

## A flexible and efficient adaptive refinement scheme using a local solution procedure

*P. Ciampolini, A. Forghieri, A. Pierantoni, M. Rudan and G. Baccarani*

Dipartimento di Elettronica, Informatica e Sistemistica  
Università di Bologna, viale Risorgimento 2, 40136 Bologna, Italy

### Abstract

*An adaptive refinement scheme which implements a local solution of the full set of semiconductor equations is presented. The software implementation allows for different error estimates, some of which are described. Some practical examples (a bidimensional p-n junction and a planar MOSFET) are presented, along with data concerning the efficiency and reliability of the scheme.*

## 1 Introduction

With the increasing availability of low-cost computational resources, adaptive mesh generation is becoming one of the most attractive features for the field-users of device simulation programs. In fact, adaptive meshing completely relieves the user from the burden of mesh generation (thus saving a considerable amount of user's time) while ensuring a grid whose properties are automatically tailored to the problem to be solved, thus avoiding the waste of CPU time occurring when simulations are performed on inadequate grids.

Since these kind of schemes usually require repeated solutions of a test problem, involving large data transfers between part of codes in charge of different tasks (typically a mesh generator, a numerical solver and an I/O manager), a scheme which aims at being used in practical applications must satisfy a number of efficiency, flexibility and reliability requisites. In particular, an adaptive mesh generator should:

- run as fast as possible, taking advantage from the knowledge of previously computed values of the selected quantities;
- make the implementation of different refinement criteria feasible;
- lead to stable solutions with respect to node addition.

In the following, we discuss a scheme that fulfills these requirements using software modules (previously existing as stand-alone programs) linked in an integrated system whose efficiency has been enhanced by means of several software and numerical techniques, among others a local solution of the semiconductor equations and an accurate first guess determination.

## 2 Software implementation

Fig. 1 shows a schematic block diagram of the software organization that implements our adaptive scheme. After defining the device geometry and physical parameters via the input preprocessor (WALHALL) the control is transferred to the mesh generator (ATMOS) which manages the iterative loop.

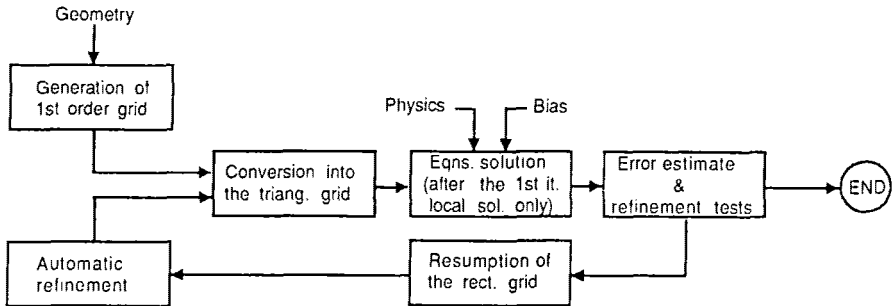


Figure 1: Block diagram of the adaptive refinement loop.

This is accomplished through the following steps:

1. a first, coarse grid is generated, either automatically or interactively;
2. the control is then transferred to the solver (HFIELDS, ) which estimates the local error and returns a list of rectangular elements which have to be refined: this test is controlled by a user-supplied parameter, which, as will be shown later, is roughly inversely proportional to the number of nodes of the final mesh;
3. ATMOS performs the refinements and loads the appropriate values of the electric potential and quasi-Fermi potentials on the mesh nodes: this means that every node belonging to the previous mesh receives the value computed by HFIELDS, while “new” nodes (due to the current refinement) receive a value that linearly interpolates the values of their neighbours. We stress that performing such an operation at this point leads to a considerable time-saving: in fact such an interpolation requires a time of order  $O(N^2)$  if performed later on (because every node requires an inclusion test on every triangular element of the “old” mesh) while the same task requires a time of order  $O(N)$  when performed by ATMOS.
4. The control is then transferred to HFIELDS which solves again the problem, but only for the “new” nodes and their neighbours (the reason of this is explained below) thus implementing what we call the “local solution technique”. The program then iterates points (2) and (3) until no more refinement is required: after that, the interactive mode is switched on, and the user can continue the session.

