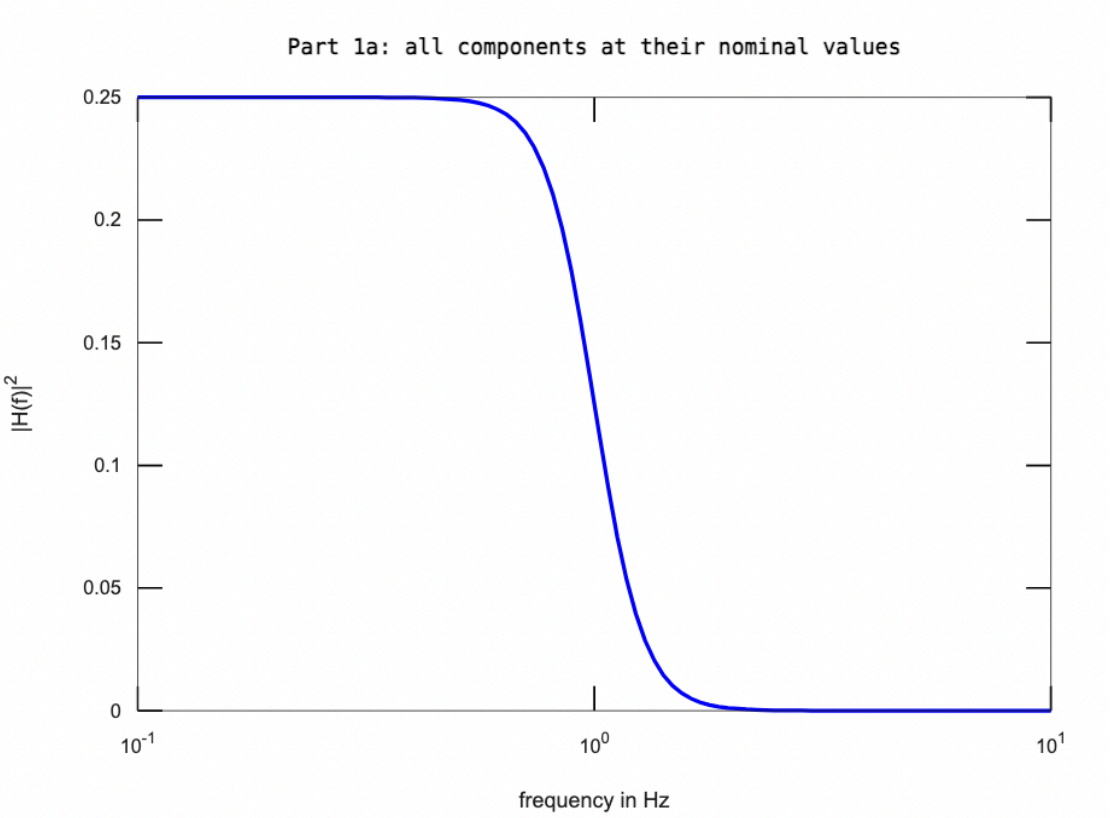
# ECE 341 | Project 2
# University of Illinois at Chicago

# Circuit Analysis & Design Applications of Probability Theory

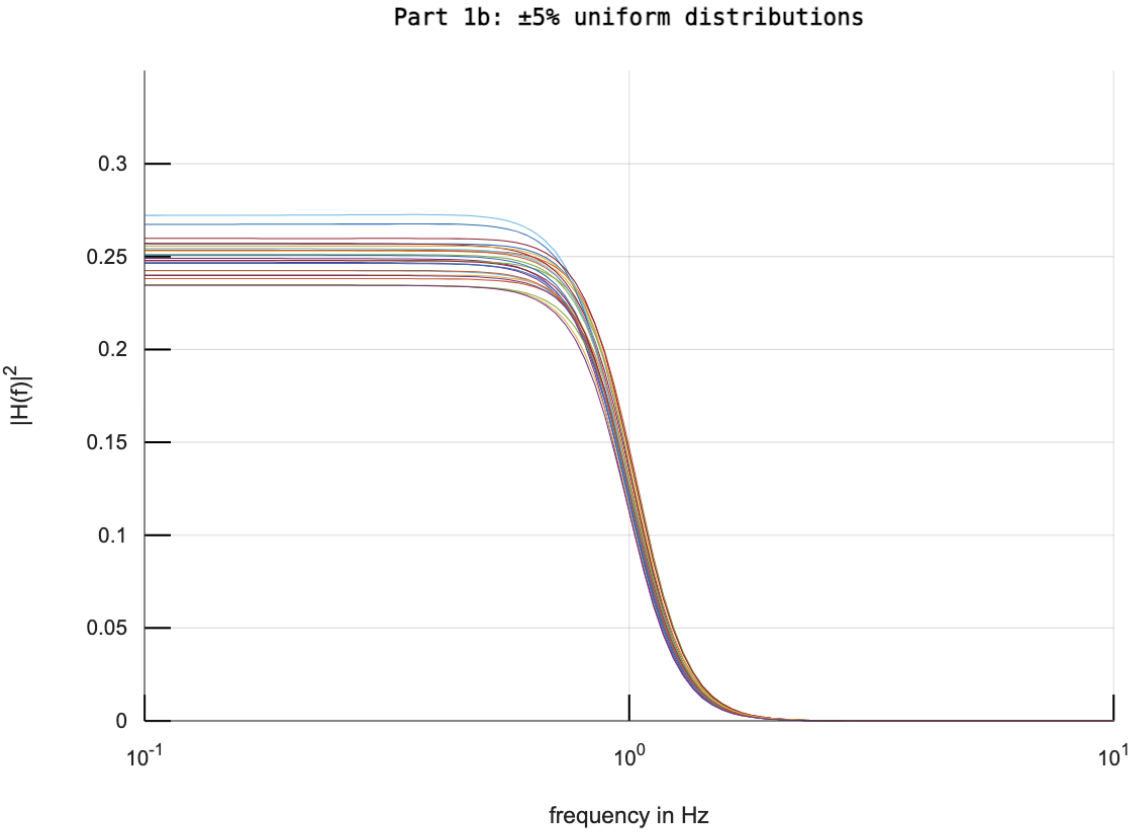
**Qudsia Sultana**

**a. Calculate |H(f)|^2 for Circuit 1, using the provided function Frequency_response1, when all component values are as specified above. Plot the result on a log frequency scale.**




```
R1 = 1.0000;
R2 = 1.0000;
C1 = 0.1218;
C2 = 0.2940;
L1 = 0.2940;
L2 = 0.1218;

% Define the frequency range
f = logspace(-1, 1, 100);
H_f_squared = Frequency_response1(R1, R2, C1, C2, L1, L2, f);

% Plot the frequency response on a log scale
figure;
semilogx(f, H_f_squared, 'b', 'LineWidth', 2);
title('Part 1a: all components at their nominal values');
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
grid on;
```

