W6

Name: Mohammed AL Shuaili

Date: 17/2/2025

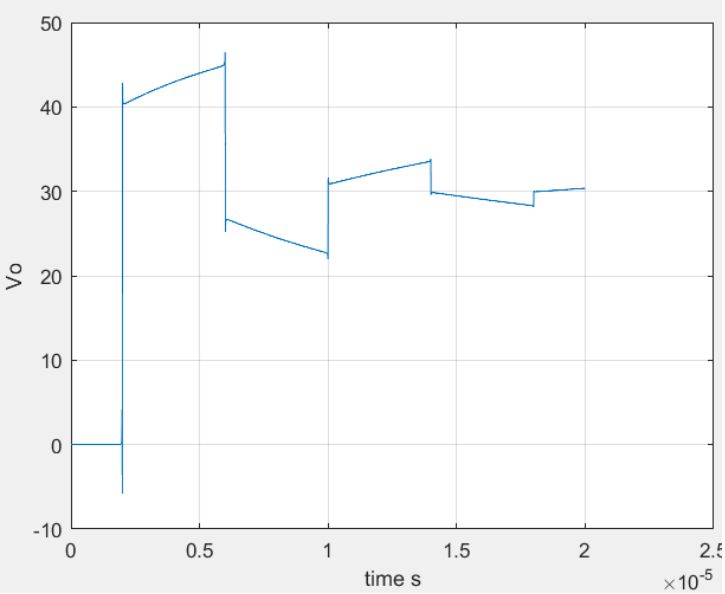
Consider the exact solution that is:

$$Y(s) = \frac{1}{\cosh\left(l\sqrt{(R+sC)(G+sC)}\right)}$$

Consider the following values for the impedance and an input of 30 volts:

$$R = 0.1\,, C = 10^{-10}, L = 2.5\,10^{-7}, l = 400, G = 0, v_{in} = 30$$

The unit step response looks as follows:



we can generate Y parameter models out of this, consider 100 points for $f = 9000$ until 9e5. And 100 points for the output TL.

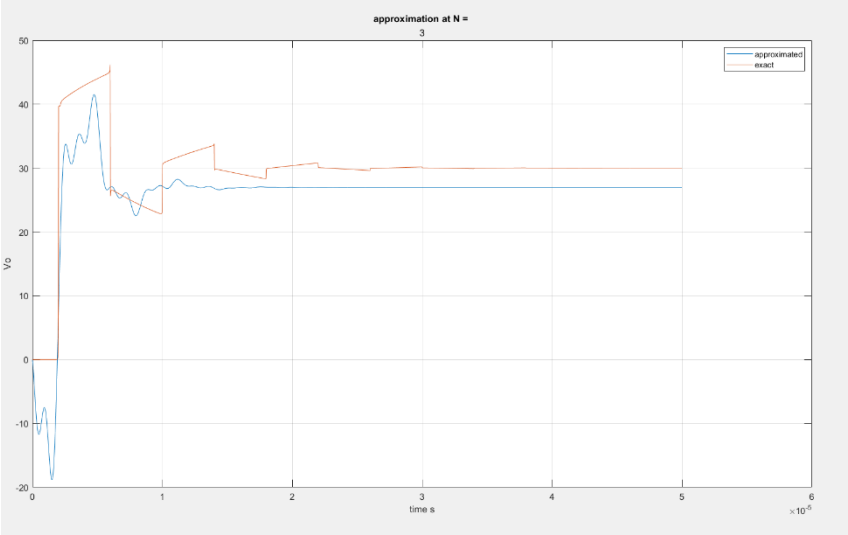
To simplify this, consider 2 models of Y parameters and follow the same steps for as many Y models.

