

**EDA/CAD para Nanoelectrónica**

2º Semestre 2020 / 2021

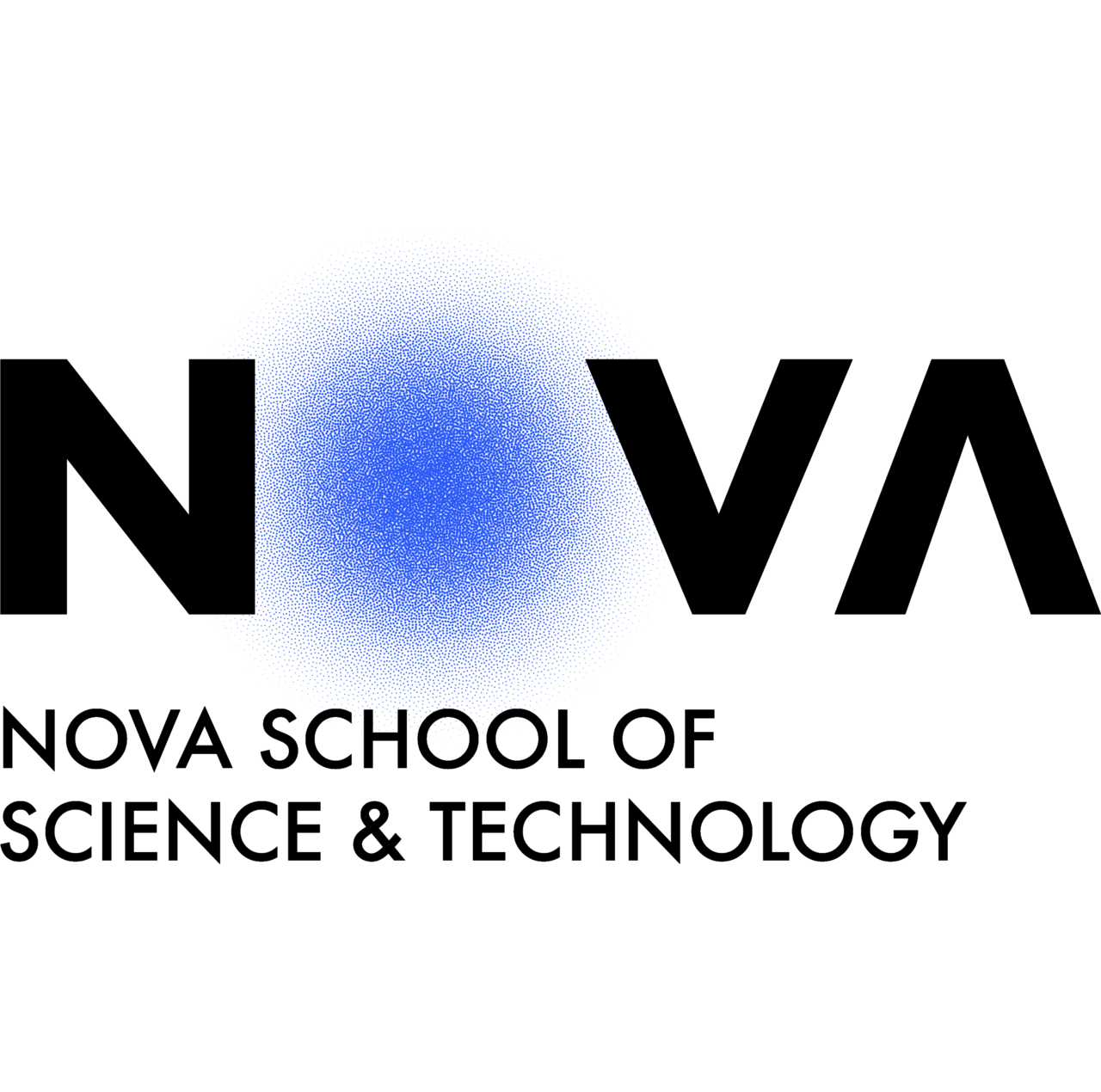
**Relatório**

*Lab#1:* Estudo e Aplicação do Modelo *NPower*

25/04/2021

|  |  |  |
| --- | --- | --- |
| **Número de Aluno** | **Nome** | **Turno Prático** |
| 50726 | Francisco Mendes Micaelo André | P1 |
| 51005 | José Maria Corrêa Arouca Cortes Tamagnini | P1 |
| 43853 | Rafael Gonçalves Feio de Oliveira | P1 |

**Docente:** Maria Helena Fino



Índice

[Introdução 4](#_Toc70171056)

[I-Objetivo 5](#_Toc70171057)

[II - Apresentação do Modelo *NPower* 6](#_Toc70171058)

[2.1. Descrição do Modelo 6](#_Toc70171059)

[III - Metodologia 7](#_Toc70171060)

[3.1. LTspice 7](#_Toc70171061)

[3.2. Python - Spyder 10](#_Toc70171062)

[IV - Resultados 12](#_Toc70171063)

[4.1. Parâmetros Obtidos 12](#_Toc70171064)

[4.2. Analise de Resultados 12](#_Toc70171065)

[V - Conclusões 16](#_Toc70171066)

[Bibliografia 17](#_Toc70171067)

[Anexos 18](#_Toc70171068)

[Trabalho1\_70nm.py 18](#_Toc70171069)

[Trabalho1\_700nm.py 21](#_Toc70171070)

Índice de Figuras

[Figura 1 Esquemático para NMOS 700n 7](#_Toc69936746)

[Figura 2 Características em função de , NMOS com W=L=700 nm 7](#_Toc69936747)

[Figura 3-Características de em função de , NMOS com W=L= 70 nm 8](#_Toc69936748)

[Figura 4 - Característica de em função de , NMOS com W=L= 700 nm 8](#_Toc69936749)

[Figura 5 - Característica de em função de , NMOS com W=L= 70 nm 9](#_Toc69936750)

[Figura 6 - Estrutura das dataframes utilizadas para armazenar: a) ; b) 10](#_Toc69936751)

[Figura 7 – Curvas , com fixo para: a) W=L=700 nm; b) W=L=70 nm 12](#_Toc69936752)

[Figura 8 - Curvas , com para: a) W=L=700 nm; b) W=L=70 nm 13](#_Toc69936753)

[Figura 9 - Curvas , com para: a) W=L=700 nm; b) W=L=70 nm 13](#_Toc69936754)

[Figura 10 - Curvas, para: a) W=L=700 nm; b) W=L=70 nm ;c) W=L=700 nm (escala logarítmica) ; d) W=L=70 nm (escala logarítmica) 14](#_Toc69936755)

[Figura 11 - Erro de em relação a , para: a) W=L=700 nm; b) W=L=70 nm 14](#_Toc69936756)

Índice de Tabelas

[Tabela 1 – Transístores utilizados 5](#_Toc69935205)

[Tabela 2 – Parâmetros obtidos 12](#_Toc69935206)

# Introdução

No âmbito da cadeira de EDA/CAD para Nanoelectrónica, é apresentado o primeiro trabalho laboratorial onde é realizado o estudo do modelo *NPower* para a caracterização de transístores MOSFETs de tecnologia nanométrica [1].

