Implement AWE

Name: Mohammed AL Shuaili

Date: 31/1/2025

Objectives:

1. Find a pattern in matrix A to relate it with the number of sections in the RLC ladder
2. Validate AWE using the theory method (as in
3. Find the unit step response by doing one step and then the unit step as in section 5
4. Code the unit step response
5. Look at the Y-parameters and try to implement it.
6. complex frequency hopping will be the novel feature.

AWE is used to find the impulse response as follows:

The following code implements AWE with up to q-th approximation where q is the order:

1. clear all

2. clc

3. % Input state-space matrices

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

5. B = [1; 0; 0; 0];

6. C = [1; 0; 0; 0];

7.

8. % Determine the order of the system

9. q = length(B);

10.

11. % Compute moments

12. num\_moments = 2 \* q;

13. moments = zeros(1, num\_moments);

14. for i = 1:num\_moments

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

16. end

17.

18. % Generalized approximation for all orders

19. for approx\_order = 1:q

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

21.

22. % Construct the moment matrix

23. moment\_matrix = zeros(approx\_order);

24. Vector\_c = -moments(approx\_order+1:2\*approx\_order)';

25.

26. for i = 1:approx\_order

27. moment\_matrix(i, :) = moments(i:i+approx\_order-1);

28. end

29. % find b matrix (deno coef)

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

31.

32. %find the ploes

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

34.

35. % determine residuses

36. % form the V matrix

37. V = zeros(approx\_order);

38. for i = 1:approx\_order

39. for j = 1:approx\_order

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

41. end

42. end

43. % form the A matrix

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

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

46. % find the residuse

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

48. %set a value for t

49. %t=0:10;

50. t = 0:0.1:5;

51. %form the impulse response

52. h =0;

53. for i = 1:approx\_order

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

55. end

56. % plot the output

57. figure(approx\_order);

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

59. plot(t,h);

60. xlabel('Time (\mus)');

61. ylabel('V Load (Volts)');

62. grid on

63. end

64.

* Now, generalise the code to implement AWE with any number of sections for the RLC ladder.

First, consider the open voltage RLC ladder for the transmission line with N=2 as follows:

A diagram of a circuit

Description automatically generated

From the circuit, we can say that:

(3)

Matrix A:

Matix B:

Matrix C:

Second , Let’s consider the following circuit (RLC ladder with N = 3 and open voltage) to find a pattern where we can link the number of sections to the state space model:

A diagram of a circuit

Description automatically generated

Let,

So, each section of the transmission line contributes two states to the state-space model:

1. For

1. For
2. For
3. For
4. For :
5. For :

So,

Matrix A:

Thirdly, if N =4, then

So, the dimensions of matrix A is N\*2 and there is a pattern.

The following code generate matrix A, B and C based on N

1. clear all

2. clc

3. Rs =1;

4. Rdz = 5;

5. Ldz = 2;

6. Cdz =10;

7. N = 4;

8. numStates = 2 \* N; % Each section has 2 states (current and voltage)

9. A = zeros(numStates, numStates);% Initialize A matrix

10. for i = 1:N

11. if i == 1

12. A(1, 1) = -(Rs + Rdz) / Ldz; % first term (Rs + Rdz)

13. else

14. A(2\*i-1, 2\*i-1) = -Rdz / Ldz;

15. end

16. if i > 1

17. A(2\*i-1, 2\*(i-1)) = 1 / Ldz;

18. A(2\*i-1, 2\*i) = -1 / Ldz;

19. end

20. if i < N

21. A(2\*i-1, 2\*i) = -1 / Ldz;

22. A(2\*i, 2\*i-1) = 1 / Cdz;

23. A(2\*i, 2\*i+1) = -1 / Cdz;

24. else

25. A(2\*i, 2\*i-1) = 1 / Cdz;

26. end

27. end

28. % Initialize B matrix

29. B = zeros(numStates, 1);

30. B(1) = 1 / Ldz;

31. %C matrix

32. C = zeros(1, numStates);

33. C(end) = 1;

34. A

35.

This code can be put in a function and just by passing R,Rs,L, length of the line and N will generate the matrices.

* To validate AWE, we can use the following method.

The impulse response of a state space model is given by:

On MATLAB we can simply use the following code to obtain the impulse response.

1. clear all

2. clc

3. %find the theorytical impulse response

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

5. B = [1; 0; 0; 0];

6. C = [1; 0; 0; 0];

7. C = transpose(C);

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

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

10. t = 0:0.01:2;

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

12. % Plot y(t)

13. plot(t, y\_values);

14.

Comparing AWE with this method, case 3 and 4 of 4th order matrix gives the best results.

A graph of a function

Description automatically generatedA graph with red dots and blue lines

Description automatically generated

* Find the unit step response by doing one step and then the unit step as in section 5

Integration Method (Section V):

To avoid computationally expensive explicit convolution, the paper employs recursive convolution based on the pole-residue representation of the transfer function. For a system described in the Laplace domain as:

Assume *x*(*t*) is piecewise constant over each time interval [Solving the above over the time interval ​using the recursive solution is:

This equation updates the output at ​ using only the previous state  and the input   eliminating the need to store the entire history of x(t) [1]*.*

Considering the previous example, let’s say the input is a unit step as x(t) = 1, and with zero initial condition. Then, the step response can be found by forming a state space and using step. Additionally, x(t) = , below is an attempt to integrate the recursive method with awe to obtain the unit step.

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

% 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);

end

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

end

% Plot Results

figure(approx\_order);

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 Impulse', '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 graph of a function

Description automatically generated with medium confidence

* Y-parameters

Will look at this after making sure that the previous code works as expected.

* Complex frequency hoping
* Final FYP report