

Síntesis de redes activas

Laboratorio N°4: Filtros Activos

Profesor Titular: Dr. Ing. Pablo Ferreyra

Profesor Adjunto: Ing. César Reale

Alumnos: Campos Mariano, Enzo Verstraete

28 de noviembre de 2024

Resumen

En base a la planilla de requerimientos suministrada, sintetizar un circuito basado en amplificadores operacionales que satisfaga esos requisitos.

1. Metodología general

En general, para cada uno de los casos particulares solicitados, se debe: A. Realizar una sintética introducción teórica. B. Analizar el circuito propuesto, su desarrollo numérico, todos los cálculos analíticos. C. Realizar simulación en LTSPICE. D. Armar el circuito y hacer las mediciones en laboratorio. E. Finalmente comparar los valores calculados, simulados y medidos, y extraer conclusiones a cerca de las diferencias. Analizar las causas. F. Presentar un informe digital y en papel.

2. Desarrollo

1.1 Aproximar la función de atenuación mediante polinomios de Chebyshev utilizando python o matlab (ver Anexo I y II). 1.2 Sintetizar un circuito que satisfaga los requerimientos del punto anterior utilizando topologías bi-cuadráticas de realimentación positiva o negativa, a elección. 1.3 Simular cada

etapa y el filtro total con LTSPICE. 1.4 Calcular la sensibilidad de la frecuencia del polo de cada bicuadrática (w_p) y del ancho de banda (w_p/Q_p). 1.5 Analizar la peor desviación si todos los elementos tienen una tolerancia del 10. 1.6 Realizar una simulación de Montecarlo de las desviaciones con LTspice. 1.7 Armar el circuito, medir experimentalmente las curvas de atenuación y desfase. Contrastarlas con las predicciones teóricas y las simulaciones.

2.1. Aproximación de la función de transferencia

Para la aproximación de la función utilizamos el siguiente código en Matlab, partimos de los parámetros de diseño, en este caso tenemos un filtro pasa banda con las siguientes características:

```

1 Parametros:
2 Fbp = [800 1250]; % Puntos en frecuencia de la banda de
  paso entre 800 y 1250 [Hz]
3 Fbr = [200 5000]; % Puntos en frecuencia de la banda de
  rechazo 0-200 [Hz] y 5000-inf [Hz]
4 Wbp = 2*pi*Fbp; % Puntos de omega [rad/s]
5 Wbr = 2*pi*Fbr; % Puntos en omega [rad/s]
6 Abp = 0.25; % Atenuacion en la banda de paso [dB]
7 Abr = 30; % Atenuacion en la banda de rechazo [dB]
  ]
8
9
```

El segundo paso es obtener la función de transferencia, por medio de la aproximación Chebychev de primer orden:

```

1 TF:
2 [n , Wp] = cheb1ord(Wbp,Wbr,Abp,Abr,"s"); %
  Determinacion del orden y la frecuencia de corte de un
  filtro de chebyshev tipo 1
3 [num , den] = cheby1(n,Abp,Wp,"s"); %
  Determinacion del numerador y denominador de la funcion de
  transferencia del filtro
4 TF_filtro = tf(num,den); % Creacion de
  la TF del filtro
5
6 % Descomposicion en bicuadraticas (debemos descomponer en
  una funcion para
7 % filtro pasa-alto y otra para el filtro pasa-bajo):
8 [Msos , Gg] = tf2sos(num,den); % Descompone
  en bicuadraticas
```

```

9      % Msos: matriz de coeficientes de la bicuadraticas ; Gg:
      ganancia global del filtro
10     TFpb = tf(Gg*Msos(1,1:3) , Msos(1,4:6)); % Funcion de
      transferencia del filtro pasa-bajo
11     % 2*Gg*Msos(1,1:3): coeficiente del numerador escalados
      por la ganancia global, Msos(1,4:6): coeficiente del
      denominador
12     TFpa = tf(Msos(2,1:3) , Msos(2,4:6)); % Funcion de
      transferencia del filtro pasa-alto
13     % 1/2*Msos(2,1:3): Coeficientes del numerador escalados
      por la ganancia global, Msos(2,4:6): Coeficientes del
      denominador
14
15     figure                                % Diagramas de bode juntos
16     hold on;
17     bode(TF_filtro);
18     bode(TFpa);
19     bode(TFpb);
20     hold off;
21

