UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Introdução a Microeletrônica

Relatório Otimização aplicada a Estágio Fonte Comum

Matheus Francisco Batista Machado

Professor:

Tiago Oliveira Webera

1 Introdução

O trabalho realizado tem como objetivo compreender o funcionamento e utilização de otimização aplicada acircuitos integrados analógicos,com simulações realizadas no ltspice e octave.

2 Simulação de um transístor do tipo NMOS

Foi utilizado um modelo $N1u$ modelo de canal longo, realizado o circuito da figura um com os valores $Vdd = 5V$, $R_D = 25k\Omega$ e a fonte de tensão entre a porta e a fonte em c.c. é chamada V_{GS} e a fonte de pequenos sinais entre a porta e a fonte é chamada v_{gs}

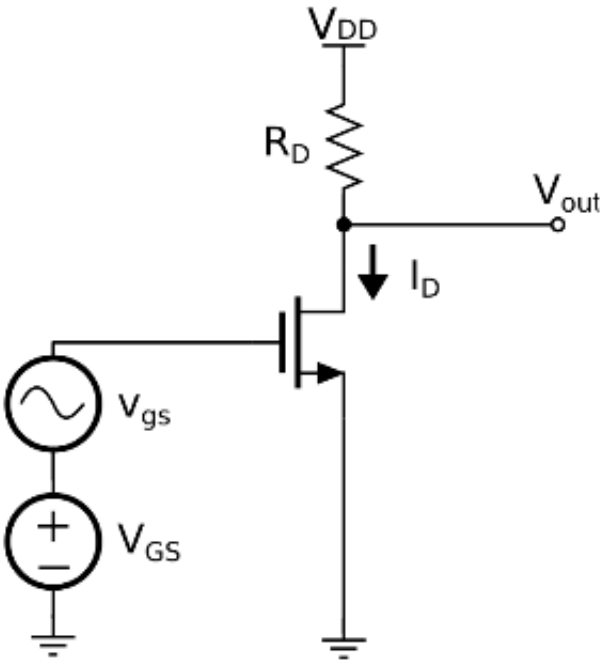

Figure 1: Circuito

3 Parte 1:Simulação do tipo AC

Faça uma simulação do tipo AC de 1 até 1MHz. Considere que:

- $RD = 25k\Omega$;
- O transistor tenha comprimento L (length) igual a $1,5um$ e largura W (width) igual a $3um$.
- a fonte vgs tenha amplitude 1V AC (tensão apenas utilizada para simulações do tipo AC)
- VGS seja 1.5 V.

Os resultados a serem obtidos são o ganho em escala linear e em dB.

Nesta etapa realizamos a montagem do circuito no ltspice, logo em seguida realizamos as simulação em busca de conseguir o ganho do circuito dado pela expressão

$A_v = \frac{V_{out}}{Vin} = -gm * (RD||ro)$,como podemos analisar nas medidas do gráfico o ganho em dB é dado pelo $20log_{10}Av = 7.63662$, para voltar de escala dB para V/V utilizamos a expressão $10^{\frac{gaindB}{20}}$