Este trabalho está dividido em várias etapas de desenvolvimento, de modo a estudar e comparar o modelo *NPower* com o modelo tradicional de *Shockley,* comumente conhecido como modelo quadrático, dada a variação quadrática da corrente de “dreno”, , com a tensão entre a “gate” e a “source”, , na zona de saturação, verificando quais as limitações de ambos os modelos.

Na primeira parte do projeto é realizada a descrição do modelo *NPower*, que permite determinar os parâmetros de funcionamento do circuito através de equações de variáveis simples, minimizado os erros associados ao cálculo das variáveis de decisão do modelo. Após a descrição do modelo teórico utilizado (*NPower*), são retiradas as características das correntes , dependendo da tensão e , através do software *LTspice*.

Na segunda parte do trabalho é feita a implementação do modelo *NPower*, obtendo as características do modelo, calculando o erro relativo entre os dados teóricos e simulados, utilizando o software *Spyder*. Por último foram retiradas as conclusões da realização do trabalho prático, com base nos dados obtidos através da abordagem teórica utilizada.

# I-Objetivo

Este projeto tem como objetivo a determinação dos parâmetros de funcionamento dos transístores MOSFET de 90nm para uma relação comprimento e largura igual (), verificando quais as limitações do modelo *NPower*. A Tabela 1 apresenta as especificações dos transístores utilizados para aplicação da respetiva metodologia.

Tabela 1 – Transístores utilizados

|  |  |  |  |
| --- | --- | --- | --- |
|  | Modelo |  |  |
| NMOS4 | 90nm\_NMOS\_bulkL70n | 70 | 70 |
| NMOS4 | 90nm\_NMOS\_bulkL700n | 700 | 700 |

O modelo de *Shockley* torna-se uma aproximação imprecisa para descrição do funcionamento de dispositivos MOSFET de dimensões nanométricas. Deste modo, este trabalho apresenta os desafios da utilização do modelo *NPower*.

# II - Apresentação do Modelo *NPower*

2.1. Descrição do Modelo

O modelo *NPower* é descrito pelo sistema equações (1), onde os parâmetros e representam as correntes no dreno e de saturação nos transístores MOSFET. A tensão de saturação no “dreno” é representada por , enquanto e descrevem a tensão de entre a “gate-source” e “dreno-source” respetivamente e a tensão de “threshold”. As seguintes equações dependem das características físicas da tecnologia utilizada, como a largura do canal e o seu comprimento de canal . Os parâmetros K e controlam as características da região linear (tensão linear de saturação), enquanto o e determinam as características da região de saturação. Por último, a variável 𝜆 expressão uma relação entre a condutância do “dreno” na zona linear de saturação.

|  |  |
| --- | --- |
|  | (1) |

Onde:

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |

# III - Metodologia

3.1. LTspice

Na primeira etapa do projeto foi utilizado o programa LTspice para simular o ponto de funcionamento do circuito CMOS apresentado na Figura 1, tendo em conta dois cenários de simulação distintos.

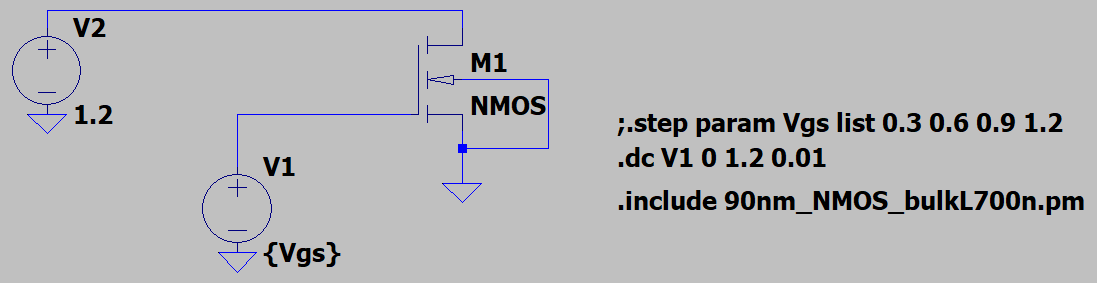


Figura 1 Esquemático para NMOS 700n

O primeiro cenário de simulação corresponde à obtenção da corrente para valores de fixos onde . Assim, recorrendo ao circuito da figura 1, foi utilizado transístor NMOS de 90nm com W=L=700nm obtendo os seguintes resultados:

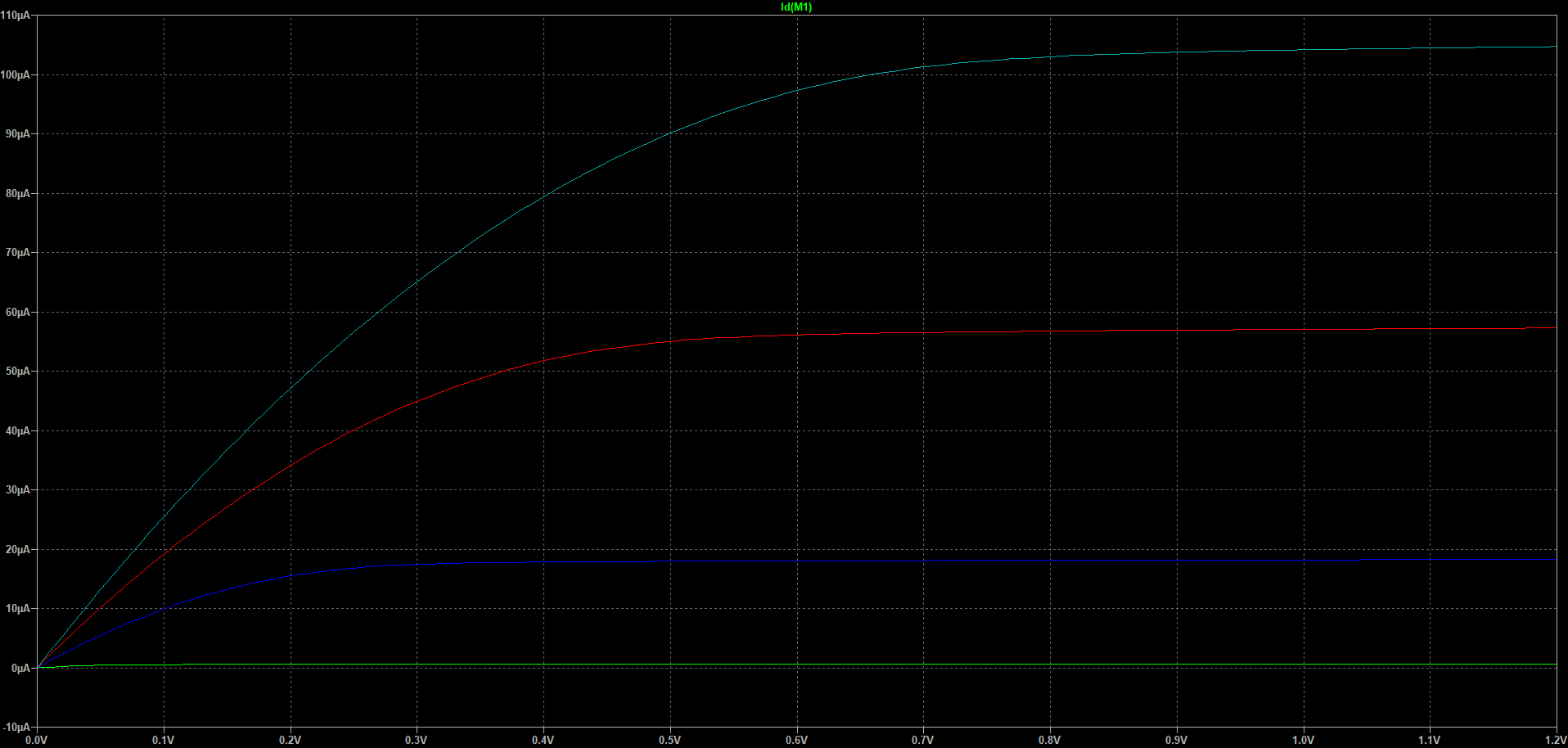


Figura 2 Características em função de , NMOS com W=L=700 nm

Na Figura 2 podemos observar que a característica de tem tendência a com o crescimento de variar entre 0.6µA e 105µA.

De seguida, foi repetida a simulação para um NMOS de 90nm para W=L=70nm. Na Figura 3 observamos que tal como na figura 2, a corrente tende a estabilizar com o aumento de no entanto a amplitude de correntes é menor, variando de 2µA a 67µA

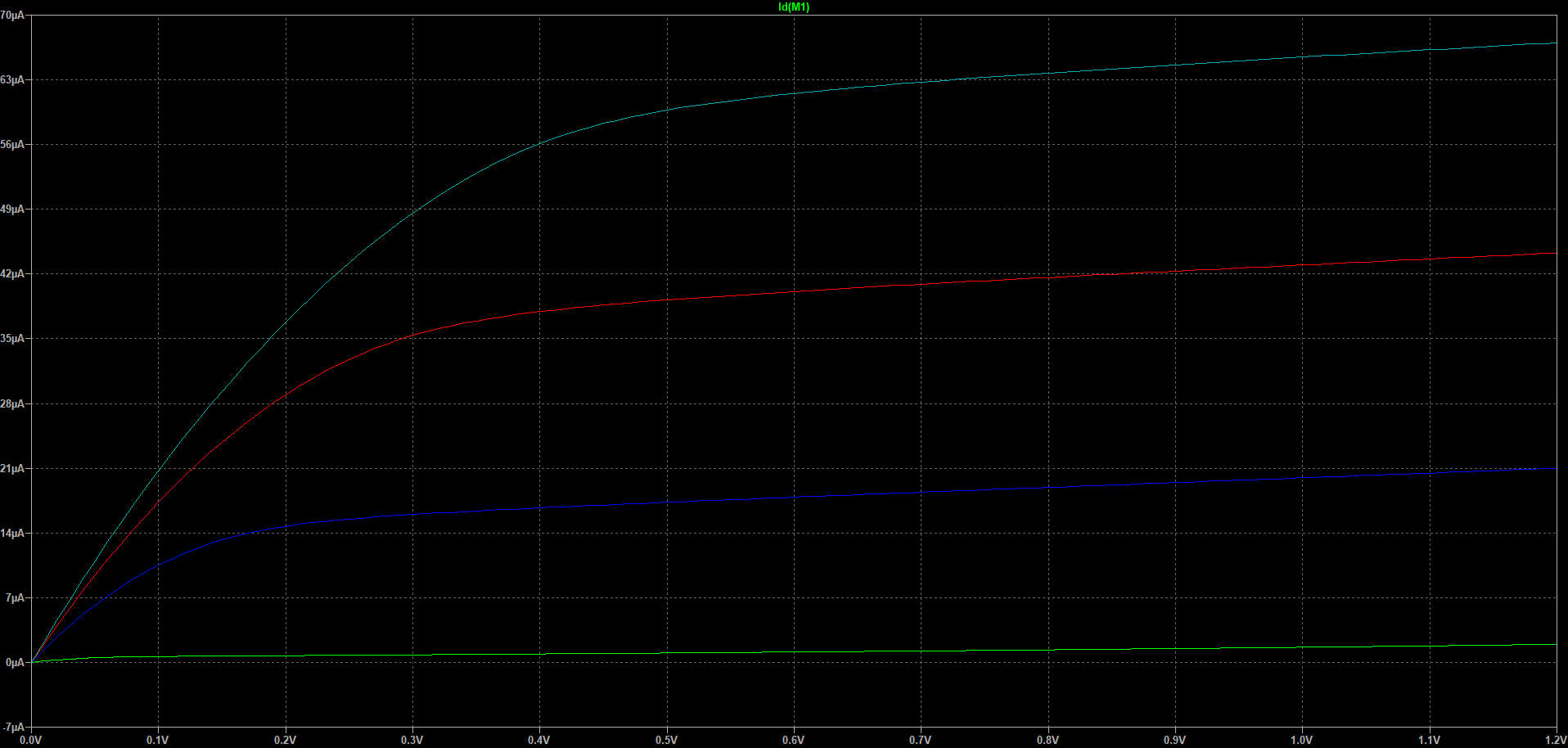


Figura 3-Características de em função de , NMOS com W=L= 70 nm

O segundo cenário de simulação corresponde à obtenção da corrente para um valor fixo de igual a 1.2V, para os dois circuitos NMOS de 90nm. Os resultados obtidos da simulação do circuito para uma relação W=L=700nm, são apresentados na Figura 4. Podemos verificar que com o aumento da tensão a corrente irá aumentar exponencialmente, atingindo um máximo em 105µA.