```

Obteniéndose como resultado las dos funciones transferencias, el filtro pasa bajo y filtro pasa alto, la suma resulta el pasa banda que estamos buscando:

```

TF_filtro =

              1.642e07 s^2
-----
s^4 + 5080 s^3 + 9.586e07 s^2 + 2.006e11 s + 1.559e15

Continuous-time transfer function.
TFpb =

      1.642e07
-----
s^2 + 3184 s + 6.632e07

Continuous-time transfer function.
TFpa =

      s^2
-----
s^2 + 1896 s + 2.35e07

Continuous-time transfer function.

```

Figura 1: Funciones de transferencias: Global, FPB, FPA

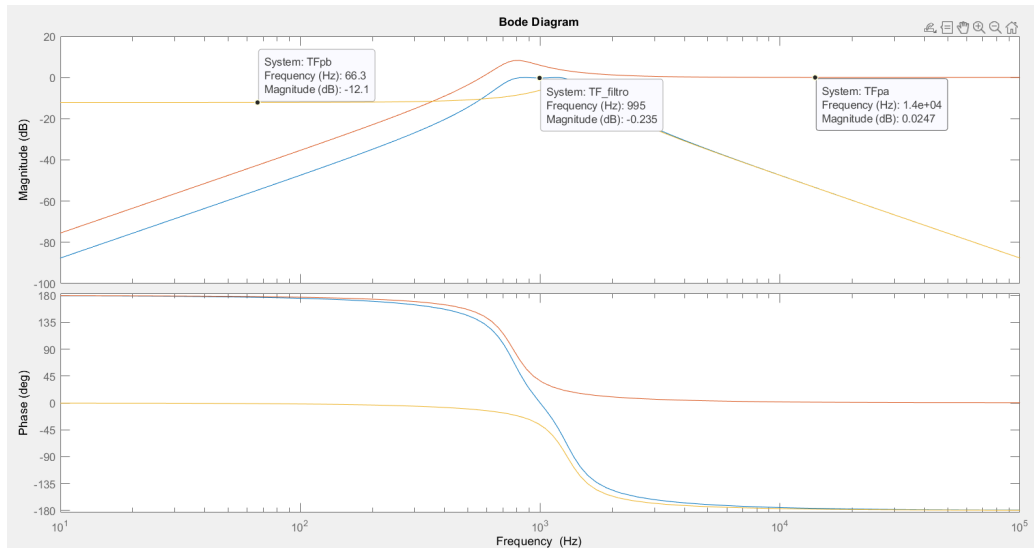


Figura 2: Diagrama de bode: Global,FPB,FPA

2.2. Síntesis del filtro

A continuación se compara con las expresiones canónicas de los filtros para calcular los coeficientes, esto nos permite obtener los valores de resistencias y capacitancias

```

1  Filtro pasa bajo:
2  [num_pb,den_pb] = tfdata(TFpb); %
   Obtencion numerador y denominador de la TFpb
3  num_pb = cell2mat(num_pb);
4  num_pb = num_pb(3);
5  den_pb = cell2mat(den_pb);
6  [Kpb,Wpb,Qpb] = obt_coef(num_pb,den_pb); %
   Obtencion de K, W y Q
7
8  % Tomamos R1 = R2 = 10000
9  syms C1pb C2pb;
10 R1pb = 10000;
   % Suponemos R1 para el FPB
11 R2pb = 10000;
   % Suponemos R2 para el FPB
12 eq1 = 1/(sqrt(C1pb*C2pb*R1pb*R2pb)) == Wpb;
   % Describimos la EC1
13 eq2 = sqrt(C1pb/C2pb)*(sqrt(R1pb*R2pb)/(R1pb+R2pb)) ==
   Qpb; % Describimos la EC2