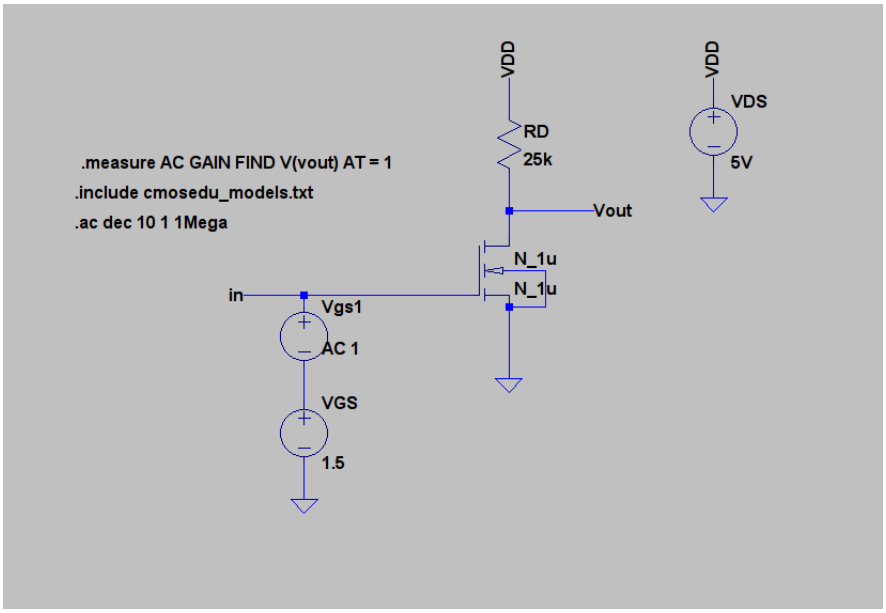

Figure 2: Circuito

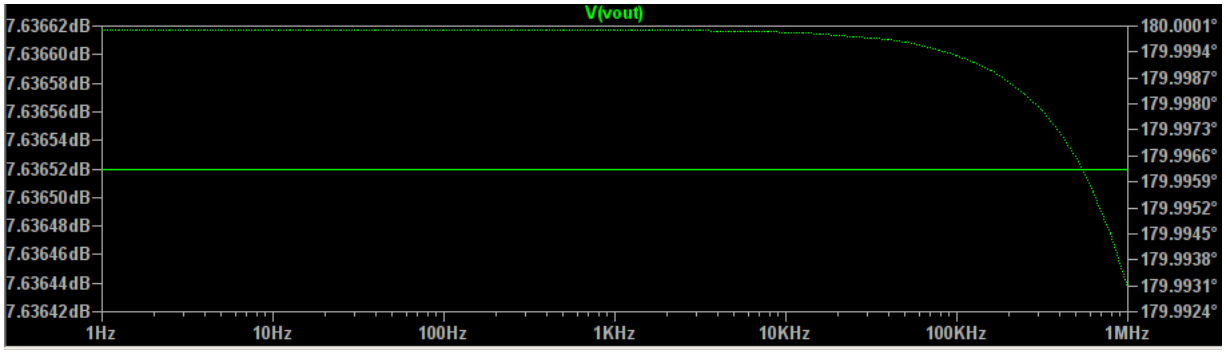

Figure 3: Ganho em dB

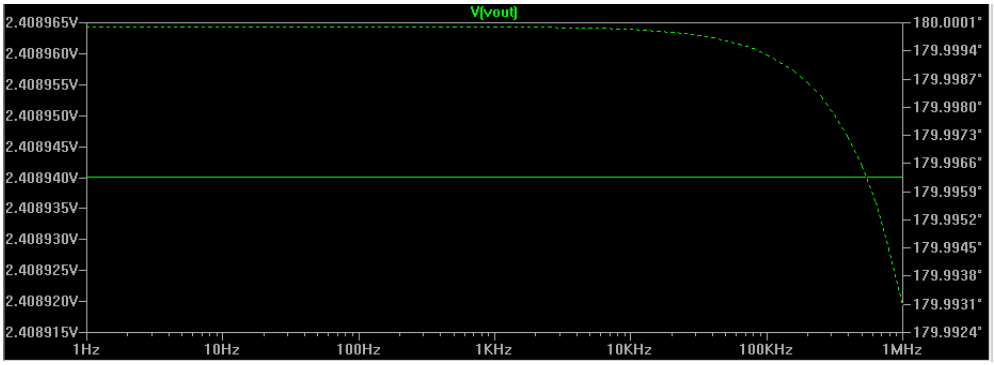

Figure 4: Ganho em V/V

Então agora iremos tentar melhorar o ganho que é de 7,64dB, mudando a resistência, vgs , comprimento e largura do transístor

4 Parte 2

- $Ganho = -gm * (RD||ro)$ (pode ser medido diretamente no simulador elétrico pela razão da saída pela entrada do amplificador)
- Limite de tensão de saída superior $V_{out,max} = V_{DD}$
- Limite de tensão de saída inferior (limitado devido ao transistor ir para região de triodo):$V_{out,min} = V_{GS} - V_{Th}$

Foi utilizado para realizar as medidas de ganho automático .measure AC GAIN FIND V(vout) AT =1. Para as análises .AC, as expressões condicionais de dados complexos são traduzidas para condições reais, convertendo a expressão em sua magnitude. O resultado é a magnitude em dB de V(vout) quando a frequência for igual a 1 logo para convertermos para ganho de tensão utilizamos a expressão matemática $10^{gain/20}$.
Os limites de tensão superior é igual 5V e o limite de tensão inferior é $V_{out,min} = 1.5 - 0.8 = 0.7V$

