AWE and Y parameters

Name: Mohammed AL Shuaili

Date: 4/2/2025

Objectives:

1. Implement Y parameters
2. Complex frequency hopping
3. Final FYP report

One can obtain the unit step response out of a state space model using recursive convolution based on the pole-residue representation of the transfer function as shown in the below code. This is to avoid explicit convolution which is computationally expensive.

clear all

clc

% Input state-space matrices

A = [-2 1 0 0; 1 -2 1 0; 0 1 -2 1; 0 0 1 -1];

B = [1; 0; 0; 0];

C = [1; 0; 0; 0];

t = 0:0.1:2;

% impulse response

%{

eat = @(t) expm(A.\*t);

y = @(t)mtimes(mtimes(transpose(C),eat(t)),B);

y\_values = arrayfun(@(t) y(t), t);

%}

% step response

sys = ss(A,B,transpose(C),0);

[y\_theory, t] = step(sys, t); % get the step input response using step

% Determine the order of the system

q = length(B);

% Compute moments

num\_moments = 2 \* q;

moments = zeros(1, num\_moments);

for i = 1:num\_moments

moments(i) = -transpose(C) \* (A^(-i)) \* B;

end

figure(1);

ii=1;

hold on

% Generalized approximation for all orders

for approx\_order = 1:q

fprintf('Case %d:\n', approx\_order);

% 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

% find b matrix (deno coef)

b\_matrix = inv(moment\_matrix)\*Vector\_c;

%find the ploes

poles = roots([transpose(b\_matrix) ,1]);

% determine residuses

% form the V matrix

V = zeros(approx\_order);

for i = 1:approx\_order

for j = 1:approx\_order

V(i, j) = 1/poles(j)^(i-1);

end

end

% form the A matrix

A\_diag = diag(1 ./ poles);

r\_moments = moments(1:approx\_order); % a helper matrix

% find the residuse

residues = -1\*inv(A\_diag)\* inv(V)\* transpose(r\_moments);

%form the impulse response

h =0;

for i = 1:approx\_order

h = h + residues(i) \* exp(poles(i) \* t);

end

% Recursive Convolution (Section V)

y\_awe = zeros(size(t)); % AWE response

% y variables for each pole

y = zeros(length(poles), 1);

% Recursive convolution loop

for n = 2:length(t)

dt = t(n) - t(n-1); % Time step

exp\_term = exp(poles \* dt); % Precompute exponentials

for i = 1:length(poles)

% Updat state using Eq. (15)

% y(i) = residues(i) \* (1 - exp\_term(i)) \* 1 + exp\_term(i)\*y(i);

y(i) = residues(i) \* (1 - exp\_term(i))/(-poles(i)) \* 1 + exp\_term(i)\*y(i);

end

y\_awe(n) = sum(y); % Total response

end

% Plot Results

subplot(2,2,ii)

ii=ii+1;

plot(t, y\_awe, 'b-'); hold on;

plot(t, y\_theory, 'ro--');

xlabel('Time (μs)');

ylabel('Response');

title(['Approximation of Order ', num2str(approx\_order)]);

legend('AWE (Recursive Convolution)', 'Theoretical step', 'Location', 'Best');

grid on;

% plot the output

%{

figure(approx\_order);

plot(t,h);

hold on

plot(t, y\_values,'ro');

xlabel('Time (\mus)');

ylabel('V Load (Volts)');

title(['Apprximation of order',num2str(approx\_order)]);

legend('Awe', 'Theory Impulse', 'Location', 'Best');

grid on

%}

end

A group of graphs with red and blue lines

Description automatically generated

The figure compares step response approximations using the AWE method (blue) with the theoretical response (red circle) for orders 1 to 4. Higher orders improve accuracy, closely matching the theoretical response.

* Y parameters:

Each is approximated with rational function:

Considering the open voltage case:

This can then be converted into a state space model as follows:

Then, converting this to time domain gives:

And,

So, let

This gives the A matrix as:

The B matrix:

The C matrix:

Let’s consider n = 2;

The state space model:

Where,

Then AWE can be implemented to get the impulse response and unit step response.

Now, Let’s consider the following example to code this in MATLAB, assume the rational approximation for :

So,

To make it strictly proper (numerator degree < denominator degree) we can say:

For a third-order system (*n*=3):

**Do we ignore the (-2s^3) or we only consider the right side of the proper one?**

Ignoring the term gives:

With considering the proper term we get:

Code:

depending on the above as obtaining the proper one will depend on the values od