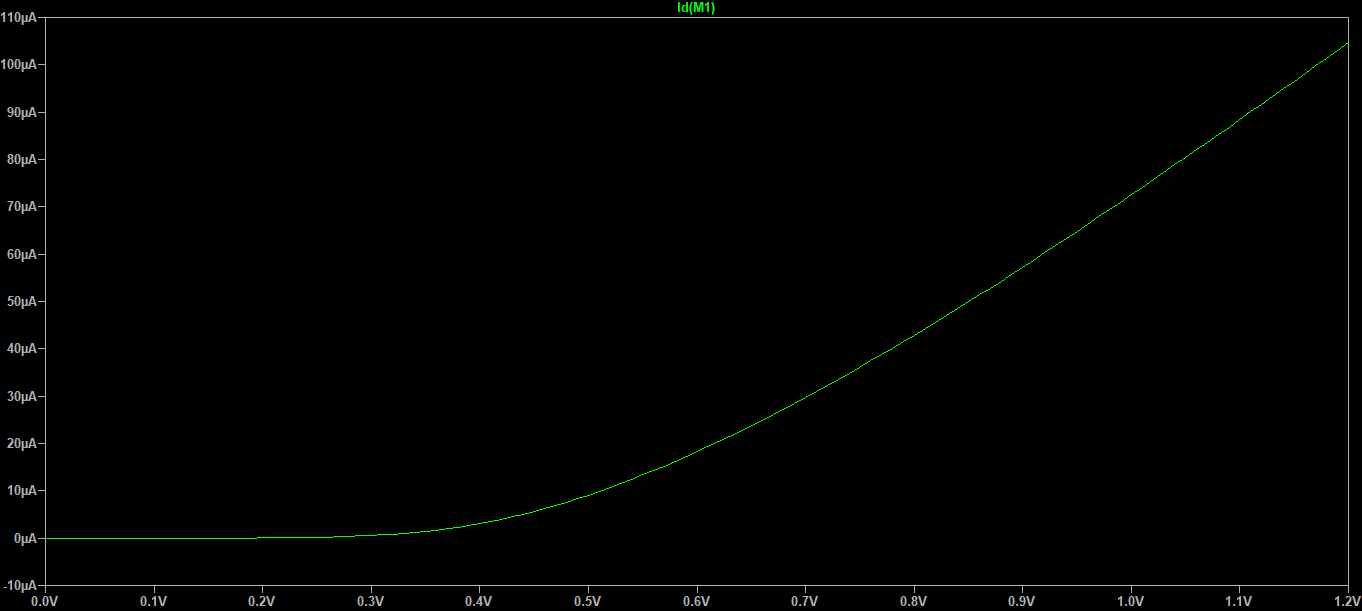


Figura 4 - Característica de em função de , NMOS com W=L= 700 nm

Repetindo a mesma simulação para o NMOS com W=L=70nm, são obtidos os resultados apresentados na figura 5. Podemos verificar a mesma relação entre a corrente e , para um valor fixo de tensão igual a 1.2V.



Figura 5 - Característica de em função de , NMOS com W=L= 70 nm

Após a simulação de ambos os circuitos para os dois cenários de simulação estabelecidos, são extraídos os dados referentes às curvas características do ponto de funcionamento do circuito, efetuando a manipulação dos dados com o auxílio do software *Spyder*, aplicando o modelo *NPower*.

3.2. Python - Spyder

Nesta secção é feita a modelação dos dados simulados, de modo a implementar a metodologia *NPower* para o estudo e análise do comportamento do circuito CMOS anteriormente. O modelo é implementado em *Python*.

Uma vez obtidos os dados referentes às curvas e , os mesmos são carregados no *Spyder* em duas *dataframes* distintas, cuja estrutura está representada na Figura 6.

|  |  |
| --- | --- |
| a) | b) |

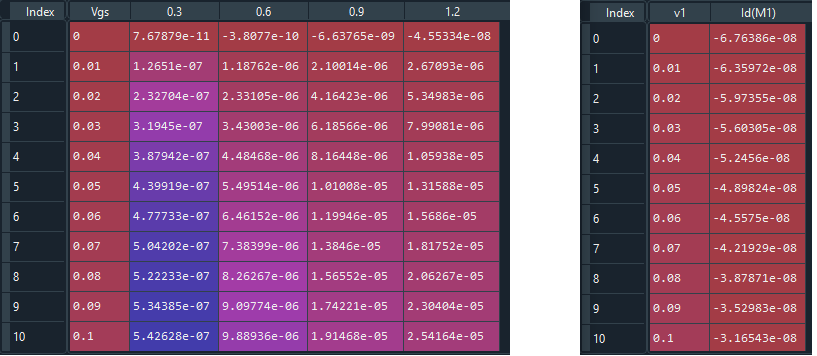


Figura 6 - Estrutura das dataframes utilizadas para armazenar: a) ; b)

De seguida foram obtidos os parâmetros .

A função *gm(Id,Vgs)* que calcula os vários valores da transcondutância , através da equação (3.1).

|  |  |
| --- | --- |
|  | (3.1) |

Após o calculo da transcondutância é efetuada a relação , variável utilizada para o *curve\_fit* à função *get\_n\_vt(Vgs, n, Vt)*, com os valores de e os valores de , a partir do qual é possível determinar os parâmetros e . A função *get\_n\_vt* realiza o cálculo de (3.2).

|  |  |
| --- | --- |
|  | (3.2) |

Onde:

|  |  |
| --- | --- |
|  | (3.3) |
|  | (3.4) |

O parâmetro é calculado partir do declive das curvas características do gráfico simulado , selecionando dois pontos na região linear de saturação, formulando a sua expressão através da equação (3.5).

|  |  |
| --- | --- |
|  | (3.5) |

Por último, é determinado o parâmetro B através de um *curve\_fit* à função *get\_B(Vgs, B)*, que recebe como parâmetros de entrada os valores de e de . A função *get\_B* realiza o cálculo apresentado em (3.6).

|  |  |
| --- | --- |
|  | (3.6) |

Calculados todos os parâmetros de entrada necessários para a implementação do modelo descrito, é estabelecida a função *get\_Id(Vgs, Vds, Vt, n, L, B)* que executa o cálculo do valores de corrente de teóricos, através do sistema de equações (3.6), armazenados no vetor Id2 referente à implementação computacional.

