IMPLEMENTATION OF A PHYSICS BASED MODEL FOR THE GaAs MESFET

J.W. Bandler, Q.J. Zhang and Q. Cai

SOS-90-3-U

January 1990

IMPLEMENTATION OF A PHYSICS BASED MODEL FOR THE GaAs MESFET

J.W. Bandler, Q.J. Zhang and Q. Cai

Abstract A physics based model for the GaAs MESFET is dynamically integrated into a linear/nonlinear simulation environment. The model is based on the work of Khatibzadeh and Trew. The program, written in C, has been embedded into our research system called McCAE. This document gives a description of all modules related to the physical model.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants STR0040923 and OGP0007239.

The authors are with the Simulation Optimization Systems Research Laboratory and the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Canada L8S 4L7.

I. INTRODUCTION

The physics based model for GaAs MESFETs has been described in our previous report [1]. The model accepts physical/geometrical/process parameters of the device as input data. Simulation and optimization can be directly applied to the physical characteristics of the devices with this model, representing a breakthrough over the conventional equivalent circuit approach. The model is very useful for both device and circuit design optimization and is quite suitable for statistical modeling and yield optimization.

This document gives a description of the program for the physics based model for the MESFET. The program is written in C. It includes three main files: (1) trew.c for the physical model, (2) spline.c for cubic spline interpolation and (3) integral.c for the calculation of integrations used by routines in trew.c, and three include files: "trew.h", "spline.h" and "integral.h". The three principal files are described in detail in the following sections.

II. DESCRIPTION OF FILE trew.c

This file contains two main functions: initrew and ctrew. initrew is devoted to all pre-calculations such as initialization of the parameters, evaluation of constants and creation of cubic spline interpolation parameters for $F_1(d)$ w.r.t. d, etc. See (21) in [1]. ctrew calculates the drain and source conduction currents I_d and I_s and accumulation charges in the gate, drain and source electrodes, i.e., Q_g, Q_d and Q_s. The equations for calculating I_d, I_s, Q_g, Q_d and Q_s are described in detail in [1].

In this section, the arguments of the subprograms are listed. They are also classified as input arguments and output arguments. The notation used in this section is consistent with our previous report [1].

A. initrew

initrew(PA)

PA is a float array of dimension 20. It contains the input parameters of the model. The values for the elements of PA must be supplied by the user. The elements of PA are defined as follows.

PA[0] gate length L (μm).

PA[1] gate width W (μm).

PA[2] channel thickness a (μm).

PA[3] relative dielectric constant ε_r.

PA[4] critical electric field E_c (V/μm).

PA[5] saturation electronic velocity v_s (μm/ns).

PA[6] built-in potential V_{bi} (V).

PA[7] low-field mobility μ_0 (μm^2/Vns).

PA[8] high-field diffusion coefficient D (μm^2/ns).

PA[9] doping density (for uniform doping) N_d (μm^{-3}).

PA[10] model parameter λ for calculating transition function (see (12) in [1]) (μm).

PA[11] model parameter α for evaluating the weight factor ω_i for calculating the average mobility (see (33) in [1]).

PA[12] time delay τ (ns).

PA[13] model parameter N_{d1} for calculating doping profile (μm^{-3}).

PA[14] model parameter Y_0 for calculating doping profile (μm).

PA[15] model parameter Y_{n0} for calculating doping profile (μm).

PA[16] model parameter Y_{n1} for calculating doping profile (μm).

The representations for three typical doping profiles are

Uniform doping

$$N(y) = N_d ,$$

exponential doping

$$N(y) = N_d \exp[- 0.5(y - Y_0)/Y_{n0}]$$

and piecewise doping

$$N(y) = \begin{cases} N_d + N_{d1} & y \le Y_0 \\ N_d \exp[-(y - Y_0)/Y_{n0}] & \\ \quad + N_{d1}\exp[-(y - Y_0)/Y_{n1}] & y > Y_0 \end{cases}$$

PA[17] model parameter A for calculating the velocity-electrical (v-E) curve (V/μm).

PA[18] model parameter B for calculating the v-E curve.

PA[19] model parameter C for calculating the v-E curve.

PA[20] model parameter H for calculating the v-E curve.

See (9) and (10) in [1] for details of A, B, C and H.

B. ctrew

ctrew(V1f, Vgsf, Vdsf, Id, Is, Qg, Qd, Qs)

V1f is a pointer to a float argument. *V1f represents the model variable V_1. (INPUT).

Vgsf is a pointer to a float argument. *Vgsf represents intrinsic gate voltage V_{gs}. (INPUT).

Vdsf is a pointer to a float argument. *Vdsf is the value of intrinsic drain voltage V_{ds}. (INPUT).

Id is a pointer to a float argument. *Id represents the conduction drain current I_d. (OUTPUT).

Is is a pointer to a float argument. *Is represents the conduction source current I_s. (OUTPUT).

Qg is a pointer to a float argument. *Qg is the value of accumulation charge in the gate electrode. (OUTPUT).

Qd is a pointer to a float argument. *Qd represents the accumulation charge in the drain electrode. (OUTPUT).

Qs is a pointer to a float argument. *Qs represents the accumulation charge in the source electrode. (OUTPUT).

