NOVA SCHOOL OF SCIENCE & TECHNOLOGY

EDA/CAD para Nanoelectrónica

2º Semestre 2020 / 2021

Relatório

Lab#1: Estudo e Aplicação do Modelo *Thin Film Transistor (TFT)*

22/05/2021

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

Índice de Figuras

Índice de Tabelas

Introdução

No âmbito da cadeira de EDA/CAD para Nanoelectrónica, é apresentado o primeiro trabalho laboratorial onde é realizado o estudo e determinação do modelo *Thin Film Transistor* (TFT), para a caracterização de transístores MOSFETs de tecnologia nanométrica [1]–[3].

Este trabalho está dividido em várias etapas de desenvolvimento, de modo a determinar de forma automática os parâmetros do modelo TFT, a partir dos ficheiros característicos do circuito para diferentes relações W/L.

Na primeira parte do projeto é realizado o modelo TFT desprezando as resistências de contacto do circuito, descobrindo os parâmetros analíticos referentes às expressões algébricas do modelo em estudo, concluindo se o modelo é escalável e preciso, tendo em conta os dados simulados em *Cadence* ???????? e os dados obtidos através da modelização do TFT.

Na segunda parte do trabalho são consideradas as resistências de contacto do modelo TFT, avaliando este em função da precisão dos resultados simulados com os dados modelados. Por último, são analisadas as duas abordagens, comparando os dados obtidos com base no erro relativo calculado entre os dados simulados e teóricos obtidos, retirando as conclusões da implementação deste modelo através do *fitting* dos parâmetros para a determinação do funcionamento de um circuito MOSFET.

I-Objetivo

Este projeto tem como objetivo a determinação dos parâmetros do modelo *Thin Film Transistor* para um circuito MOSFET para diferentes relações de comprimento e largura (W/L), verificando quais as limitações do modelo TFT. A Tabela 1 apresenta as diferentes relações W/L para o estudo do modelo TFT.

Tabela 1 – Transístores utilizados

Modelo	W (µm)	L (µm)
MOSFET	50	5
MOSFET	50	10
MOSFET	50	20

O modelo TFT é implementado para as diferentes dimensões W/L, de modo a verificar a escalabilidade do modelo e o erro relativo obtido para os diferentes cenários simulados. Deste modo este trabalho apresenta os desafios da utilização do modelo TFT.

II - Apresentação do Modelo TFT

2.1. Descrição do Modelo TFT

Parte 1

Na primeira parte do trabalho é descrito o modelo TFT, desprezando o efeito das resistências de contacto. Deste modo, o modelo TFT é caracterizado pelo sistema de equações (1), onde é definido o comportamento da corrente I_D para os transístores deste tipo. A corrente de saturação do "dreno" I_{DSAT} é dada pela equação (4), escrita na forma compacta. Porém este parâmetro depende da largura do canal W, do comprimento do efetivo do canal L_{eff}, da tensão de "threshold" V_T e do parâmetro de mobilidade efetiva e μ_{eff}.

O parâmetro de mobilidade efetiva μ_{eff} é dado pela equação (3), onde μ_0 representa a mobilidade portadora a baixa polarização ($V_{GS} \cong V_T$), Υ fator de melhoria da mobilidade e V_{AA} parâmetro empírico medido em Volts.

A equação (2) define a tensão de saturação V_{DSAT}, onde a variável α é um parâmetro adimensional que relaciona a tensão de saturação ao *overdrive* da "gate".

Simplificadas as equações, surgem os parâmetros K e m, associados à corrente de saturação e são responsáveis por controlar as características da região linear (tensão linear de saturação). Por último, a variável λ expressa uma relação entre a condutância do "dreno" na zona linear de saturação.

$$I_D = \begin{cases} \frac{WC_i}{L_{eff}}\mu_{eff}\left(V_{GS} - V_T - \frac{V_{DS}}{2\alpha}\right)V_{DS}, & V_{DS} < V_{DSAT} \\ \frac{WC_i}{L_{eff}}\mu_{eff}\alpha(V_{GS} - V_T)^2, & V_{DS} \geq V_{DSAT} \end{cases} \quad (1)$$

Onde:

$$V_{DSAT} = \alpha(V_{GS} - V_T) \quad (2)$$

$$\mu_{eff} = \mu_0 \frac{(V_{GS} - V_T)^{\Upsilon}}{V_{AA}} \quad (3)$$

$$I_{DSAT} = \mathrm{K}(V_{GS} - V_T)^m \quad (4)$$

$$\mathrm{K} = \frac{WC_i}{L_{eff}}\frac{\mu_{eff}}{V_{AA}}C_i\alpha \quad (5)$$

$$m = 2 + \Upsilon \quad (6)$$

$$I_{DS} = I_D(1 + \lambda \mathrm{V_{DS}}) \quad (7)$$

Parte 2

Na segunda parte iremos considerar as resistências de contacto, representadas na Figura 1. Assim sendo, a equação para o cálculo de I_D passa a estar dependente de R_s, enquanto a tensão V_{DS} passará a depender da soma entre as resistências R_D e R_s.

$$\mathrm{I_D} = \mathrm{K}(V_{GS} - R_S I_D - V_T)^m \tag{8}$$

$$\mathrm{V_{DS}} = V_{ds} - (R_D + R_s) I_D \tag{9}$$

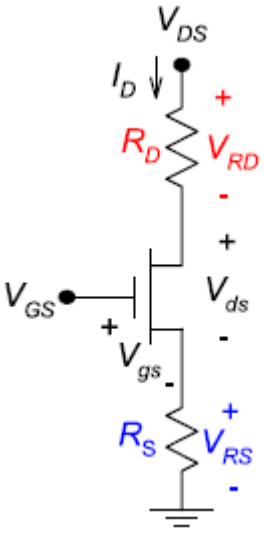


Figura 1 – Modelo TFT com resistências em série parasitas

III - Metodologia

3.1. Carregamento de dados

Nesta secção é feita a descrição da metodologia utilizada para implementar o modelo TFT, através dos dados medidos para o funcionamento de um circuito MOSFET. O modelo é implementado no software *Spyder-Python*.

Inicialmente os dados obtidos são carregados no *Spyder* em seis *dataframes* distintas, três do tipo *data_satL* e três do tipo *data_outL*, onde L assume valores entre $\{5, 10, 20\}$, e que contêm os dados para a região de saturação com V_{DS} igual a 7V e para a saída com ensaios de vários valores de V_{GS} fixos, respetivamente. A estrutura para as dataframes do tipo *data_satL* são representadas pela Figura 2. A estrutura das dataframes do tipo *data_outL* é semelhante, sendo que a diferença entre elas é o tipo de ensaio realizado. Nas dataframes do tipo *data_satL* tem-se apenas um ensaio com V_{DS} igual a 7V com V_{GS} a variar entre $[-4, 7]\ V$, enquanto nas dataframes do tipo *data_outL* têm-se nove ensaios distintos para um conjunto de valore $V_{GS} \in \{-1, 0, 1, 2, 3, 4, 5, 6, 7\}$ fixo e com V_{DS} entre $[0, 7]\ V$ variável.

data_sat5 - DataFrame

Index	VG	VD	ID	IG	IS	dIddVg	absIg	absId
0	-4	7	-6.98e-13	-8.61e-13	1.05e-13	6.2677e-12	8.61e-13	6.98e-13
1	-3.9	7	1.179e-12	-1.77e-13	-1.4e-14	7.58436e-14	1.77e-13	1.179e-12
2	-3.8	7	6.09e-13	-1.9e-13	2.02e-13	1.61905e-11	1.9e-13	6.09e-13
3	-3.7	7	-7.9e-14	6.54e-13	9.5e-14	1.06016e-12	6.54e-13	7.9e-14
4	-3.6	7	3.3e-13	-8.43e-13	3.2e-14	9.92168e-12	8.43e-13	3.3e-13
5	-3.5	7	8.3e-13	-1.34e-13	-9.7e-14	6.09385e-13	1.34e-13	8.3e-13
6	-3.4	7	1.75e-13	1.319e-12	-4.7e-14	5.60043e-14	1.319e-12	1.75e-13
7	-3.3	7	7.46e-13	1.469e-12	-1.09e-13	2.85571e-12	1.469e-12	7.46e-13
8	-3.2	7	-5.72e-13	-1.82e-13	-2.64e-13	9.95706e-14	1.82e-13	5.72e-13
9	-3.1	7	-8.59e-13	-5.42e-13	-7.4e-14	9.16e-12	5.42e-13	8.59e-13
10	-3	7	-1.854e-12	-1.054e-12	-1.68e-13	3.34376e-13	1.054e-12	1.854e-12

Figura 2 - Estrutura das dataframes para armazenar os dados

3.2. Obtenção dos parâmetros sem contabilização do efeito R_D e R_S

De seguida foram obtidos os parâmetros V_T, m, γ, λ e K, através da função *calc_parameters(data_out, data_sat)*. Esta função realiza as operações a seguir descritas. Primeiramente são calculados os vários valores da transcondutância gm, através da equação (3.1), pela função *gm(Id,Vgs)*.

$$gm = \frac{\partial I_D}{\partial V_{GS}} \tag{3.1}$$

Após o cálculo da transcondutância é efetuada a relação $\frac{I_D}{gm}$, variável utilizada para o *curve_fit* à função *get_m_vt(Vgs, n, Vt)*, com os valores de V_{GS} e os valores de $\frac{I_D}{gm}$, a partir do qual é possível determinar os parâmetros V_T, m e γ. A função *get_m_vt* realiza o cálculo descrito pela equação (3.2).

$$\frac{I_D}{gm} = \begin{cases} 0, & V_{GS} < V_T \\ \dfrac{V_{GS} - V_T}{m}, & V_{GS} \geq V_T \end{cases} \tag{3.2}$$

Onde:

$$gm = B \cdot \frac{W}{L} \cdot (V_{GS} - V_T)^{n-1}(1 + \lambda V_{DS}) \tag{3.3}$$

$$I_D = B \cdot \frac{W}{L} \cdot (V_{GS} - V_T)^{n}(1 + \lambda V_{DS}) \tag{3.4}$$

$$\gamma = m - 2 \tag{3.5}$$

O parâmetro λ é calculado a partir do declive das curvas características do gráfico simulado de $I_D(V_{GS})$, selecionando dois pontos na região linear de saturação, através da equação (3.6), recorrendo à função *get_lambda(Id1,Id2,Vds1,Vds2)*.