Por fim calculou-se o erro relativo dos novos valores de , , em relação aos valores de obtidos através do *LTSpice,,* através de (3.7)*.*

|  |  |
| --- | --- |
|  | (3.7) |

Esta metodologia foi utilizada para os dois ensaios realizados, e o código para os mesmos pode ser consultado em Anexos

# IV - Resultados

4.1. Parâmetros Obtidos

Aplicando a metodologia apresentada no Capítulo III, foram obtidos os seguintes parâmetros, registados na Tabela 2.

Tabela 2 – Parâmetros obtidos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NMOS 90nm |  |  | B |  |
| W=L=700nm |  | 1.467 |  |  |
| W=L=70nm |  | 1.031 |  |  |

4.2. Analise de Resultados

Neste Capítulo são apresentados os dados obtidos da implementação do modelo *NPower* com recurso ao software *Spyder*.

Primeiramente foi aplicada a seguinte metodologia para os transístores NMOS de 90nm com respetivas relações de igualdade W=L, para 700nm e 70nm. Na Figura 7 são apresentadas as curvas das respetivas características referentes a para .

|  |  |
| --- | --- |
| a) | b) |

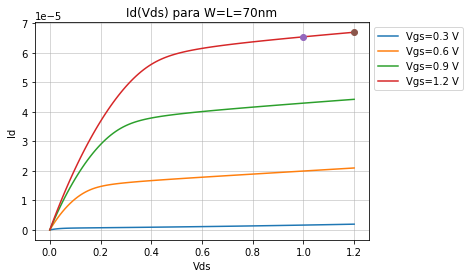
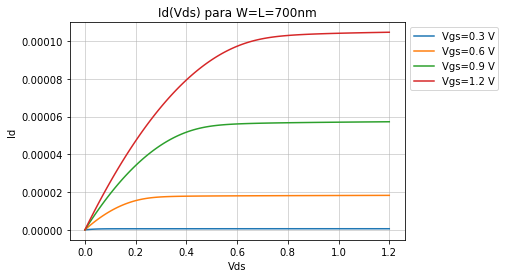
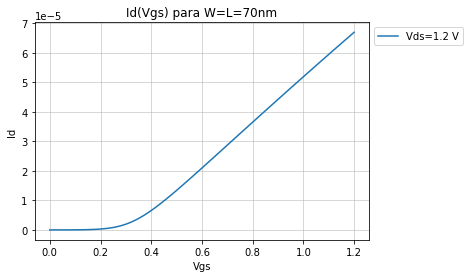
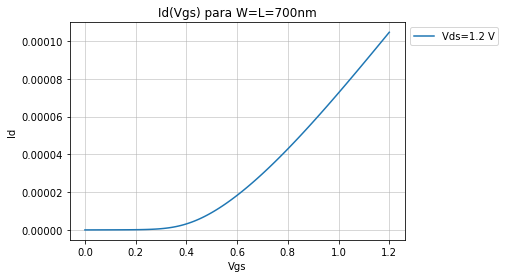


Figura 7 – Curvas , com fixo para: a) W=L=700 nm; b) W=L=70 nm

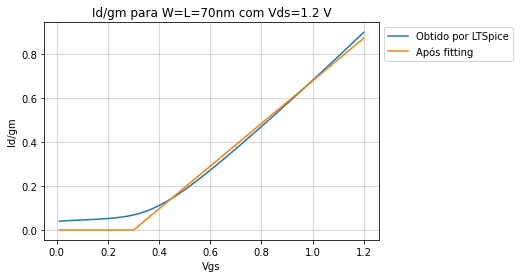
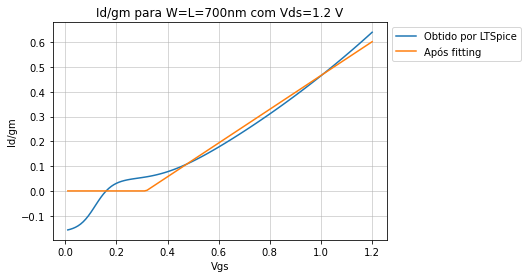
Seguidamente foi realizada a simulação do segundo cenário de operação, onde foram obtidas as curvas características para igual a 1.2V, considerando as diferentes relações W=L. A Figura 8 apresenta os resultados obtidos da simulação dos circuitos em análise.



|  |  |
| --- | --- |
| a) | b) |

Figura 8 - Curvas , com para: a) W=L=700 nm; b) W=L=70 nm

Tendo em conta os dados recolhidos referentes às simulações realizadas, foi implementada a metodologia descrita na secção anterior. Deste modo, recorrendo à equação (3.2) foram obtidos os gráficos representados na Figura 9 que expressam a relação simulado e modelado para os respetivos circuitos W=L.

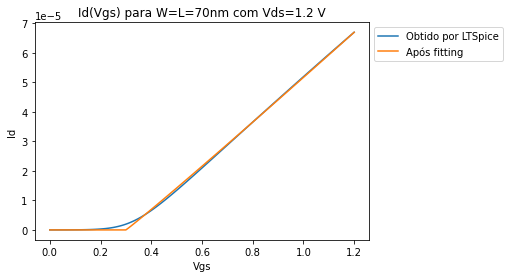
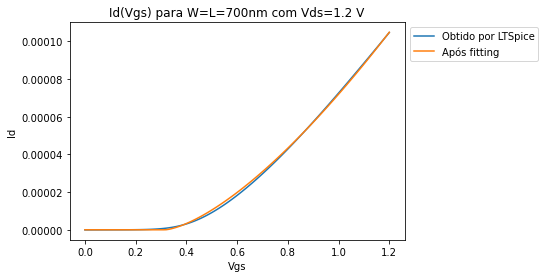


|  |  |
| --- | --- |
| a) | b) |

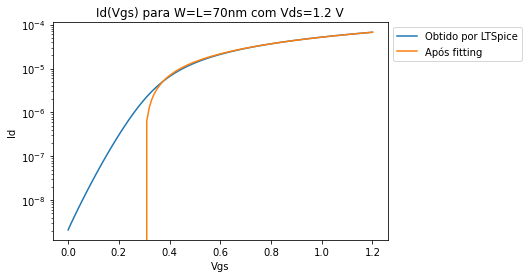
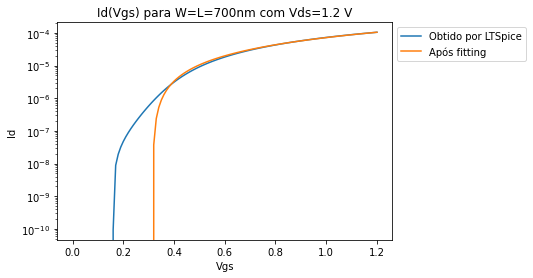
Figura 9 - Curvas , com para: a) W=L=700 nm; b) W=L=70 nm

Analisando os resultados modelados obtidos é possível verificar que as cuvas têm uma boa aproximação aos resultados simulados quando . Deste modo, podemos assumir que aproximação realizada para determinar os parâmetros e é bastante precisa.