```

```

14     solu1 = solve([eq1,eq2],[C1pb,C2pb]);
        % Obtenemos las soluciones de las ECs
15
16     C1_pb = double(solu1.C1pb(2));
        % Valor de C1
17     C2_pb = double(solu1.C2pb(2));
        % Valor de C2
18
19     % El K lo hacemos con un divisor resistivo
20
21     Filtro pasa alto:
22     [num_pa,den_pa] = tfdata(TFpa);
23     num_pa = cell2mat(num_pa);
24     num_pa = num_pa(1);
25     den_pa = cell2mat(den_pa);
26     [Kpa,Wpa,Qpa] = obt_coef2(num_pa,den_pa);
27
28
29     syms R1pa R2pa;
30     C1pa = 0.0000001;
31     C2pa = 0.0000001;
32     eq3 = 1/(sqrt(C1pa*C2pa*R1pa*R2pa)) == Wpa;
33     eq4 = 1/((sqrt(R1pa/R2pa)*((C1pa+C2pa)/sqrt(C1pa*C2pa)))
        +((1-Kpa)*sqrt((R2pa*C2pa)/(R1pa*C1pa)))) == Qpa;
34     solu2 = solve([eq3,eq4],[R1pa,R2pa]);
35
36     R1_pa = double(solu2.R1pa(2));
37     R2_pa = double(solu2.R2pa(2));
38
39

```

Como resultado del análisis tenemos que los valores de R y C para el filtro pasa bajo son $R1 = R2 = 10[Kohm]$ y los valores de capacidad $C1 = 62,8[nF]$ $C2 = 2,4[nF]$. Para el filtro pasa alto tenemos $C1 = C2 = 10[uF]$ y para las resistencias $R1 = 400[ohm]$ $R2 = 10[Kohm]$. Por ultimo para la síntesis del filtro se eligió la siguiente configuración:

Por ultimo se verifican los resultados de modulo y fase con una simulación en LTSPICE