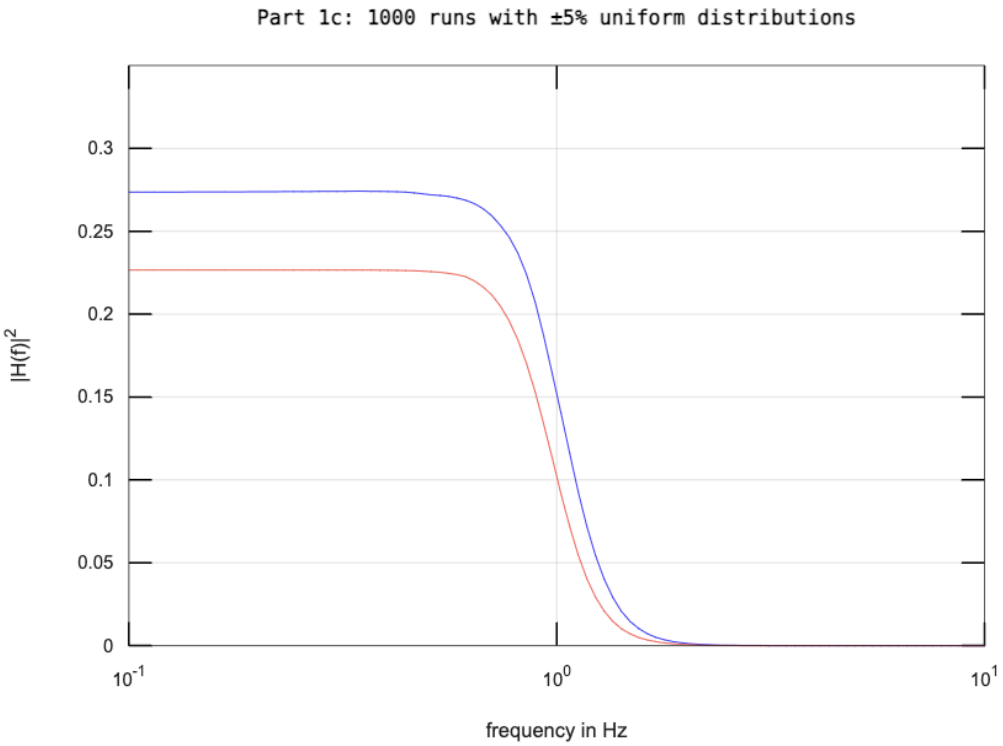
**b. Create 25 superimposed plots of |H(f)| 2 vs. f on a single graph. Randomize Circuit 1's component values independently for each plot by generating samples from continuous random variables uniformly distributed over the range [nominal component values ±5%].**




```
% Define nominal component values
R1_nominal = 1.0000;
R2_nominal = 1.0000;
C1_nominal = 0.1218;
C2_nominal = 0.2940;
L1_nominal = 0.2940;
L2_nominal = 0.1218;

% Define the frequency range
f = logspace(-1, 1, 100);

% Preallocate a matrix to hold all the H(f) values
all_H_f_squared = zeros(length(f), 25);

% Create the plots
figure;
hold on;
```

```
% Monte Carlo simulation for 25 iterations
for i = 1:25
    % Randomize component values within ±5% of their nominal values
    R1 = R1_nominal * (1 + (rand(1) - 0.5) * 0.1);
    R2 = R2_nominal * (1 + (rand(1) - 0.5) * 0.1);
    C1 = C1_nominal * (1 + (rand(1) - 0.5) * 0.1);
    C2 = C2_nominal * (1 + (rand(1) - 0.5) * 0.1);
    L1 = L1_nominal * (1 + (rand(1) - 0.5) * 0.1);
    L2 = L2_nominal * (1 + (rand(1) - 0.5) * 0.1);

    all_H_f_squared(:, i) = Frequency_response1(R1,R2,C1,C2,L1,L2,f);

    semilogx(f, all_H_f_squared(:, i));
end

% Adjust the graph to match the provided image
title('Part 1b: +/- 5% uniform distributions');
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
xlim([0.1 10]); % x-axis limits
ylim([0 0.35]); % y-axis limits
grid on; % Add a grid
set(gca, 'FontSize', 12);

hold off;
```

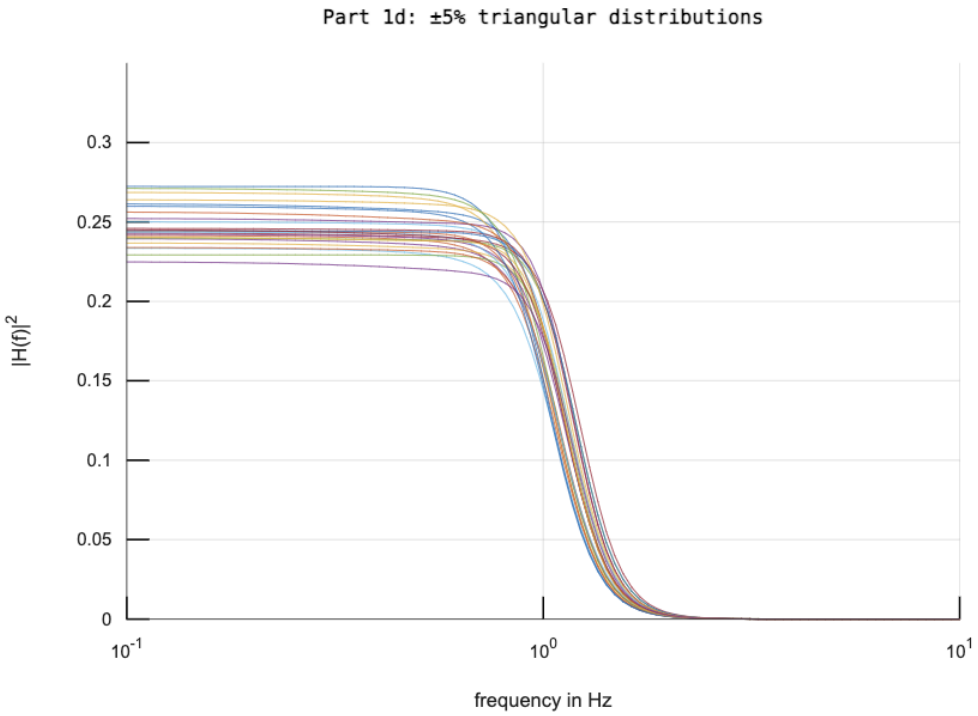
**c. Independently randomize Circuit 1's component values as you did in (b) and calculate |H(f)|^2. Repeat this 1000 times, retaining the maximum and minimum values of |H(f)|^2 at each value of f. Plot the resulting |H(f)|^2 max and |H(f)|^2 mim waveforms on the same graph.**




```
Hmag2_max = zeros(size(f));
Hmag2_min = inf(size(f));

for i = 1:1000
    R1 = R1_nominal * (1 + 0.1*rand - 0.05);
    R2 = R2_nominal * (1 + 0.1*rand - 0.05);
    C1 = C1_nominal * (1 + 0.1*rand - 0.05);
    C2 = C2_nominal * (1 + 0.1*rand - 0.05);
    L1 = L1_nominal * (1 + 0.1*rand - 0.05);
    L2 = L2_nominal * (1 + 0.1*rand - 0.05);

    Hmag2 = Frequency_response1(R1, R2, C1, C2, L1, L2, f);
    Hmag2_max = max(Hmag2_max, Hmag2);
    Hmag2_min = min(Hmag2_min, Hmag2);
end

figure;
semilogx(f, Hmag2_max, 'b', f, Hmag2_min, 'r');
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
```

```
title('Part 1c: 1000 runs with ±5% uniform distributions');
grid on;
axis([min(f) max(f) 0 0.35]);
```