$$\lambda = \frac{I_{D1} - I_{D2}}{I_{D2} \cdot V_{DS1} - I_{D1} \cdot V_{DS2}} \tag{3.6}$$

De seguida, determina-se o parâmetro K através de um *curve_fit* à função *get_K(Vgs, K)*, que recebe como parâmetros de entrada os valores da tensão V_{GS} e de $I_D(V_{GS})$. A função *get_K* realiza o cálculo apresentado em (3.7).

$$I_D = \begin{cases} 0, & V_{GS} < V_T \\ K \cdot (V_{GS} - V_T)^m \cdot (1 + \lambda \cdot V_{DS}), & V_{GS} \geq V_T \end{cases} \quad (3.7)$$

Finalmente, é calculado o parâmetro α através de um *curve_fit* à função *get_alpha(Vds, alpha),* com os valores de V_{DS} *e* $I_D(V_{DS})$ *com* V_{GS} igual a $7\,V$. A função *get_alpha* realiza o cálculo de (3.8).

$$I_D = \begin{cases} \frac{2K}{\alpha} \cdot (V_{GS} - V_T)^\gamma \cdot \left(V_{GS} - V_T - \frac{V_{DS}}{2\alpha}\right) \cdot V_{DS} \cdot (1 + \lambda V_{DS}), & V_{DS} < \alpha(V_{GS} - V_T) \\ K \cdot (V_{GS} - V_T)^m \cdot (1 + \lambda \cdot V_{DS}), & V_{GS} \geq \alpha(V_{GS} - V_T) \end{cases} \quad (3.8)$$

O procedimento para o cálculo dos parâmetros é repetido para as diferentes relações $\frac{W}{L}$, sendo os valores resultantes armazenados numa dataframe com a estrutura representada na Figura 3.

parameters - DataFrame

Index	Vt	m	gamma	K	Lambda	alpha	L
0	-0.992037	2.60151	0.601505	6.27735e-07	0.248583	0.526047	5
1	0.210607	2.40322	0.40322	8.56567e-07	0.0432715	0.742186	10
2	0.0396462	2.35389	0.353892	4.57851e-07	0.0279382	0.759722	20

Figura 3 - Estrutura da dataframe que armazena o resultado dos parâmetros obtidos

Calculados todos os parâmetros necessários para a implementação do modelo descrito, é estabelecida a função *get_Idout(Vds, Vx, alpha, K, gamma, m, L)*, que executa o cálculo do valores de corrente de I_D, através do sistema de equações (3.7), onde Vx é igual a $(V_{GS} - V_T)$. Assim, através da função *calc_Iout(data_out,params,j)* são calculados os valores das correntes de saída $I_D(V_{DS})$ para as diferentes relações $\frac{W}{L}$ e para valores de V_{GS} fixos tais que $V_{GS} \geq V_T$. Estes valores são armazenados numa dataframe *IL*, onde L tem valores de $\{5, 10, 20\}$, cuja estrutura está representada na Figura 4.

I5 - DataFrame

Index	VG	VD	ID
0	0	0	0.0
1	0	0.1	2.1834511594278225e-07
2	0	0.2	3.9988645940517724e-07
3	0	0.3	5.412568966027197e-07
4	0	0.4	6.390892937509445e-07
5	0	0.5	6.900165170653859e-07
6	0	0.6	7.06512392058872e-07
7	0	0.7	7.217955872346781e-07
8	0	0.8	7.370787824104842e-07
9	0	0.9	7.523619775862904e-07

Figura 4 - Estrutura da dataframe que armazena o resultado dos valores de $I_D(V_{DS})$

3.3. Obtenção dos parâmetros contabilizando o efeito R_D e R_S

De forma a obter os novos valores dos parâmetros V_T, m, λ e K e ainda os valores das resistências de contacto R_D e R_S, recorreu-se à função *calc_parameters2(data_out, data_sat, L, p00, bounds_Rs, bounds_Rd).*

Numa primeira parte, sem contabilizar R_D e λ, obtém-se novos valores para V_T, m, K e R_S através de um *curve_fit* à função *get_Rs(data, Vt, m, K, Rs),* com os valores de V_{GS} e I_D para os quais $V_{GS} \geq V_T$, obtidos na primeira parte. Este *curve_fit* tem como pontos iniciais os obtidos anteriormente. A função *get_Rs* realiza o cálculo da equação (3.9).

$$\mathrm{I_D} = \mathrm{K} \cdot (V_{GS} - R_S I_D - V_T)^m \tag{3.9}$$

Numa segunda parte, os novos valores obtidos para os parametros V_T, m, K e R_S são utilizados como pontos iniciais a um novo *curve_fit*, desta vez à função *get_Rd_L(data, Vt, m, K, Rs, Rd, L)*, com os valores de V_{GS}, V_{DS} e I_D na zona de saturação. Esta função realiza o cálculo dos valores de $V_T, m, K, \lambda, R_D, R_S$ através da equação (3.10).

$$\mathrm{I_D} = \mathrm{K} \cdot (V_{GS} - R_S I_D - V_T)^m \cdot (1 + \lambda \cdot (V_{ds} - (R_D + R_s) I_D)) \tag{3.10}$$

Este procedimento é repetido para as diferentes relações $\frac{W}{L}$, sendo os valores resultantes armazenados numa dataframes, com a estrutura representada na Figura 5.

parameters2 - DataFrame

Index	Vt	m	K	Rs	Rd	Lambda	L
0	-0.991815	2.6039	1.73331e-06	99.721	0	0	5
1	-0.17269	2.56973	1.16339e-06	163.959	165.474	0.16868	5
2	0.299569	2.39898	1.17185e-06	105.515	0	0	10
3	0.385934	2.37028	1.08754e-06	112.311	107.194	0.0401625	10
4	0.0619744	2.36415	5.43227e-07	99.3788	0	0	20
5	0.0619744	2.36415	5.43227e-07	99.3788	100	0.0279382	20

Figura 5 - Estrutura da dataframe que armazena o resultado dos parâmetros obtidos contabilizando as resistências de contacto

Finalmente, são calculados os novos valores de $I_D(V_{GS})$ na saturação para V_{DS} igual a $7\,V$, através das funções *get_Rs* e *get_Rd_L* utilizando os novos parâmetros obtidos. Estes valores são armazenados nos vetores do tipo *Rd_IL* e *Rs_IL,* para as diferentes dimensões L consideradas, sendo que o primeiro tipo contabiliza todos os parâmetros obtidos e o segundo não contabiliza R_D *e* λ.

3.4. Gráficos e Erros

Finalmente, são realizados os seguintes gráficos que descrevem o erro relativo para as várias relações $\frac{W}{L}$:

- Valores $I_D(V_{GS})$ medidos na saturação, para $V_{DS} = 7\ V$
- Valores de $I_D(V_{DS})$ para os vários valores fixos de V_{GS}
- Valores de $I_D(V_{GS})$ medidos vs valores obtidos através do modelo
- Valores de $I_D(V_{DS})$ medidos vs valores obtidos através do modelo
- Valores de $I_D(V_{GS})$ medidos vs modelados, contabilizando R_S
- Valores de $I_D(V_{GS})$ medidos vs modelados, contabilizando $R_S\ e\ R_D$
- Valores de $I_D(V_{DS})$ medidos vs modelados, contabilizando $R_S\ e\ R_D$

Para os gráficos em que se comparam os valores medidos com os valores modelados, calcularam-se os valores dos erros relativos, através da equação (3.11). Todos os gráficos referidos estão representados no Capítulo IV.

$$Erro = \frac{Valor\ modelado - Valor\ medido}{Valor\ medido} * 100\ [\%] \tag{3.11}$$

O código desenvolvido para implementar a metodologia apresentada neste Capítulo, está disponível em Anexos.

IV - Resultados

4.1. Parâmetros Obtidos

Parte 1

Aplicando a metodologia apresentada no Capítulo III, foram obtidos os seguintes parâmetros de entrada do modelo sem o efeito das resistências de contacto do circuito. Na Tabela 2 são apresentados os valores obtidos para as diferentes dimensões do modelo TFT.

Tabela 2 - Parâmetros do modelo TFT, sem o efeito das resistências de contacto

Modelo	W (µm)	L (µm)	V_T	m	Υ	K	λ	a
MOSFET	50	5	-0.992	2.602	0.602	$6.277 \cdot 10^{-7}$	0.249	0.526
MOSFET	50	10	0.211	2.403	0.403	$8.566 \cdot 10^{-7}$	0.043	0.742
MOSFET	50	20	0.039	2.354	0.354	$4.579 \cdot 10^{-7}$	0.028	0.760

Parte 2

Considerando o efeito das resistências de contacto do circuito, são obtidos os seguintes parâmetros referentes à aplicação da segunda parte da metodologia apresentada no Capítulo III, formulando a Tabela 3, que contabiliza apenas R_S, e a Tabela 4, que contabiliza as duas resistências de contacto.

