Multifrontal solution of sparse unsymmetric matrices arising from semiconductor equations

A. El Boukili* A. Madrane*
R. Vaillancourt*†‡

CRM-3125

June 2004

*Department of Mathematics and Statistics, University of Ottawa, Ottawa, K1N 6N5 Canada
†Corresponding author. E-mail: remi@uottawa.ca
‡Partially supported by NSERC of Canada and the Centre de recherches mathématiques of the Université de Montréal

Abstract

The multifrontal LU method is implemented to solve drift-diffusion models with large sparse matrices arising in the simulation and optimization of semiconductor devices. The performance of this method is compared with the LU algorithm without multifrontal scheme on different computers in the case of a realistic double heterojunction transistor.

Résumé

On emploie un méthode LU multifrontale pour résoudre un modèle de transport-diffusion à grande matrice creuse qu'on retrouve dans la simulation et l'optimisation de semiconducteurs. On compare la performance de cette méthode et celle de l'algorithme LU non multifrontal sur différents ordinateurs dans le cas d'un transistor à hétérojonction double.

Index Terms— sparse matrix, symmetric matrix pattern, multifrontal method, drift-diffusion model, bipolar transistor.

1 Introduction

The solution of large sparse systems of linear equations which occur in solving semiconductor device equations can account for as much as 90% of the computational time on industrial-scale problems. Thus, any reduction in the linear system solution time will result in a significant reduction of the total simulation run time. While iterative methods can be used to solve such systems, their reliability is questionable in the context of drift-diffusion semiconductor equations. In fact, in most applications the matrices arising from the discretization of these equations are ill conditioned because of the presence and dominance of advection terms. For this reason, we focus on direct methods.

In the solution of a large sparse linear system $Ax = b$ the multifrontal, the LU decomposition attempts to minimize fill-ins in the lower and upper triangular matrices L and U, respectively, in order to reduce run time. Here A is a sparse $n \times n$ matrix and x and b are column vectors of length n. We recall that the factored system $Ax = (LU)x = L(Ux) = b$ is solved in two steps. First $Ly = b$ is solved for y by forward substitution and $Ux = y$ is solved for x by backward substitution. In this paper, we focus on the multifrontal method which is an LU factorization method which is efficient for a large class of problems. While it may be efficient for unsymmetric and ill-conditioned linear systems, it was in fact originally derived for symmetric linear systems [6].

We consider the stationary drift-diffusion model of electron flow in a heterojunction semiconductor device. This model involves a nonlinear Poisson equation and two nonlinear transport equations for electrons and holes, respectively. The stationary problem is first transformed into an artificial transient nonlinear problem which, in turn, is discretized into a nonlinear implicit scheme. At each implicit step, the Newton–Raphson method used for solving nonlinear systems gives rise to large ill-conditioned unsymmetric linear algebraic systems with symmetric pattern. Since these matrices are diagonally dominant, the robustness and high efficiency of the Multifrontal LU Factorization (MLUF) can be tested without using any pivoting.

From several numerical experiments [7] using an unstructured grid [9] (see Fig. 3 below) on Dec Alpha 2100, IBM RS6000, HP 900-735 and SUN ULTRA's 1 and 10, the LU factorization algorithm without the multifrontal scheme, which will be named the Standard LU Factorization (SLUF), needs 32 hours for a single two-dimensional solution, called *characteristic*, associated with the currents across the collector and base contacts as a function of voltage for a realistic Double Heterojunction Bipolar Transistor (DHBT). On the other hand, MLUF needs only roughly 13 hours. The most costly and computationally intensive step is the numerical factorization. The performance of MLUF shows a significant improvement over SLUF.

The outline of the paper is as follows. Section 2 reviews the basics of the multifrontal method. Section 3 introduces the drift-diffusion equations, a corresponding transient problem, and the discretization scheme. Section 4 describes the physical problem of a DHBT characteristic, compares the numerical performances of MLUF and SLUF, and presents the numerical solution of the two-dimensional problem. A conclusion is drawn in Section 5.

2 Review of the Multifrontal Method

The multifrontal method is a generalization of the unifrontal method. It was originally developed for symmetric matrices. To understand the principle of this method, we first present the main idea of the unifrontal method.

2.1 Unifrontal method

In a unifrontal scheme [4], [10], the factorization proceeds as a sequence of partial factorizations and eliminations on dense submatrices, called *frontal matrices*. The unifrontal methods were originally designed for the solution of finite-element problems [10] and they can be used on assembled systems [4]. For assembled systems, the frontal matrices can be written as

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{1}$$

where all rows are fully summed (that is, there are no further contributions to come to the rows in (1)) and the first block column is fully summed. This means that the pivots can be chosen from anywhere in the first block column, and, within these columns, numerical pivoting with arbitrary row interchanges can be accommodated since all rows in the frontal matrix are fully summed. We assume, without loss of generality, that the pivots that have been chosen are in the square matrix A_{11} of (1). Then A_{11} is factored and the Gaussian elimination overwrites A_{21} and the Schur complement

$$A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{2}$$

is found using dense matrix kernels. The submatrices consisting of the rows and columns of the frontal matrix from which pivots have not yet been chosen is called the *contribution block*. In the case above, this is the same as the Schur complement matrix (2). At the next stage, further rows from the original matrix are assembled with the Schur complement to form another frontal matrix.