While a deeper insight of ATMOS and HFIELDS may be found in [1,2] we give here a short outline of their main features.

HFIELDS (Hybrid Finite Element Device Simulator) is a numerical device simulator based on a classical BIM (Box Integration Method) scheme: since it makes use of triangular elements, it is capable to deal with completely general geometries.

ATMOS (Automatic Triangular Mesh Optimization System) is a program which generates triangular meshes on the basis of a "rectangular approach" that is, performing all required refinements on a rectangular-element grid, only roughly conforming to the device geometry, and carrying out the conversion into triangles and the adaption to geometry at the very last step.

## 2.1 Initial guess determination and local solution

The loop described in the previous section involves a number of operations intrinsically time consuming which may eventually make the mesh generation process insufferably lengthy. Improvements may be achieved both by speeding up the convergence rate of the solver and by decreasing the dimension of the problem which is repeatedly solved. As described above, our mesh-generation scheme requires the iterated solution of the non-linear set of the semiconductor equations: it is well known that the convergence rate of such a problem is strongly affected by the accuracy of the initial guess. Needless to say, the chance of using the solution evaluated on a coarser mesh to find an accurate initial guess allows one to considerably shorten the solution times. Plots in Fig. 2 show solution times versus number of iterations for a planar MOSFET. The dot-dashed line refers to a global solution and to a rough initial guess, in which electric potential and quasi-Fermi potentials are initialized in a step-like fashion; the dashed line refers to an "interpolated" initial guess, which sets old mesh nodes to the values computed in the previous solution, while the newly generated nodes are initialized by means of a linear interpolation: the CPU time saving is evident. Initialization times may be shortened by evaluating the initial guess while adding nodes to the mesh, as stated above: in this case, it is in fact possible to take advantage from the knowledge of the "names" of the new nodes neighbours.

Another feature that saves a considerable amount of time is the "local solution". Since a lower accuracy of intermediate solutions may be allowed at the expense of computing a complete solution on the final mesh our program solves for potential and carrier concentrations only over a sub-set of nodes in each intermediate mesh. The solid line in Fig. 2 refers to solution times when both local solution and interpolated initial guess are used. The selected sub-set includes both the new nodes and their neighbours, while concentrations and potential values of remaining nodes are regarded as boundary conditions of the problem. Solving equations only at the newly-generated nodes may cause some evil effects such as the node clustering shown in Fig. 3 (left). New nodes cluster around nodes belonging to the earliest generations — whose values have not been updated since the very first iterations — and are, there-

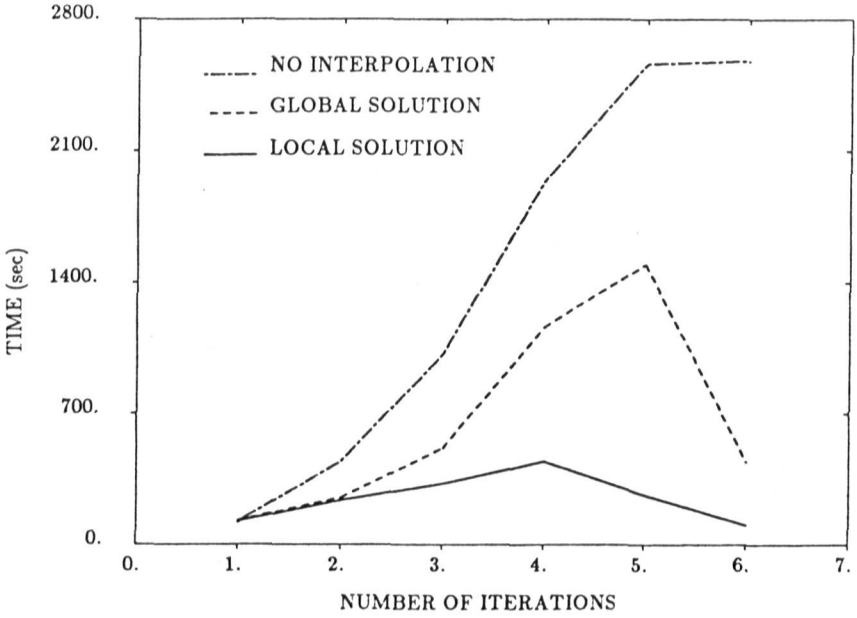


Figure 2: *Solution times (on a MicroVAX II) versus number of iterations, for a planar MOSFET.*