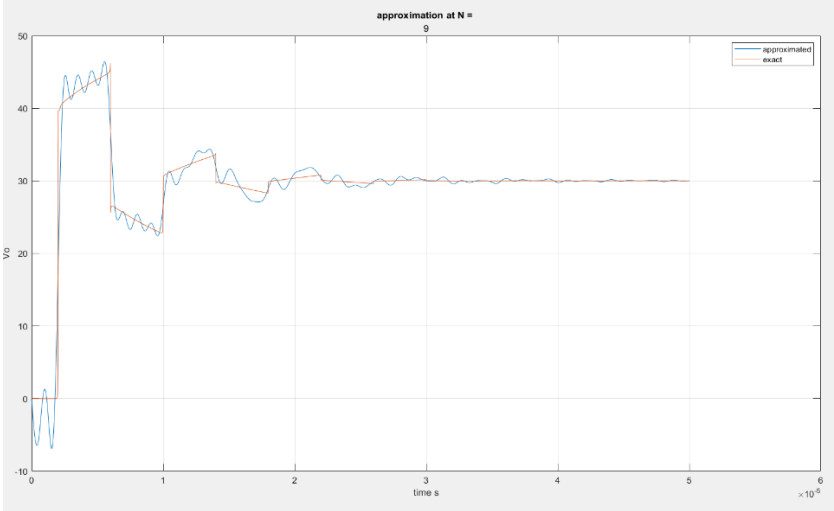
1. $Y_1 = generated\ from\ the\ first\ 50\ pints.$
2. $evaluate\ Y_1 at\ the\ last\ 50\ points\ (w_1).$
3. $subtract\ the\ obtain\ values\ from\ the\ exact\ given\ value\ at\ (w_1)\ (i.e, Y_1(w_1) - exact(w_1).$
4. $use\ the\ results\ to\ obtain\ Y_2\ ans\ so\ on.$

The following code will implement this.

```
clear
clc
f = 9e5;
f = linspace(0,f,100+1);
w = 2*pi*f(2:end);
s = i *w;
vo =1./(cosh(400.*(0 + 1e-10.*s).^(1/2).*(0.1 + 2.5e-7.*s).^(1/2)));
N=1; % number of section basicly 100/N
H_total = @(s)0;
section_size = ceil(length(vo)/N);  % Points per section (except last)
for i = 1:N
    start_idx = (i-1)*section_size + 1;
    end_idx = min(i*section_size, length(w));
    seg_idx = start_idx:end_idx;
    w_i = w(seg_idx);
    s_i = 1i * w_i;
    vo_i = vo(seg_idx);
    H_prev_eval = H_total(s_i);

    % Calculate residual response
    vo_residual = vo_i - H_prev_eval;

    % Fit new model to residual
    Hi = generate_yp2(real(vo_residual), imag(vo_residual), w_i);

    % Add to total transfer function
    H_total = @(s) H_total(s) + Hi(s);
end
H = @(s) H_total(s) * 30 ./ s;
v =@(s)30./(s.*cosh(400.*(0 + 1e-10.*s).^(1/2).*(0.1 + 2.5e-7.*s).^(1/2)));
[y,t]=niltcv(H,50e-6,'pt1');
[y1,t1]=niltcv(v,50e-6,'pt1');
RMSE = sqrt(sum((y-y1).^2)/length(y1));
plot(t,y,t1,y1)
grid on
xlabel('time s')
ylabel('Vo')
legend('approximated','exact');
```

N (number of Y models)	RMSE
1	25.5234
2	27.8445
3	4.9876
4	1.8179
5	8.3383
6	3.6791
7	3.1194
8	3.5229
9	1.6736
10	4.9612
11	Code error





We can see that the accuracy depends on the number of models used but not at all times as there are other things that could affect this. This was done using NILTcv,

- Implementation using AWE.