The entire sequence of frontal matrices is held in the same working array. Data movement is limited to assembling rows of the original matrix into the frontal matrix, and storing rows and columns as they become pivotal. One important advantage of the method is that only this single working array needs to reside in memory. A detailed description of frontal methods for assembled matrices is given in [14].

An example is shown in Table 1, where two pivot steps have already been performed on a 5×7 frontal matrix (computing the first two rows of U and columns of L, respectively), the columns are in pivotal order.

Entries in L and U are shown in lower case. Row 6 has just been assembled into the current 4×7 frontal matrix (shown as a solid box). Columns 3 and 4 are now fully summed (a column is fully summed if it has all of its nonzero elements in the frontal matrix) and can be eliminated. After this step, rows 7 and 8 must both be assembled before columns 5 and 6 can be eliminated (the dashed box, a 4×6 frontal matrix containing rows 5 through 8 and columns 5, 6, 7, 8, 9, and 12). The frontal matrix is, of course, stored without the zero columns, that is, columns 6 and 7 in the dashed box. The dotted box shows the state of the frontal matrix where the next four pivots can be eliminated. To factor the 12×12 sparse matrix in Table 1, a 5×7 (dense) working array is sufficient to hold all the frontal matrices.

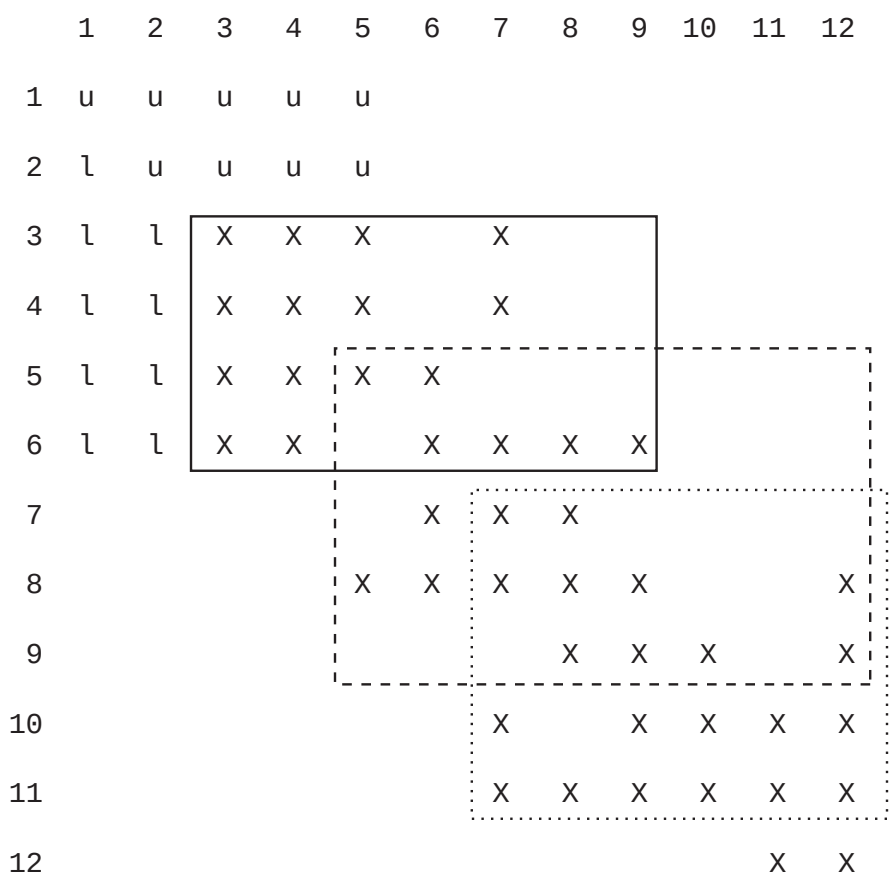
The unifrontal method works well for matrices with small pattern. But, for matrices with large pattern, the frontal matrices may be very large and an unacceptable amount of fill-in may occur.

2.2 Multifrontal method

Like the unifrontal method, the multifrontal method also exploits low-level parallelism and vectorization using the dense matrix kernels on frontal matrices. However, the frontal matrices in the multifrontal method are generally smaller and denser than in the unifrontal method.

We are considering the multifrontal method for matrices with a symmetric sparsity pattern [6]. We perform the following three phases for both SLUF and MLUF:

Table 1: Two pivot steps already performed on a 5×7 frontal matrix.



- a reordering phase to reduce fill-in by means of the minimum degree algorithm,
- a symbolic factorization,
- a numerical factorization.

Orderings, as by the minimum degree algorithm, tend to reduce fill-in much more than profile reduction orderings. The ordering is combined with a symbolic analysis to generate an assembly tree in the case of a multifrontal method.