5 Parte 3- Otimizar o ganho do circuito

- variáveis de entrada:R_D, V_{GS}, WeL.
- valores mínimos das variáveis de entrada (respectivamente) $= 100, 1, 3e-6, 1.5e-6$
- valores máximos das variáveis de entrada (respectivamente) $= 100k, 5, 100e-6, 10e-6$
- Medições: Ganho em baixa frequência (em medida do LTSPICE, medirem freq $= 1$)
- Objetivo: maximizar o ganho
- Como fazer a função custo: quanto maior o ganho, menor o resultado da função. Fica a critério do aluno como fazer esta função
- Critério de parada do algoritmo: a critério do aluno

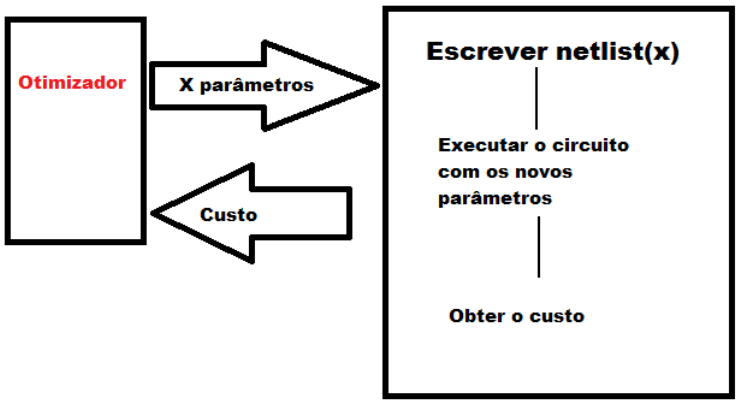

Figure 5: Otimizador

Foi utilizado uma técnica para otimizar o circuito, alterando os valores da entrada será retornado o custo, com isso nosso objetivo é minimizar o custo. Primeiramente foi criado um netlist com os valores de entrada para o otimizador, esses valores são resistência (RD), VGS, largura (L) e comprimento (W) do transístor, após receber os parâmetros é executado o netlist e recolhido um ganho com isso é calculado o novo custo para cada alteração de vetor de parâmetro x. A função custo utilizada foi$y = (meas - objetivo)^2$, meas é a medida do ganho o objetivo foi definido como 100dB logo quando mais proximo de 100dB o valor da função custo ira ser menor, foi utilizado funções custo do tipo $y = 1 - objetivo/meas$, porém havia a necessidade de converter o meas de dB

para Volt/V, assim em ambos os teste podemos chegar a um ganho máximo de 31.07dB. Foi utilizado um algoritmo de Hill Climbing é uma técnica simples de busca local (não armazena o caminho percorrido até a solução corrente e sim a solução propriamente dita como estado), relacionada ao método do gradiente, que não requer que seja conhecido o gradiente ou sua direção - novos candidatos são gerados posteriormente na região da solução atual. O algoritmo inicia com uma solução randômica, potencialmente ruim, e iterativamente efetua pequenas modificações nesta solução, buscando melhorias no resultado da função objetivo. O algoritmo termina quando não encontra nenhuma melhoria possível em uma iteração. A solução, ao término do algoritmo, é idealmente ótima, mas não há qualquer garantia de otimalidade. A função montada hill climbing foi fornecida e alterada apenas os limites inferiores e superiores, assim o algoritmo ira sempre gerar novos valores aleatórios para aquela faixa de valores com um máximo de iterações.