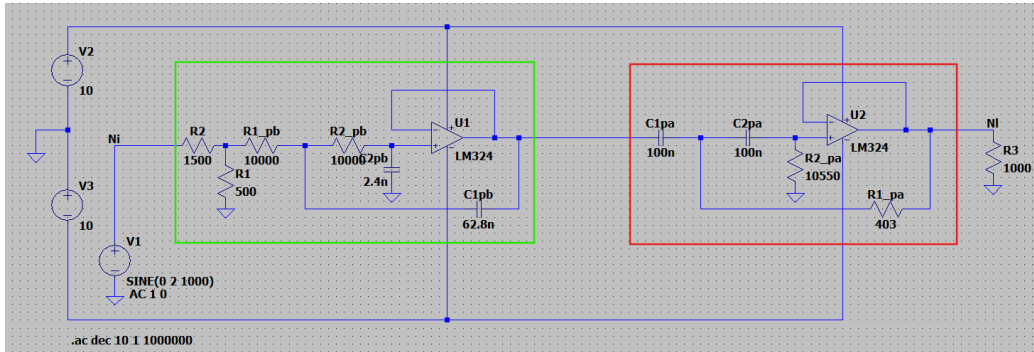


Figura 3: Síntesis del filtro pasa-banda FPB+FPA

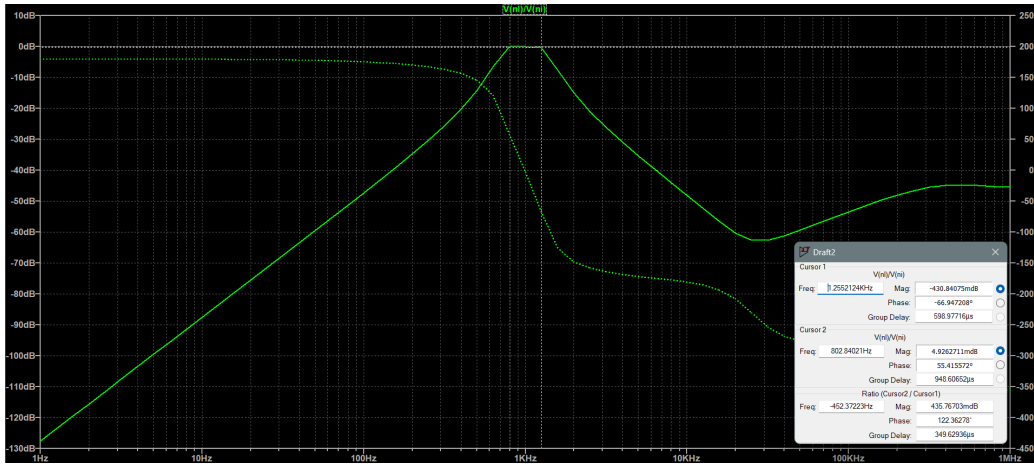


Figura 4: Respuesta en frecuencia del filtro

2.3. Análisis de sensibilidad

El análisis de sensibilidad en el diseño de filtros activos es crucial porque evalúa cómo las variaciones en los parámetros de los componentes afectan el desempeño del circuito. Dado que los filtros activos están contruidos con resistencias, capacitores, y amplificadores operacionales, las imperfecciones o desviaciones de estos elementos pueden alterar las características del filtro, como la frecuencia de corte, el factor de calidad, y la ganancia.

Partimos del análisis de sensibilidad para el filtro pasa bajo, la expresión

general del filtro es:

$$FPB(s) = \frac{K}{s^2 + \frac{wp}{Qp}s + wp^2} \quad (1)$$

La definición de sensibilidad es:

$$S_x^p = \frac{x}{p} \frac{dp}{dx} \quad (2)$$

Donde p es el parámetro y x es la variable. Teniendo en cuenta el circuito anterior mente presentados, podemos calcular la sensibilidad para wp y wp/Qp con el siguiente script de matlab:

```

1  %%Filtro pasa bajo
2  % Declarar las variables simbolicas
3  syms R1 R2 C1 C2
4  % Definir la ecuacion para wp
5  wp = sqrt(1 / (R1 * R2 * C1 * C2))
6  % Sensibilidades de wp respecto a R1, R2, C1 y C2
7  S_R1_wp = (R1 / wp) * diff(wp, R1);
8  S_R2_wp = (R2 / wp) * diff(wp, R2);
9  S_C1_wp = (C1 / wp) * diff(wp, C1);
10 S_C2_wp = (C2 / wp) * diff(wp, C2);
11 % Simplificar las sensibilidades
12 S_R1_wp = simplify(S_R1_wp);
13 S_R2_wp = simplify(S_R2_wp);
14 S_C1_wp = simplify(S_C1_wp);
15 S_C2_wp = simplify(S_C2_wp);
16 % Definir la ecuacion para wp/Qp
17 wp_Qp = (1 / (R1 * C1)) + (1 / (R2 * C1))
18 % Sensibilidades de wp/Qp respecto a R1, R2, C1 y C2
19 S_R1_wp_Qp = (R1 / wp_Qp) * diff(wp_Qp, R1);
20 S_R2_wp_Qp = (R2 / wp_Qp) * diff(wp_Qp, R2);
21 S_C1_wp_Qp = (C1 / wp_Qp) * diff(wp_Qp, C1);
22 S_C2_wp_Qp = (C2 / wp_Qp) * diff(wp_Qp, C2);
23 % Simplificar las sensibilidades
24 S_R1_wp_Qp = simplify(S_R1_wp_Qp);
25 S_R2_wp_Qp = simplify(S_R2_wp_Qp);
26 S_C1_wp_Qp = simplify(S_C1_wp_Qp);
27 S_C2_wp_Qp = simplify(S_C2_wp_Qp);
28 % Valores especificos
29 R1_val = 10e3; % 10 kOhm
30 R2_val = 10e3; % 10 kOhm
31 C1_val = 62.8e-9; % 62.8 nF
32 C2_val = 2.4e-9; % 2.4 nF
33 % Evaluar wp y wp/Qp numericamente