Após o cálculo de todos os parâmetros de entrada para a implementação do modelo *NPower,* foram determinados as curvas características para ambos os circuitos W=L. Na figura 10 são representadas as respetivas curvas simuladas e modeladas (NPower) para ambos os circuitos W=L, tanto em escala linear como em escala logarítmica.



|  |  |
| --- | --- |
| a) | b) |



|  |  |
| --- | --- |
| c) | d) |

Figura 10 - Curvas, para: a) W=L=700 nm; b) W=L=70 nm ;c) W=L=700 nm (escala logarítmica) ; d) W=L=70 nm (escala logarítmica)

Estabelecendo uma comparação de modo grosseiro entre os resultados simulados e modelos, podemos afirma que o modelo *NPower* tem uma boa aproximação para a região linear (saturação, ).

Contudo, de modo a determinar qual o grau de precisão dos resultados obtidos, foi calculado o valor do erro relativo entres os valores simulados e modelados, observando a variação do valor do erro tendo em consideração a tensão do componente NMOS. Na Figura 11 são apresentados os gráficos referentes aos erros relativos para ambos os circuitos W=L, considerando apenas erro pera valores .

|  |  |
| --- | --- |
| a) | b) |

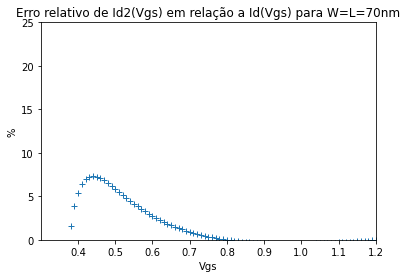
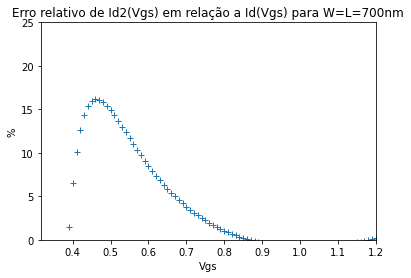


Figura 11 - Erro de em relação a , para: a) W=L=700 nm; b) W=L=70 nm

Analisando os dados da Figura 11, podemos verificar que o erro relativo é superior para transístores com canais de maiores dimensões. Uma vez que apenas se considerou a região , é visível que o maior de valor de erro relativo acontece no limiar da zona de saturação.

# V - Conclusões

Com a realização deste trabalho laboratorial foram abordados vários conceitos que nos permitem retirar algumas conclusões em cada uma das partes de desenvolvimento do projeto, descrevendo e analisando o funcionamento de um circuito NMOS 90nm com uma relação W=L.

Assim sendo, inicialmente realizou-se a análise teórica do circuito executando uma simulação para dois pontos de operação diferentes variando a dimensão do transístor, mantendo a relação W=L. Deste modo, segundo a simulação do primeiro ensaio que corresponde à obtenção da corrente para valores de fixos onde , verificamos que para a corrente se encontra na região de saturação. Por outro lado, executando o segundo cenário de simulação, que corresponde à obtenção da corrente para um valor fixo de igual a 1.2V, observamos que a corrente cresce linearmente ao entrar na região de saturação ().

Após as simulações do *LTSpice* é descrito e implementado um modelo *NPower* que determina os parâmetros de saída simulados no *LTSpice,* analisandoa precisão e incerteza do modelo. Observando os parâmetros resultantes do fitting das curvas de simulação, podemos afirmar que o valor da constante *n* não corresponde exatamente ao fator 2, característica do modelo de *Shockley*,o que revela uma certa incerteza por parte deste último, Tabela 2.

É possível constatar também que o valor de *B* é responsável pela conversão da corrente em tensão onde o seu valor é maior quanto maior forem as dimensões do transístor MOSFET.

Analisando o valor de λ obtido por regressão linear, retirando dois pontos da curva característica na região de saturação, podemos averiguar que o valor do λ depende da relação onde os efeitos da modulação do canal são mais notórios para transístores de canais mais curtos.

Através do cálculo do erro relativo é visível que este converge para 0 com o aumento do valor de . Desta maneira, observando os gráficos do erro relativo, para um transístor com W=L=70nm o erro é menor do que para um transístor de dimensões maiores (com a mesma relação W=L), logo o modelo *NPower* é mais preciso/adequado para transístores com canais mais curtos.

Contudo, podemos afirmar que o modelo *NPower* é um modelo simples e compacto que apresenta uma aproximação robusta para o cálculo dos parâmetros de saída do modelo, nomeadamente da corrente , concluindo também que as aproximações efetuadas por fitting dos parâmetros correspondem a uma boa aproximação para a implementação do modelo.

Finalizando, o modelo *NPower* apresenta algumas falhas para transístores com diferentes dimensões, com W=L, apresentando um comportamento não escalável.

# Bibliografia

[1] T. Sakurai and A. Richard Newton, “A Simple MOSFET Model for Circuit Analysis,” *IEEE Trans. Electron Devices*, vol. 38, no. 4, pp. 887–894, 1991, doi: 10.1109/16.75219.

# Anexos

Trabalho1\_70nm.py

# -\*- coding: utf-8 -\*-

"""

Created on Mon Apr 12 10:59:50 2021

@authors: Rafael, Francisco, José

"""

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import curve\_fit

os.chdir('G:\\O meu disco\\MIEEC\\5º Ano\\2º Semestre\\EDA\\Trabalho 1')

############################################ PARTE 1 ########################################

########################## 1 - CARREGAMENTO DE DADOS #######################################

#Dados para Id(Vgs,1.2)

vgs12\_70=pd.read\_csv('Id\_Vgs\_12\_70nm.txt',sep='\t')

#Dados para Id(Vds), com os 4 ensaios realizados (0.3, 0.6, 0.9, 1.2)

vds\_70=pd.read\_csv('Id\_Vds\_70nm.txt',sep='\t')

#Carregar os dados de Id(Vds) para uma dataframe, de modo a ter os 4 ensaios separados

U=[]

U.append(vgs12\_70['v1'])

for i in range(0,4,1):

U.append(vds\_70['Id(M1)'][1+i\*122:122+i\*122].to\_numpy())