Tabela 3 - Parâmetros do modelo TFT, com efeito da resistência R_S

Modelo	W (µm)	L (µm)	V_T	m	K	R_S	R_D	λ
MOSFET	50	5	-0.992	2.604	$1.733 \cdot 10^{-6}$	99.721	-	-
MOSFET	50	10	0.300	2.399	$1.172 \cdot 10^{-6}$	105.515	-	-
MOSFET	50	20	0.062	2.364	$5.432 \cdot 10^{-7}$	99.379	-	-

Tabela 4 – Parâmetros do modelo TFT, com efeito das resistências R_S e R_D

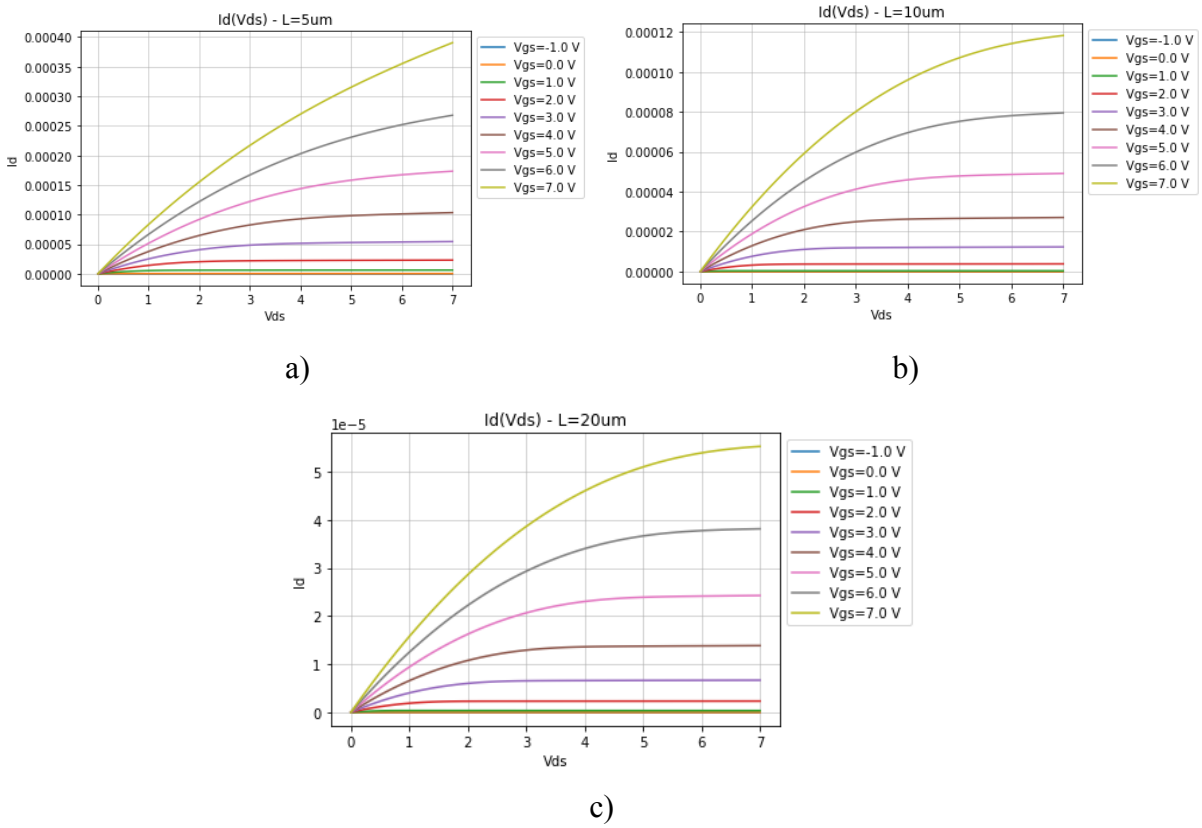
Modelo	$W\ (\mu m)$	$L\ (\mu m)$	V_T	m	K	R_S	R_D	λ
MOSFET	50	5	-0.173	2.570	$1.163 \cdot 10^{-6}$	163.959	165.474	0.169
MOSFET	50	10	0.386	2.370	$1.088 \cdot 10^{-6}$	112.311	107.194	0.040
MOSFET	50	20	0.386	2.364	$5.432 \cdot 10^{-7}$	99.379	100	0.028

4.2. Analise de Resultados

Parte 1

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

Primeiramente foram simulados no software *Cadence* os circuitos para as diferentes dimensões (W/L), obtendo as curvas características da corrente $I_D(V_{DS})$ para $V_{DS} \in [0; 7]V$. Na Figura 6 estão representados os diferentes gráficos simulados para as respetivas dimensões.



a) b)

c)

Figura 6 – Curvas características $I_D(V_{DS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Seguidamente foram obtidas as curvas características simuladas em *Cadence* da corrente $I_D(V_{GS})$ para V_{DS} igual a 7V, considerando as diferentes relações W/L. A Figura 7 apresenta os resultados obtidos da simulação dos circuitos em análise.

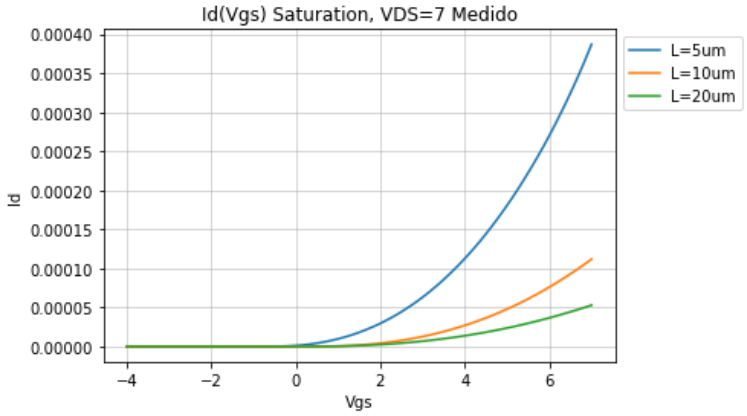
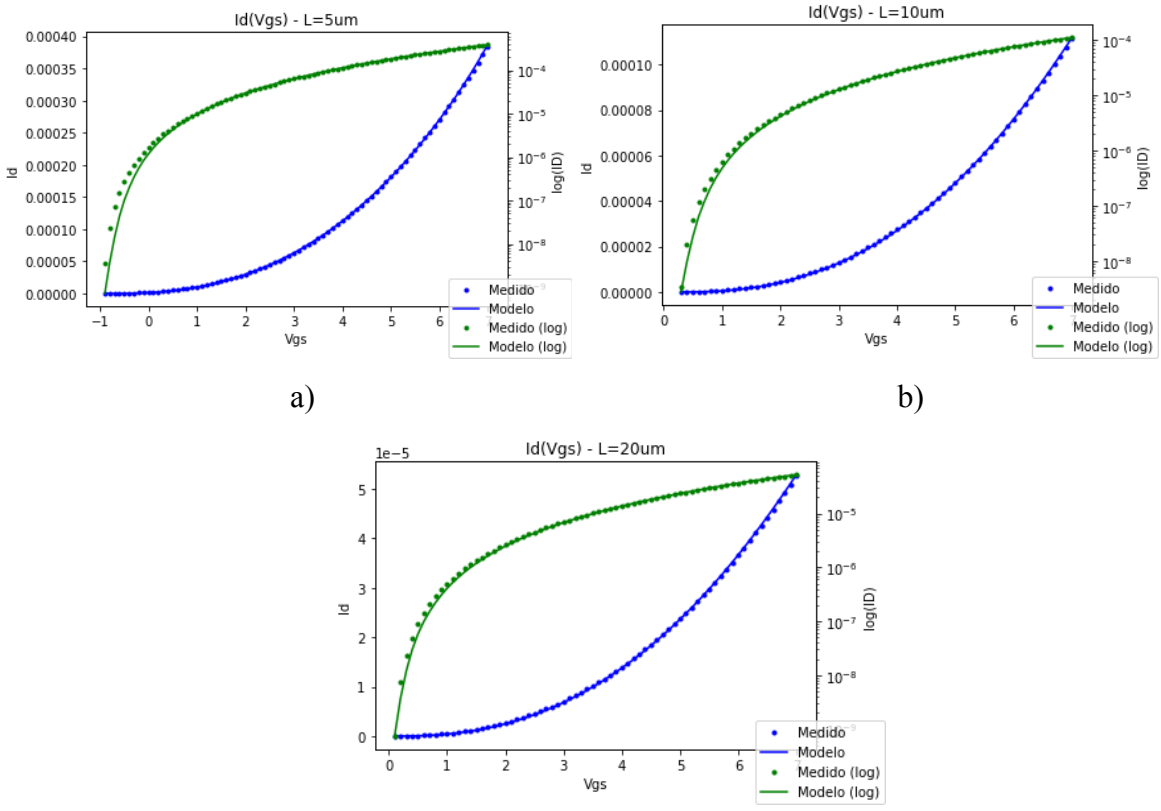


Figura 7 – Curvas características $I_D(V_{GS})$ de saturação, para $V_{DS} = 7V$

Tendo em conta os dados recolhidos referentes às simulações realizadas, foi implementada a metodologia descrita na secção anterior. Deste modo, recorrendo à equação (3.7) foram obtidos os gráficos representados na Figura 8, que expressam a relação $I_D(V_{GS})$ simulado e modelado para os respetivos circuitos W/L.



a) b)

c)

Figura 8 - Curvas características $I_D(V_{GS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Analisando os resultados modelados obtidos é possível verificar que as cuvas têm uma boa aproximação aos resultados simulados quando $V_{GS} \geq V_T$. Deste modo, podemos assumir que a aproximação realizada para a determinação dos parâmetros do modelo TFT, sem o efeito da resistência de contato, é bastante precisa. Através do cálculo do erro relativo entre os valores medidos e modelados podemos constatar essa mesma precisão para a região de saturação, onde o erro relativo converge para 0 quando V_{GS} aumenta. Na Figura 9 estão representados os erros relativos calculados para as diferentes dimensões consideradas.