```

```

34     wp_val = sqrt(1 / (R1_val * R2_val * C1_val * C2_val));
35     wp_Qp_val = (1 / (R1_val * C1_val)) + (1 / (R2_val *
36     C1_val));
37     % Evaluar sensibilidades numericas de wp
38     S_R1_wp_val = subs(S_R1_wp, [R1, R2, C1, C2], [R1_val,
39     R2_val, C1_val, C2_val]);
40     S_R2_wp_val = subs(S_R2_wp, [R1, R2, C1, C2], [R1_val,
41     R2_val, C1_val, C2_val]);
42     S_C1_wp_val = subs(S_C1_wp, [R1, R2, C1, C2], [R1_val,
43     R2_val, C1_val, C2_val]);
44     S_C2_wp_val = subs(S_C2_wp, [R1, R2, C1, C2], [R1_val,
45     R2_val, C1_val, C2_val]);
46     % Evaluar sensibilidades numericas de wp/Qp
47     S_R1_wp_Qp_val = subs(S_R1_wp_Qp, [R1, R2, C1, C2], [
48     R1_val, R2_val, C1_val, C2_val]);
49     S_R2_wp_Qp_val = subs(S_R2_wp_Qp, [R1, R2, C1, C2], [
50     R1_val, R2_val, C1_val, C2_val]);
51     S_C1_wp_Qp_val = subs(S_C1_wp_Qp, [R1, R2, C1, C2], [
52     R1_val, R2_val, C1_val, C2_val]);
53     S_C2_wp_Qp_val = subs(S_C2_wp_Qp, [R1, R2, C1, C2], [
54     R1_val, R2_val, C1_val, C2_val]);
55     % Mostrar resultados numericos
56     fprintf('\nSensibilidades numericas de wp:\n');
57     fprintf('S_R1_wp = %.2f\n', double(S_R1_wp_val));
58     fprintf('S_R2_wp = %.2f\n', double(S_R2_wp_val));
59     fprintf('S_C1_wp = %.2f\n', double(S_C1_wp_val));
60     fprintf('S_C2_wp = %.2f\n', double(S_C2_wp_val));
61     fprintf('\nSensibilidades numericas de wp/Qp:\n');
62     fprintf('S_R1_wp_Qp = %.2f\n', double(S_R1_wp_Qp_val));
63     fprintf('S_R2_wp_Qp = %.2f\n', double(S_R2_wp_Qp_val));
64     fprintf('S_C1_wp_Qp = %.2f\n', double(S_C1_wp_Qp_val));
65     fprintf('S_C2_wp_Qp = %.2f\n', double(S_C2_wp_Qp_val));

```

Obteniéndose como resultado:

Realizando un estudio similar al anterior pero esta vez teniendo en cuenta la síntesis del filtro pasa alto, realizamos el análisis de sensibilidad:

```

1     %%Filtro pasa alto:
2     % Declarar las variables simbolicas
3     syms R1 R2 C1 C2 k
4
5     % Definir la ecuacion para wp
6     wp = sqrt(1 / (R1 * R2 * C1 * C2))
7
8     % Sensibilidades de wp respecto a R1, R2, C1 y C2

```



```

wp =

$$\sqrt{\frac{1}{C_1 C_2 R_1 R_2}}$$

wp_Qp =

$$\frac{1}{C_1 R_1} + \frac{1}{C_1 R_2}$$

Sensibilidades numéricas de wp:
S_R1_wp = -0.50
S_R2_wp = -0.50
S_C1_wp = -0.50
S_C2_wp = -0.50
Sensibilidades numéricas de wp/Qp:
S_R1_wp_Qp = -0.50
S_R2_wp_Qp = -0.50
S_C1_wp_Qp = -1.00
S_C2_wp_Qp = 0.00

```

Figura 5: Sensitividad filtro pasa bajo

```

9      S_R1_wp = (R1 / wp) * diff(wp, R1);
10     S_R2_wp = (R2 / wp) * diff(wp, R2);
11     S_C1_wp = (C1 / wp) * diff(wp, C1);
12     S_C2_wp = (C2 / wp) * diff(wp, C2);
13
14     % Simplificar las sensibilidades
15     S_R1_wp = simplify(S_R1_wp);
16     S_R2_wp = simplify(S_R2_wp);
17     S_C1_wp = simplify(S_C1_wp);
18     S_C2_wp = simplify(S_C2_wp);
19
20     % Definir la ecuacion para wp/Qp con k = 1
21     wp_Qp = -((k - 1) * sqrt((C2 * R2) / (C1 * R1)) - sqrt(R1
22 / R2) * (C1 + C2)) / sqrt(C1 * C2 * R1 * R2)
23
24     % Sustituir k = 1 en la expresion
25     wp_Qp = subs(wp_Qp, k, 1);
26
27     % Sensibilidades de wp/Qp respecto a R1, R2, C1 y C2
28     S_R1_wp_Qp = (R1 / wp_Qp) * diff(wp_Qp, R1);
29     S_R2_wp_Qp = (R2 / wp_Qp) * diff(wp_Qp, R2);
30     S_C1_wp_Qp = (C1 / wp_Qp) * diff(wp_Qp, C1);
31     S_C2_wp_Qp = (C2 / wp_Qp) * diff(wp_Qp, C2);