So in this section AWE is used to get the step response and H(s) where :

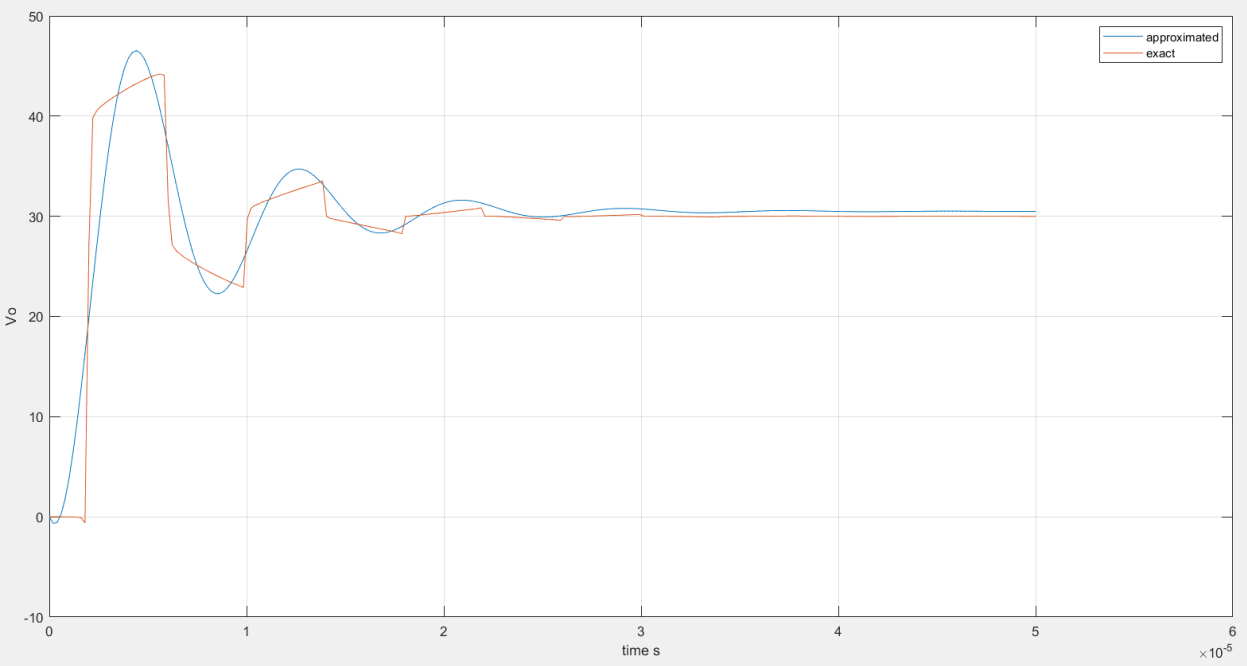
$$H(s) = \frac{residue}{p_1 - s} + \cdots ..$$

The frequency passed to AWE is the beginning of each model. The following code is an example on only using one model.

```
clear
clc
% generate 100 points.
f = 9e5;
f = linspace(0,f,100);
w = 2*pi*f;
s = i *w;
vo =1./(cosh(400.*(0 + 1e-10.*s).^(1/2).*(0.1 + 2.5e-7.*s).^(1/2)));
v =@(s)30./(s.*cosh(400.*(0 + 1e-10.*s).^(1/2).*(0.1 + 2.5e-7.*s).^(1/2)));
%H is as H = @(s) a1s+a0/s^2+b1s+b0;
[H,num,deno] = generate_yp2(real(vo(1:13)),imag(vo(1:24)),w(1:13));
[A,B,C,D] = create_state_space(num,deno);
%HAWE is Hs= @(s) resdue/s-pole + ...;30 is the inout, 50e-6 is t for plot
[h_impulse,HAWE, y, t] = AWE(A,B,C,D,w(1),30,50e-6);
[y1,t1]=niltcv(v,50e-6,'pt1');
RMSE = sqrt(sum((y-y1).^2)/length(y1));
plot(t,y,t1,y1);
grid on
xlabel('time s')
ylabel('Vo')
legend('approximated','exact');
abs(RMSE)
```

in this code, 100 points are generated for the exact solution and the first 13 (i.e, $f = 1.0909e + 05$) points were used to generate Y rational approximation using code 1. Then it's converted to state space mode using code 2 in the appendix.