III. DESCRIPTION OF FILE spline.c

This file is used for cubic spline interpolation to a set of data. Given a set of data (x_i, y_i), i = 1, 2, •••, N, which satisfy

$$x_1 < x_2 < \cdots < x_N .$$

A cubic spline s(x) interpolates the above partition in the interval $[x_1, x_N]$. It is a function for which $s(x_i) = y_i$. The function consists of N-1 cubic polynomials f_i defined in the range $[x_i, x_{i+1}]$, respectively. In addition the cubic polynomials f_i are joined at x_i (i = 2, 3, •••, N-1) in such a way that s(x) is twice differentiable. In our program the cubic polynomials are chosen to be of the form

$$f_i(x) = A_i(x - x_i)^3 + B_i(x - x_i)^2 + C_i(x - x_i) + D_i \qquad x_i \le x \le x_{i+1}$$

where A_i, B_i, C_i and D_i are the coefficients to be determined. Given the values of the first or second derivatives at the left-end and right-end points: $y'(x_1)$ and $y'(x_N)$ or $y''(x_1)$ and $y''(x_N)$, the coefficients A_i, B_i, C_i and D_i can be uniquely determined [2].

In this file there is one key module, CubicSplineWithDeriv(), and two functions, CubicSplineValue() and CubicSplineDeriv. Module CubicSplineWithDeriv() is used to calculate the coefficients A_i, B_i, C_i and D_i. Function CubicSplineValue() is applied to evaluate the value of y at any given point x. Function CubicSplineDeriv() is used to calculate the derivatives of y w.r.t. x at any given point x.

A. CubicSplineWithDeriv

CubicSplineWithDeriv(Ndata, Xdata, Fdata, Ideriv, Lderiv, Rderiv, Coef)

Ndata is an integer argument denoting the number of data points N. (INPUT).

Xdata is a pointer to a float array of dimension Ndata which contains all the data of

x_i. (INPUT).

Fdata is a pointer to a float array of dimension Ndata which contains all the data of y_i. (INPUT).

Ideriv is an integer argument which indicates the order of derivatives in the left-end (minimum of x_i) and right-end (maximum of x_i) points. It must be in the range $0 \leq$ Ideriv ≤ 3. (INPUT).

Ideriv = 0	$y''(x_1) = 0$	$y''(x_N) = 0$
Ideriv = 1	$y'(x_1)$ = Lderiv	$y'(x_N)$ = Rderiv
Ideriv = 2	$y''(x_1)$ = Lderiv	$y''(x_N)$ = Rderiv
Ideriv = 3	$y''(x_1)$ = Lderiv•$y''(x_2)$	$y''(x_N)$ = Rderiv•$y''(x_{N-1})$

Lderiv is a float argument containing the derivative at the left-end point. (INPUT).

Rderiv is a float argument containing the derivative at the right-end point. (INPUT).

Coef is a matrix containing all the coefficients A_i, B_i, C_i and D_i. (OUTPUT).

B. Function CubicSplineValue

CubicSplineValue(x, Ndata, Xdata, Coef)

x is a float argument indicating the x point at which the value of y is to be found. (INPUT).

Ndata is an integer argument containing the number of data points N. (INPUT). The same argument is also used in CubicSplineWithDeriv().

Xdata is a pointer to a float array of dimension Ndata containing all the data of x_i. (INPUT). The same argument is used in CubicSplineWithDeriv().

Coef is the coefficient matrix set up by CubicSplineWithDeriv(). (INPUT).

The return value of CubicSplineValue() is y corresponding to the point x.

C. Function CubicSplineDeriv

CubicSplineDeriv(x, Ndata, Xdata, Coef, Order)

We can use this function to evaluate the derivatives of y w.r.t. x at any given point x by cubic spline interpolation. The arguments are the same as in CubicSplineValue() except that "Order" is an integer argument (INPUT) containing the order of derivatives to be computed. The return value is the derivative of y w.r.t. x at the given point.

IV. DESCRIPTION OF FILE integral.c

This file contains the modules used to calculate a single or double integration. Simpson's rule is used in this file to calculate the integrations. There are three functions in this file: (1) Double_Integral(), which is used to evaluating the double integration, (2) Single_Integral(), which is applied to calculate the single integration using variable step, and (3) Single_Integral_Fix(), which is used to compute the single integration using a fixed step.

A. Double_Integral

Double_Integral(Lower, Upper, Eps, Fun, FunY, K, Sum)

This function is used to calculate the double integration of the type

$$I = \int_a^b dx \int_{y_1(x)}^{y_2(x)} f(x, y)dy$$

The arguments follow.

Lower — is a float argument containing the lower boundary of the first integration, i.e., a. (INPUT).

Upper — is a float argument containing the upper boundary of the first integration, i.e., b. (INPUT).

Eps — is a float argument indicating the permit error of the integration used to check the convergence of the integral. (INPUT).

Fun — is a pointer to the function to be integrated (i.e. f(x, y)). Given the value of x

and y, the function returns the value of f(x, y). The user must supply the routine for this function. (INPUT).

FunY — is a pointer to the function for calculating $y_1(x)$ and $y_2(x)$ supplied by the user. The function must be of the type

FunY(x, y_1, y_2)

where x is a input float argument, y_1 and y_2 are the pointers to the output float arguments which contain the values for $y_1(x)$ and $y_2(x)$, respectively. (INPUT).

K — is an integer argument used to determine the step at which the convergence is tested for the first time. (INPUT).

Sum — is a pointer to a float argument containing the result of the integration I. (OUTPUT).

B. Single_Integral

Single_Integral(Lower, Upper, Eps, Fun, K, Sum)

This function is applied to calculate the single integration of the type

$$I = \int_a^b f(x)dx$$

The arguments follow.

Lower — is a float argument containing the lower boundary of the integration, i.e., a. (INPUT).

Upper — is a float argument containing the upper boundary of the integration, i.e., b. (INPUT).

Eps — is a float argument denoting the allowable error of the integration. It is used to check the convergence of the integral. (INPUT).

Fun — is a pointer to the function to be integrated (i.e. f(x)). Given the value of x, the function returns the value of f(x). The user must supply this function. (INPUT).