```

```

30     S_C2_wp_Qp = (C2 / wp_Qp) * diff(wp_Qp, C2);
31
32     % Simplificar las sensibilidades
33     S_R1_wp_Qp = simplify(S_R1_wp_Qp);
34     S_R2_wp_Qp = simplify(S_R2_wp_Qp);
35     S_C1_wp_Qp = simplify(S_C1_wp_Qp);
36     S_C2_wp_Qp = simplify(S_C2_wp_Qp);
37
38     % Nuevos valores especificos
39     R1_val = 400; % 400 Ohm
40     R2_val = 10e3; % 10 kOhm
41     C1_val = 10e-6; % 10 uF
42     C2_val = 10e-6; % 10 uF
43
44     % Evaluar wp y wp/Qp numericamente
45     wp_val = sqrt(1 / (R1_val * R2_val * C1_val * C2_val));
46     wp_Qp_val = subs(wp_Qp, [R1, R2, C1, C2], [R1_val, R2_val,
47     C1_val, C2_val]);
48
49     % Evaluar sensibilidades numericas de wp
50     S_R1_wp_val = subs(S_R1_wp, [R1, R2, C1, C2], [R1_val,
51     R2_val, C1_val, C2_val]);
52     S_R2_wp_val = subs(S_R2_wp, [R1, R2, C1, C2], [R1_val,
53     R2_val, C1_val, C2_val]);
54     S_C1_wp_val = subs(S_C1_wp, [R1, R2, C1, C2], [R1_val,
55     R2_val, C1_val, C2_val]);
56     S_C2_wp_val = subs(S_C2_wp, [R1, R2, C1, C2], [R1_val,
57     R2_val, C1_val, C2_val]);
58
59     % Evaluar sensibilidades numericas de wp/Qp
60     S_R1_wp_Qp_val = subs(S_R1_wp_Qp, [R1, R2, C1, C2], [
61     R1_val, R2_val, C1_val, C2_val]);
62     S_R2_wp_Qp_val = subs(S_R2_wp_Qp, [R1, R2, C1, C2], [
63     R1_val, R2_val, C1_val, C2_val]);
64     S_C1_wp_Qp_val = subs(S_C1_wp_Qp, [R1, R2, C1, C2], [
65     R1_val, R2_val, C1_val, C2_val]);
66     S_C2_wp_Qp_val = subs(S_C2_wp_Qp, [R1, R2, C1, C2], [
67     R1_val, R2_val, C1_val, C2_val]);
68
69     % Mostrar resultados numericos
70     fprintf('\nSensibilidades numericas de wp:\n');
71     fprintf('S_R1_wp = %.2f\n', double(S_R1_wp_val));
72     fprintf('S_R2_wp = %.2f\n', double(S_R2_wp_val));
73     fprintf('S_C1_wp = %.2f\n', double(S_C1_wp_val));
74     fprintf('S_C2_wp = %.2f\n', double(S_C2_wp_val));

```

```

66
67 fprintf('\nSensibilidades numericas de wp/Qp:\n');
68 fprintf('S_R1_wp_Qp = %.2f\n', double(S_R1_wp_Qp_val));
69 fprintf('S_R2_wp_Qp = %.2f\n', double(S_R2_wp_Qp_val));
70 fprintf('S_C1_wp_Qp = %.2f\n', double(S_C1_wp_Qp_val));
71 fprintf('S_C2_wp_Qp = %.2f\n', double(S_C2_wp_Qp_val));
72

```

Los resultados obtenidos son los siguientes:

$$\begin{aligned}
 wp &= \sqrt{\frac{1}{C_1 C_2 R_1 R_2}} \\
 wp_{Qp} &= \frac{\sqrt{\frac{R_1}{R_2}} (C_1 + C_2) - (k-1) \sqrt{\frac{C_2 R_2}{C_1 R_1}}}{\sqrt{C_1 C_2 R_1 R_2}}
 \end{aligned}$$

Sensibilidades numéricas de wp:

S_R1_wp = -0.50
S_R2_wp = -0.50
S_C1_wp = -0.50
S_C2_wp = -0.50

Sensibilidades numéricas de wp/Qp:

S_R1_wp_Qp = 0.00
S_R2_wp_Qp = -1.00
S_C1_wp_Qp = 0.00
S_C2_wp_Qp = 0.00

Figura 6: Sensitividad filtro pasa alto

2.4. Simulación de Montecarlo

La simulación de Montecarlo es una técnica estadística que utiliza múltiples iteraciones para analizar cómo la variabilidad en los parámetros afecta el rendimiento de un sistema. En el caso de los filtros activos, esta técnica permite evaluar la sensibilidad y robustez del diseño frente a las imperfecciones en los componentes reales, como resistencias y capacitores. Montecarlo

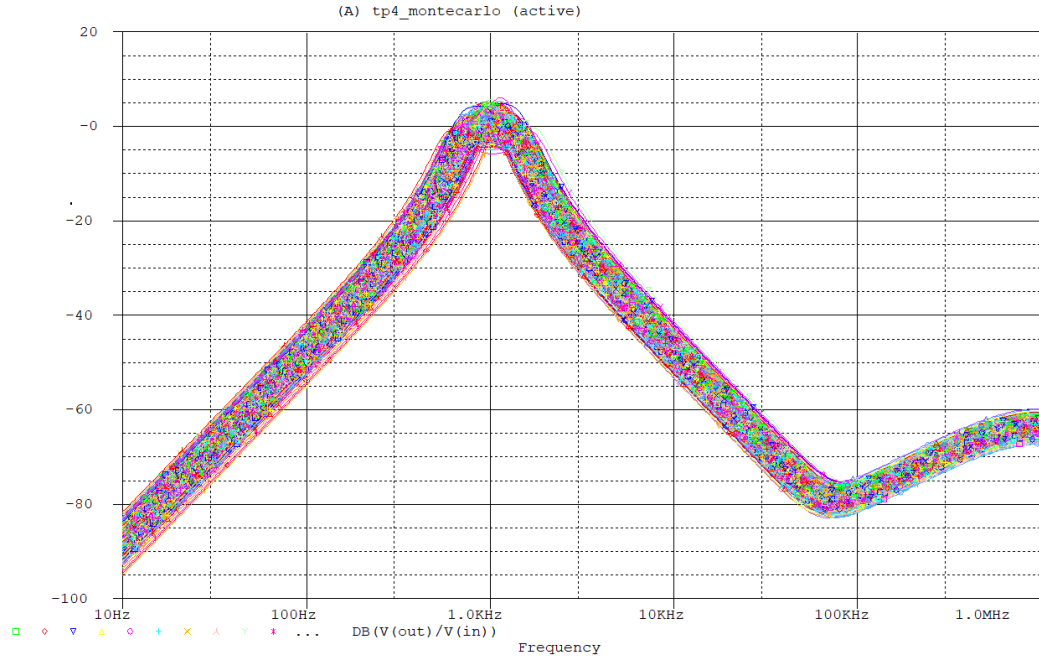


Figura 7: Análisis de Montecarlo

ayuda a simular miles de combinaciones posibles dentro de estas tolerancias y analiza el impacto de estas variaciones en parámetros clave del filtro, como la frecuencia de corte (ω_p), el factor de calidad (Q_p), o la ganancia.

En la simulación corremos 1000 veces sobre la variable $V(out)$ (distribución gaussiana) todos los componentes tienen una tolerancia del diez por ciento, con esto obtenemos los datos para el análisis.

Como tenemos un filtro pasa banda nos interesa hacer un análisis de rendimiento en la banda de paso, para eso utilizamos la siguiente función *Bandwidth3dB()*: Encuentre la diferencia entre los valores X donde la traza cruza primero su valor máximo menos 3 dB ($Y_{max}-3dB$) con una pendiente positiva y luego con una pendiente negativa. (es decir, encuentre el ancho de banda de 3 dB de una señal).