5.1 MATLAB/Octave

```
1  % Cost function
2  % Esta funcao ira receber um vetor onde sao os valores de
       entrada respectivamente RD, VGS, W, L em seguida ira
       escrever um netlist, complilar o netlist no simulador
       ltspice e ler o log buscando o ganho quando a freq = 1
3  function [y gain] = cost_function(x)
4          printf('\n');
5
6          write_netlist(x);
7
8          run_netlist();
9
10         meas = read_log;
11         objetivo =100;
12         fprintf(1,'\n Ganho: %.4fdB \n',meas);
13
14     y = (meas-objetivo)^2;
15         %fprintf(1,'\n Custo: %.4f \n',y);
16          gain = meas;
17
18 endfunction
19 % Funcao que escreve o netlist do circuito substituindo o
       parametros
20 function[retval] = write_netlist(x)
21
22
23 fid =fopen( 'generated_netlist.cir','w');
24
```

```
25 fprintf(fid,'*Netlist_teste\n');
26
27 fprintf(fid,'Vgs1 in N001 AC 1 \n');
28 fprintf(fid,'VGS N001 0 %.4f \n',x(2));
29 fprintf(fid,'MSN_1u Vout in 0 0 N_1u l=%.4fu w=%.4fu \n',x(4),
     x(3));
30 fprintf(fid,'RD VDD Vout %.4f \n',x(1));
31 fprintf(fid,'VDS VDD 0 5V \n');
32 fprintf(fid,'.model NMOS NMOS \n');
33 fprintf(fid,'.model PMOS PMOS \n');
34 fprintf(fid,'.lib C:\\Program Files (x86)\\LTC\\LTspiceIV\\lib
     \\cmp\\standard.mos \n');
35 fprintf(fid,'.include cmosedu_models.txt \n');
36 fprintf(fid,'.ac dec 10 1 1Mega\n');
37 fprintf(fid,'.measure AC GAIN FIND V(vout) AT = 1 \n');
38 fprintf(fid,'.backanno \n');
39 fprintf(fid,'.end \n');
40
41
42 fclose(fid);
43
44 endfunction
45 %
46 %
47 % Funcao para otimizar o circuito
48 %
49 %
50
51 function [xoptim ,fval] = hill_climbing(iterations)
52 %=========================================
53 %+++++ Minimos valores ++++++++++++++++++
54 %=========================================
55 RD_lb = 100;
56 Vgs_lb =1;
57 W_lb = 3;
58 L_lb = 1.5;
59 %=========================================
60 %+++++ Maximos valores ++++++++++++++++++
61 %=========================================
62 RD_ub = 100000;
63 Vgs_ub = 5;
64 W_ub = 100;
65 L_ub=10;
66 %=========================================
67 %=========================================
68 %+ vetor de  valores minimos e maximos +
69 %=========================================
70 lb =[RD_lb Vgs_lb W_lb L_lb];
```

```
71  ub = [RD_ub Vgs_ub W_ub L_ub];
72  %=========================================
73  %+ Iniciando vetores aleatorios ++++++++
74  %=========================================
75
76  RD  = RD_lb + (RD_ub - RD_lb)*rand;
77  Vgs = Vgs_lb + (Vgs_ub - Vgs_lb)*rand;
78  W   = W_lb + (W_ub - W_lb)*rand;
79  L   = L_lb + (L_ub - L_lb)*rand;
80  %=========================================
81  %lim = [RD_lb,RD_ub,Vgs_lb,Vgs_ub,W_lb,W_ub,L_lb,L_ub];
82
83  x = [RD,Vgs,W,L];
84  %=========================================
85
86  % Calculando o primeiro custo
87  [custo_atual,gain] = cost_function(x);
88
89  printf('\t Primeiro custo: %f \t Ganho Atual: %f \t\n',
        custo_atual,gain);
90  y_history_cost = zeros(1,iterations);
91
92  for i=1:iterations
93          % Valor de convergencia da exponencial
94      conv = 3*i/iterations;
95
96      % cria um vetor com zero
97      x_modificado = zeros(1,numel(x));
98      % Modificar os parametros para verificar o novo custo
99      for k=1:numel(x)
100         x(k) = x(k)+ exp(-conv)*randn*(ub(k) - lb(k));
101          while (x(k) < lb(k) || x(k) > ub(k))
102              x(k) = x(k) + exp(-conv)*randn*(ub(k) - lb(k));
103          endwhile
104         x_modificado(k) = x(k);
105     endfor
106     % Verifica o novo custo e o novo ganho, nosso objetivo e
        minimzar o custo e maximizar o ganho
107         [custo_novo,gain] = cost_function(x_modificado);
108     fprintf(1,'iteration: %d \t Custo Novo %f \t Novo Ganho: %
        f \t\n',i,custo_novo,gain);
109     y_history_cost(i)= custo_novo;
110     fflush(stdout);
111     if (custo_novo < custo_atual)
112        bestGain = gain;
113        custo_atual = custo_novo;
114        best_X = x_modificado;
115        x = x_modificado;
```

```
116
117         endif
118         fprintf(1,'\n *** Melhor Custo: %f *** \t',custo_atual);
119
120 endfor
121 plot(y_history_cost,'.');
122 hold on;
123 grid;
124 xoptim = best_X;
125 fval = custo_atual;
126
127 fprintf(1,'\n= Fim de Simulacao =\n Ganho:%.2fdB \n com RD
      =%.2f VGS=%.2f W=%.2f L=%.2f\n',bestGain, xoptim(1),
      xoptim(2),xoptim(3),xoptim(4))
128
129 endfunction
```