2.3 Multifrontal elimination tree

In this subsection, we define the elimination tree based on graph theory and show how it is linked to Gaussian elimination. With any sparse symmetric matrix A, we associate a nondirected graph (V, E), where V is a set of nodes and E is the set of pairs (or edges) expressing the connections between the elements of V. If n is the order of A, we can label the nodes of the graph by integers $1, 2, \ldots$, such that $(i, j) \in E$ if and only if a_{ij} or a_{ji} are nonzero. A sequence of distinct edges $(i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k)$ is a *path*. A path is a *cycle* if $i_k = i_1$ for $k > 2$. An *acyclic graph* is a graph with no cycles. If there is a path between any pair of nodes, the graph is *connected*. A *tree* is defined as an acyclic connected graph. A *rooted tree* is defined by choosing a node, called *root*, which gives an orientation to the tree. For each tree, we can define *father* nodes, *son* nodes and *leaf* nodes (that is, nodes without sons).

A tree associated with Gaussian elimination is called an *elimination tree*. Several properties of trees associated with a tree generation algorithm are also given in [1]. We recall here two interesting properties. Let $A^{(k)}$ be the modified matrix from rows and columns k to n after elimination by pivots $1, 2, 3, \ldots, k-1$. The essence of the multifrontal approach to Gaussian elimination is that the elimination operations corresponding to pivot k can be performed as soon as all the entries in the first row and first column of $A^{(k)}$ have been calculated. Its power comes from eliminating several variables at a node. Therefore, there is no need to wait until all the entries in $A^{(k)}$ have been computed.

Theorem 2.1 ([1]) *The entries which are modified by the sons of node k are included in a submatrix of $A^{(k)}$ determined by the rows and columns with nonzeros in row k and column k of $A^{(k)}$.*

The entries that are modified by the sons correspond to nonzeros in the son's pivot row and column. Theorem 2.1 is equivalent to saying that for any son j, say, of node k, the sparsity structure of column j in rows k to n is included in the sparsity structure of column k of $A^{(k)}$.

Corollary 2.1 ([1]) *Computation corresponding to tree nodes which are not ancestors or descendants of one another are independent.*

Finite-element problems occur with matrices of the form

$$A = \sum_m B_m, \tag{3}$$

where each B_m is the contribution from a single finite element and is zero except in a small number of rows and columns. An example of an assembly tree corresponding to the summation

$$[[B_1 + B_2] + [B_3 + B_4]] + [[B_5 + B_6] + [B_7 + B_8]] \tag{4}$$

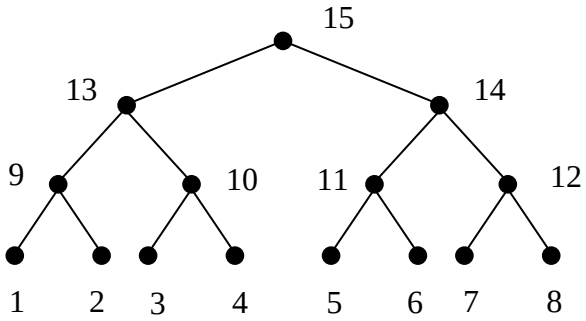


Figure 1: Assembly tree corresponding to bracketing (4).

is shown in Fig. 1.

Using Corollary 2.1, one can exploit the parallelism of the elimination tree [5].

The multifrontal method associates with each node/edge of the elimination tree a process that will exploit the properties of the tree. Thus, each edge of the elimination tree corresponds to summing contributions from the sons of the node with the rows and columns of the node itself. Consider a node k, say, which is not a leaf node. Then the contributions of the sons of this node will change the entries in $A^{(k)}$ corresponding to the nonzero entries in column k. We associate with each node k the matrix, called *frontal matrix*, determined by the nonzero elements in the pivot row and column, respectively, such that the contributions from the previous eliminations of the descendants of node k can be summed in the matrix. The outer products from the elimination by pivot k are then performed within the frontal matrix of node k.

We refer the reader to [2] for more details.

3 Physical Model and Discretization Schemes

3.1 Drift-diffusion equations

We consider the drift-diffusion model of mobile carrier transport in semiconductor devices [7], [12]. The complete system involves a coupled set of three strongly nonlinear partial differential equations. The first equation is Poisson's equation for the electrostatic potential ϕ, and the second and third equations are the transport equations for the quasi-Fermi levels ϕ_n and ϕ_p for electrons and holes, respectively. The mixed boundary conditions correspond to applied potential differences on the Ohmic contact portions of the device and to insulation in the remaining parts. We have to solve the following stationary boundary value problem:

$$\begin{aligned}
&-\operatorname{div}\big[\varepsilon_r(x)\nabla\phi\big] + \frac{q}{\varepsilon_0}\big[N(x,\phi,\phi_n) - P(x,\phi,\phi_p) - d_{\mathrm{op}}(x)\big] = 0 && \text{in} \quad \Omega,\\
&-\operatorname{div}\big[q\mu_n(x)N(x,\phi,\phi_n)\nabla\phi_n\big] + q\,GR(x,\phi,\phi_n,\phi_p) = 0 && \text{in} \quad \Omega,\\
&-\operatorname{div}\big[q\mu_p(x)P(x,\phi,\phi_p)\nabla\phi_p\big] - q\,GR(x,\phi,\phi_n,\phi_p) = 0 && \text{in} \quad \Omega,\\
&\phi = g_1, \qquad \phi_n = \phi_p = g_2 && \text{on} \quad \partial\Omega_D,\\
&\frac{\partial\phi}{\partial\boldsymbol{\nu}} = \frac{\partial\phi_n}{\partial\boldsymbol{\nu}} = \frac{\partial\phi_p}{\partial\boldsymbol{\nu}} = 0 && \text{on} \quad \partial\Omega_N.
\end{aligned} \tag{5}$$

Here, Ω is a bounded domain in $\mathbf{R}^2$, $\boldsymbol{\nu}$ is the outward normal vector to the boundary $\partial\Omega$, and $\partial\Omega_N$ and $\partial\Omega_D$ are disjoint parts of $\partial\Omega$ such that $\partial\Omega_N \cup \partial\Omega_D = \partial\Omega$. The function ε_r is the semiconductor relative dielectric permittivity and ε_0 is the permittivity of vacuum. The constant q is the electron

charge and $N(x,\phi,\phi_n)$ and $P(x,\phi,\phi_p)$ are electron and hole concentrations, respectively. Typical models for these strongly nonlinear functions and the mobilities μ_n and μ_p can be found in [7], [11]. The function $d_{\rm op}$ describes the doping profile, and the function GR is the so-called generation-recombination term which takes the possible "birth" and "death" of positive and negative charges into account. The Shockley–Read–Hall [7] model is used for this term.

To clarify the procedure, we list the steps that will be used in the solution of the nonlinear stationary system (5) which we write in general form as

$$F(u) = 0 \qquad \text{on } \Omega. \tag{6}$$

When dealing with the solution of such nonlinear systems, a key point is the choice of iterative methods to compute the solution. For this purpose, one often uses variants of Newton–Raphson's method [7] whose quadratic convergence is achieved only sufficiently near the desired solution. Hence, the choice of the initial value is critical. Since in the applications very little *a priori* knowledge about the solution is available, Newton–Raphson's method may fail because of poor starting values. As a remedy, one can use time deformation by transforming (6) into an artificial transient problem of the form

$$\frac{\partial u}{\partial t} + F(u) = 0 \qquad \text{on } \Omega. \tag{7}$$

The final solution is constructed step by step instead of computing it directly. This computing strategy provides a "good" initial guess. The solution of our problem is then obtained as the limit solution of an artificial nonstationary problem.

Problem (7) is then discretized in time to yield the system

$$\frac{u^{k+1} - u^k}{\Delta t} + F(u^{k+1}) = 0 \qquad \text{on } \Omega, \tag{8}$$

which we rewrite as

$$G(u^{k+1}) = 0 \qquad \text{on } \Omega. \tag{9}$$

Next, one uses a variational method and looks for a weak solution in the space X to the problem

$$(G(u^{k+1}), v) = 0 \qquad \forall\, v \in X. \tag{10}$$

Spatial discretization of this problem is achieved using a finite element method. This produces the discrete nonlinear system

$$(G(u_h^{k+1}), v_h) = 0 \qquad \forall\, v_h \in X_h. \tag{11}$$

Finally, this problem is solved iteratively by Newton's method, giving rise to the linear system

$$G'(u_h^k) u_h^{k+1} = u_h^k - G(u_h^k), \tag{12}$$

which is solved by the multifrontal LU decomposition.

3.2 Mixed formulation

In the case in hand to approximate the solution of (10) we use a mixed finite element (MFE) method [12], [3], based on the dual mixed formulation. The basic idea behind the mixed formulation of (5) is to introduce with the solution (ϕ,ϕ_n,ϕ_p) of (5), the displacement D and the current densities $\mathbf{j}_n$, $\mathbf{j}_p$ as independent variables defined by the expressions

$$\begin{aligned} D &= \varepsilon_r(x)\nabla\phi, \\ \mathbf{j}_n &= q\mu_n(x)N(x,\phi,\phi_n)\nabla\phi_n, \\ \mathbf{j}_p &= q\mu_p(x)P(x,\phi,\phi_p)\nabla\phi_p, \end{aligned}$$

and to derive a suitable variational formulation. In many applications, the flux variables $\mathbf{j}_n$ and $\mathbf{j}_p$, introduced above, are more important than the primal variables ϕ, ϕ_n, and ϕ_p.