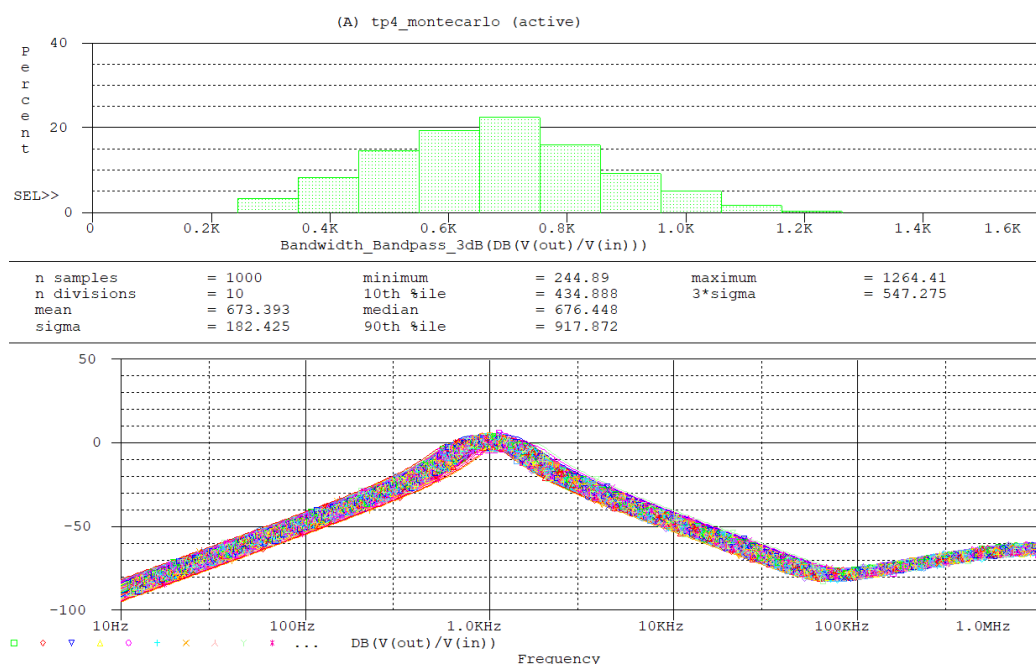


Figura 8: Análisis de banda de paso con Montecarlo

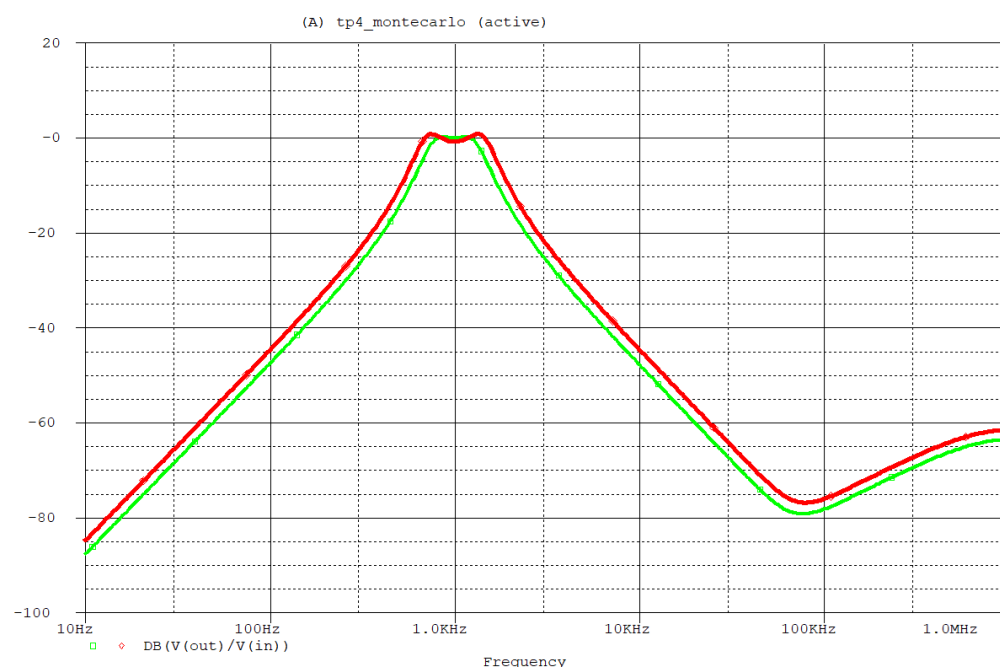


Figura 9: Análisis para el peor de los casos