O algoritmo utilizado é conhecido como hill climbing, foi realizado 1000 iteração para uma determinada faixa de valores, assim conseguindo obter em escala decibéis $29.12dB$ com isso tivemos um maior ganho e um menor custo. Os parâmetros encontrados foi $RD = 96813.97\Omega$ $V_{GS} = 1.10$, $W = 70.60u$, $L = 7.03u$

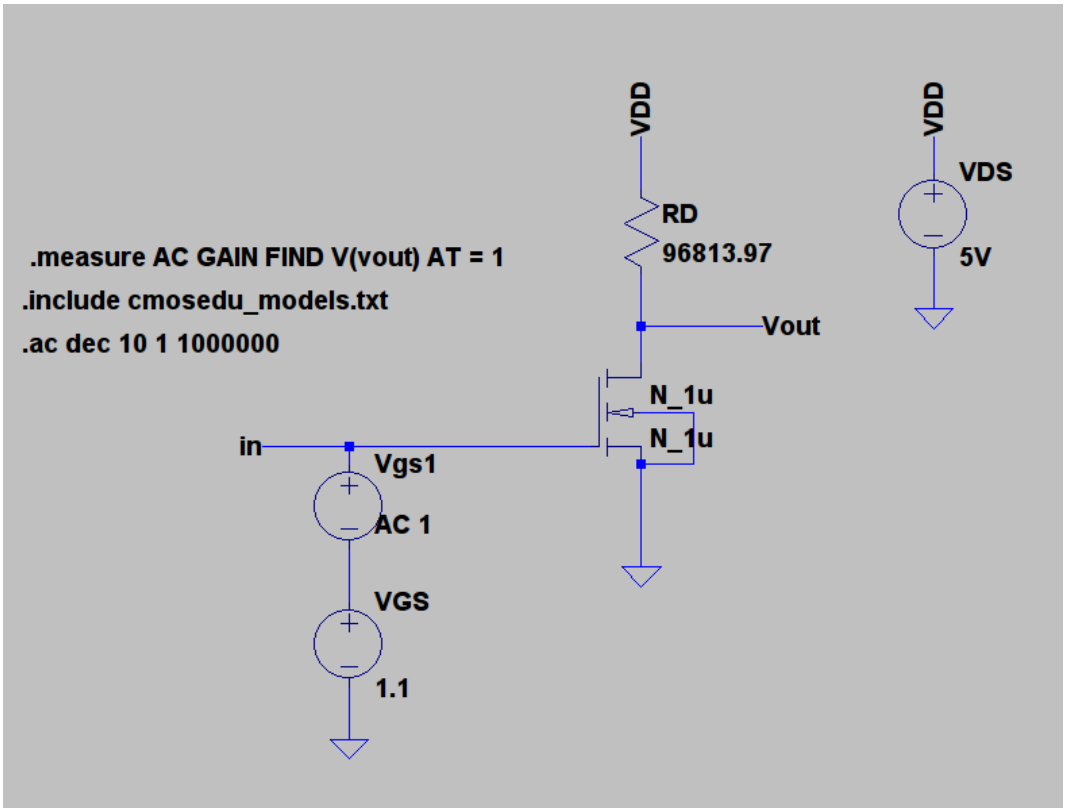

Figure 6: Circuito otimizado

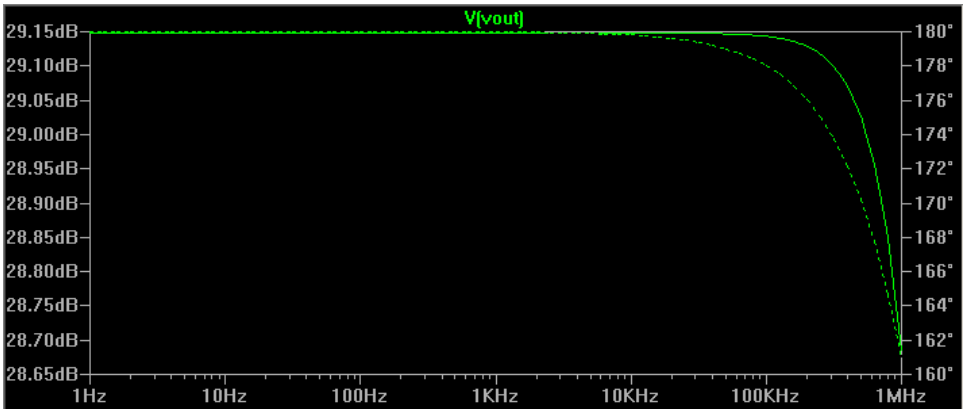

Figure 7: Gráfico de ganho em dB

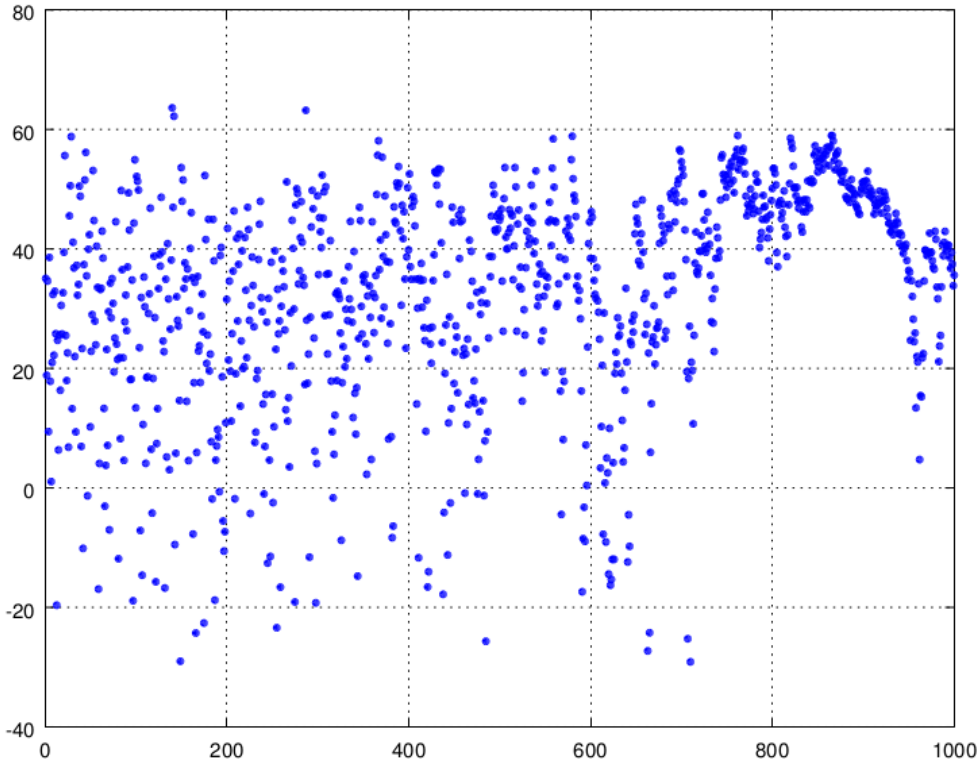

Figure 8: Gráfico de custos

6 Parte 4: Otimizar o ganho e a excursão de saída

- Medições: Ganho em baixa frequência (em medida do LTSPICE, medirem freq = 1) e Excursão Máxima de Saída.

- Objetivo: maximizar o ganho e a excursão de saída;

- Como fazer a função custo: uma função agregativa. Supondo que a função custo

do ganho seja f1 e a função custo da excursão de saída seja f2: $f(x) = f1(x)\Delta w1 + f2(x)\Delta w2$

Nesta parte buscaremos simular para conseguir minimizar a excursão de saída juntamente maximizar o ganho, ou seja temos para a excursão de saída $exc = V_{DD} - (V_{GS} - V_{TH})$, ou seja a excursão é a diferença máxima entre os dois picos do sinal que ira garantir que o transístor esteja na zona de saturação.