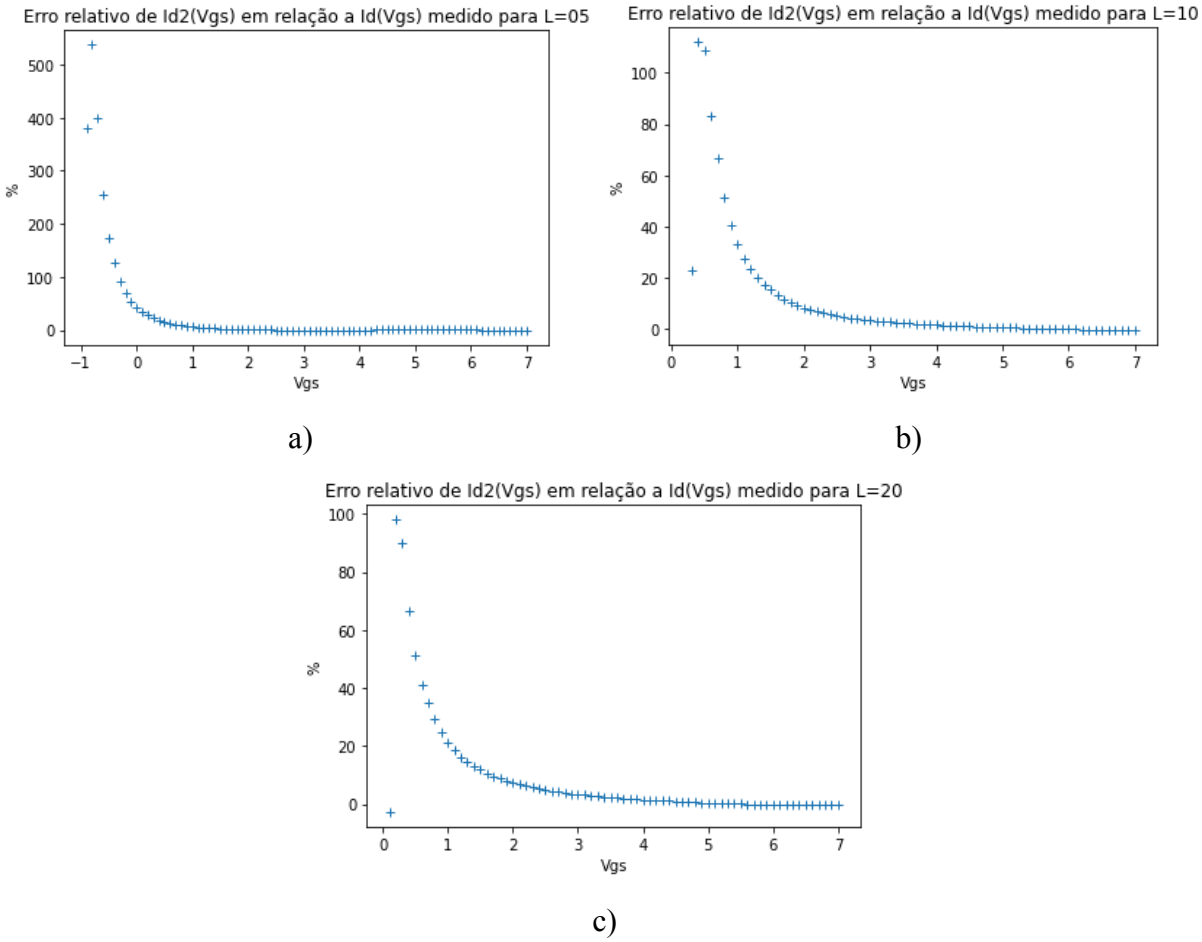
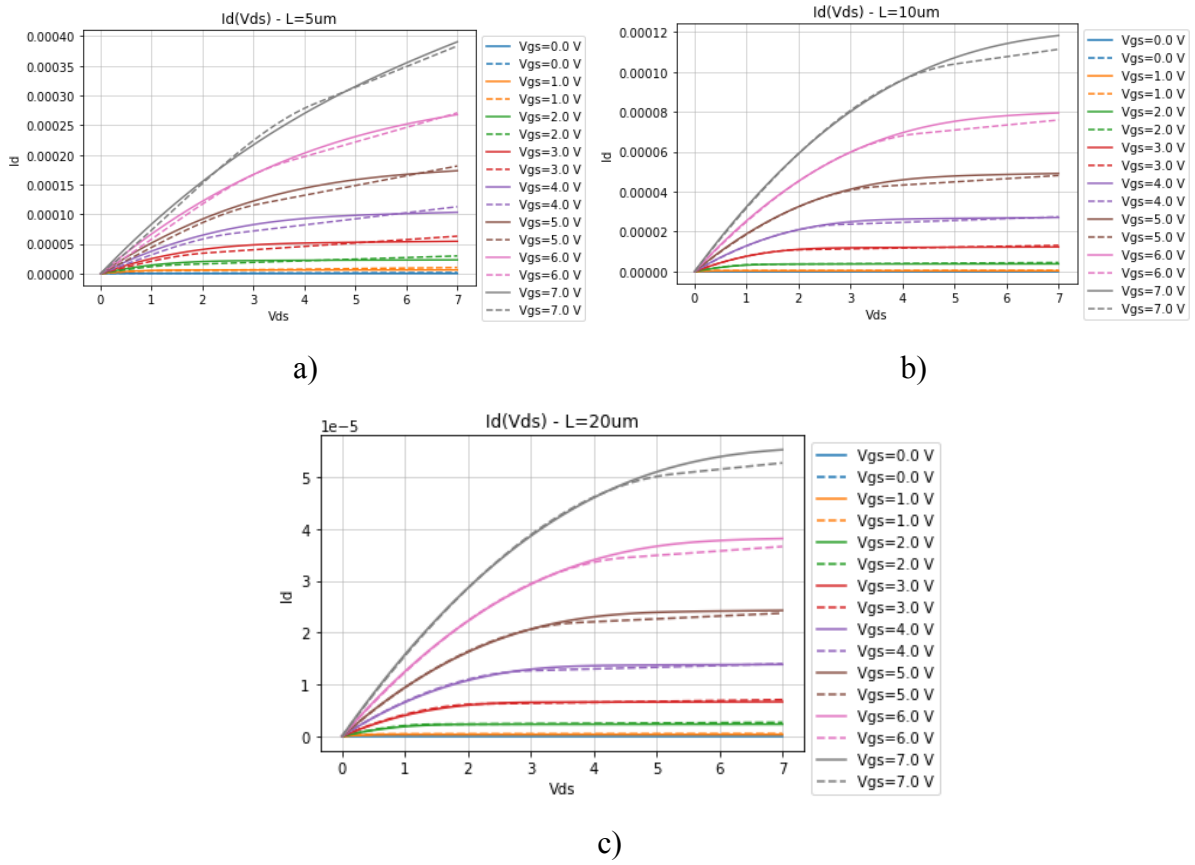


Figura 9 - Erro relativo de $I_D(V_{GS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Após o cálculo de todos os parâmetros de entrada para a implementação do modelo TFT, foram determinados as curvas características $I_D(V_{DS})$ para ambos os circuitos W/L. Na Figura 10 são representadas as respetivas curvas $I_D(V_{DS})$ simuladas e modelada para respetivas dimensões.



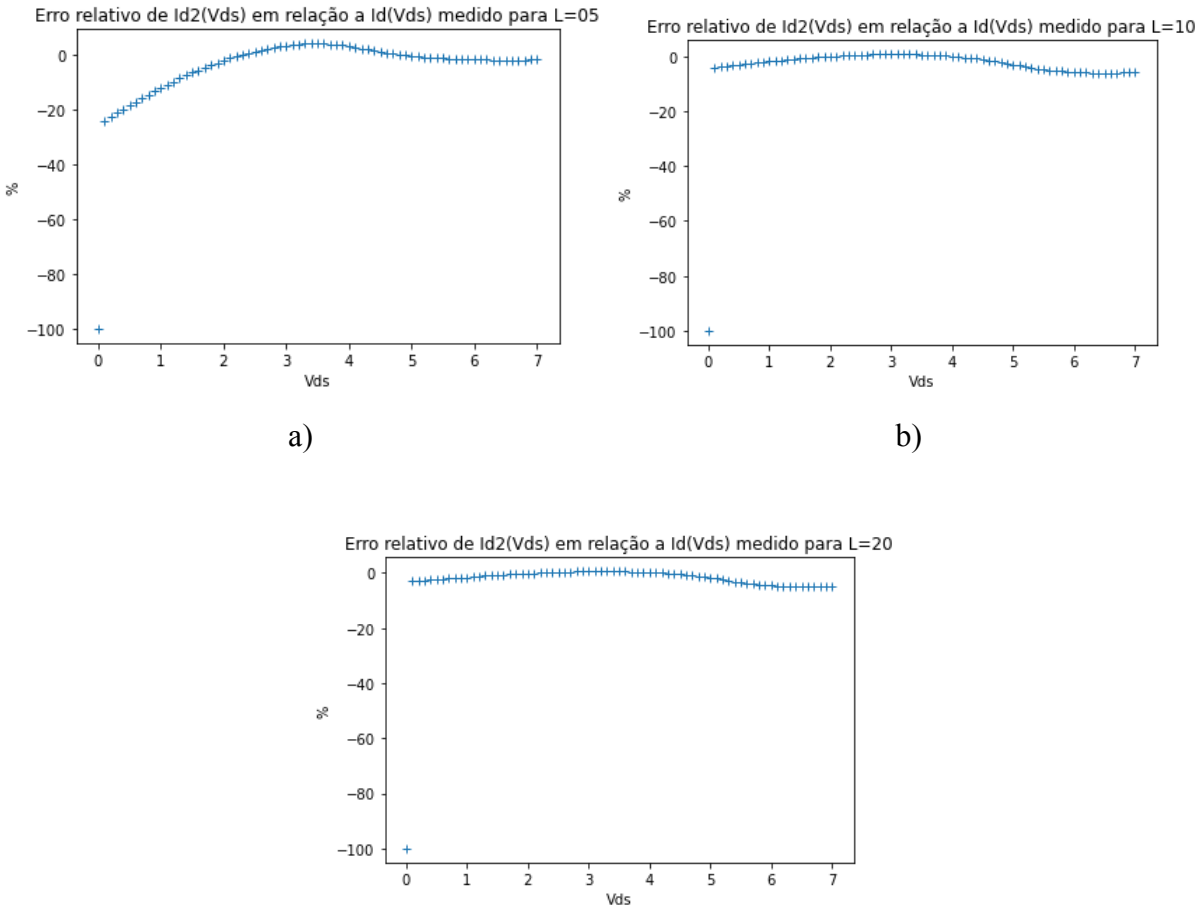
a) b)

c)

Figura 10 - Curvas características $I_D(V_{DS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Estabelecendo uma comparação de modo grosseiro entre os resultados simulados e modelados, podemos afirmar que o modelo *TFT* tem uma boa aproximação, com a exceção dos últimos intervalos em que o *fitting* realizado aos parâmetros não é suficientemente poderoso para aproximar todos os pontos simulados do circuito. Esta característica representa uma desvantagem da utilização do método de aproximação (*fitting*) ao modelo *TFT*.

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 V_{DS} dos respetivos circuitos MOSFET. Na Figura 11 são apresentados os gráficos referentes aos erros relativos para as dimensões $\frac{W}{L} \in \{10,5,2.5\}V$.



a) b)

c)

Figura 11 - Erro relativo de $I_D(V_{DS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Analisando os dados da Figura 11, podemos verificar que o erro relativo é superior para transístores com canais de menores dimensões. Uma vez que apenas se considerou a região $V_{GS} \geq V_T$, é visível que o maior de valor de erro relativo acontece no limiar da zona de saturação.

Parte 2

Considerando, o efeito da resistência de contacto do circuito MOSFET para as mesmas dimensões simuladas anteriormente, é aplicada a segunda parte da metodologia do Capítulo III. Primeiramente é simulada aproximação do modelo TFT com o efeito da resistência R_S para o cálculo das curvas características $I_D(V_{GS})$. A Figura 12 apresenta os resultados obtidos da aplicação do modelo TFT com efeito da resistência R_S, em comparação com os resultados medidos.

