944
```

Figure 10

After calculation of the roots and the corresponding b_m values, we have to get a_m values as below:

```
>> Out_object.A

ans =

   7.999999999999957
  24.000000000000043
```

Figure 11

- Also, the same as class notes!

$$\begin{bmatrix} 1 & 1 \\ \mu_1 & \mu_2 \end{bmatrix} \cdot \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 32 \\ 20 \end{bmatrix} \Rightarrow$$

$A_1 = 8$, $A_2 = 24$

تعمیم ، الآن باید نگران باشیم یا مشکلی نیست؟

$$f(x) = 8(1)^x + 24\left(\frac{1}{2}\right)^x = 8 + 24e^{-0.69x}$$

اگر ریشه های μ منفی شوند ، B ها مختلط خواهند شد.

Figure 12

Comparison of estimated function and the original function:

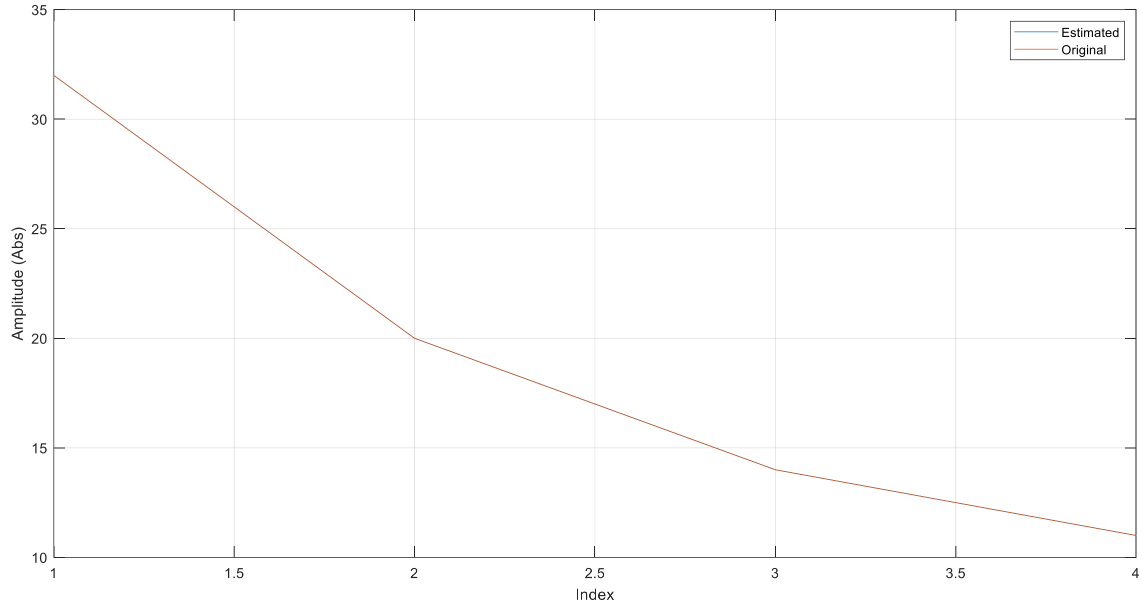


Figure 13

Both plots overlap each other and are exactly the same.

Test Case 2:

- A more complex scenario is implemented below:

X	0.5	2+1.5j	2.5	2+3.5j
F(X)	-0.500000000000000 – 1.95815868232297i	-0.884546873465022 – 0.164218689894810i	-0.500000000000000 – 0.166136708627264i	-0.986863706853306 – 0.0267614030201496i

Where F(X) is the function:

$$F(X) = \frac{e^{-jX}}{1 - e^{-jX}}$$

- Obtained roots and their corresponding coefficients:

μ	+ 1.11166334232769 - 0.702764626824133i	- 0.0980814568368275 - 0.422256022318346i
b_m	+ 0.273966333323648 - 0.563741499013032i	- 0.835869148542786 – 1.79902869470424i

a_m	+ 0.0194726171218779 - 0.464514103360527i	- 0.519472617121878 – 1.49364457896244i