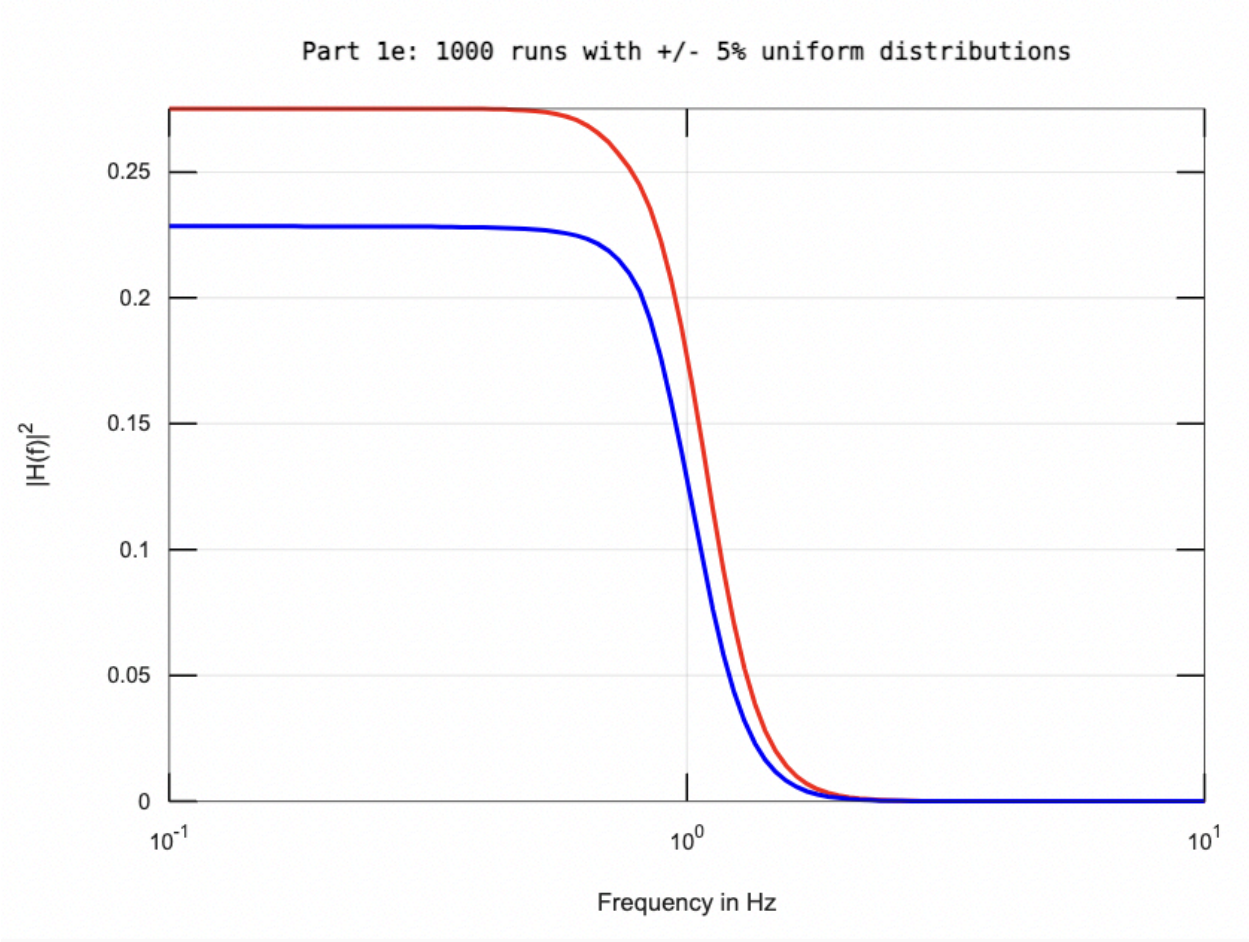
**d. Randomize the Circuit1 component values according to a triangular distribution over the range [nominal values ±5%] by repeating b.**




```
figure;
hold on;

% Define the nominal component values
R1_nominal = 1.0000;
R2_nominal = 1.0000;
C1_nominal = 0.1218;
C2_nominal = 0.2940;
L1_nominal = 0.2940;
L2_nominal = 0.1218;

% Define the frequency range
f = logspace(-1, 1, 100);

% Generate and plot 25 random frequency responses
for i = 1:25
    R1 = R1_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
```

```
    R2 = R2_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    C1 = C1_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    C2 = C2_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    L1 = L1_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    L2 = L2_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));

    % Calculate the magnitude-squared frequency response
    Hmag2 = Frequency_response1(R1, R2, C1, C2, L1, L2, f);

    % Plot the response
    semilogx(f, Hmag2);
end

% Customize the plot
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
title('Part 1d: ±5% triangular distributions');
grid on;
axis([min(f) max(f) 0 0.35]);
hold off;

% Triangular distribution random number generator
function r = triangular_rnd()
    u = rand();
    if u < 0.5
        r = sqrt(2*u) - 1;
    else
        r = 1 - sqrt(2*(1-u));
    end
end
```

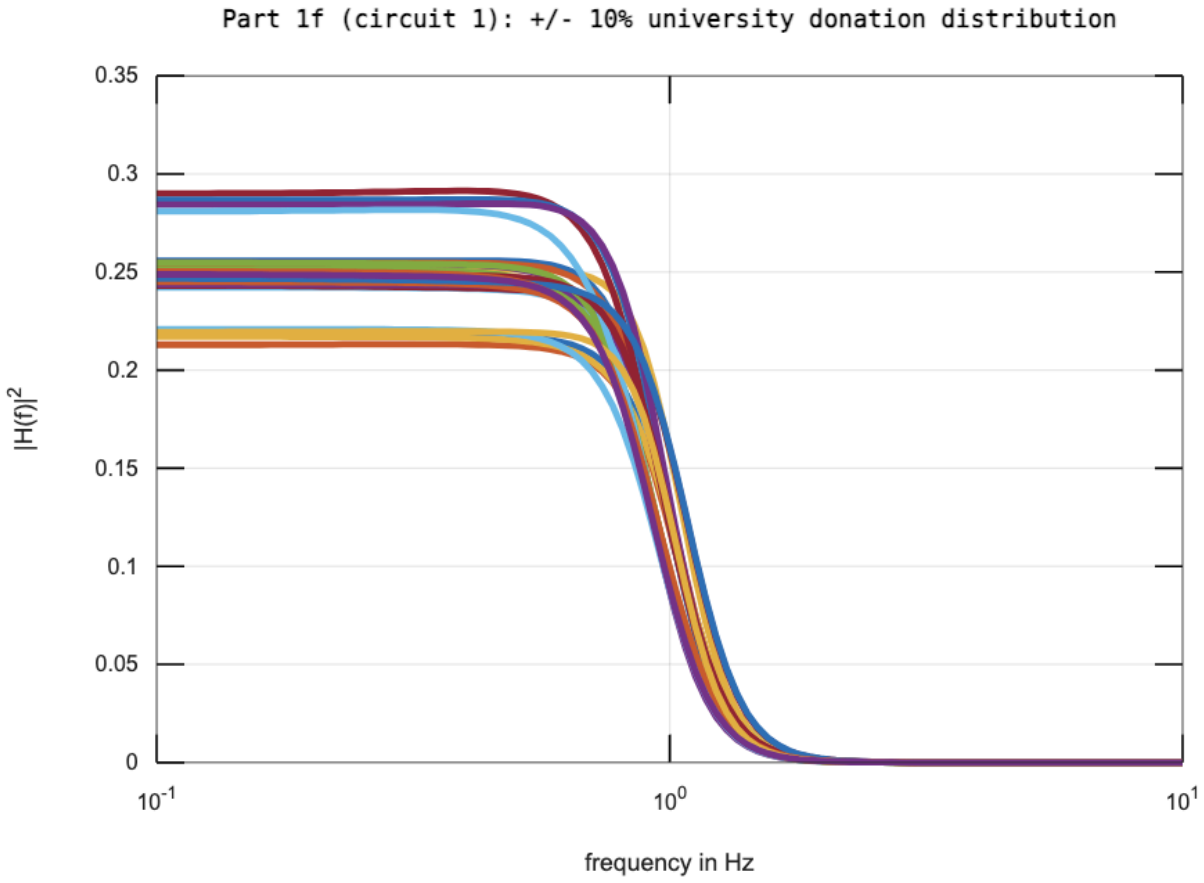
**e. Randomize the Circuit1 component values according to a triangular distribution over the range [nominal values ±5%] by repeating c.**




```
R1_nominal = 1.0000;
R2_nominal = 1.0000;
C1_nominal = 0.1218;
C2_nominal = 0.2940;
L1_nominal = 0.2940;
L2_nominal = 0.1218;

% Define the frequency range
f = logspace(-1, 1, 100);

% Preallocate arrays for the maximum and minimum magnitude-squared
responses
H_f_squared_max = zeros(size(f));
H_f_squared_min = inf(size(f));
```

```
% Perform 1000 runs with triangular distribution of component values
for i = 1:1000
    % Generate component values with ±5% variation using the mean of two
uniform distributions
    R1 = R1_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    R2 = R2_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    C1 = C1_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    C2 = C2_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    L1 = L1_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    L2 = L2_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));

    % Calculate the magnitude-squared frequency response
    H_f_squared = Frequency_response1(R1, R2, C1, C2, L1, L2, f);

    % Update the maximum and minimum values
    H_f_squared_max = max(H_f_squared_max, H_f_squared);
    H_f_squared_min = min(H_f_squared_min, H_f_squared);
end

% Plotting the maximum and minimum values
figure;
plot(f, H_f_squared_max, 'r', 'LineWidth', 2); % Max response in red
hold on;
plot(f, H_f_squared_min, 'b', 'LineWidth', 2); % Min response in blue
hold off;

% Formatting the plot
xlabel('Frequency in Hz');
ylabel('|H(f)|^2');
title('Part 1e: 1000 runs with +/- 5% uniform distributions');
grid on;
axis([min(f) max(f) 0 max(H_f_squared_max)]);
set(gca, 'XScale', 'log');
```