# i\*122 - offset entre ensaios, contabilizando a linha da string

vds\_70 = pd.DataFrame(U).transpose()

vds\_70.columns = ['Vgs','0.3','0.6','0.9', '1.2']

del U, i

###########################################################################################

############################### 2 - PLOT ID(VDS) PARA OS VÁRIOS VGS ############################

plt.figure()

plt.title('Id(Vds) para W=L=70nm')

plt.ylabel('Id')

plt.xlabel('Vds')

for vgs in np.arange(0.3,1.5,0.3):

plt.plot(vds\_70['Vgs'], vds\_70[str(round(vgs,2))],label='Vgs=%.1f V' %vgs)

plt.plot(vds\_70['Vgs'][100], vds\_70['1.2'][100],'o', vds\_70['Vgs'][120], vds\_70['1.2'][120],'o'

plt.legend(bbox\_to\_anchor=(1, 1))

plt.grid(linewidth=0.5)

del vgs

###########################################################################################

############################### 3 - PLOT ID(VGS) COM VDS = 1.2 V ###############################

plt.figure()

plt.title('Id(Vgs) para W=L=70nm')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_70['v1'],vgs12\_70['Id(M1)'], label='Vds=1.2 V')

plt.legend(bbox\_to\_anchor=(1, 1))

plt.grid(linewidth=0.5)

###########################################################################################

############################# 4 - OBTENÇÃO DOS PARÂMETROS LAMBDA, VT, N e B ##################

############ Definição de funções #################

def gm(Id,Vgs): #gm= dId/dVgs

return np.diff(Id)/np.diff(Vgs)

def get\_n\_vt(Vgs, n, Vt):

return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0, lambda Vgs:(Vgs-Vt)/n])

def get\_lambda(Id1,Id2,Vds1,Vds2):

return (Id1-Id2)/(Id2\*Vds1-Id1\*Vds2)

def get\_B(Vgs, B):

return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0, lambda Vgs:B\*((Vgs-Vt)\*\*n)\* (1+L\*1.2)])

###########################################################################################

# Obtenção dos valores de gm

gm\_70 = gm(vgs12\_70['Id(M1)'],vgs12\_70['v1'])

# Obtenção dos valores de Id/gm

id\_gm\_70=(vgs12\_70['Id(M1)'][1:]/gm\_70).to\_numpy()

#Obtenção de n e Vt

xx,xy=curve\_fit(get\_n\_vt,vgs12\_70['v1'][1:].values,id\_gm\_70)

n=xx[0]

Vt=xx[1]

#Obtenção dos valores Id/gm após o fitting

id\_gm\_new=get\_n\_vt(vgs12\_70['v1'][1:].values,n,Vt)

#Obtenção de Lambda

L=get\_lambda(vds\_70['1.2'][100], vds\_70['1.2'][120], vds\_70['Vgs'][100], vds\_70['Vgs'][120])

#Obtenção de B

xx,xy=curve\_fit(get\_B,vgs12\_70['v1'].values,vgs12\_70['Id(M1)'].values)

B=xx[0]

del xx, xy

######################## PLOTS #############################################################

#Id/gm antes e após fitting

plt.figure()

plt.title('Id/gm para W=L=70nm com Vds=1.2 V')

plt.ylabel('Id/gm')

plt.xlabel('Vgs')

plt.plot(vgs12\_70['v1'][1:],id\_gm\_70, label='Obtido por LTSpice')

plt.plot(vgs12\_70['v1'][1:],id\_gm\_new, label='Após fitting')

plt.legend(bbox\_to\_anchor=(1, 1))

plt.grid(linewidth=0.5)

#Id(Vgs) com Vds = 1.2 V antes e após fitting

plt.figure()

plt.title('Id(Vgs) para W=L=70nm com Vds=1.2 V')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_70['v1'],vgs12\_70['Id(M1)'], label='Obtido por LTSpice')

plt.plot(vgs12\_70['v1'],get\_B(vgs12\_70['v1'].values,B),label='Após fitting')

plt.legend(bbox\_to\_anchor=(1, 1))

#Id(Vgs) com Vds = 1.2 V antes e após fitting em escala logarítmica

plt.figure()

plt.title('Id(Vgs) para W=L=70nm com Vds=1.2 V')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_70['v1'],vgs12\_70['Id(M1)'], label='Obtido por LTSpice')

plt.plot(vgs12\_70['v1'],get\_B(vgs12\_70['v1'].values,B),label='Após fitting')

plt.yscale('log')

plt.legend(bbox\_to\_anchor=(1, 1))

############################################ PARTE 2 ########################################

############################ 1 - get\_ID FUNCTION #############################################

def get\_Id(Vgs, Vds, Vt, n, L, B):

return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0, lambda Vgs:B\*((Vgs-Vt)\*\*n)\* (1+L\*Vds)])

##################### 2 - Id2(Vgs,Vds) matrix ###################################################

Id2=get\_Id(np.arange(0,1.21,0.01),1.2,Vt,n,L,B)

##################### 3 - Id2(Vgs) plot, with Vds = 1.2 V #############################

plt.figure()

plt.title('Id(Vgs) para W=L=70nm com Vds=1.2 V')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_70['v1'],get\_B(vgs12\_70['v1'].values,B),label='Após fitting')

plt.plot(vgs12\_70['v1'],Id2, label='através de get\_Id')

plt.yscale('log')

plt.legend(bbox\_to\_anchor=(1, 1))

##################### 4 - Relative Error of Id2(Vgs) in relation to Id1(Vgs) ##########

# i.e., Error (Vgs) = (Id2 (Vgs) -Id1 (Vgs)) / Id1 (Vgs).

E\_rel=((Id2-vgs12\_70['Id(M1)'])/vgs12\_70['Id(M1)'])\*100

plt.figure()

plt.ylabel('%')

plt.xlabel('Vgs')

plt.title('Erro relativo de Id2(Vgs) em relação a Id(Vgs) para W=L=70nm')

plt.plot(vgs12\_70['v1'],E\_rel,'+')

plt.axis([Vt,1.2,0,25])

Trabalho1\_700nm.py

# -\*- coding: utf-8 -\*-

"""

Created on Mon Apr 12 10:59:50 2021

@authors: Rafael, Francisco, José

"""

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import curve\_fit

os.chdir('G:\\O meu disco\\MIEEC\\5º Ano\\2º Semestre\\EDA\\Trabalho 1')

############################################ PARTE 1 ########################################

########################## 1 - CARREGAMENTO DE DADOS #######################################

#Dados para Id(Vgs,1.2)

vgs12\_700=pd.read\_csv('Id\_Vgs\_12\_700nm.txt',sep='\t')