6.1 MATLAB/Octave

```
% Cost function
% Esta funcao ira receber um vetor onde sao os valores de entrada
% respectivamente RD, VGS, W, L em seguida ira escrever um netlist, complilar o
% netlist no simulador ltspice e ler o log buscando o ganho quando a freq = 1
function [y gain exc] = cost_function(x)
  printf('\n');
  vgs = x(2);
  vdd =5;
  write_netlist(x);

  run_netlist();

  meas = read_log;
  objetivo = 100;
  obj=2;
  fprintf(1,'\n Ganho: %.4fdB \n',meas);
  exc = vdd -(vgs-0.8);
  y = (0.5*(meas-objetivo)^2 + 0.5*(exc-obj)^2);
  %fprintf(1,'\n Custo: %.4f \n',y);
  gain = meas;

endfunction


% Funcao para otimizar o circuito
%
%
function [xoptim ,fval, excotim] = hill_climbing(iterations)
%=================================
%+++++ Minimos valores +++++++++++++++++
%=================================
RD_lb = 100;
Vgs_lb =1;
W_lb = 3;
```

```
L_lb = 1.5;
%=========================================
%+++++ Maximos valores ++++++++++++++++++
%=========================================
RD_ub = 100000;
Vgs_ub = 5;
W_ub = 100;
L_ub=10;
%=========================================
%=========================================
%+ vetor de  valores minimos e maximos +
%=========================================
lb =[RD_lb Vgs_lb W_lb L_lb];
ub = [RD_ub Vgs_ub W_ub L_ub];
%=========================================
%+ Iniciando vetores aleatorios ++++++++
%=========================================

RD  = RD_lb + (RD_ub - RD_lb)*rand;
Vgs = Vgs_lb + (Vgs_ub - Vgs_lb)*rand;
W   = W_lb + (W_ub - W_lb)*rand;
L   = L_lb + (L_ub - L_lb)*rand;
%=========================================
%lim = [RD_lb,RD_ub,Vgs_lb,Vgs_ub,W_lb,W_ub,L_lb,L_ub];

x = [RD,Vgs,W,L];
%=========================================

% Calculando o primeiro custo
[custo_atual,gain, exc] = cost_function(x);

printf('\t Primeiro custo: %f \t Ganho Atual: %f \t Excursao
   saida: %.2f\n',custo_atual,gain,exc);
y_history_cost = zeros(1,iterations);

for i=1:iterations
    conv = 3*i/iterations;
    x_modificado = zeros(1,numel(x));

    for k=1:numel(x)
        x(k) = x(k)+ exp(-conv)*randn*(ub(k) - lb(k));
         while (x(k) < lb(k) || x(k) > ub(k))
             x(k) = x(k) + exp(-conv)*randn*(ub(k) - lb(k));
         endwhile
        x_modificado(k) = x(k);
    endfor

```

```
81              [custo_novo,gain, exc] = cost_function(x_modificado)
                    ;
82      fprintf(1,'iteration: %d \t Custo Novo %f \t Novo Ganho: %
           f \t Nova Excursao de Saida: %f\n',i,custo_novo,gain,
           exc);
83      y_history_cost(i)= custo_novo;
84      fflush(stdout);
85      if (custo_novo < custo_atual)
86          bestGain = gain;
87          custo_atual = custo_novo;
88          best_X = x_modificado;
89          x = x_modificado;
90          otima = exc;
91      endif
92      fprintf(1,'\n *** Melhor Custo: %f *** \t',custo_atual);
93
94  endfor
95  plot(y_history_cost,'.');
96  hold on;
97  grid;
98  xoptim = best_X;
99  fval = custo_atual;
100 excotim = otima;
101
102 fprintf(1,'\n *** Excursao de saida: %f *** \t',excotim);
103
104 fprintf(1,'\n= Fim de Simulacao =\n Ganho:%.2fdB \n com  RD
       =%.2f   VGS=%.2f   W=%.2f   L=%.2f\n',bestGain, xoptim(1),
       xoptim(2),xoptim(3),xoptim(4))
105
106 endfunction
```

Após realizar a simulação do algoritmo de hill climbing com a excursão de saída temos que o nosso melhor *Ganho* : $28.75dB$, com RD=92977.28 VGS=1.01 W=79.65 L=5.97, temos uma excursão de saída 4.7856.

7 Conclusão

Este trabalho capacito para a integração da ferramenta ltspice juntamente com octave, buscando otimizar transístores do tipo NMOS, utilizando algoritmo de hill climbing porém foi apresentando diversos algoritmos. Esta integração pode facilitar na busca por melhores parâmetros para um transístor fazendo com que ele ainda satisfaça as regras para ser utilizado na zona de saturação.