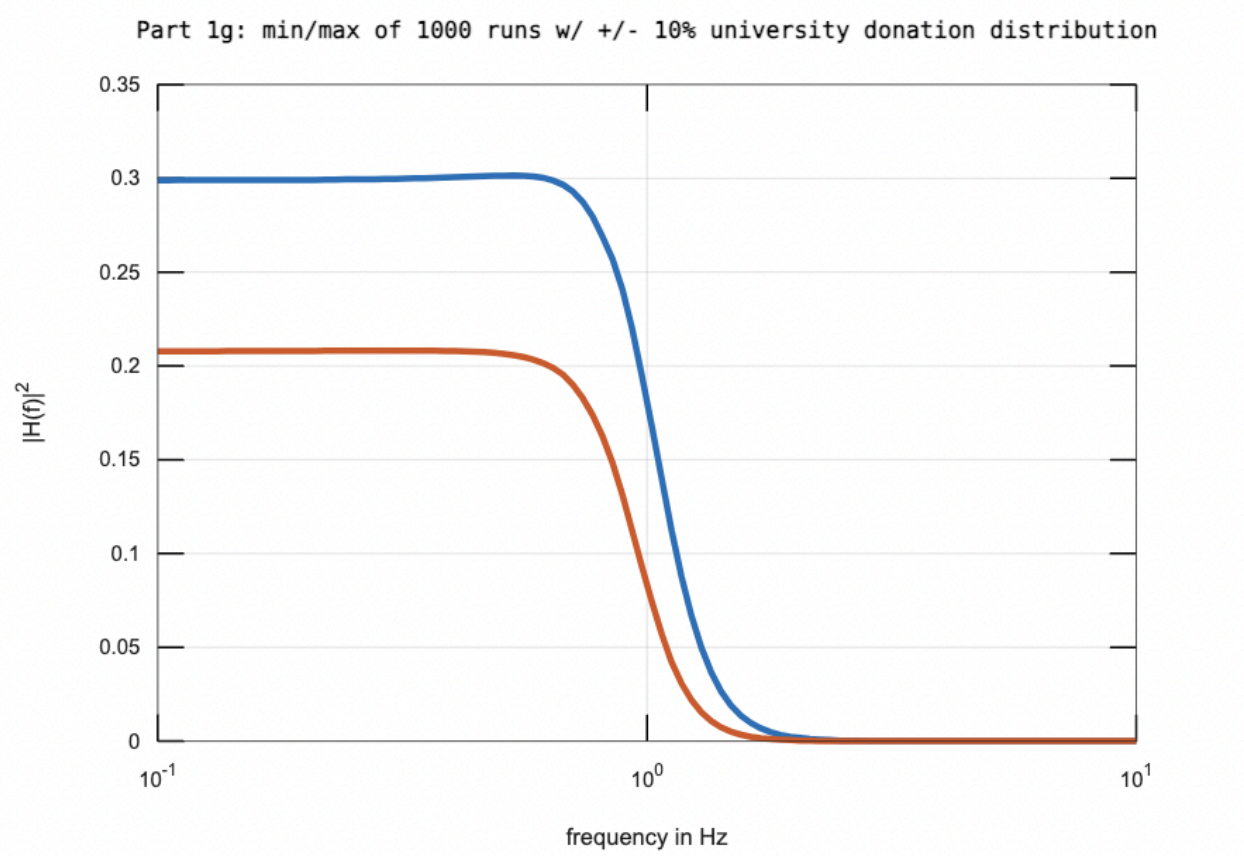
**f. Randomizing the Circuit1 component values according to a "university donation" distribution: triangular over the range [nominal values ±10%], then removing the [nominal values ±5%] portion using b**




```
R1 = 1.0000;
R2 = 1.0000;
C1 = 0.1218;
C2 = 0.2940;
L1 = 0.2940;
L2 = 0.1218;

f = logspace(-1,1, 1e2);

for n = 1:25
    x = [];
    while length(x) <9
        x = 1 + 0.10*(rand(1,100)+rand(1,100)-1);
        x((x>0.95)&(x<1.05)) =[];
    end
    r1 = R1*x(1);
    r2 = R2*x(2);
```

```
    c1 = C1*x(3);
    c2 = C2*x(4);
    l1 = L1*x(5);
    l2 = L2*x(6);
    Hmag2 = Frequency_response1 (r1,r2,c1,c2,l1,l2,f);
    semilogx (f,Hmag2,'linewidth',3);hold on
end
hold off

title('Part 1f (circuit 1): +/- 10% university donation distribution')
xlabel ('frequency in Hz')
ylabel ('|H(f)|^2 ')
grid on
```

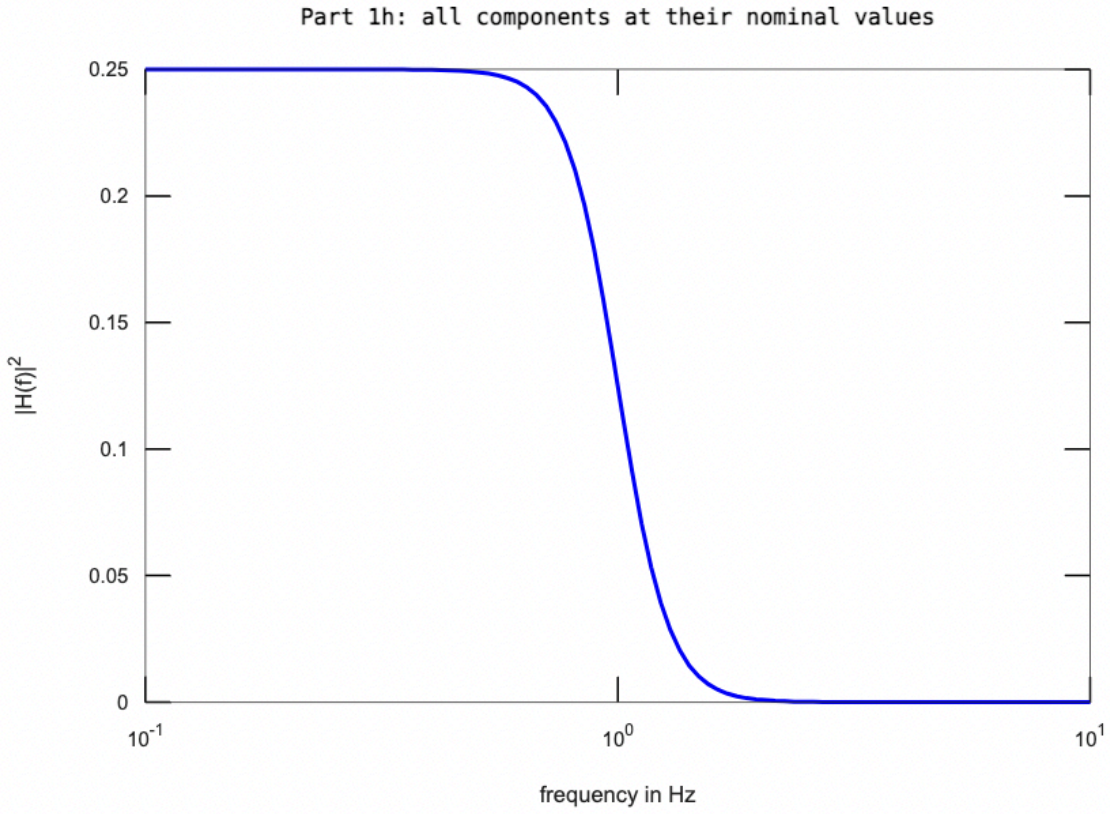
**g. Randomizing the Circuit1 component values according to a "university donation" distribution: triangular over the range [nominal values ±10%], then removing the [nominal values ±5%] portion using c**




```
R1 = 1.0000;
R2 = 1.0000;
C1 = 0.1218;
C2 = 0.2940;
```

```
L1 = 0.2940;
L2 = 0.1218;

f = logspace(-1,1, 1e2);

Hmag2_max = -Inf*ones(size(f));
Hmag2_min = Inf*ones(size(f));

for n= 1:1000
    x=[];
    while length(x)< 9
        x = 1 + 0.10*(rand(1,100) +rand(1,100)-1);
        x((x>0.95) & (x<1.05)) = [];
    end
    r1 = R1*x(1);
    r2 = R2*x(2);
    c1 = C1*x(3);
    c2 = C2*x(4);
    l1 = L1*x(5);
    l2 = L2*x(6);
    Hmag2 = Frequency_response1 (r1,r2,c1,c2,l1,l2,f);

    Hmag2_max = max(Hmag2_max, Hmag2);
    Hmag2_min = min(Hmag2_min, Hmag2);
end

semilogx(f,Hmag2_max,'linewidth',3);
hold on;
semilogx(f,Hmag2_min,'linewidth',3);
hold off;

title('Part 1g: min/max of 1000 runs w/ +/- 10% university donation
distribution')
xlabel ('frequency in Hz')
ylabel ('|H(f)|^2 ')
grid on
```