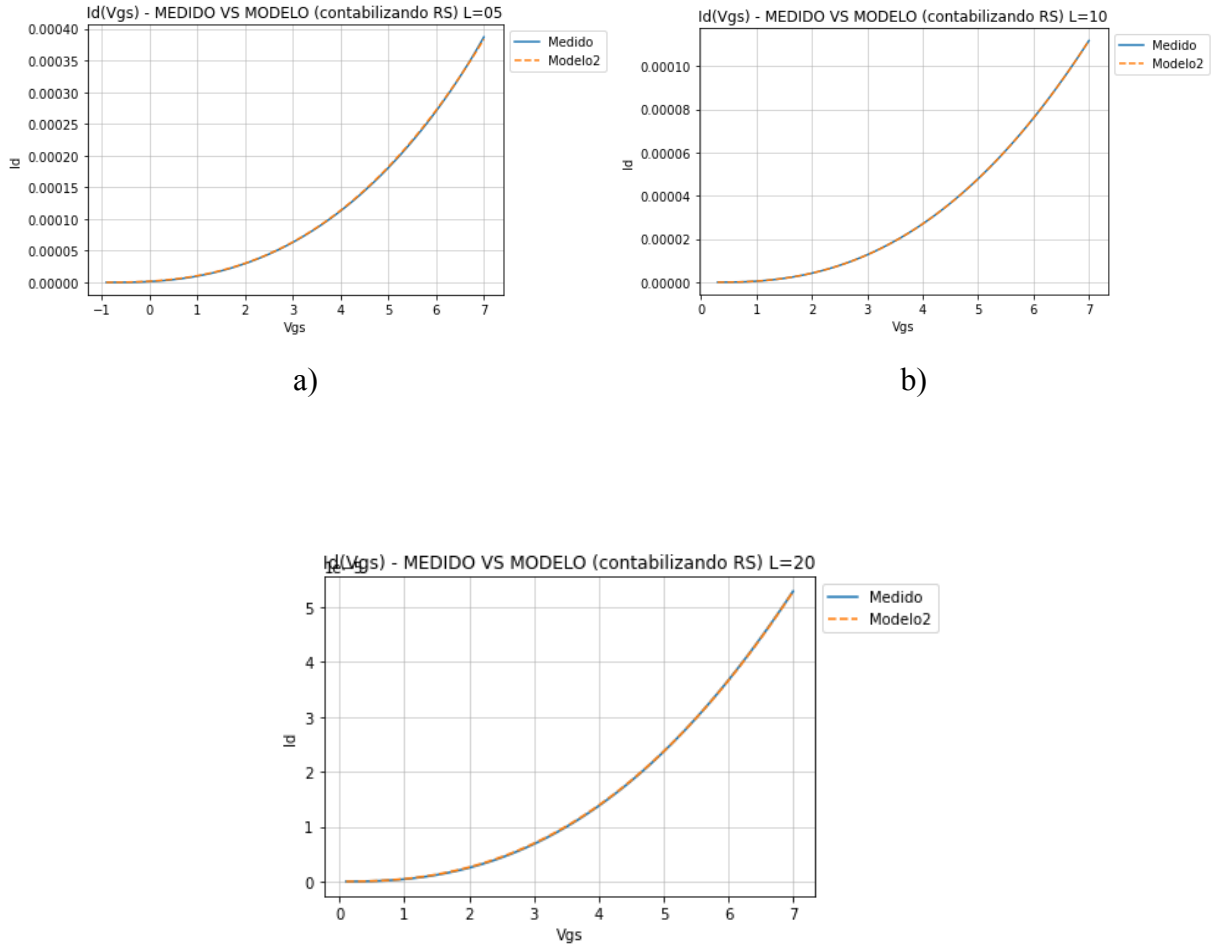
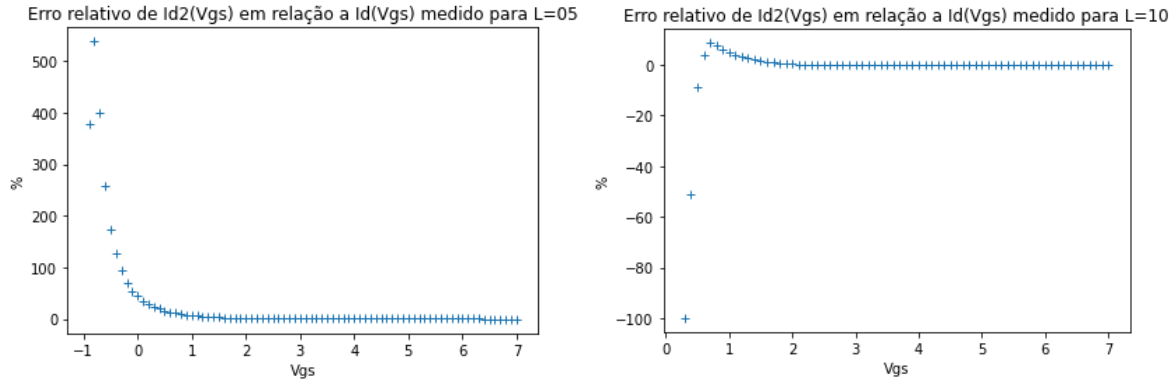
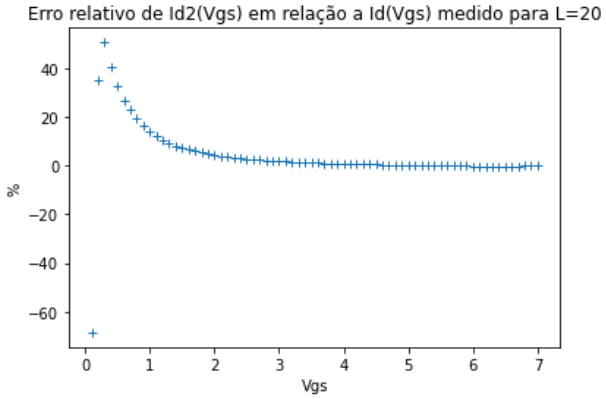


Figura 12 – Curvas características de $I_D(V_{GS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Aplicando o cálculo do erro relativo às curvas características obtidas, podemos observar que o erro diminui com o aumento da tensão V_{DS}, o que mostra a viabilidade do modelo considerando o efeito da resistência R_S. É possível estabelecer também uma relação entre o L e o erro, dado que a convergência do erro para a assintota acontece praticamente no mesmo ponto de tensão V_{DS}. Na Figura 13 são representados os gráficos referentes ao erro relativo das curvas características $I_D(V_{GS})$.



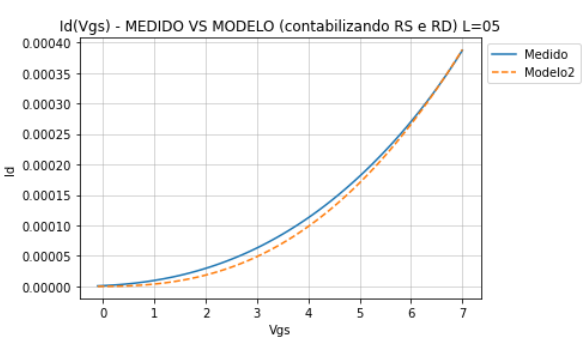
a) b)



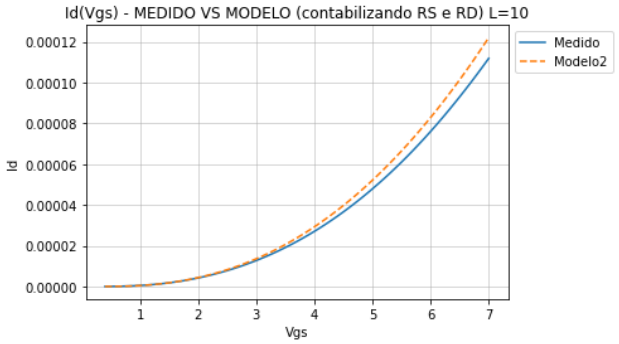
c)

Figura 13 - Erro relativo de $I_D(V_{GS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

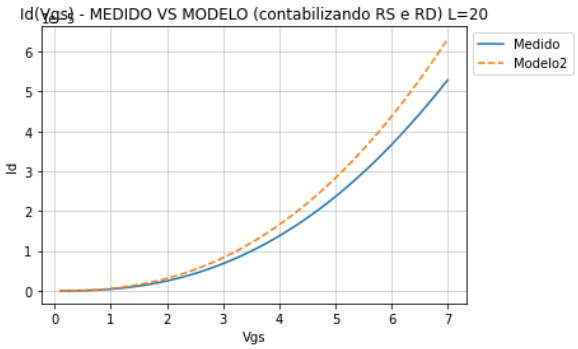
De seguida é simulado o efeito das resistências R_S e R_D para o cálculo das curvas características da $I_D(V_{GS})$, formulando a Figura 14 com os respetivos resultados simulados.



a)



b)



c)

Figura 14 – Curvas características de $I_D(V_{GS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Com a adição da resistência R_D o modelo apresenta algumas lacunas de precisão, o que pode estar relacionado com o método de aproximação utilizado, *fitting* aos parâmetros do modelo TFT. Repetindo o cálculo do erro relativo às curvas características $I_D(V_{GS})$, é apresentada a Figura 15 com os resultados obtidos.