For time discretization, following previous work [7], [8], we use a nonlinear implicit scheme with local time steps in which the system is decoupled into three nonlinear subsystems. Spatial discretization is based on the Raviart–Thomas finite element of lowest order [7], [13]. We have to solve the following problem:

Find $(\phi^{k+1}, D^{k+1}, \phi_n^{k+1}, \mathbf{j}_n^{k+1}, \phi_p^{k+1}, \mathbf{j}_p^{k+1})$ such that, in Ω,

$$\begin{aligned}
&A_1(x)\frac{\phi^{k+1}-\phi^k}{\delta t_1} - \operatorname{div} D^{k+1} + \frac{q}{\varepsilon_0}\left[Nx, \phi^{k+1}, \phi_n^{k+1}) - P(x, \phi^{k+1}, \phi_p^{k+1}) - d_{\mathrm{op}}(x)\right] = 0,\\
&D^{k+1} = \varepsilon_r(x)\nabla\phi^{k+1},\\
&A_2(x)\,\frac{\phi_n^{k+1}-\phi_n^k}{\delta t_2} - \operatorname{div} j_n^{k+1} + q\,GR(x, \phi^{k+1}, \phi_n^{k+1}, \phi_p^k) = 0,\\
&\mathbf{j}_n^{k+1} = q\mu_n(x)N(x, \phi^{k+1}, \phi_n^{k+1})\nabla\phi_n^{k+1},\\
&A_3(x)\,\frac{\phi_p^{k+1}-\phi_p^k}{\delta t_3} - \operatorname{div} \mathbf{j}_p^{k+1} - q\,GR(x, \phi^{k+1}, \phi_n^{k+1}, \phi_p^{k+1}) = 0,\\
&\mathbf{j}_p^{k+1} = q\mu_p(x)P(x, \phi^{k+1}, \phi_p^{k+1})\nabla\phi_p^{k+1},
\end{aligned} \tag{13}$$

and

$$\begin{aligned}
&\phi^{k+1} = g_1, \qquad \phi_n^{k+1} = \phi_p^{k+1} = g_2 \qquad \text{on } \partial\Omega_D,\\
&D^{k+1}\cdot\boldsymbol{\nu} = j_n^{k+1}\cdot\boldsymbol{\nu} = j_p^{k+1}\cdot\boldsymbol{\nu} = 0 \qquad \text{on } \partial\Omega_N,
\end{aligned} \tag{14}$$

where the preconditioners A_1, A_2, and A_3 are uniformly bounded positive functions. The local time steps are defined by

$$\delta t^i(x) = A_i^{-1}(x)\delta t_i, \qquad i = 1, 2, 3,$$

where δt_i, $i = 1, 2, 3$, are dimensionless parameters and the time dimension is introduced in A_i^{-1}, $i = 1, 2, 3$.

In solving each nonlinear subsystem by the Newton–Raphson algorithm one obtains a linear system with coefficient matrix M. In the case of the linearized Poisson equation, M is of the form

$$M = \begin{bmatrix} A & B^t \\ B & -C \end{bmatrix},$$

where A is associated with a perturbation of the identity operator in $L^2(\Omega)]^2$, B is associated with the divergence operator and C is associated with the source terms. We point out that M is symmetric but not positive definite since it is associated with a saddle-point problem. Since the approximation is of lowest order, the primal variables are constant on each triangle, so they can be eliminated before the assembly. Thereby, we obtain a positive definite symmetric matrix which can be written *formally* as

$$A + B^t C^{-1} B.$$

The LL^t Cholesky factorization is used to solve this first system.

The matrices associated with the linearized transport equations have the form

$$M = \begin{bmatrix} A & B^t + J \\ B & -C \end{bmatrix},$$

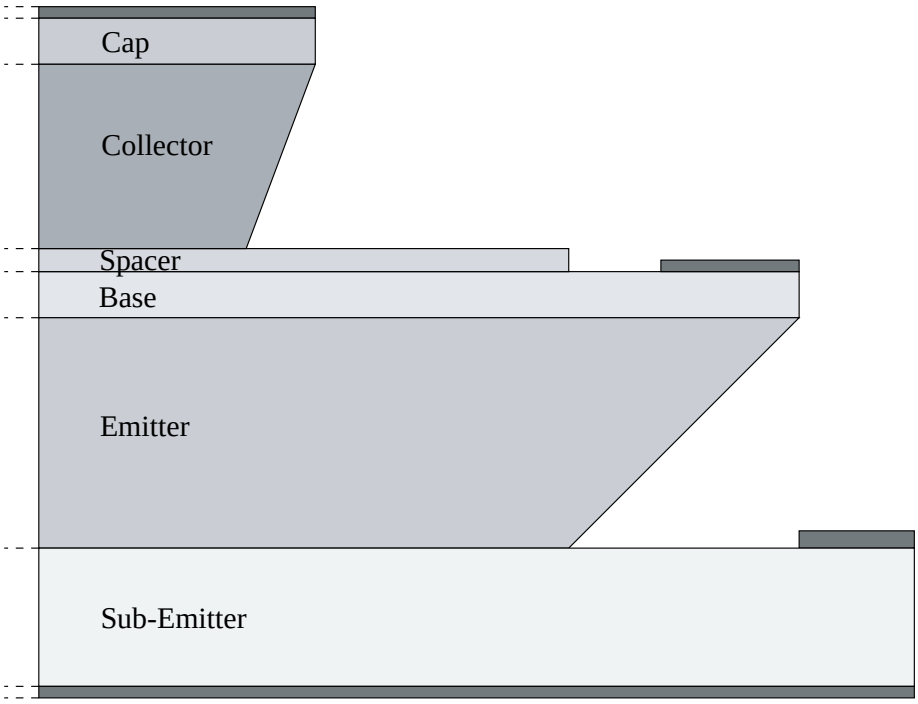


Figure 2: Double heterojunction bipolar transistor with collector-up.