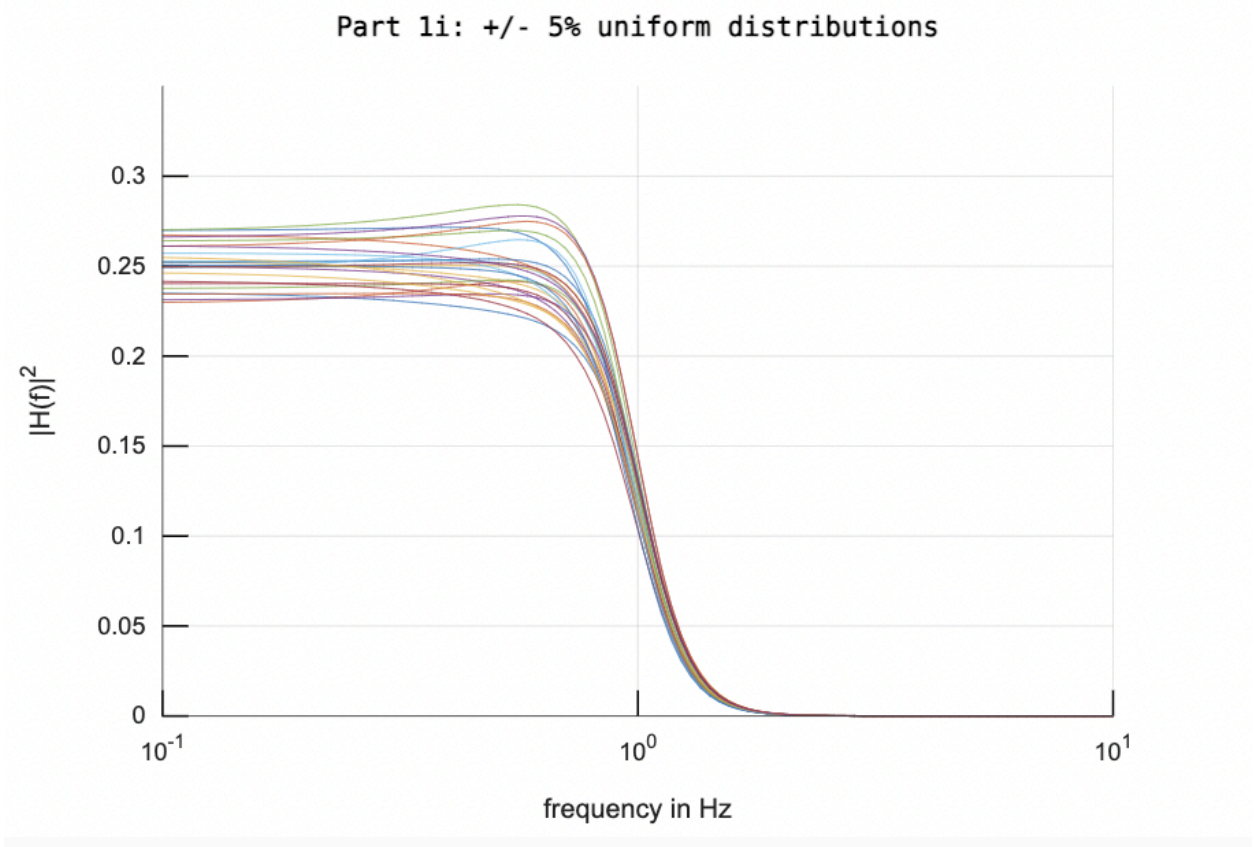
**h. Calculate |H(f)|^2 for Circuit 2, using the provided function Frequency_response2, when all component values are as specified above. Plot the result on a log frequency scale.**




```
R1 = 1.0000;
R2 = 1.0000;
R3 = 1.0000;
R4 = 2.0000;
R5 = 2.0000;
C1 = 0.1722;
C2 = 0.1471;
C3 = 0.4160;
C4 = 0.0609;

f = logspace(-1, 1, 100);

H_f_squared = Frequency_response2(R1,R2,R3,R4,R5,C1,C2,C3,C4,f);

% Plot the frequency response on a log scale
figure;
semilogx(f, H_f_squared, 'b', 'LineWidth', 2); % Blue line with a line
width of 2
title('Part 1h: all components at their nominal values');
xlabel('frequency in Hz');
```

```
ylabel('|H(f)|^2');
xlim([min(f) max(f)]); % Set x-axis to cover the range of f
ylim([0 max(H_f_squared)*1.1]); % Set y-axis slightly above the max value
of H_f_squared
grid on;
```

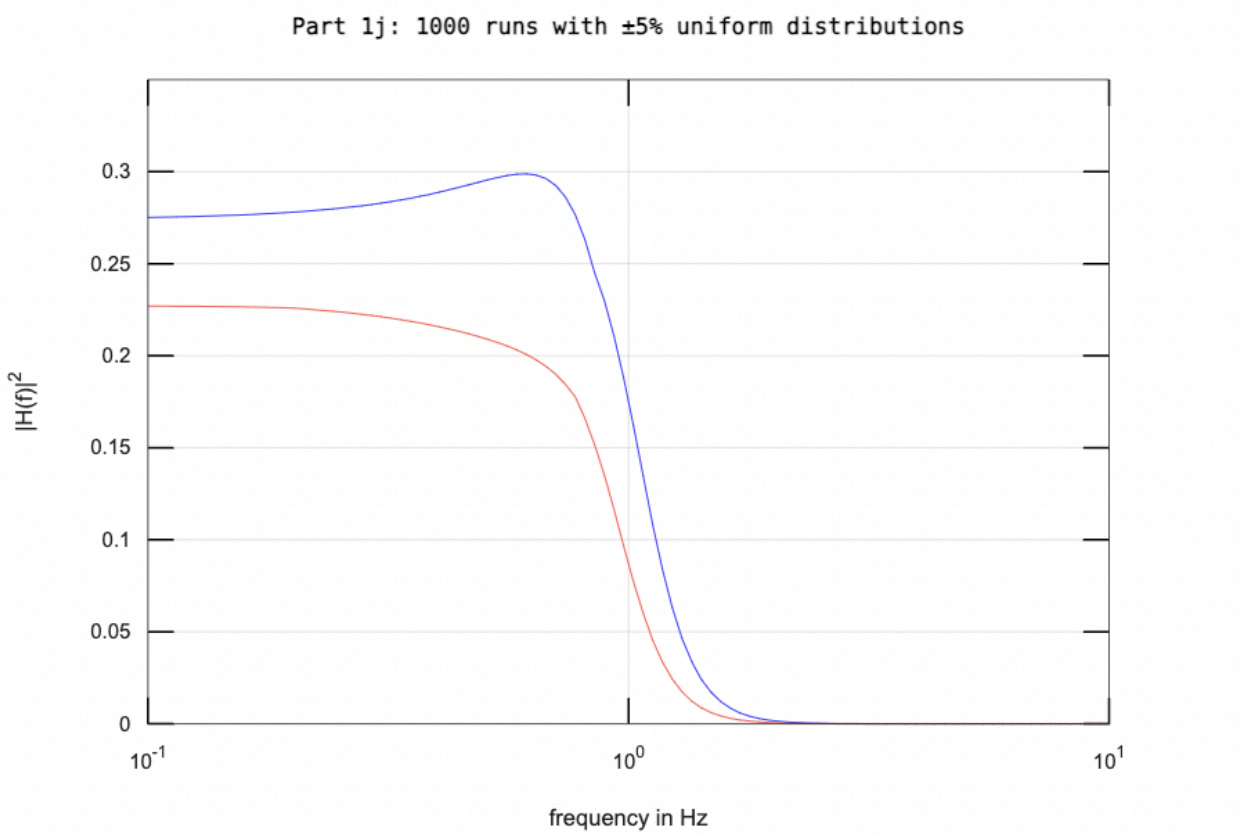
**i. Create 25 superimposed plots of |H(f)|^2 vs. f on a single graph. Randomize Circuit 2's component values independently for each plot by generating samples from continuous random variables uniformly distributed over the range [nominal component values ±5%].**




```
R1_nominal = 1.0000;
R2_nominal = 1.0000;
R3_nominal = 1.0000;
R4_nominal = 2.0000;
R5_nominal = 2.0000;
C1_nominal = 0.1722;
C2_nominal = 0.1471;
C3_nominal = 0.4160;
C4_nominal = 0.0609;

% Define the frequency range
f = logspace(-1, 1, 100);
```

```
% Preallocate a matrix to hold all the H(f) values
all_H_f_squared = zeros(length(f), 25);

% Create the plots
figure;
hold on; % Hold on to plot multiple lines

% Monte Carlo simulation for 25 iterations
for i = 1:25
    % Randomize component values within ±5% of their nominal values
    R1 = R1_nominal * (1 + (rand(1) - 0.5) * 0.1);
    R2 = R2_nominal * (1 + (rand(1) - 0.5) * 0.1);
    R3 = R3_nominal * (1 + (rand(1) - 0.5) * 0.1);
    R4 = R4_nominal * (1 + (rand(1) - 0.5) * 0.1);
    R5 = R5_nominal * (1 + (rand(1) - 0.5) * 0.1);
    C1 = C1_nominal * (1 + (rand(1) - 0.5) * 0.1);
    C2 = C2_nominal * (1 + (rand(1) - 0.5) * 0.1);
    C3 = C3_nominal * (1 + (rand(1) - 0.5) * 0.1);
    C4 = C4_nominal * (1 + (rand(1) - 0.5) * 0.1);

    % Calculate |H(f)|^2 using the Frequency_response1 function with
randomized components
    all_H_f_squared(:, i) =
Frequency_response2(R1,R2,R3,R4,R5,C1,C2,C3,C4,f);

    % Plot the current |H(f)|^2
    semilogx(f, all_H_f_squared(:, i));
end

% Adjust the graph to match the provided image
title('Part 1i: +/- 5% uniform distributions');
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
xlim([0.1 10]); % x-axis limits
ylim([0 0.35]); % y-axis limits
grid on; % Add a grid
set(gca, 'FontSize', 12); % Set font size

hold off;
```