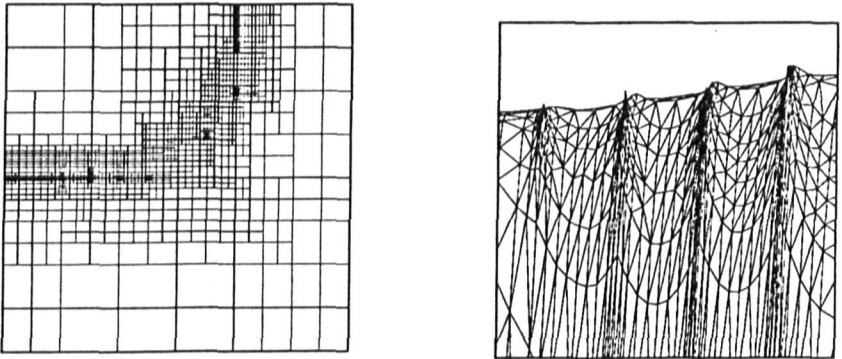


Figure 3: *Influence of local solution on adaptive refinement, when solving the equations only at newly-generated nodes.*

fore, affected by the largest errors. Fig. 3 (right) shows a particular of the solution computed at this stage: although the height of the “ridges” is small, they strongly modify the second order differential quantities (e.g. curvature), which are taken into account by the refinement test. On the other hand, solving the equations also on the nearest neighbours of the new nodes allows one to adjust the boundary conditions of the local solution by moving the boundaries of the reduced domain toward less perturbed regions. This results in smoother surfaces and better meshes, as shown later in Fig. 11.

Fig. 4 summarizes the behaviour of the two features above; the norm  $\Delta$  is defined as follows:

$$\Delta = \max_{i=1,N} \{ |\varphi_i - \psi_i| \}$$

where  $N$  is the number of nodes,  $\varphi_i$  is the value of the electric potential computed at the  $i$ -th node through a “global” solution and  $\psi_i$  is either the corresponding value supplied by the “local” solution (dashed line) or the interpolated initial guess (solid line). As the procedure approaches convergence, the quality of the initial guess improves, while a good agreement between “local” and “global” solutions is maintained.

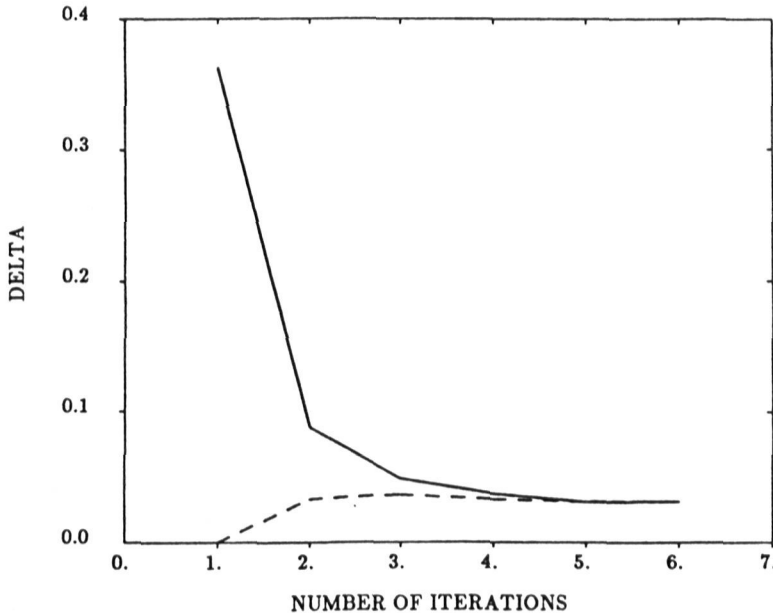


Figure 4:  $\Delta$  versus number of iterations

## 2.2 The refinement algorithm

The principle informing our refinement strategy is — quite obviously — that a higher node density is needed in the regions where the selected physical quantities deviate from a linear behaviour. A measure of this deviation is given by the differential moments of an appropriate additive combination of