where J comes from the advection term. If we use, as in the first case, an elimination procedure, we obtain a matrix of the form

$$A + B^t C^{-1} B + J C^{-1} B$$

which is invertible and diagonally dominant (see [7]), but is neither symmetric nor positive definite.

For comparison purposes, in subsection 4.2 below, the two linear systems associated with the transport equations are solved by means of the standard (**SLUF**) and the multifrontal (**MLUF**) LU factorizations, respectively.

4 Numerical Results

In this section, we first describe the structure of a double heterojunction bipolar transistor with collector-up and define its computed characteristics and physical objective. Then, we compare the performance of **SLUF** and **MLUF** on single-processor SUN ULTRA's 1 and 10.

4.1 Double heterojunction bipolar transistor with collector-up

We consider a realistic device whose structure is shown in Fig. 2. This device could work as a high-power amplifier. For the present simulation, the sub-emitter is omitted.

The materials and the doping values were provided by Thomson Laboratory [7]. Let V_C, V_B, and V_E be the applied voltage at the collector, base, and emitter contacts, respectively. The characteristic is associated with the values of the current I_C across the collector contact and the current I_B across the base contact, as V_C and V_B increase from 0.2 to 1.3 volts. The applied voltage $V_E = 0$ in our case. The mesh for the present simulation is presented in Fig. 3.

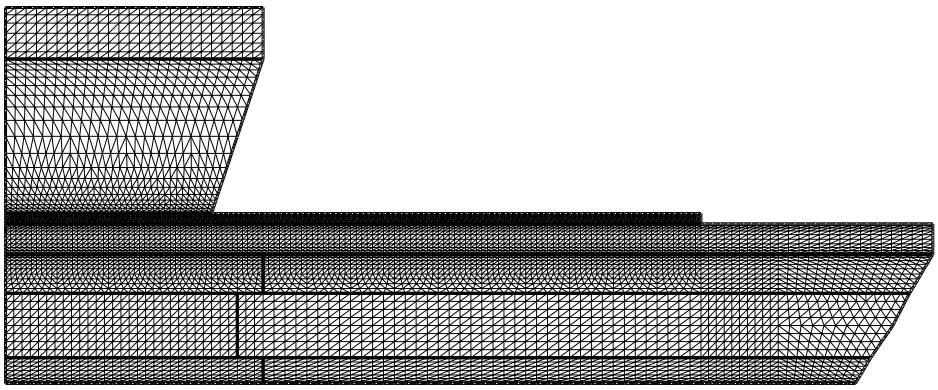


Figure 3: Mesh containing 11347 elements, 17200 edges, and 5854 vertices.

4.2 Comparison of SLUF and MLUF

In this subsection, we compare the performance of the **SLUF** and **MLUF** methods. These two methods use the same storage strategy, the minimum degree algorithm and symbolic factorization. The two phases (reordering and symbolic factorization) are not really time consuming and are performed only once. Since no pivoting is used and hence the sparsity is preserved, then there is no need to perform these two preprocessing steps more than once. We are only comparing the overall run time to perform the numerical factorization (FACTOR) and to solve a single linear system (SOLVE) as presented in Table 3.

The matrices in this problem are of order 17,200. The number of nonzero entries after factorization and the number of floating-point operations (flops) needed for the LU factorization step using both **SLUF** and **MLUF** are listed in Table 2.

Table 2: Number of nonzero entries and megaflops with the variable band and the multifrontal methods.

	SLUF	**MLUF**
Number of nonzero entries	2,348,962	593,204
Number of megaflops	375	20

When 0.2 volts are applied to the structure, 76 LU factorizations are needed to solve the nonlinear system associated with the transport electron equation. Table 3 presents the run times for

Table 3: Performance of the standard and the multifrontal methods for ULTRA's 1 and 10.

	SUN	**SLUF**	**MLUF**
FACTOR	ULTRA 1	4415.00 s	136.07 s
	ULTRA 10	1602.06 s	47.38 s
SOLVE	ULTRA 1	237.24 s	18.66 s
	ULTRA 10	88.15 s	5.58 s

this number of double-precision factorizations on SUN ULTRA's 1 and 10 whose CPU and RAM specifications are presented in Table 4.

Table 4: Specifications of uniprocessor SUN ULTRA's 1 and 10.