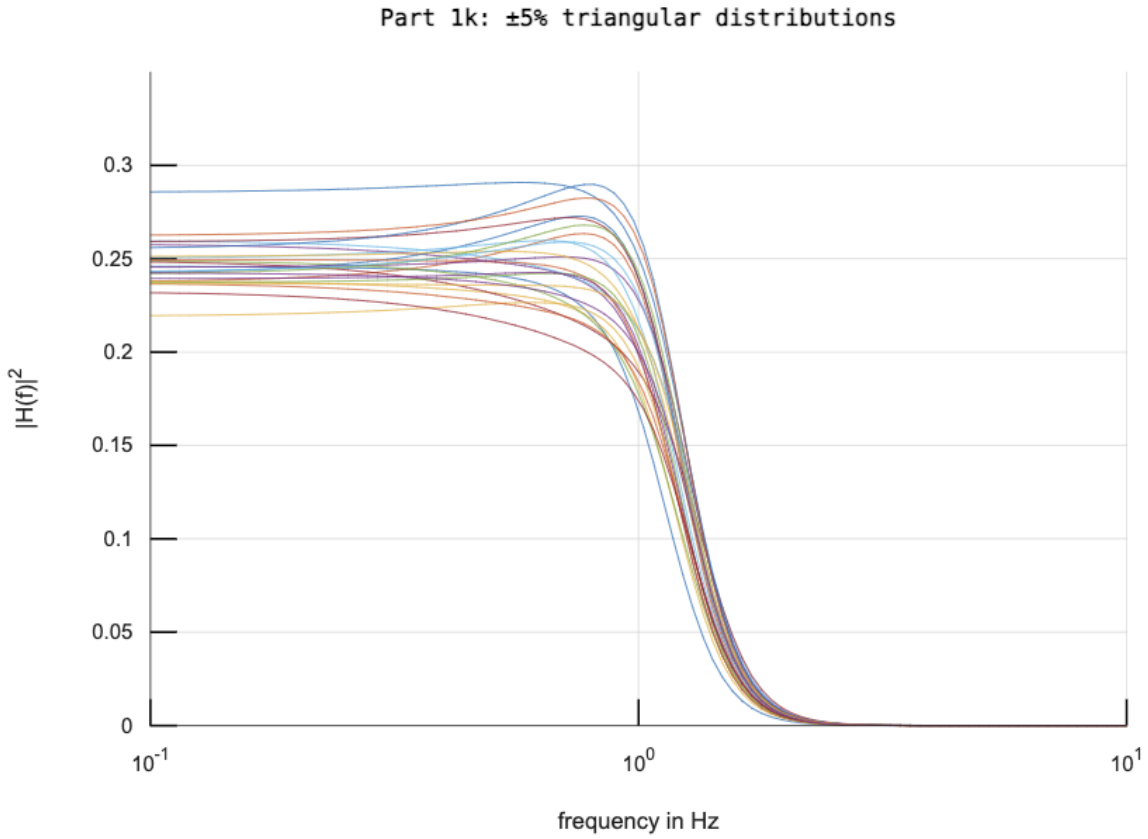
**j. Independently randomize Circuit 2's component values as you did in (i) and calculate |H(f)|^2. Repeat this 1000 times, retaining the maximum and minimum values of |H(f)|^2 at each value of f. Plot the resulting |H(f)|^2 max and |H(f)|^2 min waveforms on the same graph.**




```
R1_nominal = 1.0000;
R2_nominal = 1.0000;
R3_nominal = 1.0000;
R4_nominal = 2.0000;
R5_nominal = 2.0000;
C1_nominal = 0.1722;
C2_nominal = 0.1471;
C3_nominal = 0.4160;
C4_nominal = 0.0609;

Hmag2_max = zeros(size(f));
Hmag2_min = inf(size(f));

for i = 1:1000
    R1 = R1_nominal * (1 + 0.1*rand - 0.05);
    R2 = R2_nominal * (1 + 0.1*rand - 0.05);
    R3 = R3_nominal * (1 + 0.1*rand - 0.05);
    R4 = R4_nominal * (1 + 0.1*rand - 0.05);
```

```
    R5 = R5_nominal * (1 + 0.1*rand - 0.05);
    C1 = C1_nominal * (1 + 0.1*rand - 0.05);
    C2 = C2_nominal * (1 + 0.1*rand - 0.05);
    C3 = C3_nominal * (1 + 0.1*rand - 0.05);
    C4 = C4_nominal * (1 + 0.1*rand - 0.05);

    Hmag2 = Frequency_response2(R1,R2,R3,R4,R5,C1,C2,C3,C4,f);
    Hmag2_max = max(Hmag2_max, Hmag2);
    Hmag2_min = min(Hmag2_min, Hmag2);
end

figure;
semilogx(f, Hmag2_max, 'b', f, Hmag2_min, 'r');
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
title('Part 1j: 1000 runs with ±5% uniform distributions');
grid on;
axis([min(f) max(f) 0 0.35]);
```

**k. Randomize the Circuit2 component values according to a triangular distribution over the range [nominal values ±5%] by repeating i.**