```
>> Out_object.Mu

ans =

  1.111663342327695 - 0.702764626824133i
 -0.098081456836827 - 0.422256022318346i

>> Out_object.B_m

ans =

  0.273966333323648 - 0.563741499013032i
 -0.835869148542786 - 1.799028694704241i
```

Figure 14

- The final output looks like:

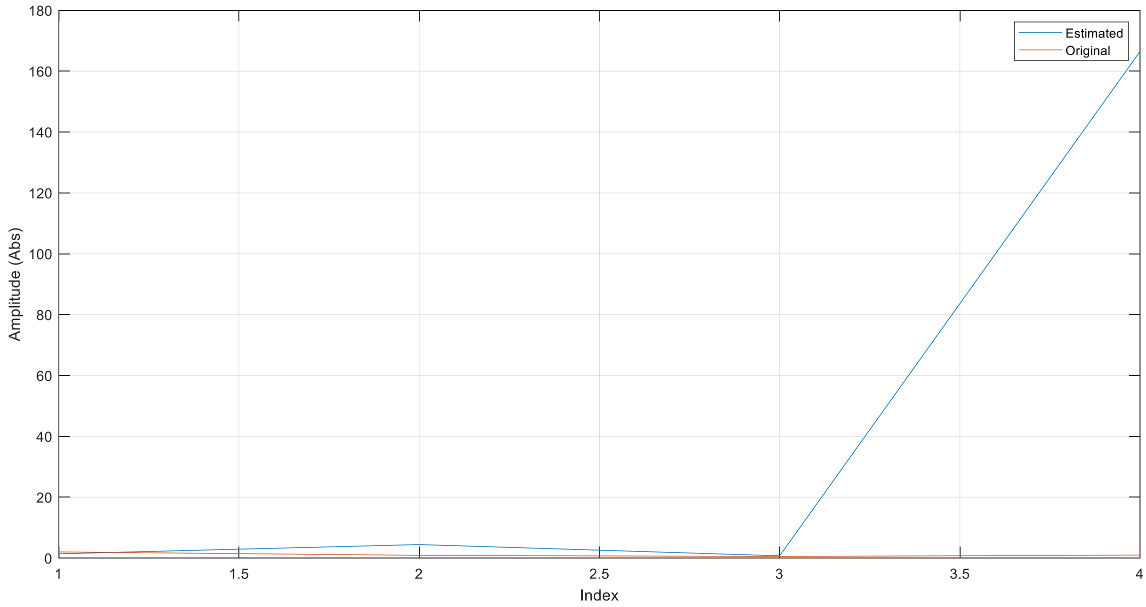


Figure 15

- Obtained Coefficients Matrix for acquiring a_m has been brought in Table:

1.00000000000000 + 0.00000000000000i	1.00000000000000 + 0.00000000000000i
1.11166334232769 - 0.702764626824133i	-0.0980814568368275 - 0.422256022318346i

Which is the same as:

$$\begin{bmatrix} 1 & 1 \\ \mu_1 & \mu_2 \end{bmatrix}.$$

Figure 16

For illustration, another function is estimated:

$$F(X) = \frac{e^{-X}}{1 - e^{-X}}$$

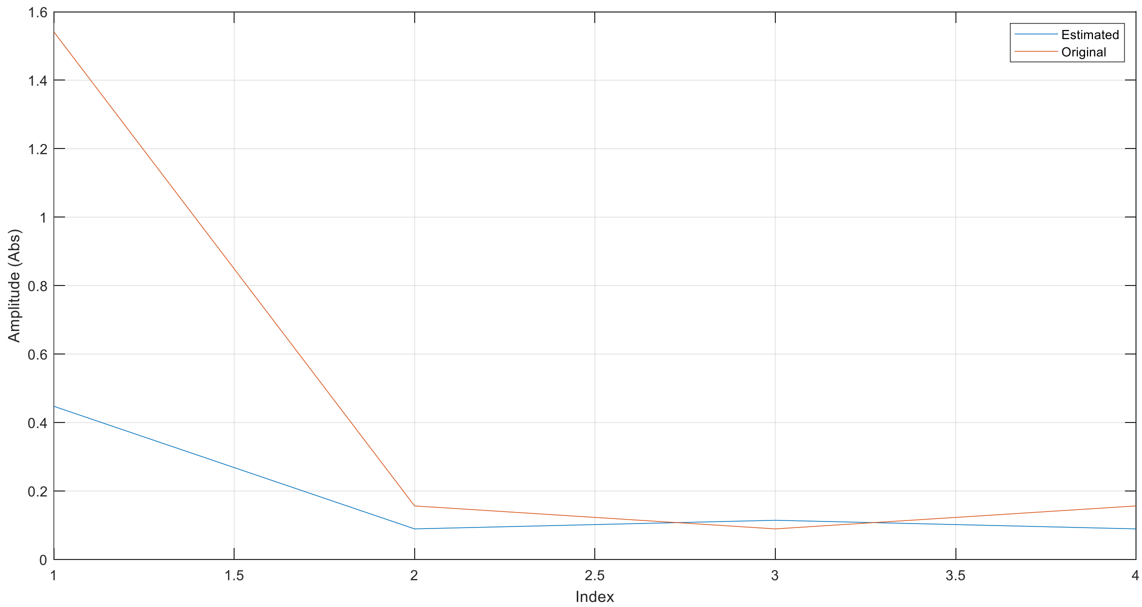


Figure 17

An extended version for more points for the same function:

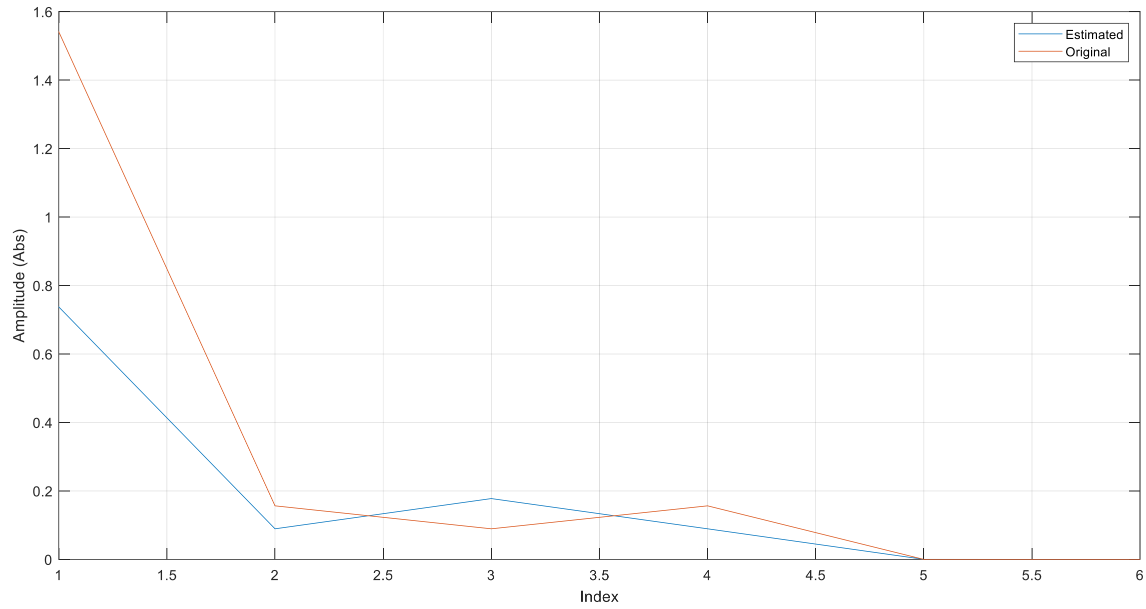


Figure 18

- Corresponding figures are depicted in the pdf file attached.

Part B:

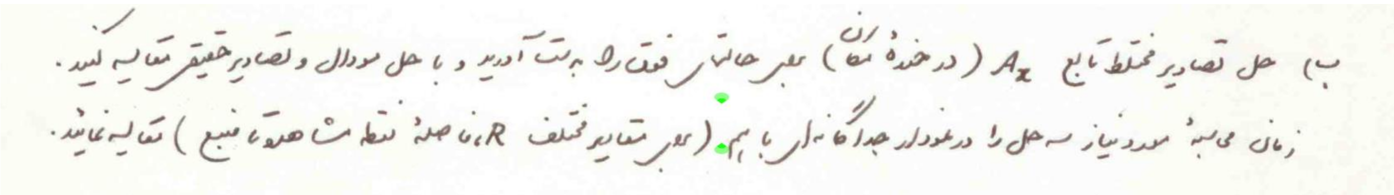

ب) حل تصاویر مختلط تابع A_x (در حوزه مکان) برای حالتهای فوق را بدست آورید و با حل مودال و تصاویر حقیقی مقایسه کنید.
زمان محاسبه مورد نیاز سه حل را در نمودار جداگانه‌ای با هم (برحسب مقادیر مختلف R، فاصله نقطه مشاهده تا منبع) مقایسه نمائید.

Figure 19

Real Images Solution:

Starting from spectral domain form of green function below:

$$\tilde{g}\Big|_{z>z'} = \frac{\sin(\beta_z z')\sin(\beta_z(a-z))}{\beta_z \cdot \sin(\beta_z a)} = \frac{\left(e^{j\beta_z z'} - e^{-j\beta_z z'}\right)\left(e^{j\beta_z(a-z)} - e^{-j\beta_z(a-z)}\right)}{2j\beta_z\left(e^{j\beta_z a} - e^{-j\beta_z a}\right)}$$

Figure 20

Which eventually leads to:

$$= \frac{e^{-j\beta_z(z-z')} - e^{-j\beta_z(z+z')} - e^{-j\beta_z(2a-z-z')} + e^{-j\beta_z(2a-z+z')}}{2j\beta_z\left(1 - e^{-j2\beta_z a}\right)}$$

Figure 21

Which by using Maclaurin expansion below:

$$\frac{1}{1-z} = 1 + z + z^2 + \cdots = \sum_{n=0}^{\infty} z^n \quad ; \quad |z| < 1$$

Figure 22

We have:

$$\frac{1}{1-e^{-2j\beta_z a}} = \sum_{n=0}^{\infty} e^{-j\beta_z 2na}$$

$$\tilde{g}(k_x, k_y, z|z') = \sum_{n=0}^{\infty} e^{-j2\beta_z na} \cdot \frac{\overbrace{e^{-j\beta_z(z-z')} - e^{-j\beta_z(z+z')} + e^{-j\beta_z(2a-z+z')} - e^{-j\beta_z(2a-z-z')}}^{\text{W yl Identity}}}{2j\beta_z}$$

Figure 23

And then by using Weyl Identity we get:

$$g(\vec{r},\vec{r}') = \sum_{n=0}^{\infty} \sum_{l=0}^{3} (-1)^{l} \frac{e^{-jkR_l}}{4\pi R_l} \quad ; \quad R_l = \sqrt{(x-x')^2+(y-y')^2+(z_l+2na)^2}$$

$$z_l = \begin{cases} z-z' & ; l=0 \\ z+z' & ; l=1 \\ 2a-z+z' & ; l=2 \\ 2a-z-z' & ; l=3 \end{cases} \xrightarrow[\text{بیان حوزهٔ مکان}]{(z<z') \quad z \to z'} z_l = \begin{cases} |z-z'| & ; l=0 \\ z+z' & ; l=1 \\ 2a-|z-z'| & ; l=2 \\ 2a-(z+z') & ; l=3 \end{cases}$$

Figure 24

MATLAB Code:

```
%% Init:
c  = 3e+08;
fc = c ;
Lambda = fc/c;
K = 2*pi/Lambda;
T = 1;


a = Lambda/5;

z = (0.001:a/100:10*a/10)';
y = a;
x = a;

z_p = 8*a/10;
x_p = 0;
y_p = 0;



%% Real Images Solution:

% Max_iter = 10;
g_real_images = zeros(length(z),Max_iter);

for n=1:Max_iter
    for l=0:3
        Z_L = [ abs(z-z_p) , z+z_p , 2*a-abs(z-z_p), 2*a-(z+z_p) ];
        R_nl =sqrt( (x-x_p)^2 + (y-y_p)^2 + (Z_L(:,l+1) + 2*n*a).^2     ) ;
        g_real_images(:,n) = g_real_images(:,n) + ((-1)^l )*   exp(-1j*K*R_nl)./(4*pi*R_nl) ;
    end
end
```

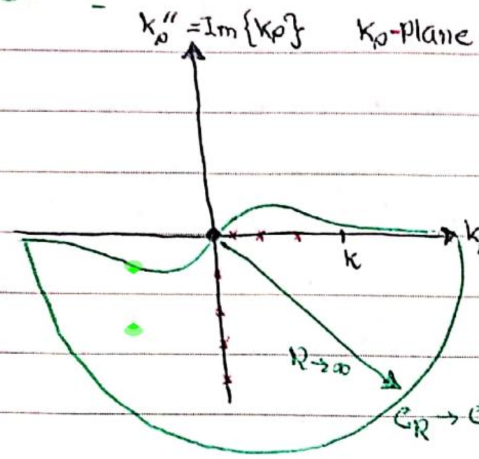
Modal Solution:

Again, we start from spectral domain solution:

$$g(x,y,z|x',y',z') = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{\sin(\beta_z z')\sin(\beta_z(a-z))}{\beta_z \cdot \sin(\beta_z a)} H_0^{(2)}(k_\rho \rho)\, k_\rho \, dk_\rho$$

Figure 25

And then using Hankel Transform to get to the spatial domain form and applying contour integration over a specific path we get:



بنابراین چنین مسیری را دنبال خواهیم کرد و در نتیجه

حاصل انتگرال چنین خواهد شد:

$$g = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{\sin(\beta_z z')\sin(\beta_z(a-z))}{\beta_z \cdot \sin(\beta_z a)} H_0^{(2)}(k_\rho \rho)\, k_\rho \, dk_\rho$$

$$= -j2\pi \sum_{n=1}^{\infty} \mathrm{Res}\Big|_{k_\rho = k_{\rho_n}} [F(k_\rho)]$$

Figure 26

Finally, Modal solution form becomes:

فرم مودال تابع گرین به صورت زیر خواهد شد:

$$g(\rho,z|\rho',z') = \sum_{n=1}^{\infty} \frac{\sin(\frac{n\pi}{a}z') \cdot \sin(\frac{n\pi}{a}z)}{j2a} H_0^{(2)}(k_{\rho_n}\rho) \;;\; k_{\rho_n} = \sqrt{k^2 - (\frac{n\pi}{a})^2}$$

Figure 27

- Hankel function in MATLAB can be called via:

Description

`H = besselh(nu,Z)` computes the Hankel function of the first kind $H_\nu^{(1)}(z) = J_\nu(z) + iY_\nu(z)$ for each element in array `Z`. example

`H = besselh(nu,K,Z)` computes the Hankel function of the first or second kind $H_\nu^{(K)}(z)$, where `K` is `1` or `2`, for each element of array `Z`. example

`H = besselh(nu,K,Z,scale)` specifies whether to scale the Hankel function to avoid overflow or loss of accuracy. If `scale` is `1`, then Hankel functions of the first kind $H_\nu^{(1)}(z)$ are scaled by e^{-iZ}, and Hankel functions of the second kind $H_\nu^{(2)}(z)$ are scaled by e^{+iZ}. example

Figure 28

MATLAB Code:

```
%% Modal Solution:


Max_iter = 100;
g_Modal = zeros(length(z),Max_iter);


for n=1:Max_iter
    k_rho_n = sqrt( K^2-(n*pi/a)^2 ) ;
    Rho =  sqrt( (x-x_p)^2 + (y-y_p)^2 )  ;
    H_02 = besselh(0,2,k_rho_n * Rho);
    g_Modal(:,n) =  g_Modal(:,n) + H_02*sin(n*pi/a*z_p)*sin(n*pi/a*z)/(1j*2*a);

end
```

Complex Images Solution:

Comparison:

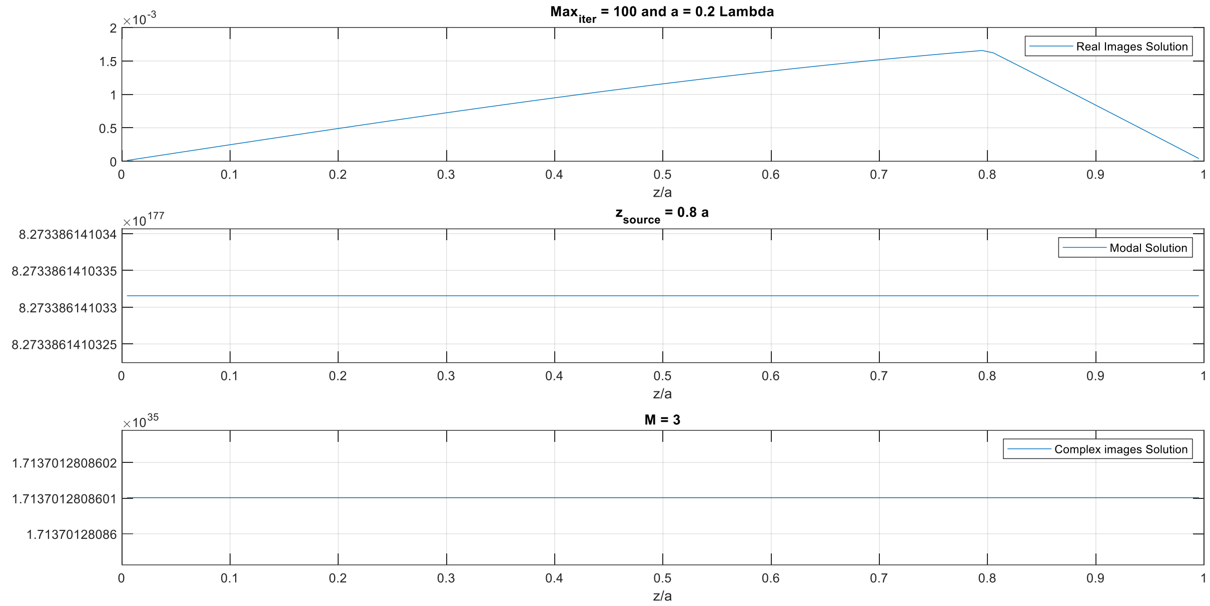


Figure 29

And with fewer iterations we have:

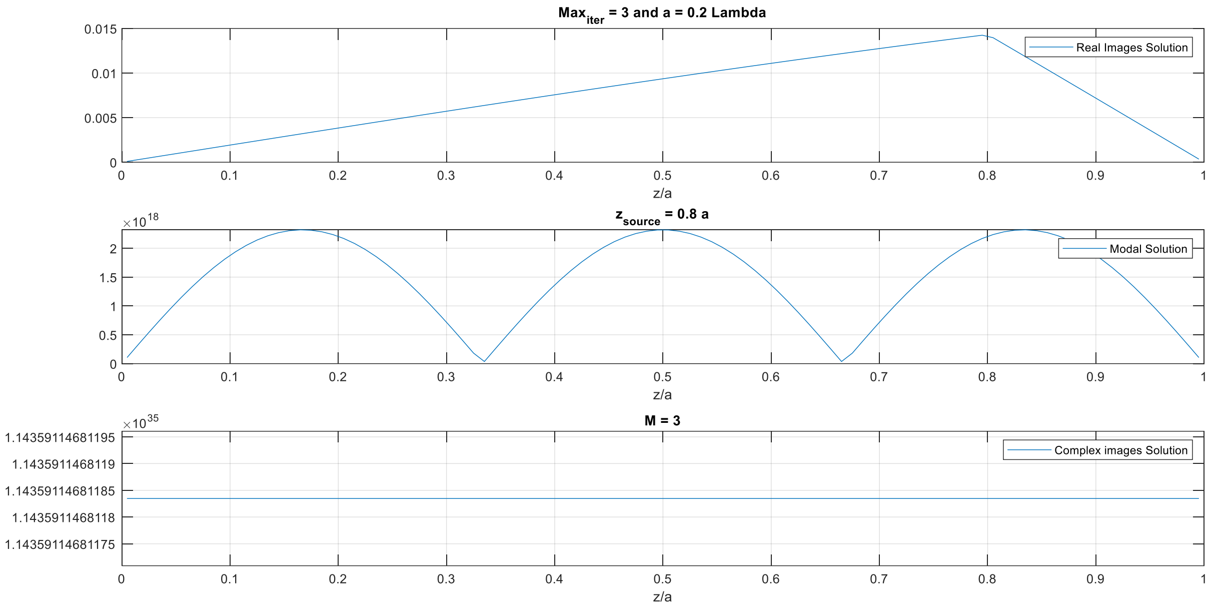


Figure 30

Time Analysis:

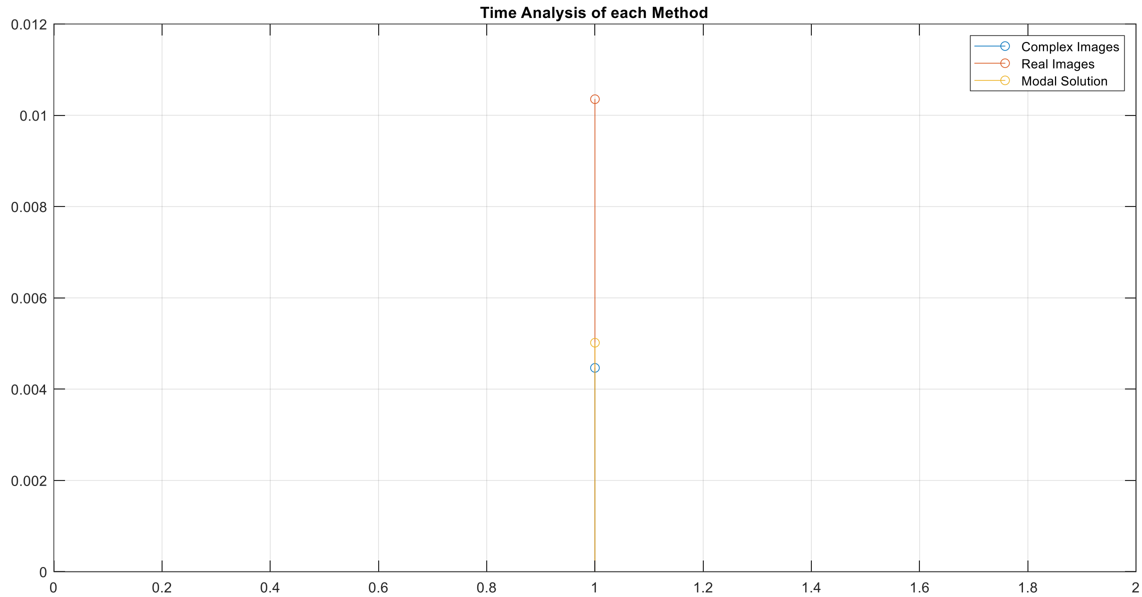
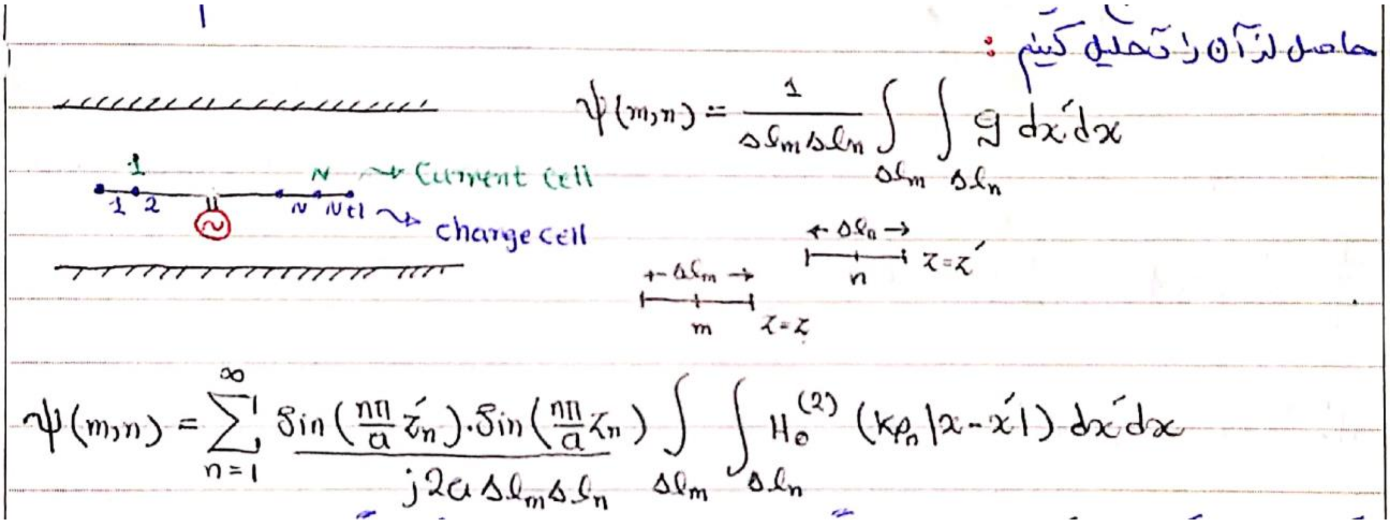


Figure 31

Part C:

To obtain new values for this ψ function, we have to replace the formula below:



And then use Numerical integration to its values:

Even

$$\psi(m,n) = \sum_{n=1}^{\infty} \frac{\sin\left(\frac{n\pi}{a} z_n'\right) \sin\left(\frac{n\pi}{a} z_n\right)}{j2a} \left\{ \sum_{i=1}^{M_m} \sum_{j=1}^{M_n} H_i H_j H_0^{(2)}\left(k\rho_n |x_i - x_j|\right) \right\}$$

← Gauss-Legendre's Quadrature Formula →

Figure 32

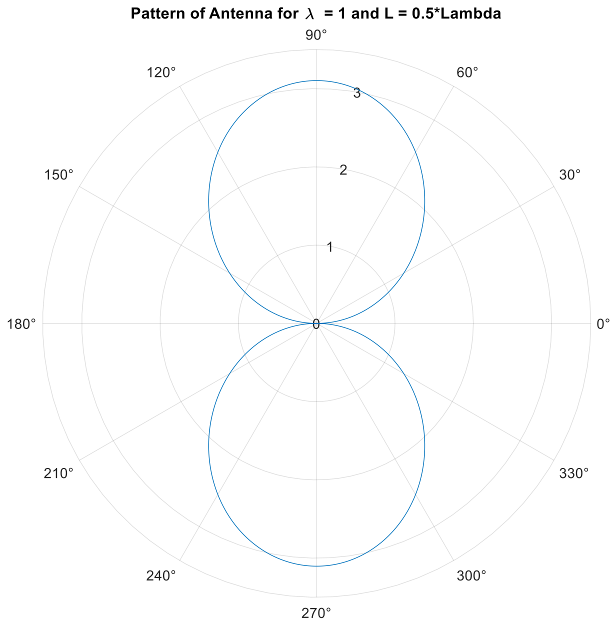


Figure 33

Input Impedance:

0.0006 + 0.1029i

Problem-2:

The problem is defined as below:

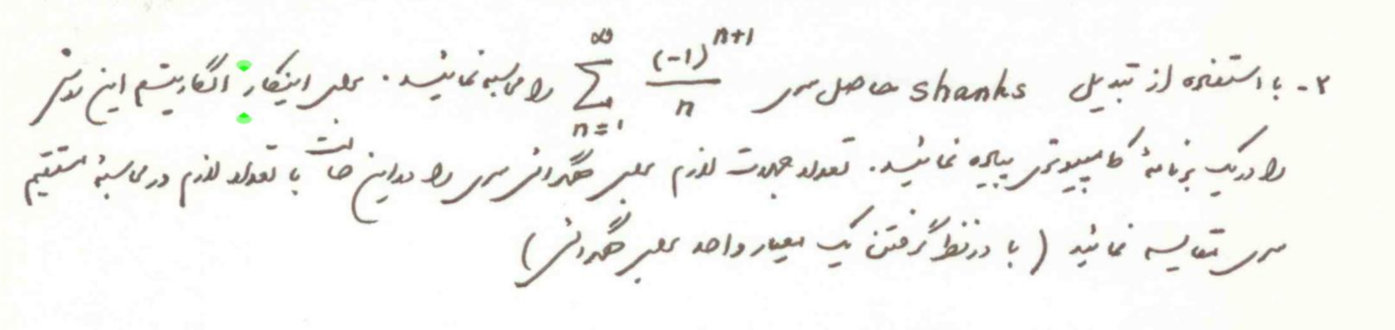

۲- با استفاده از تبدیل shanks حاصل سری $\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}$ را محاسبه نمائید. برای اینکار، الگوریتم این روش را در یک برنامه کامپیوتری پیاده نمائید. تعداد جملات لازم جهت همگرائی سری را در دامنه خاص با تعداد لازم در محاسبه مستقیم سری مقایسه نمائید (با درنظر گرفتن یک معیار واحد جهت همگرائی)

Figure 34

Convergence is ensured in this series so that last 2 consecutive term get close enough that their difference gets lower than the given threshold.

For a proper convergence, threshold is considered to be 10^{-3};

Shanks's Transform formulation implementation in MATLAB comes below:

```
function S_T1 = perform_Shanks(S_n)
      M = length(S_n);


      S_T1 = zeros(1,M-1-2+1);
      for i=1:M-2
          A = S_n(i) ;
          B = S_n(i+1);
          C = S_n(i+2);
          S = Shanks_Trans_nodes(A,B,C);
          S_T1(i) = S ;


      end
end


function  S = Shanks_Trans_nodes(A,B,C)
S = (C*A-B^2)/(C+A-2*B) ;
end
```

After finding the right number of terms to start from, the below table is achieved:

0									
1	…	0.69314012 7374205							
0.50000000 0000000	…	0.69314886 9333182	0.69314711 9737629						
0.83333333 3333333	…	0.69314668 1971590	0.69314719 6108489	0.69314717 9506449					
0.58333333 3333333	…	0.69314735 4031818	0.69314717 6120171	0.69314717 7877719	0.69314718 6043765				
0.78333333 3333333	…	0.69314711 1911928	0.69314718 1988220	0.69314718 7773986	0.69314718 3766039	0.69314718 7455125			
0.61666666 6666667	…	0.69314721 0655419	0.69314717 9616545	0.69314717 0801418	0.69314717 4291965	0.69314717 1031682	0.69314717 7677940		
0.75952380 9523810	…	0.69314716 6238412	0.69314718 0970217	0.69314717 6203286	0.69314717 1181333	0.69314718 2694033	0.69314718 0819805	0.69314719 7854007	
0.63452380 9523810	…	0.69314718 7892628	0.69314717 9603046	0.69314714 6411038	0.69314715 9085680	0.69314717 2323245	0.69314718 4389785	0.69314718 3370418	0.69314729 1985178
0.74563492 0634921	…	0.69314717 6565422	0.69314718 0810286	0.69314716 8228574	0.69314703 3274481	0.69314710 1079378	0.69314714 4181797	0.69314716 6770018	
0.64563492 0634921	…	0.69314718 2765315	0.69314717 5736741	0.69314719 3467518	0.69314718 1514178	0.69314720 9548667	0.69314719 4193413		
0.73654401 1544012	…	0.69314717 9326246	0.69314716 8094248	0.69314717 6658810	0.69314720 5652950	0.69314719 0835223			
0.65321067 8210678	…	0.69314718 1431651	0.69314719 0988102	0.69314714 2022397	0.69314716 5730256	0.69314718 7455125			
0.73013375 5133755	…	0.69314718 0247842	0.69314723 4477407	0.69314721 9318417					
0.65870518 3705184	…	0.69314718 1001345	0.69314721 2628626						
0.72537185 0371851	…	0.69314718 0604408							
0.66287185 0371851	…								
0.72169537 9783615	…								
0.66613982 4228060	…								
0.71877140 3175428	…								
0.66877140 3175428	…								
0	…								
1	…								
0.50000000 0000000	…								
0.83333333 3333333	…								
0.58333333 3333333	…								
0.78333333 3333333	…								
0.61666666 6666667	…								
0.75952380 9523810	…								
0.63452380 9523810	…								
0.74563492 0634921	…								
0.64563492 0634921	…								

0.736544011544012	…								
0.653210678210678	…								
0.730133755133755									

We can see that digits are solid for final columns:

0.693147291985178 VS 0.693147166770018

And

0.693147166770018 VS 0.693147194193413

The error is less than threshold for both of these terms! =>

The convergence criterion has been met!

- Maximum Number of iterations is found so that we meet convergence criterion and if it has not been met, 2 more terms will be added to the initial calculated series.

- Initial number of series to be calculated is considered an odd number so that the final column contains only 1 term.

- Convergence happens for:

    - 6^{th} iteration of the algorithm.
    - Starting from 7 terms, we end up summing up 7 terms.
    - Error_1 = $0.00068151393452775 5$; Error_2 = $3.437101543557475e-05$ where Error_1 is the difference between final term of third column and second column and Error_2 is of that first and second column correspondingly.

The original series met this criterion after calculation of 1002 number of Terms!

This shows 1002-7 = 995 improvement!

The Exact solution of this series is:

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} = \log(2) \approx 0.69315$$

$\log(x)$ is the natural logarithm

Figure 35

والسلام على من اتبع الهدى