the significant physical quantities, sometimes called a “monitor surface” (see for instance [3]). We found that, in most cases, the choice of the electric potential is sufficient to appropriately depict the main features of a device behaviour. More sophisticated mathematical models may, however, require the consideration of different quantities (see, e.g., [4]). A “weight function” can thus be defined, whose general form, in terms of ordinary derivatives, is:

$$\sum_{i=1}^n w_i (D^i f) \quad (1)$$

where  $w_i$  are numerical “weights” and  $D^i$  is the  $i$ -th order differential operator. We choose to take into account only the second order terms for a number of reasons, among which:

- every differential term in (1) has to be evaluated numerically: the accuracy with which differential terms of order higher than the second may be computed is therefore low;
- we do not account for first order terms since these would bring to high values of the weight function even in regions where the behaviour of the monitor surface is strictly linear, their discretization therefore requiring a low nodes density .

A rectangular element will therefore be refined if the following condition is fulfilled ( $\beta$  being supplied by the user):

$$\frac{1}{2\Delta\varphi_{max}} \left( \frac{d^2\varphi}{dl^2} \right)_{max} (\Delta l)^2 > \beta \quad (2)$$

Here, the expression  $(d^2\varphi/dl^2)_{max}$  stands for the directional second derivative of  $\varphi$  and is given by:

$$\left( \frac{d^2\varphi}{dl^2} \right)_{max} = f_{xx} \cos^2 \alpha + 2f_{xy} \sin \alpha \cos \alpha + f_{yy} \sin^2 \alpha \quad (3)$$

The algorithm implementing this test may be considerably simplified if only the terms  $f_{xx}$ ,  $f_{yy}$  are taken in consideration to determine horizontal and vertical refinement: on the other hand, this would lead to exceedingly poor meshes in regions where  $f_{xy}$  is dominant. It is for instance easy to see that the function  $f(x, y) = xy$  will not trigger the refinement test if  $f_{xy}$ , which is the only non-zero second derivative, is not accounted for. The geometric interpretation of this criterion is shown in Fig. 5, where  $1/\rho = |g'_l| / (1 + (g'_l)^2)^{3/2}$  is the curvature of  $g(l)$  and  $\delta = |g''_l (\Delta x)^2| / \sqrt{1 + (g'_l)^2}$  is the radial distance between the curve and its osculating circumference. The refinement test (2) amounts therefore to the checking of the condition  $\delta / \cos \chi < \beta$ .

An alternative test may be implemented by requiring:

$$\delta < \beta \quad (4)$$

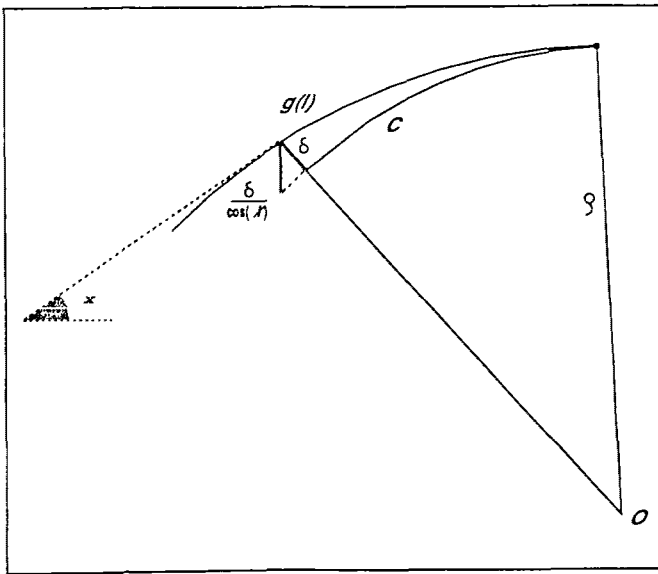


Figure 5: *Geometrical meaning of the refinement criterion.*

instead of (2). This amounts to consider a weight function of the form (1), but using differential operators in the intrinsic surface coordinates. Although this approach would be preferable from a geometrical standpoint, we have found significant differences between the two approaches only when the number of nodes becomes so large as to exceed practical limits in terms of storage and CPU time: since the methods are comparable as far as the quality of the generated meshes and computational efficiency are concerned, they are, in practice, interchangeable.