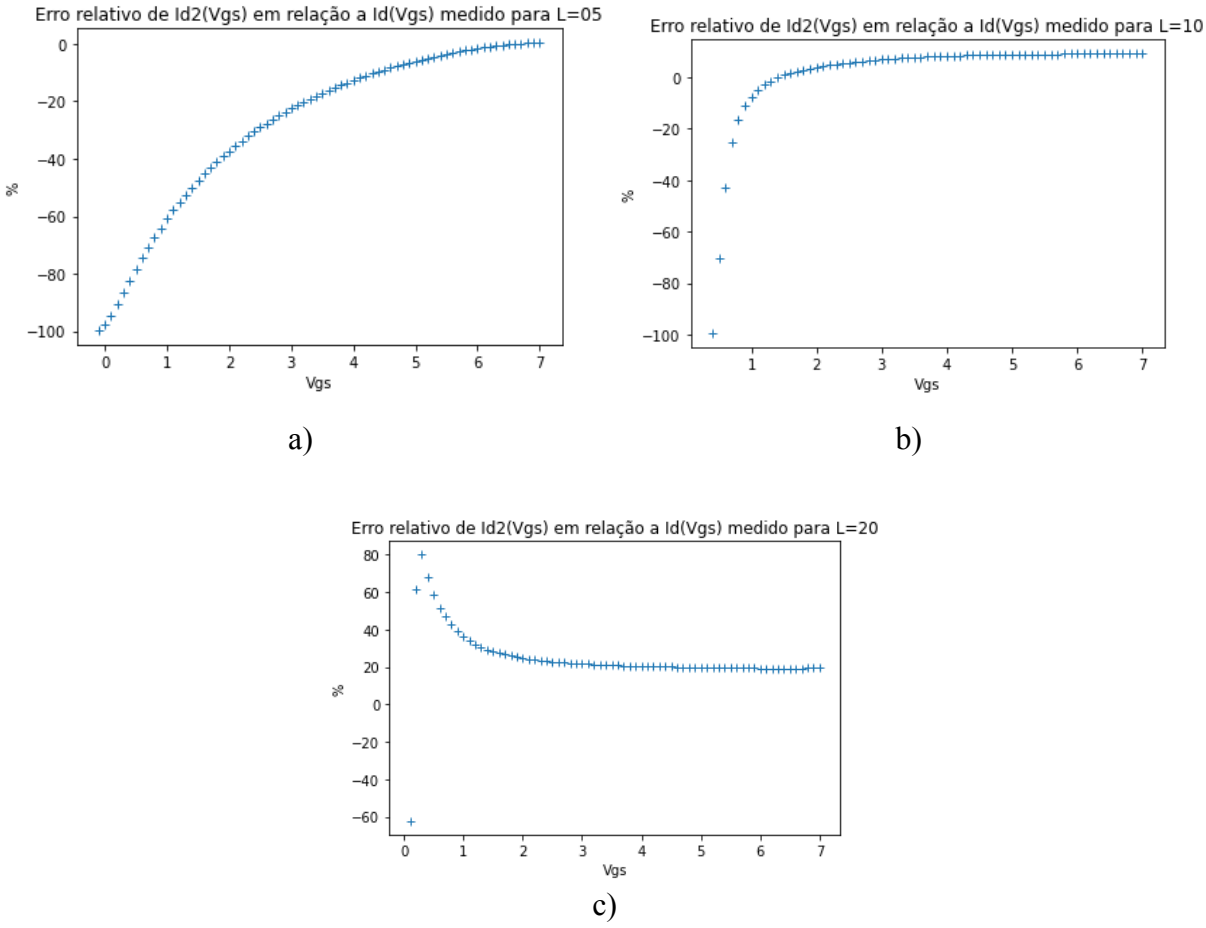


Figura 15 - Erro relativo de $I_D(V_{GS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Analisando o erro relativo entre os dados obtidos e os dados medidos, verifica-se que o modelo TFT tem uma melhor precisão para transístores de menor comprimento, considerando o efeito das resistências de contacto no circuito.

Por fim, foram calculadas as curvas $I_D(V_{DS})$ para uma tensão V_{GS} igual a 7 V, considerando as diferentes relações W/L. A Figura 16 apresenta os resultados obtidos da aplicação do modelo TFT com o efeito das resistências R_S e R_D.

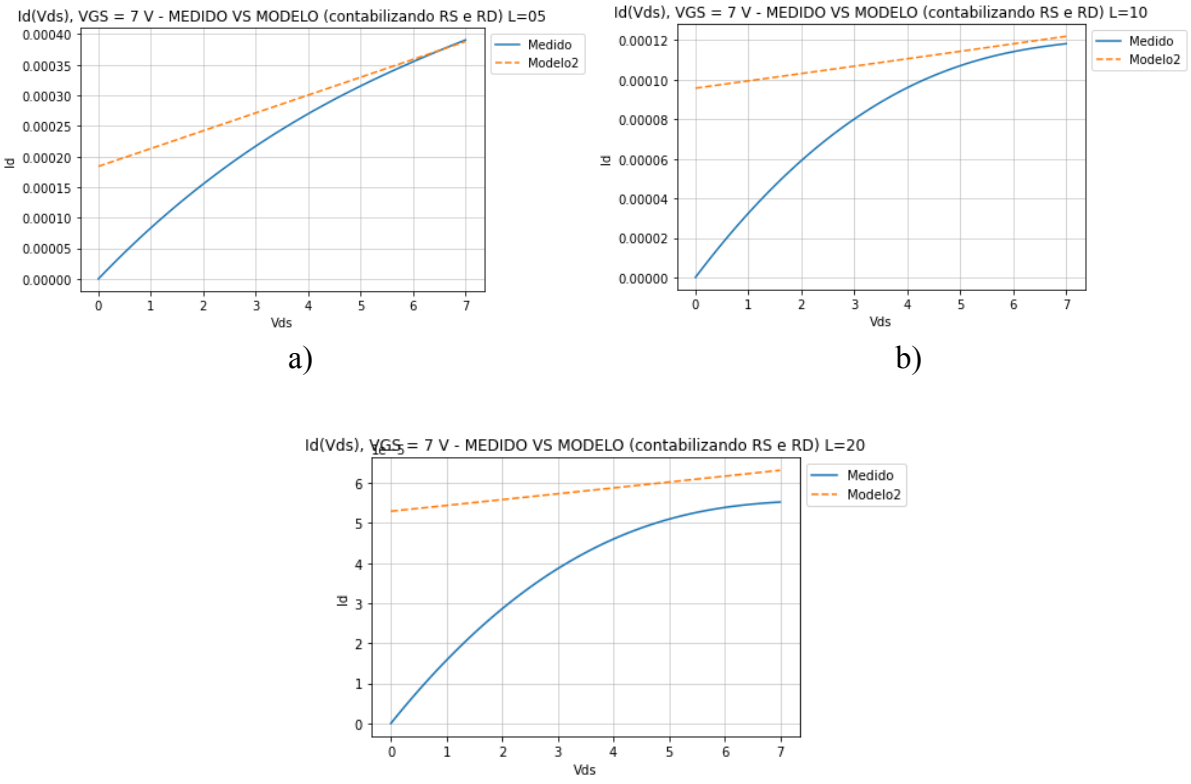


Figura 16 – Curvas características $I_D(V_{DS})$, a) $L = 5um$; b) $L = 10um$; c) $L = 20um$

Como podemos verificar, contrariamente às curvas características $I_D(V_{GS})$, a precisão do modelo para a estimação da corrente $I_D(V_{DS})$, não se verifica tão exata. Este efeito pode estar relacionado com a dificuldade de sintonização dos parâmetros do modelo para a descrição das curvas de corrente para as diferentes tensões.

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 MOSFET para diferentes relações W/L, $\frac{W}{L} \in \{10,5,2.5\}V$.

Assim sendo, inicialmente realizou-se a descrição do modelo teórico TFT, procedendo à análise de resultados para dois cenários operação, um desprezando o efeito da resistência de contacto e um segundo contabilizando o mesmo. Deste modo, através do cálculo dos parâmetros do modelo TFT, foi feita a comparação entre o valores obtidos e os modelados, relacionado a evolução do valor de cada parâmetro com a dimensão do componente MOSFET em estudo. Esta relação permite-nos concluir que o modelo não é escalável pois os valores de K e α não variam proporcionalmente com a relação W/L do transístor. Para além deste facto, os restantes parâmetros variam irregularmente com a dimensão do transístor.

Outro fator que pode influenciar negativamente a aproximação do modelo aos valores simulados deve-se à consideração de pontos em que as medições não são precisas, o que por sua vez pode afetar o *fitting* aos parâmetros de entrada do modelo TFT.

Contudo, é possível concluir que o modelo descreve razoavelmente o funcionamento de um circuito MOSFET, onde através do cálculo do erro relativo se verifica que para valores de V_{GS} maiores o erro é bastante reduzido, no entanto para tensões V_{GS} mais pequenos o valor de I_D é demasiado volátil e torna-se difícil de o prever o comportamento do circuito através de modelo TFT. De notar, que a consideração do efeito das resistências de contato influenciam o comportamento do modelo, sendo mais complexa a estimação dos parâmetros do mesmo. Assim, para os resultados obtidos do modelo com efeito das resistências parasitas em série, verificou-se uma quebra de precisão na aproximação das curvas características $I_D(V_{GS})$, porém ainda é possível assegurar/estimar o comportamento do circuito para as diferentes dimensões.

Bibliografia

[1] R. Rodriguez-Davila, A. Ortiz-Conde, C. Avila-Avendano, and M. A. Quevedo-Lopez, "A New Integration-Based Procedure to Extract the Threshold Voltage, the Mobility Enhancement Factor, and the Series Resistance of Thin-Film MOSFETs," *IEEE Trans. Electron Devices*, vol. 66, no. 7, pp. 2979–2985, 2019, doi: 10.1109/TED.2019.2913699.

[2] E. D. A. C. A. D. F. O. R. Nano-electronics, M. H. Fino, and E. Cad, "Eda/cad for nano-electronics tp6-2020/21," 2020.

[3] E. D. A. Cad, F. O. R. Nanoelctronics, M. H. Fino, and E. Cad, "2020 / 21," 2020.