#Dados para Id(Vds), com os 4 ensaios realizados (0.3, 0.6, 0.9, 1.2)

vds\_700=pd.read\_csv('Id\_Vds\_700nm.txt',sep='\t')

#Carregar os dados de Id(Vds) para uma dataframe, de modo a ter os 4 ensaios separados

U=[]

U.append(vgs12\_700['v1'])

for i in range(0,4,1):

U.append(vds\_700['Id(M1)'][1+i\*122:122+i\*122].to\_numpy())

# i\*122 - offset entre ensaios, contabilizando a linha da string

vds\_700 = pd.DataFrame(U).transpose()

vds\_700.columns = ['Vgs','0.3','0.6','0.9', '1.2']

del U, i

######################################################################################

########################## 2 - PLOT ID(VDS) PARA OS VÁRIOS VGS ############################

plt.figure()

plt.title('Id(Vds) para W=L=700nm')

plt.ylabel('Id')

plt.xlabel('Vds')

for vgs in np.arange(0.3,1.5,0.3):

plt.plot(vds\_700['Vgs'], vds\_700[str(round(vgs,2))],label='Vgs=%.1f V' %vgs)

plt.plot(vds\_700['Vgs'][100], vds\_700['1.2'][100],'o', vds\_700['Vgs'][120], vds\_700['1.2'][120],'o')

plt.legend(bbox\_to\_anchor=(1, 1))

plt.grid(linewidth=0.5)

del vgs

###########################################################################################

############################### 3 - PLOT ID(VGS) COM VDS = 1.2 V ###############################

plt.figure()

plt.title('Id(Vgs) para W=L=700nm')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_700['v1'],vgs12\_700['Id(M1)'], label='Vds=1.2 V')

plt.legend(bbox\_to\_anchor=(1, 1))

plt.grid(linewidth=0.5)

###########################################################################################

############################# 4 - OBTENÇÃO DOS PARÂMETROS LAMBDA, VT, N e B ##################

############ Definição de funções #################

def gm(Id,Vgs): #gm= dId/dVgs

return np.diff(Id)/np.diff(Vgs)

def get\_n\_vt(Vgs, n, Vt):

return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0, lambda Vgs:(Vgs-Vt)/n])

def get\_lambda(Id1,Id2,Vds1,Vds2):

return (Id1-Id2)/(Id2\*Vds1-Id1\*Vds2)

def get\_B(Vgs, B):

return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0, lambda Vgs:B\*((Vgs-Vt)\*\*n)\*(1+L\*1.2)])

###########################################################################################

# Obtenção dos valores de gm

gm\_700 = gm(vgs12\_700['Id(M1)'],vgs12\_700['v1'])

# Obtenção dos valores de Id/gm

id\_gm\_700=(vgs12\_700['Id(M1)'][1:]/gm\_700).to\_numpy()

#Obtenção de n e Vt

xx,xy=curve\_fit(get\_n\_vt,vgs12\_700['v1'][1:].values,id\_gm\_700)

n=xx[0]

Vt=xx[1]

#Obtenção dos valores Id/gm após o fitting

id\_gm\_new=get\_n\_vt(vgs12\_700['v1'][1:].values,n,Vt)

#Obtenção de Lambda

L=get\_lambda(vds\_700['1.2'][100], vds\_700['1.2'][120], vds\_700['Vgs'][100], vds\_700['Vgs'][120])

#Obtenção de B

xx,xy=curve\_fit(get\_B,vgs12\_700['v1'].values,vgs12\_700['Id(M1)'].values)

B=xx[0]

del xx, xy

######################## PLOTS ##############################

#Id/gm antes e após fitting

plt.figure()

plt.title('Id/gm para W=L=700nm com Vds=1.2 V')

plt.ylabel('Id/gm')

plt.xlabel('Vgs')

plt.plot(vgs12\_700['v1'][1:],id\_gm\_700, label='Obtido por LTSpice')

plt.plot(vgs12\_700['v1'][1:],id\_gm\_new, label='Após fitting')

plt.legend(bbox\_to\_anchor=(1, 1))

plt.grid(linewidth=0.5)

#Id(Vgs) com Vds = 1.2 V antes e após fitting

plt.figure()

plt.title('Id(Vgs) para W=L=70nm com Vds=1.2 V')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_700['v1'],vgs12\_700['Id(M1)'], label='Obtido por LTSpice')

plt.plot(vgs12\_700['v1'],get\_B(vgs12\_700['v1'].values,B),label='Após fitting')

plt.legend(bbox\_to\_anchor=(1, 1))

#Id(Vgs) com Vds = 1.2 V antes e após fitting em escala logarítmica

plt.figure()

plt.title('Id(Vgs) para W=L=700nm com Vds=1.2 V')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_700['v1'],vgs12\_700['Id(M1)'], label='Obtido por LTSpice')

plt.plot(vgs12\_700['v1'],get\_B(vgs12\_700['v1'].values,B),label='Após fitting')

plt.yscale('log')

plt.legend(bbox\_to\_anchor=(1, 1))

############################################ PARTE 2 ########################################

############################ 1 - get\_ID FUNCTION #############################################

def get\_Id(Vgs, Vds, Vt, n, L, B):

return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0, lambda Vgs:B\*((Vgs-Vt)\*\*n)\*(1+L\*Vds)])

############################### 2 - Id2(Vgs,Vds) matrix #########################################

Id2=get\_Id(np.arange(0,1.21,0.01),1.2,Vt,n,L,B)

################################## 3 - Id2(Vgs) plot, with Vds = 1.2 V #############################

plt.figure()

plt.title('Id(Vgs) para W=L=700nm com Vds=1.2 V')

plt.ylabel('Id')

plt.xlabel('Vgs')

plt.plot(vgs12\_700['v1'],get\_B(vgs12\_700['v1'].values,B),label='Após fitting')

plt.plot(vgs12\_700['v1'],Id2, label='através de get\_Id')

plt.yscale('log')

plt.legend(bbox\_to\_anchor=(1, 1))

############################ 4 - Relative Error of Id2(Vgs) in relation to Id1(Vgs) #####################

# i.e., Error (Vgs) = (Id2 (Vgs) -Id1 (Vgs)) / Id1 (Vgs).

E\_rel=((Id2-vgs12\_700['Id(M1)'])/vgs12\_700['Id(M1)'])\*100

plt.figure()

plt.ylabel('%')

plt.xlabel('Vgs')

plt.title('Erro relativo de Id2(Vgs) em relação a Id(Vgs) para W=L=700nm')

plt.plot(vgs12\_700['v1'],E\_rel,'+')

plt.axis([Vt,1.2,0,25])