	ULTRA 1	ULTRA 10
CPU	135 MHertz	440 MHertz
RAM	64 MBytes	1 GByte

Note that the results listed in Table 3 are obtained without pivoting, because it is known, *a priori* that the matrices are diagonally dominant.

The total run time (CPU on an ULTRA 10) of our program using **SLUF** and **MLUF** to compute one DHBT-characteristic is given in Table 5. This run time corresponds to 627 implicit iterations

Table 5: Run time for one DHBT characteristic with SLUF and MLUF on ULTRA 10.

ULTRA 10	**SLUF**	**MLUF**
Time	32.52 hours	13.15 hours

and 2266 LU factorizations.

The numerical solution for the electron densities N is presented in Fig. 4 where the vertical axis is $Z = \log N$. The numerical solution for the hole densities P is presented in Fig. 5 where the vertical axis is $Z = \log P$. In both figures, $V_B = V_C = 1.3\,V$ and $V_E = 0\,V$.

5 Conclusion

We have reviewed the multifrontal method and presented results on its high-performance when applied to a realistic drift-diffusion device. This application will be attractive for industrial development of semiconductor devices involving large unsymmetric matrices as done, for instance, at the Thomson Laboratory. Future work will consider the application of a parallel version of the multifrontal method (see, for example, [5]) and the parallelization of other parts of the code.

Acknowledgement

The authors thank the referee for numerous deep and constructive comments which led to a substantial improvement of this paper.

References

[1] P. Amestoy, *Factorisation de grandes matrices creuses non symétriques basée sur une méthode multifrontale dans un environnement multiprocesseur*, Doctoral dissertation, Université de Toulouse, 1990.

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl. **23**(1) (2001) 15–41.

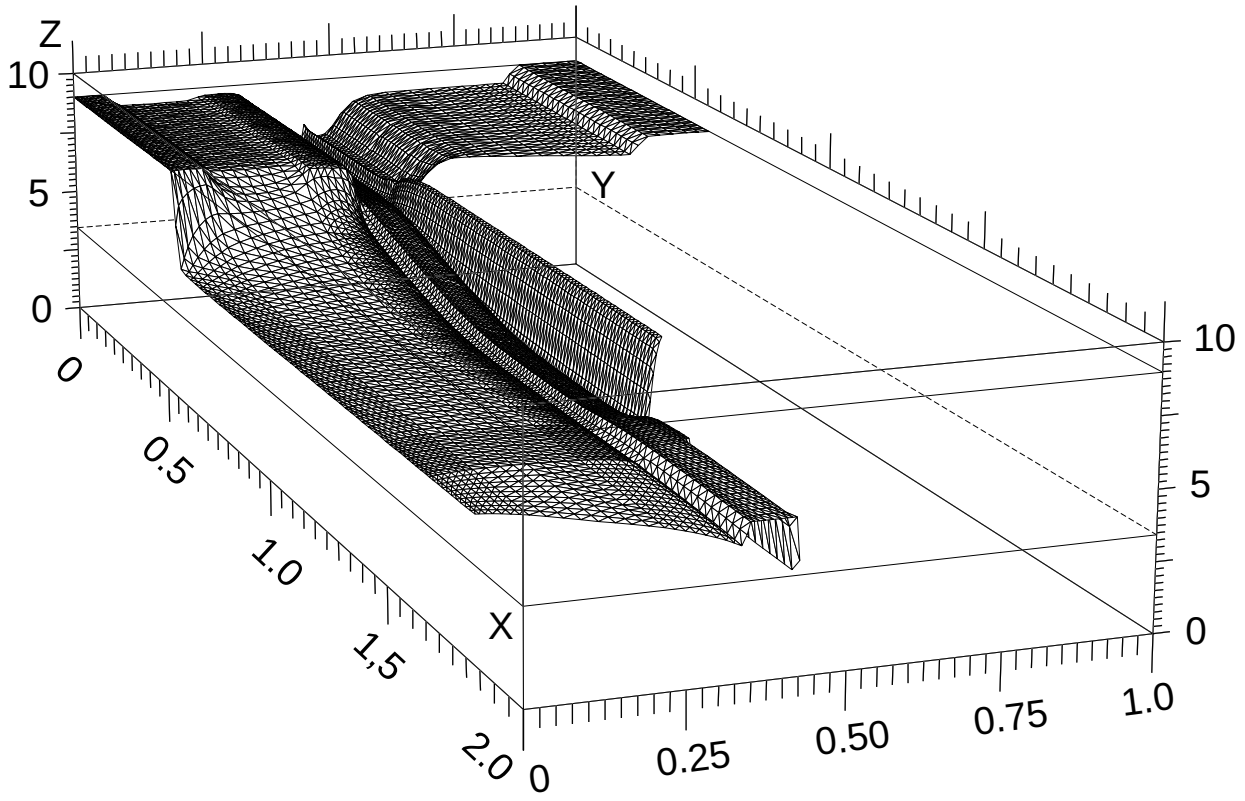


Figure 4: Electron densities N, where $Z = \log N$, for $V_B = V_C = 1.3\,V$ and $V_E = 0\,V$.