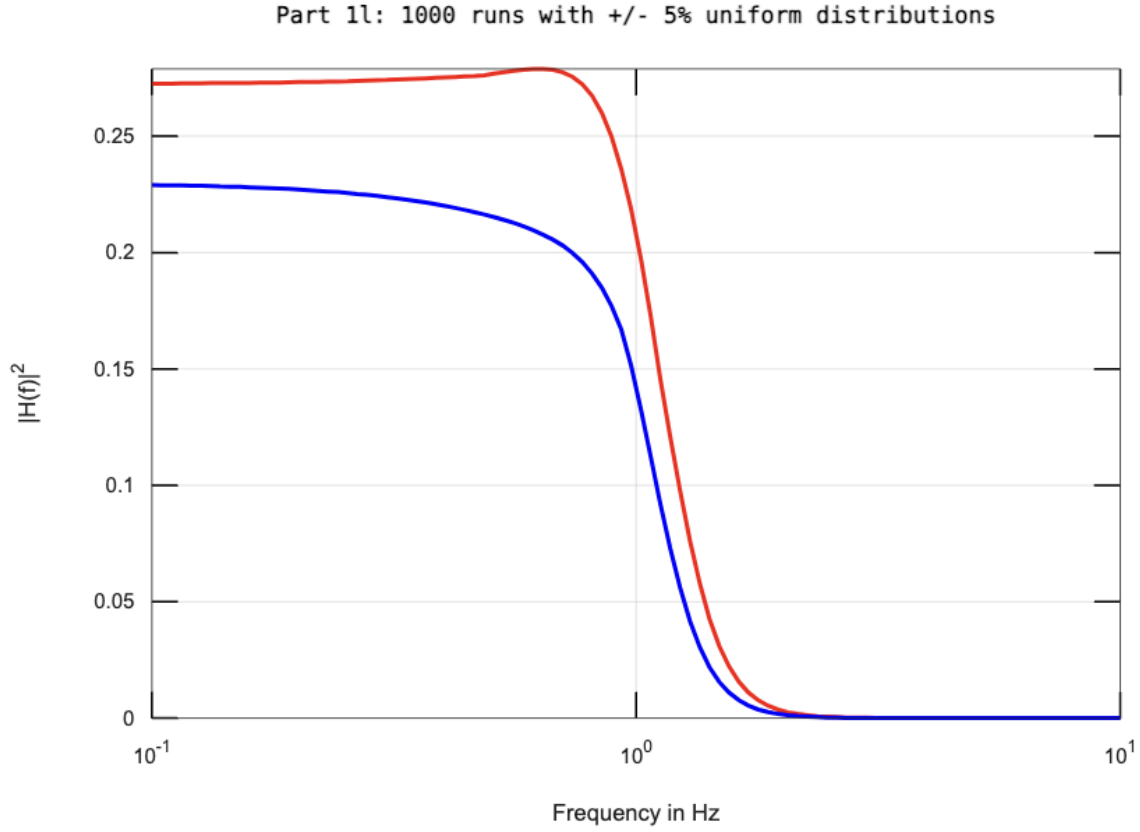



```
figure;
```

```
hold on;

% Define the nominal component values
R1_nominal = 1.0000;
R2_nominal = 1.0000;
R3_nominal = 1.0000;
R4_nominal = 2.0000;
R5_nominal = 2.0000;
C1_nominal = 0.1722;
C2_nominal = 0.1471;
C3_nominal = 0.4160;
C4_nominal = 0.0609;


% Define the frequency range
f = logspace(-1, 1, 100);

% Generate and plot 25 random frequency responses
for i = 1:25
    R1 = R1_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    R2 = R2_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    R3 = R3_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    R4 = R4_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    R5 = R5_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    C1 = C1_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    C2 = C2_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    C3 = C3_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));
    C4 = C4_nominal * (1 - 0.05 + 0.1 * (triangular_rnd() - 0.5));

    % Calculate the magnitude-squared frequency response
    Hmag2 = Frequency_response2(R1,R2,R3,R4,R5,C1,C2,C3,C4,f);

    % Plot the response
    semilogx(f, Hmag2);
end

% Customize the plot
xlabel('frequency in Hz');
ylabel('|H(f)|^2');
title('Part 1k: ±5% triangular distributions');
grid on;
axis([min(f) max(f) 0 0.35]);
hold off;

% Triangular distribution random number generator
function r = triangular_rnd()
    u = rand();
```

```
    if u < 0.5
        r = sqrt(2*u) - 1;
    else
        r = 1 - sqrt(2*(1-u));
    end
end
```

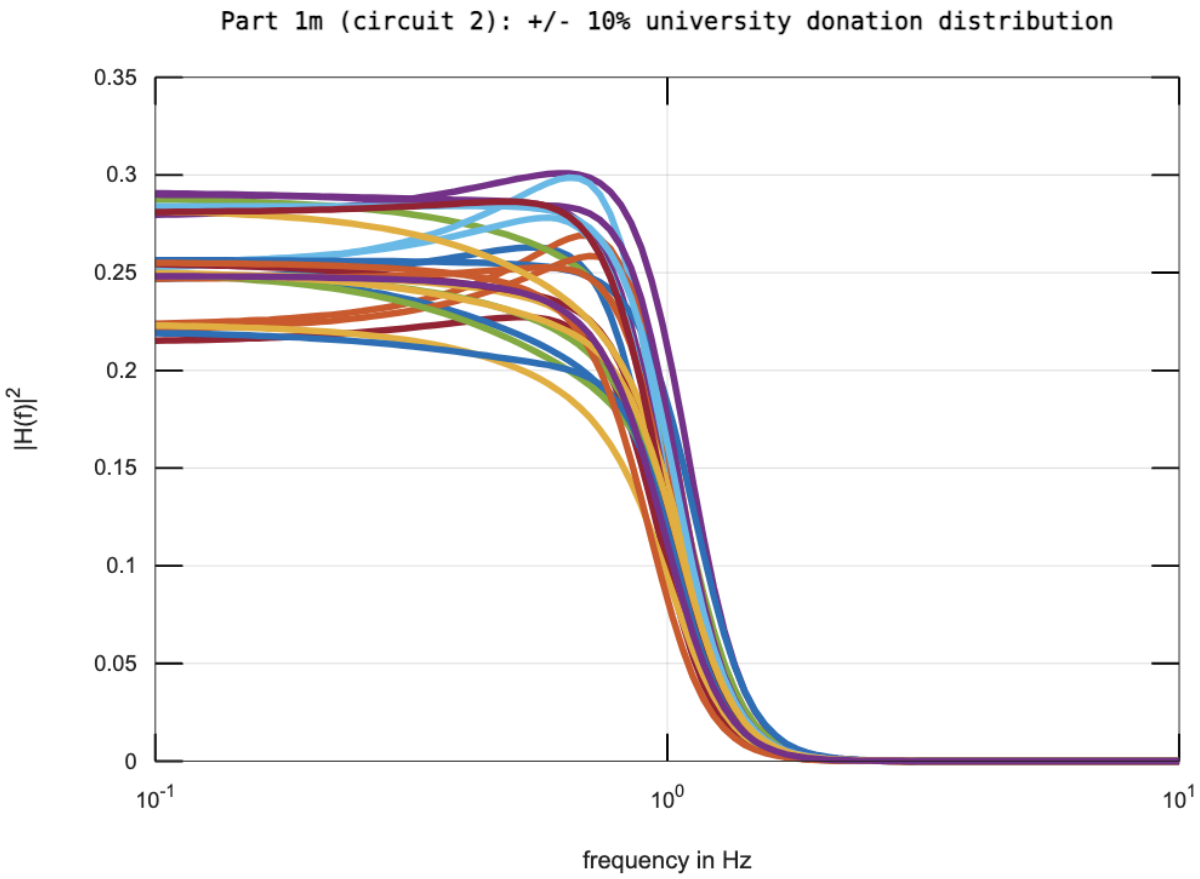
**l. Randomize the Circuit2 component values according to a triangular distribution over the range [nominal values ±5%] by repeating j.**




```
R1_nominal = 1.0000;
R2_nominal = 1.0000;
R3_nominal = 1.0000;
R4_nominal = 2.0000;
R5_nominal = 2.0000;
C1_nominal = 0.1722;
C2_nominal = 0.1471;
C3_nominal = 0.4160;
C4_nominal = 0.0609;


% Define the frequency range
f = logspace(-1, 1, 100);
```

```
% Preallocate arrays for the maximum and minimum magnitude-squared
responses
H_f_squared_max = zeros(size(f));
H_f_squared_min = inf(size(f));

% Perform 1000 runs with triangular distribution of component values
for i = 1:1000
    % Generate component values with ±5% variation using the mean of two
uniform distributions
    R1 = R1_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    R2 = R2_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    R3 = R3_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    R4 = R4_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    R5 = R5_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    C1 = C1_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    C2 = C2_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    C3 = C3_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));
    C4 = C4_nominal * (1 - 0.05 + 0.1 * mean(rand(2,1) - 0.5));

    % Calculate the magnitude-squared frequency response
    H_f_squared = Frequency_response2(R1,R2,R3,R4,R5,C1,C2,C3,C4,f);

    % Update the maximum and minimum values
    H_f_squared_max = max(H_f_squared_max, H_f_squared);
    H_f_squared_min = min(H_f_squared_min, H_f_squared);
end

% Plotting the maximum and minimum values
figure;
plot(f, H_f_squared_max, 'r', 'LineWidth', 2); % Max response in red
hold on;
plot(f, H_f_squared_min, 'b', 'LineWidth', 2); % Min response in blue
hold off;

% Formatting the plot
xlabel('Frequency in Hz');
ylabel('|H(f)|^2');
title('Part 1l: 1000 runs with +/- 5% uniform distributions');
grid on;
axis([min(f) max(f) 0 max(H_f_squared_max)]);
set(gca, 'XScale', 'log');
```