Anexos

```
# -*- coding: utf-8 -*-
"""
Created on Mon May  3 15:44:23 2021

@author: Rafael
"""

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\\5° Ano\\2° Semestre\\EDA\\Trabalho 2\\Measurement Files')

################################################################################
######################### CARREGAMENTO DE DADOS ################################

#W=50 L=05

data_sat5=pd.read_csv('P1G2_W50L5_Transfer_saturation.csv',sep=',', skiprows=251)
data_sat5.drop('DataName', axis='columns', inplace=True)
data_out5=pd.read_csv('P1G2_W50L5_Output.csv',sep=',', skiprows=256)
data_out5.drop('DataName', axis='columns', inplace=True)

#W=50 L=10

data_sat10=pd.read_csv('P1G2_W50L10_Transfer_saturation.csv',sep=',', skiprows=251)
data_sat10.drop('DataName', axis='columns', inplace=True)
data_out10=pd.read_csv('P1G2_W50L10_Output.csv',sep=',', skiprows=256)
data_out10.drop('DataName', axis='columns', inplace=True)

#W=50 L=20

data_sat20=pd.read_csv('P1G2_W50L20_Transfer_saturation.csv',sep=',', skiprows=251)
data_sat20.drop('DataName', axis='columns', inplace=True)
data_out20=pd.read_csv('P1G2_W50L20_Output.csv',sep=',', skiprows=256)
data_out20.drop('DataName', axis='columns', inplace=True)

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

################################################################################
################################## FUNCS #######################################

def plotter_Id_vds(L_val, data_out):
    plt.figure()
    plt.title('Id(Vds) - L=%02d' %(L_val))
    plt.ylabel('Id')
    plt.xlabel('Vds')
    for i in data_out[' VG'].unique():
        plt.plot(data_out[' VD'][data_out[' VG']==i], data_out[' ID'][data_out[' VG']==i],label='Vgs=%.1f V'
 %(i))
    plt.legend(bbox_to_anchor=(1, 1))
    plt.grid(linewidth=0.5)

def plotter_Id_vgs(params,i,data_sat):
    # c) Plot Id(Vgs) from the model, againts Id from measurements
    Id2=get_Id(np.arange(-4,7.1,0.1),7,params['Vt'][i],params['m'][i],params['Lambda'][i],params['K'][i])

    # create figure and axis objects with subplots()
    fig,ax = plt.subplots()

    plt.title('Id(Vgs) - Medidos vs Modelo 1 - L=%02d' %(5*(2**i)))

    l1=ax.plot(data_sat[' VG'][data_sat[' VG'] > params['Vt'][i]], Id2[data_sat[' VG'] > params['Vt'][i]], '
b.')[0]
    l2=ax.plot(data_sat[' VG'][data_sat[' VG'] > params['Vt'][i]], data_sat[' ID'][data_sat[' VG'] > params[
'Vt'][i]], 'b')[0]
    ax.set_xlabel("Vgs")
    ax.set_ylabel("Id")

    # d) Repeat c) in logaritmic scale
    # twin object for two different y-axis on the sample plot
    ax2=ax.twinx()
    l3=ax2.semilogy(data_sat[' VG'][data_sat[' VG'] > params['Vt'][i]], Id2[data_sat[' VG'] > params['Vt'][i
]], 'g.')[0]
```

```
    l4=ax2.semilogy(data_sat[' VG'][data_sat[' VG'] > params['Vt'][i]], data_sat[' ID'][data_sat[' VG'] > pa
rams['Vt'][i]], color='g')[0]
    ax2.set_ylabel("log(ID)")

    line_labels = ["Medido", "Modelo", "Medido (log)", "Modelo (log)"]
    fig.legend([l1, l2, l3, l4],     # The line objects
           labels=line_labels,   # The labels for each line
           loc="lower right",   # Position of legend
           borderaxespad=0.1,    # Small spacing around legend box
           )
    plt.show()
    del ax, ax2, fig, l1, l2, l3, l4

def plotter_Id_vds_vs(L_val,data_out, I_out):
    plt.figure()
    plt.title('Id(Vds) - L=%02d' %(L_val))
    plt.ylabel('Id')
    plt.xlabel('Vds')
    for i in I5[' VG'].unique():
        plt.plot(data_out[' VD'][data_out[' VG']==i], data_out[' ID'][data_out[' VG']==i],label='Vgs=%.1f V'
 %(i))
        plt.plot(I_out[' VD'][I_out[' VG']==i], I_out[' ID'][I_out[' VG']==i],"--
",label='Vgs=%.1f V' %(i), color=plt.gca().lines[-1].get_color())
    plt.legend(bbox_to_anchor=(1, 1))
    plt.grid(linewidth=0.5)

def plotter_Id_model2(data_sat, data_out, Rs_Ix, Rd_Ix, parameters, i):

    plt.figure()
    plt.title('Id(Vgs) - MEDIDO VS MODELO (contabilizando RS) L=%02d' %parameters['L'][i])
    plt.ylabel('Id')
    plt.xlabel('Vgs')
    plt.plot(data_sat[' VG'][data_sat[' VG']>=parameters['Vt'][i]],
             data_sat[' ID'][data_sat[' VG']>=parameters['Vt'][i]],
             label="Medido")
    plt.plot(data_sat[' VG'][data_sat[' VG']>=parameters['Vt'][i]],
             Rs_Ix,
             "--",label="Modelo2")
    plt.legend(bbox_to_anchor=(1, 1))
    plt.grid(linewidth=0.5)


    plt.figure()
    plt.title('Id(Vgs) - MEDIDO VS MODELO (contabilizando RS e RD) L=%02d' %parameters['L'][i])
    plt.ylabel('Id')
    plt.xlabel('Vgs')
    plt.plot(data_sat[' VG'][data_sat[' VG']>=parameters['Vt'][i+1]],
             data_sat[' ID'][data_sat[' VG']>=parameters['Vt'][i+1]],
             label="Medido")
    plt.plot(data_sat[' VG'][data_sat[' VG']>=parameters['Vt'][i+1]],
             Rd_Ix,
             "--",label="Modelo2")
    plt.legend(bbox_to_anchor=(1, 1))
    plt.grid(linewidth=0.5)

    plt.figure()
    plt.title('Id(Vds), VGS = 7 V - MEDIDO VS MODELO (contabilizando RS e RD) L=%02d' %parameters['L'][i])
    plt.ylabel('Id')
    plt.xlabel('Vds')
    plt.plot(data_out[' VD'][568:],data_out[' ID'][568:], label="Medido")
    plt.plot(data_out[' VD'][568:],
             get_Rd_L((data_out[' VG'][568:],data_out[' VD'][568:],data_out[' ID'][568:]),
                      parameters['Vt'][i+1], parameters['m'][i+1], parameters['K'][i+1],
                      parameters['Rs'][i+1], parameters['Rd'][i+1], parameters['Lambda'][i+1]),
             "--",label="Modelo2")
    plt.legend(bbox_to_anchor=(1, 1))
    plt.grid(linewidth=0.5)

def plotter_error(a, b, xaxis, xlabel, title):
    E_rel=((a-b)/b)*100
    plt.figure()
    plt.ylabel('%')
    plt.xlabel(xlabel)
    plt.title(title)
    plt.plot(xaxis, E_rel, '+')

def gm(Id,Vgs): #gm= dId/dVgs
    return np.diff(Id)/np.diff(Vgs)
```

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

def get_K(Vgs, K):
    return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0,
                                                     lambda Vgs:K*((Vgs-Vt)**m)*(1+L*7)])

def get_lambda(Id1,Id2,Vds1,Vds2):
    return (Id1-Id2)/(Id2*Vds1-Id1*Vds2)

def get_Id(Vgs, Vds, Vt, m, L, K):
    return np.piecewise(Vgs, [Vgs < Vt, Vgs >= Vt],[lambda Vgs:0,
                                                     lambda Vgs:K*((Vgs-Vt)**m)*(1+L*Vds)])

def get_alpha(Vds, alpha):
    return np.piecewise(Vds, [Vds<alpha*(7-Vt), Vds>=alpha*(7-Vt)],
                        [lambda Vds: (2*K/alpha)*((7-Vt)**gamma)*(7-Vt-(Vds/(2*alpha)))*Vds*(1+L*Vds),
                         lambda Vds: K*((7-Vt)**(2+gamma))*(1+L*Vds)])

def calc_parameters(data_out, data_sat):

    global m, Vt, gamma, K, L, alpha
    # Obtenção dos valores de gm
    gmx= gm(data_sat[' ID'], data_sat[' VG'])

    # Obtenção dos valores de Id/gm
    id_gm=(data_sat[' ID'][1:]/gmx).to_numpy()

    #Obtenção de m, Vt e gamma= m - 2
    xx,xy=curve_fit(get_m_vt,data_sat[' VG'][1:].values,id_gm)

    m=xx[0]
    Vt=xx[1]
    gamma= m - 2
    #Obtenção de Lambda
    L=get_lambda(data_out[' ID'][629], data_out[' ID'][638],
                 data_out[' VD'][629], data_out[' VD'][638])

    #Obtenção de K=K_gsat*(W/L)
    xx,xy=curve_fit(get_K,data_sat[' VG'].values,data_sat[' ID'].values)
    K=xx[0]

    xx, xy=curve_fit(get_alpha,
                     data_out[' VD'][data_out[' VG']==7].values,
                     data_out[' ID'][data_out[' VG']==7].values)
    alpha=xx[0]
    U=[Vt, m , gamma, K, L, alpha]
    del Vt, m , gamma, K, L, alpha
    return U

def get_Idout(Vds, Vx, alpha, K, gamma, m, L):
    return np.piecewise(Vds, [Vds<alpha*(Vx), Vds>=alpha*(Vx)],
                        [lambda Vds: (2*K/alpha)*((Vx)**gamma)*(Vx-(Vds/(2*alpha)))*Vds*(1+L*Vds),
                         lambda Vds: K*(Vx**m)*(1+L*Vds)])

def calc_Iout(data_out,params,j):
    I=[]
    for i in range(0,len(data_out),1):
        if(data_out[' VG'][i]>=params['Vt'][j]):
            I.append([data_out[' VG'][i], data_out[' VD'][i],
                      get_Idout(data_out5[' VD'][i], data_out5[' VG'][i]-params['Vt'][j],
                                params['alpha'][j],params['K'][j],params['gamma'][j],
                                params['m'][j],params['Lambda'][j])])
    I_df = pd.DataFrame(I)
    I_df.columns=[' VG',' VD',' ID']
    return I_df

def get_Rs(data, Vt, m, K, Rs):
    Vgs,Id=data
    return K*((Vgs-Rs*Id-Vt)**m)

def get_Rd_L(data, Vt, m, K, Rs, Rd, L):
    Vgs,Vds,Id=data
    return K*((Vgs-Rs*Id-Vt)**m)*(1+L*(Vds-(Rd+Rs)*Id))

def calc_parameters2(data_out, data_sat, L, p00, bounds_Rs, bounds_Rd):
    U=[]
```