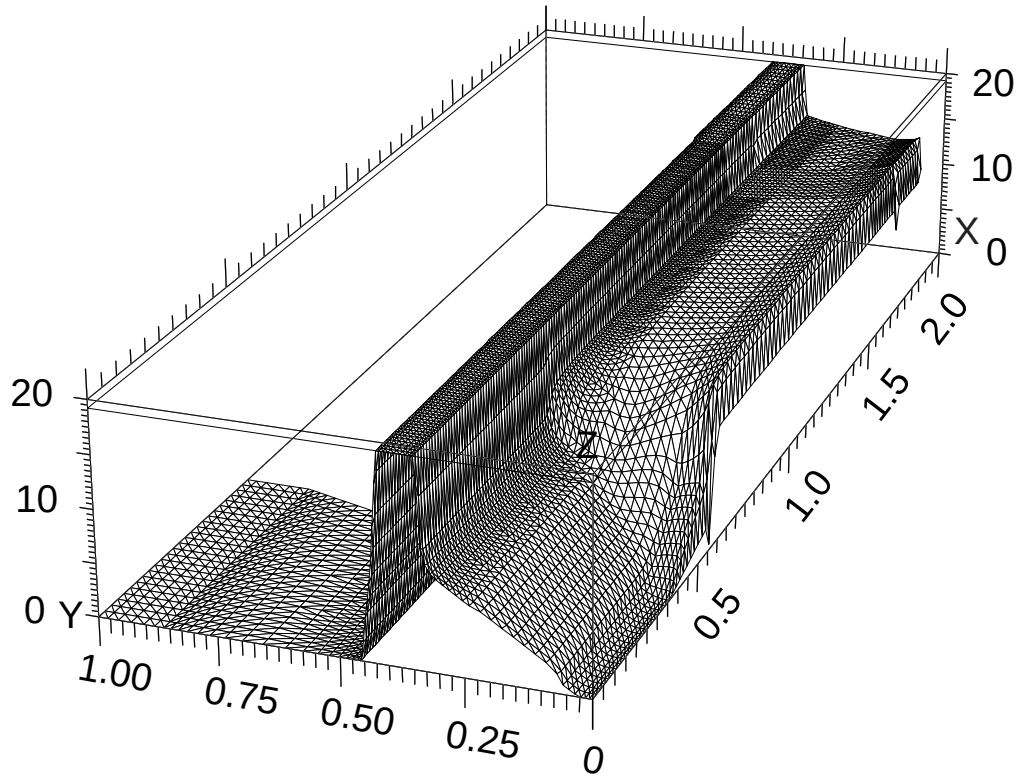


Figure 5: Hole densities P, where $Z = \log P$, for $V_B = V_C = 1.3\,V$ and $V_E = 0\,V$.

[3] F. Brezzi, *On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers*, R.A.I.R.O., Sér. Rouge **8**(R-2) (1974) 129–151.

[4] I. S. Duff, *Design features of a frontal code for solving sparse unsymmetric linear systems out-of-core*, SIAM J. Sci. Comput. **5**(2) (1984) 270-280.

[5] I. S. Duff, N. I. M. Gould, M. Lescrenier, and J. K. Reid, *The multifrontal method in a parallel environment*, in : M. G. Cox and S. Hammarling eds., Reliable Numerical Computation (Oxford, Oxford University Press, 1990) 93–111.

[6] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear systems*, ACM Trans. Math. Software **9** (1983) 302–325.

[7] A. El Boukili, *Analyse mathématique et simulation numérique bidimensionnelle des équations des semiconducteurs par l'approche éléments finis mixtes*, Doctoral dissertation, Université Paris 6, 1995.

[8] A. El Boukili, *Arclength continuation method and its application to 2D drift-diffusion semiconductor equations with mixed finite element approach*, Compel **15**(4) (1996) 36–47.

[9] A. El Boukili, A. Madrane and R. Vaillancourt, *Unstructered grid adaptation for convection-dominated semiconductor equations*, Can. Appl. Math. Quarterly, **10**(4) (Winter 2002) 447–472.

[10] B. M. Irons, *A frontal solution program for finite element analysis*, Internat. J. Numer. Methods Eng. **2** (1970) 5–32.

[11] P. A. Markowich, *The stationary semiconductor device equations*, in: Computational Microelectronics (Vienna, Springer-Verlag, 1986).

[12] M. S. Mock, *Analysis of mathematical models of semiconductors devices*, (Dublin, Boole Press, 1983).

[13] P.-A. Raviart and J.-M. Thomas, *A mixed finite element method for 2nd order elliptic problems*, in: I. Gal ligani and E. Magenes eds., Mathematical Aspects of Finite Element Methods, Lecture Notes in Math., Vol. 606 (Berlin, Springer-Verlag, 1977) 292–315.

[14] S. E. Zitney, *Sparse matrix methods for chemical process separation calculations on supercomputers* In Proc. Superecomputing 92, Minneapolis MN, Nov. 1992, IEEE Computer Society Press, pp. 414–423.