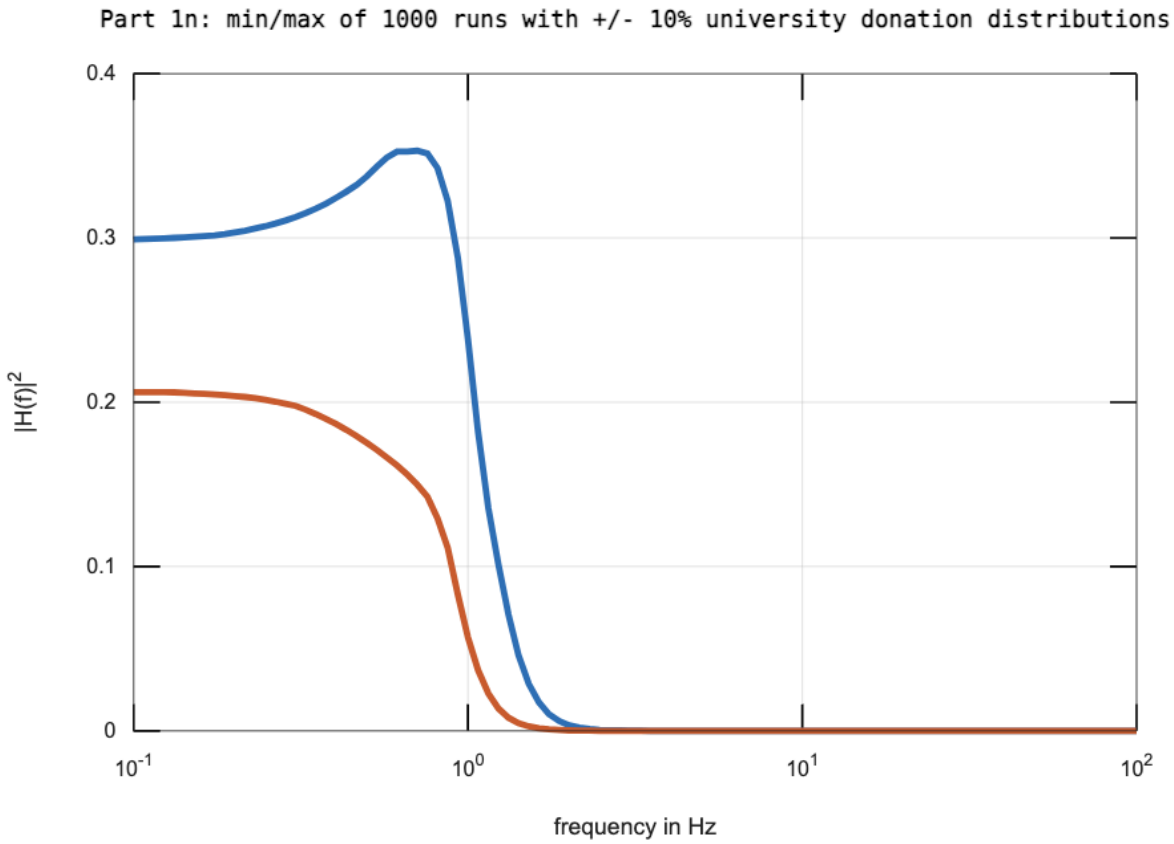
**m. Randomizing the Circuit2 component values according to a "university donation" distribution: triangular over the range [nominal values ±10%], then removing the [nominal values ±5%] portion using i.**




```
R1 = 1.0000;
R2 = 1.0000;
R3 = 1.0000;
R4 = 2.0000;
R5 = 2.0000;
C1 = 0.1722;
C2 = 0.1471;
C3 = 0.4160;
C4 = 0.0609;
f = logspace(-1,1, 1e2);

for n = 1:25
    x = [];
    while length(x) <9
        x = 1 + 0.10*(rand(1,100)+rand(1,100)-1);
        x((x>0.95)&(x<1.05)) =[];
    end
    r1 = R1*x(1);
```

```
    r2 = R2*x(2);
    r3 = R3*x(3);
    r4 = R4*x(4);
    r5 = R5*x(5);
    c1 = C1*x(6);
    c2 = C2*x(7);
    c3 = C3*x(8);
    c4 = C4*x(9);
    Hmag2 = Frequency_response2 (r1,r2,r3,r4,r5,c1,c2,c3,c4,f);
    semilogx (f,Hmag2,'linewidth',3);hold on
end
hold off
title('Part 1m (circuit 2): +/- 10% university donation distribution')
xlabel ('frequency in Hz')
ylabel ('|H(f)|^2 ')
grid on
```

**n. Randomizing the Circuit2 component values according to a "university donation" distribution: triangular over the range [nominal values ±10%], then removing the [nominal values ±5%] portion using j.**



```
% Define nominal component values for Circuit 2
R1_nom = 1.0000;
R2_nom = 1.0000;
R3_nom = 1.0000;
R4_nom = 2.0000;
R5_nom = 2.0000;
C1_nom = 0.1722;
C2_nom = 0.1471;
C3_nom = 0.4160;
C4_nom = 0.0609;

% Define frequency vector, logarithmically spaced between 10^(-1) and
10^(2)
f = logspace(-1, 2, 100);

% Initialize arrays to store the max and min |H(f)|^2 values for each
frequency
Hmag2_max = -Inf * ones(size(f));
Hmag2_min = Inf * ones(size(f));

% Monte Carlo simulation for 1000 runs
for n = 1:1000
    % Initialize placeholder for random component values
    x = [];

    % Generate 9 random component factors using uniform distribution
    % with the range of ±10% about their nominal value
    while length(x) < 9
        % Generate a random number between 0.9 and 1.1, which is ±10%
        % If the number falls outside of 0.95 to 1.05 (±5%), discard it
        x = [x; 1 + 0.1 * (rand(1, 100) + rand(1, 100) - 1)];
        x((x > 0.95) & (x < 1.05)) = [];
    end

    % Assign random values to components
    r1 = R1_nom * x(1);
    r2 = R2_nom * x(2);
    r3 = R3_nom * x(3);
    r4 = R4_nom * x(4);
    r5 = R5_nom * x(5);
    c1 = C1_nom * x(6);
    c2 = C2_nom * x(7);
    c3 = C3_nom * x(8);
    c4 = C4_nom * x(9);

    % Calculate the frequency response for the randomized component values
    Hmag2 = Frequency_response2(r1, r2, r3, r4, r5, c1, c2, c3, c4, f);
```

```
    % Update the max and min |H(f)|^2 values
    Hmag2_max = max(Hmag2_max, Hmag2);
    Hmag2_min = min(Hmag2_min, Hmag2);
end

% Plot the max and min |H(f)|^2 values on a semilogarithmic scale
semilogx(f, Hmag2_max, 'LineWidth', 3); % Max response curve
hold on;
semilogx(f, Hmag2_min, 'LineWidth', 3); % Min response curve
hold off;

% Title and labels for the plot
title('Part 1n: min/max of 1000 runs with +/- 10% university donation
distributions');
xlabel('Frequency in Hz');
ylabel('|H(f)|^2');

grid on;
```

**Conclusion:**

In my project, I conducted a thorough analysis of two distinct circuits to explore the effects of component variability on their frequency responses. Initially, I determined the expected frequency response for each circuit using their respective standard component values as benchmarks. I then embarked on a series of Monte Carlo simulations to mimic real-world inconsistencies in component values. For each circuit, I ran two types of simulations: one where component values had equal chances of varying within a certain range and another that more closely resembled manufacturing reality, with variations tending to stay near the standard values.

These simulations generated a series of visual plots that clearly illustrated the potential variations in each circuit's performance due to changes in component values. The data vividly showed which components influenced the frequency response most significantly. My conclusion from these exercises is that Monte Carlo simulations are invaluable for predicting the range of performance a circuit may exhibit when facing component tolerances. This predictive power is instrumental for circuit design, as it aids in making circuits more reliable. By identifying the critical components that have a pronounced impact on circuit behavior, I can make more informed decisions in the design and manufacturing phases to ensure that the circuits perform consistently and within the expected parameters.