Computer Methods in Applied Mechanics and Engineering 110 (1993) 17–24
North-Holland

Finite difference, boundary-fitted grid generation for arbitrarily shaped two-dimensional simulation areas

Claus Fischer[a], Gerd Nanz[b] and Siegfried Selberherr[a]

[a]*Technical University Vienna, Institute for Microelectronics, Gußhausstraße 27–29, A-1040 Vienna, Austria*
[b]*Digital Equipment Corporation, Campus-based Engineering Center, Favoritenstraße 7, A-1040 Vienna, Austria*

Received 27 April 1990
Revised manuscript received 8 January 1993

The algorithm presented analyzes the generation of a Cartesian orthogonal finite difference grid on arbitrary polygons under the restriction that the grid lines should match exactly on the boundaries. This restriction, which is in some cases required by the discretization scheme, can make grid generation impossible. If grid generation is possible, the algorithm automatically generates the finite difference grid. For some other geometries, suggestions are made as to how to change the structure by an arbitrarily small amount to make grid generation possible. If a grid cannot be constructed at all, an exact description of the reason why this is so is delivered. The algorithm has been implemented in the two-dimensional semiconductor device simulation program BAMBI, and is used there to generate the initial grid.

1. Introduction

Grid points in two-dimensional orthogonal finite difference grids usually have four neighbour points in the positive and negative coordinate directions. This means that a set of lines parallel to the x-axis and another set parallel to the y-axis cross at the grid points.

In general, if the boundaries of the integration domain are not parallel to the coordinate axes, a rectangle has to be put around the solution area and a finite difference grid has to be applied. However, some problems require that the grid lines cross exactly at the borders [1].

This requirement causes a severe problem. Wherever a grid line meets a slanted border line, it produces another grid line. When the latter reaches another slanted border line, it creates again a new line. Thus, grid lines are generated ad infinitum. This happens, for example, if one tries to apply finite difference grids to a square which is tilted against the coordinate axes by arbitrary angles (excluding multiples of 45°, Fig. 1). The dashed lines show the square that is approached by the grid lines.

The square is an example of structures which cannot be analyzed with a Cartesian finite difference grid. Some of them can be matched to a Cartesian finite difference grid by a slight change in the geometry (for example, the parallel translation of a boundary line). In this context, 'slight' means that the maximum alteration of the structure lies within user-defined limits and is small compared to the length of the line. There are also structures which look much more complicated and can easily be analyzed without a change.

In general, if arbitrary geometries are desirable, one will certainly try to use finite element programs. Still, there are some undeniable advantages in finite difference discretizations, such as speed and simplicity [2]. Consequently, maximum effort has to be made to apply finite difference methods wherever possible.

In many cases, finite difference grids can be adapted to complex geometries by using transformation

Correspondence to: Dr. C. Fischer, Technical University Vienna, Institute for Microelectronics, Institute 360, Gußhausstraße 27–29, A-1040 Vienna, Austria. Tel. ++43/222/58801-3713.

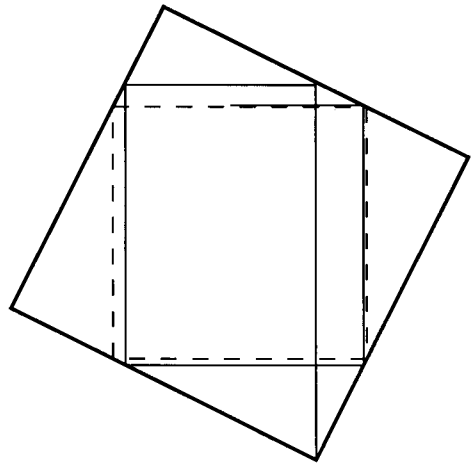

Fig. 1. The grid lines of a tilted square.

methods (curvilinear grids, [3, 4]), or multigrid methods. However, there are problems which require a 'plain finite difference grid'. Such was the case in the device simulation program BAMBI, where the high nonlinearity of the system of semiconductor equations together with the special features of the discretization scheme prohibited the use of these methods. Therefore a new algorithm was developed to extend the possibilities of Cartesian finite difference grids to their utmost limits.

The new algorithm will only fail if it is sure that no grid can be generated. Nevertheless, even in this case it sometimes suggests how a change of the structure would make grid generation possible. It has been implemented in BAMBI (Basic Analyzer of MOS and Bipolar devices) [5–7] to generate the initial grid.

2. The method

2.1. Representation of the geometry on a one-dimensional axis

First, the integration area is divided into proper subsections. A subsection must meet the condition of quasi-convexity, which is a somewhat lower form of convexity, meaning that any line which is parallel to a coordinate axis must have at most one continuous intersection with the subsection. In other words, any two points of the subsection with the same x- (or the same y-) coordinate can be connected with a straight line lying fully inside the subsection. In convex subsections, this condition is fulfilled for any direction, not only for the coordinate directions, thus a partitioning into convex subsections would be sufficient. In the examples of this text, however, and in the implemented program, we used quasi-convex subsections because there are fewer of them.

For each subsection, a surrounding rectangle is determined. It is characterized by the two intervals which are generated by projecting it on the coordinate axes. All these intervals (the x as well as the y) for all the subsections are then mapped on a one-dimensional coordinate axis successively. This is done by linear transformation.