### 3 Results

Many efficiency tests have been performed on the various techniques described above: Fig. 6 shows the typical behaviour of the program in terms of total number of nodes, number of nodes on which the program actually solves (unknowns), solution times and non-linear loops needed to achieve the convergence for Poisson's equation. The dramatic increase of CPU times due to the switching off the local solution and the initial-guess determination is shown in Fig. 2. The experiments we performed showed that our method is "convergent", that is, it eventually stops adding nodes and claims that the generation process is terminated. While this may be regarded as a quite obvious feature, we more surprisingly observed that, if appropriate values of the parameters are chosen ( $\beta = 0.01$  is often a good choice), the process terminates as the current sets to a stable value: Fig. 7 eloquently illustrates this phenomenon. When reasonable refinement levels are reached, the current ceases to be sen-

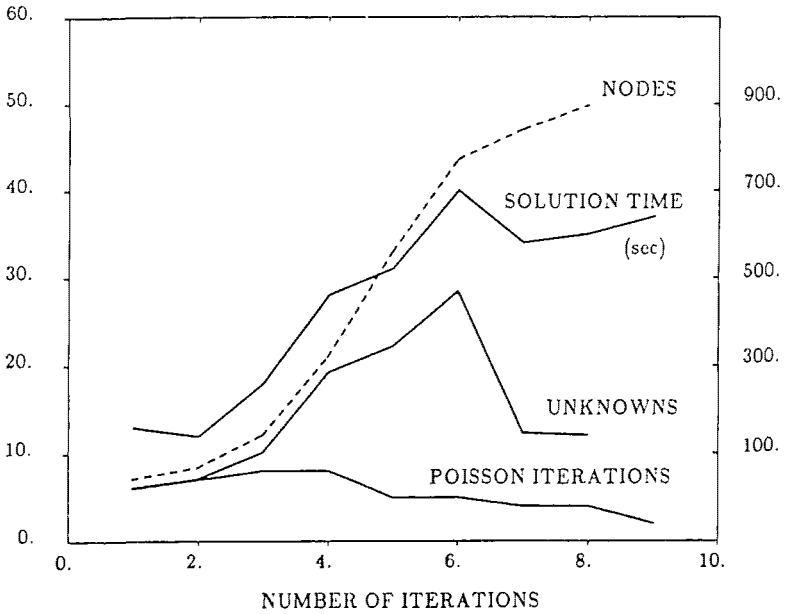


Figure 6: Plots of the solution times, of the number of unknowns and of the number of Poisson iterations. Left scale refers to times and iterations, right scale to nodes and unknowns.

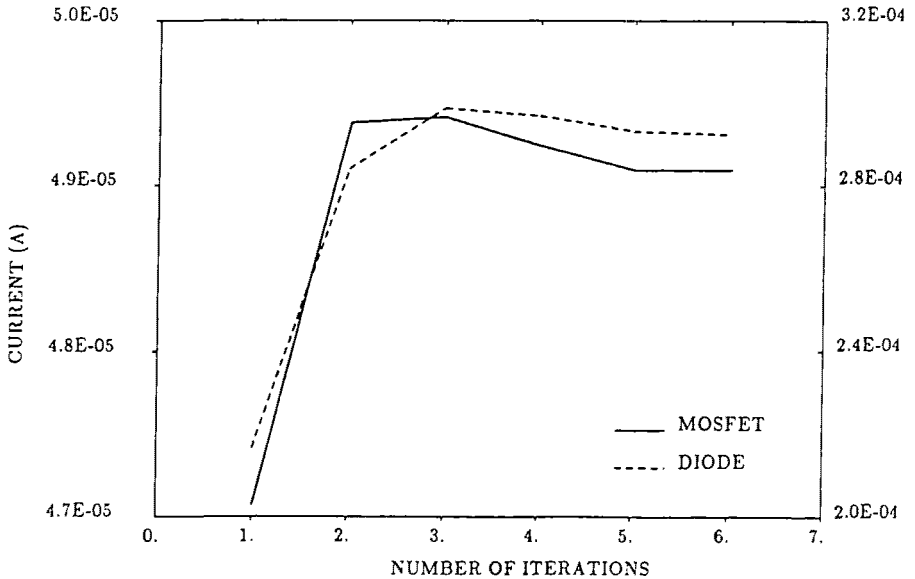


Figure 7: Computed currents versus number of iterations for both a planar MOSFET and a p-n cylindrical junction.