AWE is used with w(1) (i.e, $w = 0$). Code 3. The following unit step response was obtained.

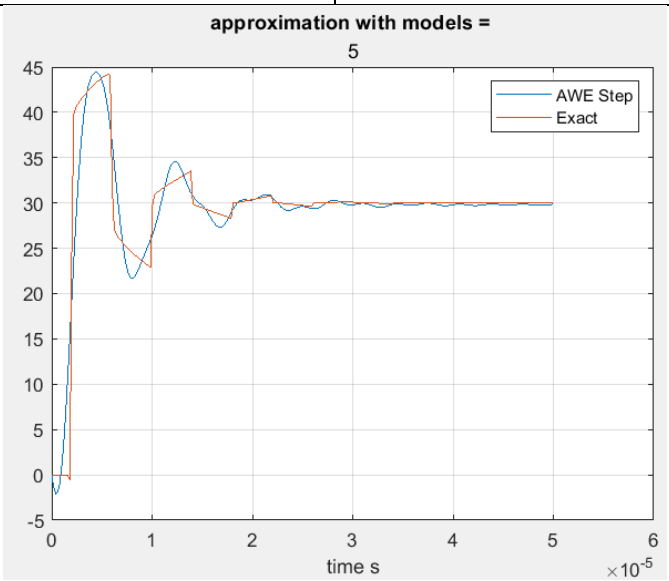


With RMSE = 2.6373.

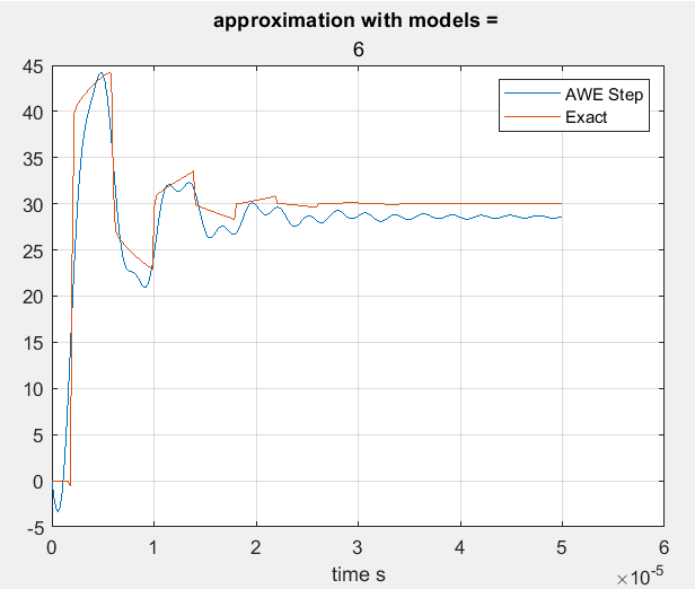
Now consider 2 Y rational approximations.

This when w = 0 for all sections, and 5 points for each section

N (number of Y models)	RMSE
1	2.6373
2	5.0380
3	2.6248
4	2.6426
5	2.3683
6	2.6128
7	6.2926



But,



Using this code:

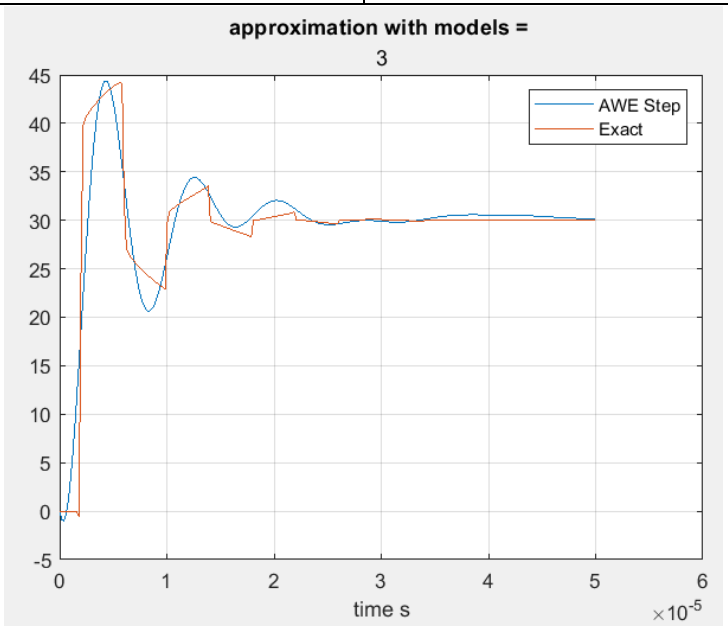
```
clear
clc
% generate 100 points.
f = 9e5;
f = linspace(0,f,100);
w = 2*pi*f;
s = i *w;
vo =1./(cosh(400.*(0 + 1e-10.*s).^(1/2).*(0.1 + 2.5e-7.*s).^(1/2)));
v =@(s)30./(s.*cosh(400.*(0 + 1e-10.*s).^(1/2).*(0.1 + 2.5e-7.*s).^(1/2)));
%H is as H = @(s) a1s+a0/s^2+b1s+b0;
[Hi,num,deno] = generate_yp2(real(vo(1:15)),imag(vo(1:15)),w(1:15));
[A,B,C,D] = create_state_space(num,deno);
%HAWE is Hs= @(s) resdue/s-pole + ...;30 is the inout, 50e-6 is t for plot
[h_impulse,HAWEi, y, t] = AWE2(A,B,C,D,w(1),30,50e-6);
models =1;
%N = ceil(70/models); %8
minr = 99;
for j=1:10 % this loop only to print results for this report
N =5; % number of points per section or model
range = 16; % starting point of the second mmodel
for i=1:models
    range = range(end):N + range(end); % raange of frequency and exact values
   % range
    H_diff = vo(range)-HAWEi(s(range));
    [Hj,numi,denoi] = generate_yp2(real(H_diff),imag(H_diff ),w(range));
    [A,B,C,D] = create_state_space(numi,denoi);
    [h_impulse,HAWEj, yi, ti] = AWE2(A,B,C,D,w(1),30,50e-6);
    HAWEi = @(s) HAWEi(s)+HAWEj(s);
    Hi = @(s) Hi(s)+Hj(s);
    y0=y+yi;
end
[y1,t1]=niltcv(v,50e-6,'pt1');
RMSE = sqrt(sum((y0-y1).^2)/length(y1));
%figure
%plot(t,y0,t1,y1)
abs(RMSE)
if (abs(RMSE) < minr)
minr = abs(RMSE);
nModel = models + 1;
end
%grid on
%xlabel('time s')
%legend('AWE Step', 'Exact');
%title('approximation with models = ',num2str(models+1));
models = models+1;
end
```

if w = the beginning point of each section as w(range(1));

N (number of Y models)	RMSE
1	2.6373
2	8.8963
3	4.1976e+05
4	5.0586e+09
5	3.7439e+30

If the code is changed to evaluate the points around Hi (generated as ypram) Best results with 3

N (number of Y models)	RMSE
1	2.6373
2	8.8963
3	1.2406
4	4.4665
5	6.0659



- Adjust AWE.