```
    xx,xy=curve_fit(get_Rs, (data_sat[' VG'][data_sat[' VG']>=p00['Vt']],data_sat[' ID'][data_sat[' VG']>=p0
0['Vt']]),
                    data_sat[' ID'][data_sat[' VG']>=p00['Vt']],maxfev=4000,
                    p0=[p00['Vt'], p00['m'], p00['K'], 100],
                    bounds=bounds_Rs)
    U.insert(0,[xx[0],xx[1],xx[2],xx[3],0,0,L])

    xx,xy=curve_fit(get_Rd_L, (data_out[' VG'][629:],data_out[' VD'][629:],data_out[' ID'][629:]),
                    data_out[' ID'][629:],maxfev=4000,
                    p0=[xx[0],xx[1],xx[2],xx[3],100,p00['Lambda']],
                    bounds=bounds_Rd)
    U.insert(1,[xx[0],xx[1],xx[2],xx[3],xx[4],xx[5],L])

    return U

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

# Cálculo Parâmetros Modelo 1

U=[]
U.append(calc_parameters(data_out5,data_sat5))
U.append(calc_parameters(data_out10,data_sat10))
U.append(calc_parameters(data_out20,data_sat20))
parameters = pd.DataFrame(U)
parameters.columns = ['Vt','m','gamma','K', 'Lambda', 'alpha']
parameters['L'] = [5,10,20]
del U

# Cálculo das correntes de saída Modelo 1
I5=calc_Iout(data_out5,parameters,0)
I10=calc_Iout(data_out10,parameters,1)
I20=calc_Iout(data_out20,parameters,2)

# Cálculo Parâmetros Modelo 2
parameters2 = pd.DataFrame()
parameters2= parameters2.append(calc_parameters2(data_out5,data_sat5,5,
                                                 parameters.loc[0],
                                                 ([-2,2.5,0,0],[1,4,0.001,300]),
                                                 ([-2,2.5,0,0,0,0],[1.5,4,0.001,300,300,1])))
parameters2= parameters2.append(calc_parameters2(data_out10,data_sat10,10,
                                                 parameters.loc[1],
                                                 ([-2,2,0,0],[1,3,0.001,200]),
                                                 ([-2,2,0,0,0,0],[1.5,3,0.001,200,200,1])))
parameters2= parameters2.append(calc_parameters2(data_out20,data_sat20,20,
                                                 parameters.loc[2],
                                                 ([-2,2,0,0],[1,3,0.001,200]),
                                                 ([-2,2,0,0,0,0],[1.5,3,0.001,200,200,1])))
parameters2.columns = ['Vt','m','K', 'Rs', 'Rd', 'Lambda','L']
parameters2 = parameters2.reset_index(drop=True)

# Cálculo das Correntes tendo em conta Rs e Rd (Saturação) Modelo 2
Rs_I5=get_Rs((data_sat5[' VG'][data_sat5[' VG']>=parameters2['Vt'][0]],
              data_sat5[' ID'][data_sat5[' VG']>=parameters2['Vt'][0]]),
             parameters2['Vt'][0], parameters2['m'][0], parameters2['K'][0],parameters2['Rs'][0])

Rs_I10=get_Rs((data_sat10[' VG'][data_sat10[' VG']>=parameters2['Vt'][2]],
               data_sat10[' ID'][data_sat10[' VG']>=parameters2['Vt'][2]]),
              parameters2['Vt'][2], parameters2['m'][2], parameters2['K'][2],parameters2['Rs'][2])

Rs_I20=get_Rs((data_sat20[' VG'][data_sat20[' VG']>=parameters2['Vt'][4]],
               data_sat20[' ID'][data_sat20[' VG']>=parameters2['Vt'][4]]),
              parameters2['Vt'][4], parameters2['m'][4], parameters2['K'][4],parameters2['Rs'][4])

Rd_I5=get_Rd_L((data_sat5[' VG'][data_sat5[' VG']>=parameters2['Vt'][1]],
                data_sat5[' VD'][data_sat5[' VG']>=parameters2['Vt'][1]],
                data_sat5[' ID'][data_sat5[' VG']>=parameters2['Vt'][1]]),
                parameters2['Vt'][1], parameters2['m'][1], parameters2['K'][1],
                parameters2['Rs'][1], parameters2['Rd'][1], parameters2['Lambda'][1])

Rd_I10=get_Rd_L((data_sat10[' VG'][data_sat10[' VG']>=parameters2['Vt'][3]],
                 data_sat10[' VD'][data_sat10[' VG']>=parameters2['Vt'][3]],
                 data_sat10[' ID'][data_sat10[' VG']>=parameters2['Vt'][3]]),
                parameters2['Vt'][3], parameters2['m'][3], parameters2['K'][3],
                parameters2['Rs'][3], parameters2['Rd'][3], parameters2['Lambda'][3])

Rd_I20=get_Rd_L((data_sat20[' VG'][data_sat20[' VG']>=parameters2['Vt'][5]],
                 data_sat20[' VD'][data_sat20[' VG']>=parameters2['Vt'][5]],
                 data_sat20[' ID'][data_sat20[' VG']>=parameters2['Vt'][5]]),
```

```
                parameters2['Vt'][5], parameters2['m'][5], parameters2['K'][5],
                parameters2['Rs'][5], parameters2['Rd'][5], parameters2['Lambda'][5])
##########################################################################
################################### PLOTS ################################

#Plot Id(Vgs) Saturation

plt.figure()
plt.title('Id(Vgs) Saturation, VDS=7 V - MEDIDOS')
plt.ylabel('Id')
plt.xlabel('Vgs')
plt.plot(data_sat5[' VG'],data_sat5[' ID'],label='L=5')
plt.plot(data_sat10[' VG'],data_sat10[' ID'],label='L=10')
plt.plot(data_sat20[' VG'],data_sat20[' ID'],label='L=20')
plt.legend(bbox_to_anchor=(1, 1))
plt.grid(linewidth=0.5)

#Plot Id(Vds) for several Vgs

plotter_Id_vds(5, data_out5)
plotter_Id_vds(10, data_out10)
plotter_Id_vds(20, data_out20)

#Plot Id(Vgs) from the model, againts Id from measurements

plotter_Id_vgs(parameters,0,data_sat5)
Id2=get_Id(np.arange(-
4,7.1,0.1),7,parameters['Vt'][0],parameters['m'][0],parameters['Lambda'][0],parameters['K'][0])
plotter_error(Id2[data_sat5[' VG'] > parameters['Vt'][0]],
              data_sat5[' ID'][data_sat5[' VG'] > parameters['Vt'][0]],
              data_sat5[' VG'][data_sat5[' VG'] > parameters['Vt'][0]],
              "Vgs", "Erro relativo de Id2(Vgs) em relação a Id(Vgs) medido para L=05")

plotter_Id_vgs(parameters,1,data_sat10)
Id2=get_Id(np.arange(-
4,7.1,0.1),7,parameters['Vt'][1],parameters['m'][1],parameters['Lambda'][1],parameters['K'][1])
plotter_error(Id2[data_sat10[' VG'] > parameters['Vt'][1]],
              data_sat10[' ID'][data_sat10[' VG'] > parameters['Vt'][1]],
              data_sat10[' VG'][data_sat10[' VG'] > parameters['Vt'][1]],
              "Vgs", "Erro relativo de Id2(Vgs) em relação a Id(Vgs) medido para L=10")

plotter_Id_vgs(parameters,2,data_sat20)
Id2=get_Id(np.arange(-
4,7.1,0.1),7,parameters['Vt'][2],parameters['m'][2],parameters['Lambda'][2],parameters['K'][2])
plotter_error(Id2[data_sat20[' VG'] > parameters['Vt'][2]],
              data_sat20[' ID'][data_sat20[' VG'] > parameters['Vt'][2]],
              data_sat20[' VG'][data_sat20[' VG'] > parameters['Vt'][2]],
              "Vgs", "Erro relativo de Id2(Vgs) em relação a Id(Vgs) medido para L=20")

# Plot Id(Vds) for several Vgs, , againts Id from measurements
plotter_Id_vds_vs(5,data_out5, I5)
plotter_error(I5[' ID'][I5[' VG']==7].values,
              data_out5[' ID'][data_out5[' VG']==7].values,
              data_out5[' VD'][data_out5[' VG']==7],
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=05")

plotter_Id_vds_vs(10,data_out10, I10)
plotter_error(I10[' ID'][I10[' VG']==7].values,
              data_out10[' ID'][data_out10[' VG']==7].values,
              data_out10[' VD'][data_out10[' VG']==7],
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=10")

plotter_Id_vds_vs(20,data_out20, I20)
plotter_error(I20[' ID'][I20[' VG']==7].values,
              data_out20[' ID'][data_out20[' VG']==7].values,
              data_out20[' VD'][data_out20[' VG']==7],
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=20")

# Plots Id
plotter_Id_model2(data_sat5,data_out5, Rs_I5, Rd_I5, parameters2, 0)
plotter_error(Rs_I5.values,
              data_sat5[' ID'][data_sat5[' VG']>=parameters2['Vt'][0]].values,
              data_sat5[' VG'][data_sat5[' VG']>=parameters2['Vt'][0]].values,
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=05")
plotter_error(Rd_I5.values,
              data_sat5[' ID'][data_sat5[' VG']>=parameters2['Vt'][1]].values,
              data_sat5[' VG'][data_sat5[' VG']>=parameters2['Vt'][1]].values,
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=05")
```

```
plotter_Id_model2(data_sat10,data_out10, Rs_I10, Rd_I10, parameters2, 2)
plotter_error(Rs_I10.values,
              data_sat10[' ID'][data_sat10[' VG']>=parameters2['Vt'][2]].values,
              data_sat10[' VG'][data_sat10[' VG']>=parameters2['Vt'][2]].values,
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=10")
plotter_error(Rd_I10.values,
              data_sat10[' ID'][data_sat10[' VG']>=parameters2['Vt'][3]].values,
              data_sat10[' VG'][data_sat10[' VG']>=parameters2['Vt'][3]].values,
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=10")


plotter_Id_model2(data_sat20,data_out20, Rs_I20, Rd_I20, parameters2, 4)
plotter_error(Rs_I20.values,
              data_sat20[' ID'][data_sat20[' VG']>=parameters2['Vt'][4]].values,
              data_sat20[' VG'][data_sat20[' VG']>=parameters2['Vt'][4]].values,
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=20")
plotter_error(Rd_I20.values,
              data_sat20[' ID'][data_sat20[' VG']>=parameters2['Vt'][5]].values,
              data_sat20[' VG'][data_sat20[' VG']>=parameters2['Vt'][5]].values,
              "Vds", "Erro relativo de Id2(Vds) em relação a Id(Vds) medido para L=20")
################################################################################
################################################################################
```