sible to node addition, as shown in Fig. 8 for both a planar MOSFET and a p-n junction: too coarse meshes may instead cause wild behaviours. Another

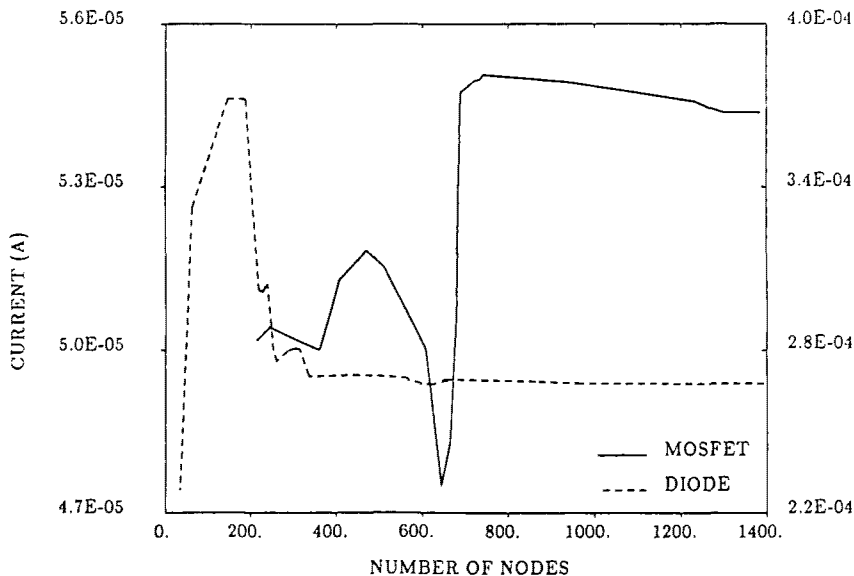


Figure 8: Computed currents versus number of nodes for the same devices as above.

interesting phenomenon is illustrated in Fig. 9, where relationship between  $\beta$  and node number is shown, for a p-n cylindrical junction. For each bias, the points fit a different hyperbola: the spread of the values around the “least squares” curve is fairly small. Similar curves, drawn for different devices, are compared in Fig. 10, and corroborate the hypothesis of a functional link between  $\beta$  and the resulting number of nodes. This inverse proportionality may be explained by observing that, as it is well known, the discretization error of a BIM method is of the order of  $o(h^2)$  ( $h$  being a characteristic dimension of the mesh) and therefore of the order  $o(1/N_{nodes})$ . Although  $\beta$  cannot be straightforwardly identified with the error involved in the discretization method, Fig. 9 and 10 show that a non-casual link between these quantities can on good grounds be supposed. Furthermore, this phenomenon may be advantageously used as a “thumb-rule” for the determination of a value of  $\beta$  appropriate to the user’s needs.

Lastly, Fig. 11 shows meshes generated for the p-n cylindrical junction above, for a planar MOSFET and for a non-planar gated diode, along with the 3-D perspective plots of the electric potential inside the devices. In all cases, the distribution of nodes allows for an accurate description of the electric potential, while unnecessary refinements in neutral and ohmic regions have been avoided.