I tried to adjust AWE which I believe is the cause of the issue when increasing sections

Code 1:

```
function [H_impulse,num,deno]=generate_yp2(realV,imagV,wo)
Yr = realV;  % Real part of Y11
Yi = imagV; % Imaginary part of Y11
w = wo;
A = [];
C = [];
% Loop through each frequency point to construct A and C
A = [];
C = [];
% Loop through each frequency point to construct A and C
for k = 1:length(w)
    wk = w(k);
    Yr_k = Yr(k);
    Yi_k = Yi(k);
    % Construct rows for A and C
    A_row1 = [-1, Yr_k, 0, -wk*Yi_k]; % Real part
    A_row2 = [0, Yi_k, -wk, wk*Yr_k]; % Imaginary part

    % Append to A
    A = [A; A_row1; A_row2];
    % C
    C_row1 = wk^2 * Yr_k; % Real part
    C_row2 = wk^2 * Yi_k; % Imaginary part
    % Append to C
    C = [C; C_row1; C_row2];
end

% Solve for B = [a0; b0; a1; b1]
B = A \ C;
% get cof
a0 = B(1);
b0 = B(2);
a1 = B(3);
b1 = B(4);
num = [a1,a0];
deno = [1,b1,b0];
% generated H
H_impulse =@(s) (a1*s+a0)./(s.^2+b1*s+b0);
end
```

Code 2:

```
function [A,B,C,D]= create_state_space(nem,deno)
N = nem;
D = deno;
% Perform polynomial division to make it strictly proper
[Q, R] = deconv(N, D);
% check leading coefficient (assumed to be 1)
if D(1) ~= 1
    D=D/D(1);
    N = N/D(1);
    [Q, R] = deconv(N, D);
end
% Extract coefficients for state-space representation
g = D(2:end);  % Exclude leading coefficient ( g terms )
% f terms
if R(1) ==0
    f=R(2:end);
else
    f = R;
end
```

```
% Construct state-space matrices
n = length(g); % Order of system
A = [zeros(n-1,1), eye(n-1); -flip(g)];
B = [zeros(n-1,1); 1];
C = flip(f);
D = Q;
end
```

Code 3:

```
function [h_impulse,h_s, y_step, t] = AWE(A,B,C,D,w,input,time)
    t = linspace(0,time,250);
    q = length(B);
    num_moments = 2 * q;
    s0=1i*w;
    moments = zeros(1, num_moments);
    [r,c]=size(C);
    if r~=1
        C= C';
    end
    for k = 1:num_moments
        moments(k) = (-1)^(k-1) * C * (s0 * eye(size(A)) - A)^-(k) * B;
    end
    moments(1)=moments(1)+D;
    approx_order = length(B);

    % Construct the moment matrix
    moment_matrix = zeros(approx_order);
    Vector_c = -moments(approx_order+1:2*approx_order)';

    for i = 1:approx_order
        moment_matrix(i, :) = moments(i:i+approx_order-1);
    end

    % Solve for denominator coefficients
    b_matrix = moment_matrix^-1 * Vector_c;

    % Compute poles
    poles = roots([b_matrix', 1]); % I tried to add s0

    % Compute residues
    V = zeros(approx_order);
    for i = 1:approx_order
        for j = 1:approx_order
            V(i, j) = 1 / poles(j)^(i-1); % I tried to change this to s0-p ot s0+p
        end
    end

    A_diag = diag(1 ./ poles); % I tried to change this to s0-p ot s0+p
    r_moments = moments(1:approx_order);
    residues = -1 * (A_diag \ (V \ r_moments'));

    % Impulse response
    h_impulse = zeros(size(t));
    for i = 1:approx_order
        h_impulse = h_impulse + residues(i) * exp(poles(i) * t);
    end
    h_s =@(s) 0;
    for i = 1:length(poles)
        h_s =@(s) h_s(s) + residues(i)./((s-s0)-poles(i));
    end
```

```
    % Step response using recursive convolution
    y_step = zeros(size(t));
    y = zeros(length(poles), 1);

    for n = 2:length(t)
        dt = t(n) - t(n-1);
        exp_term = exp(poles * dt);
        for i = 1:length(poles)
            y(i) = residues(i) * (1 - exp_term(i))/(-poles(i)) * input + exp_term(i) * y(i);
        end
        y_step(n) = sum(y);
    end
end
```