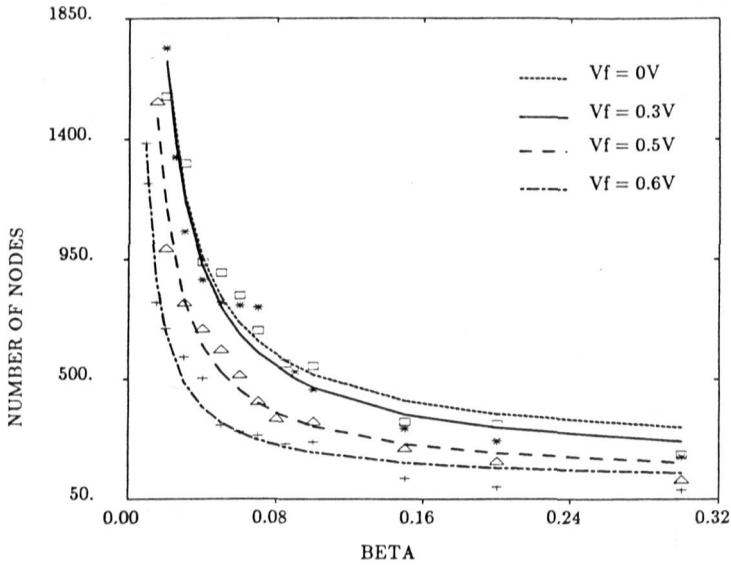


Figure 9: Relationship between  $\beta$  and number of nodes, for a cylindrical  $p$ - $n$  junction, with different forward biases: at the equilibrium ( $\square$ ), and with 0.3V (\*), 0.5V ( $\triangle$ ) and 0.6V (+) applied. Continuous curves are obtained through linear regression of data points.

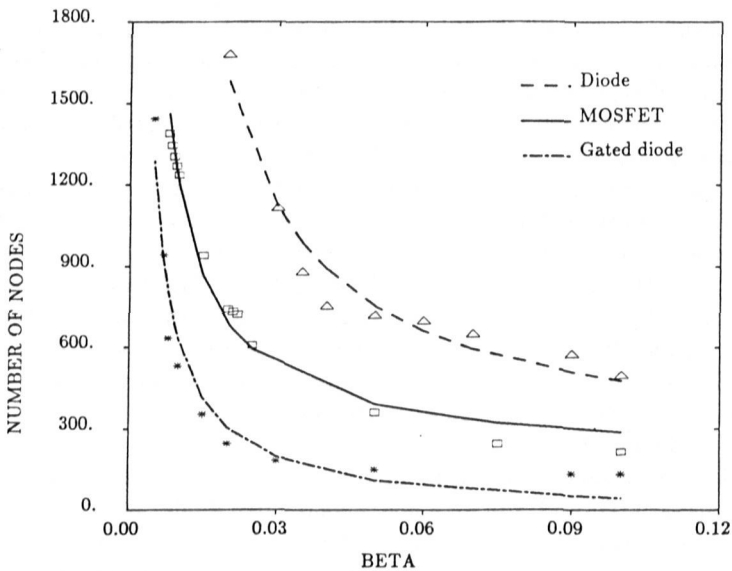


Figure 10: Same curves of previous picture, but related to different devices:  $p$ - $n$  cylindrical junction ( $\triangle$ ), a planar MOSFET ( $\square$ ) and a non-planar gated diode (\*).

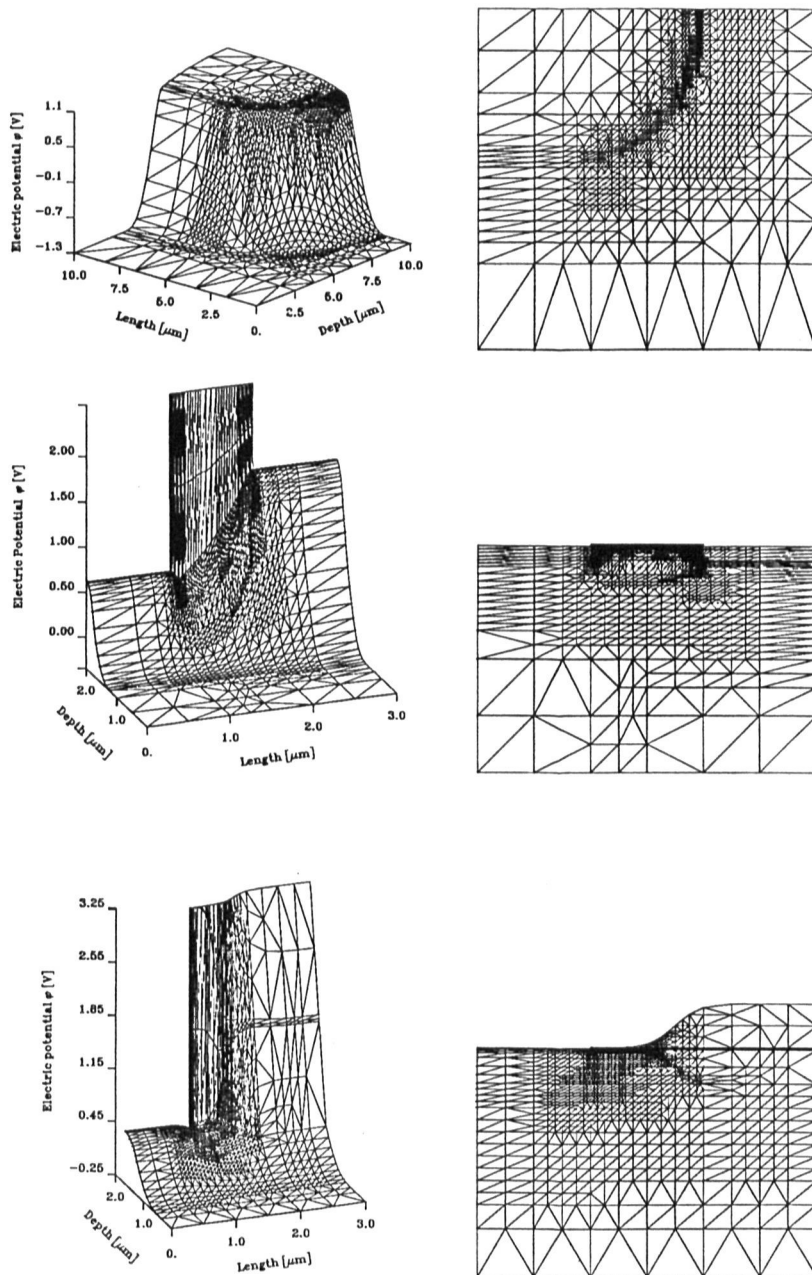


Figure 11: Automatically-generated meshes (right column) and related plots of electric potentials for the three devices of Fig. 10

## 4 Summary and conclusions

An integrated software tool has been presented, which allows generation of meshes that automatically conform to the geometrical and physical features of the problem to be solved. Special care has been devoted to avoid exceedingly large computation times, by adopting several techniques, among which a local solution algorithm and a procedure to evaluate an accurate initial guess. Different approaches to the refinement algorithm have been described, along with their mathematical foundations. Furthermore, a number of results have been presented, concerning reliability, efficiency and accuracy of the scheme.

It is our opinion that adaptive refinement schemes will both improve the friendliness of the user interface of numerical simulation programs, this simplifying their use as practical engineering tools, and constitute a key for a better understanding of the role played by meshes in numerical device simulation.

## Acknowledgements

This work has been partially supported by the Italian National Research Council (Progetto Finalizzato "Materiali e Dispositivi per l'Elettronica") and by the European Economic Community, in the framework of the ESPRIT 962 - EVEREST project. Support from SGS-Thomson is also gratefully acknowledged.

## References

- [1] P. Ciampolini, A. Pierantoni, A. Gnudi, M. Rudan and G. Baccarani, "Adaptive mesh generation preserving the quality of the initial grid" *Proc. of the NUPAD II Workshop*, S.Diego, 1988.
- [2] G. Baccarani, R. Guerrieri, P. Ciampolini and M. Rudan, "HFIELDS: a Highly-Flexible 2-D Semiconductor-Device Analysis Program," in *Proc. of the NASECODE IV Conference*, J. J. H. Miller Ed., Dublin: Boole, pp. 3-12, 1985.
- [3] G. Erlebacher and P. R. Eisman, "Adaptive triangular mesh generation," *AAIA*, vol. 25, no. 10, pp. 1356-1364, 1987.
- [4] A. Forghieri, R. Guerrieri, P. Ciampolini, A. Gnudi, M. Rudan and G. Baccarani, "A new discretization strategy of the semiconductor equations comprising momentum and energy balance," *IEEE Trans. on CAD of ICAS*, vol. 7, no. 2, pp. 231-242, 1988.