Any grid line in a subsection is now represented by a single location within one of the intervals on the one-dimensional axis. Now, every slanted boundary line gives rise to a linear relation between the coordinates of the x- and y-grid lines which match on the boundary. So, the boundary line can be interpreted as a linear relation between two intervals on the one-dimensional axis, which are subsets of the original intervals of that region. In the following, this is called an 'interval relation'. Not only the boundary lines but also the separation lines between the subsections produce such interval relations. Grid lines which cross the separation lines are continued in the other subsections. This continuation is represented by an an interval relation. Consequently, the representations of the x-coordinates of subsections Ia and Ib in Fig. 4 must be linked by an interval relation, as must the y-coordinates.

The one-dimensional coordinate axis together with the interval relations is fully equivalent to the geometrical structure in all matters pertinent to grid generation.

2.2. Nature of interval relations

An interval relation consists of two intervals which additionally have a 'direction'. In this respect, they differ from conventional intervals. Here we use the notation

$$[a; b] - [c; d] ,$$

for representing an interval relation, meaning that the coordinate a is related to c and b to d. Since the direction is important, it must be allowed that $a > b$ or $c > d$, because otherwise there would be no possibility to relate the smaller coordinate of $[a; b]$ to the larger one of $[c; d]$.

Expressed in words, the correct meaning of the interval relation $[a; b] - [c; d]$ is: whenever a grid line exists which is represented by a coordinate within the interval $[a; b]$, there exists another grid line which is represented by the related coordinate within $[c; d]$. Related coordinates can be expressed by the formula

$$a + x \cdot (b - a) \text{ is related to } c + x \cdot (d - c) , \quad 0 \leq x \leq 1 .$$

Having represented the grid generation problem on a one-dimensional axis with intervals and interval relations, we can begin a more general discussion of the nature of interval relations, which will finally result in specific mechanisms for automatizing the grid generation.

There are some types of interval relations which prevent grid generation. For instance, if one of the two intervals of the interval relation is contained within the other one (Fig. 2(a)), grid lines are generated ad infinitum. Every grid line in the larger interval creates a grid line in the smaller interval. Since the latter automatically lies also in the larger interval, the process of grid line generation is infinite. The same situation occurs if the two intervals have different directions and overlap (Fig. 2(b)). In this case, a part of the interval relation can be split with one interval inside the other one.

Beside these 'forbidden' interval relations, there are many 'allowed' types. One can reduce them into a normalized form by exchanging intervals or interval bounds. The coordinates of all allowed interval relations then fulfill the conditions

$$c < d , \quad \min(a, b) < c , \quad \max(a, b) < d .$$

In the discussion that follows, the interval $[c; d]$ is called the 'right' interval and $[a; b]$ the 'left' one. Table 1 shows five types of normalized interval relations which can be distinguished. The names 'arithmetical' and 'geometrical' are derived from the mathematical type of the (finite) series of grid line coordinates produced by these interval relations. The identical interval relation, which might be created during the analysis of the system of interval relations, can be omitted without consequences because it creates no new grid lines. In the original description of the geometrical structure there are inverting and noninverting interval relations only. The other types are created by the analysis.

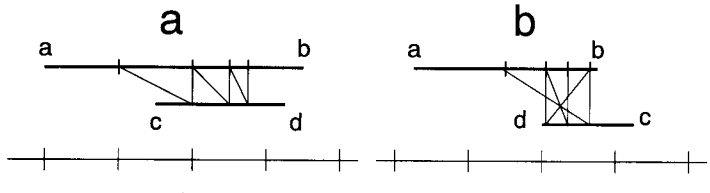

Fig. 2. Forbidden interval relations.

Table 1
Types of normalized interval relations

Definition	Type
$a<b<c<d$	Noninverting
$b<a<c<d$	Inverting
$a=c<b=d$	Identical
$a<c<b<d, b-a=d-c$	Arithmetical
$a<c<b<d, b-a\neq d-c$	Geometrical

2.3. Basic manipulation of interval relations

The original system of interval relations representing the geometry structure can be manipulated to find out whether grid line generation is continued indefinitely. There are three kinds of basic manipulations.

2.3.1. Splitting interval relations

It is possible to substitute an interval relation by two new relations. If the proportion

$$(e-a):(b-a)=(f-c):(d-c)$$

holds with $e\in[a,b]$, the interval relations

$$[a;e]-[c;f] \quad \text{and} \quad [e;b]-[f;d]$$

can be used instead of $[a;b]-[c;d]$. Thus, the relations can be split into parts.

2.3.2. Transferring interval relations

If two interval relations have one interval in common, one of them may be transferred. For example, in

$$[a;b]-[c;d], \qquad [e;f]-[c;d]$$

one can transfer the second interval of the first relation and obtain

$$[a;b]-[e;f]$$

as an equivalent substitute. Of course, the second relation could have been transferred as well. The possibility of this transfer results from abbreviating

$$\text{Line at } a \Leftrightarrow \text{Line at } c \Leftrightarrow \text{Line at } e$$

as

$$\text{Line at } a \Leftrightarrow \text{Line at } e\,.$$

2.3.3. Merging interval relations

If the right intervals of two arithmetical interval relations overlap, they can be merged into one relation. The translation width $c-a$ of the new relation is the 'greatest common divisor' of the translation widths of the original interval relations (see Fig. 3), that is, the original translation widths must be integer multiples of the new translation width. This point needs some further examination. It says, in other words, that if the first arithmetic relation requires that grid lines lie at a distance of 6 units from one another, and another relation requires a distance of 10 units, the resulting grid lines will be at distances of 2 units.

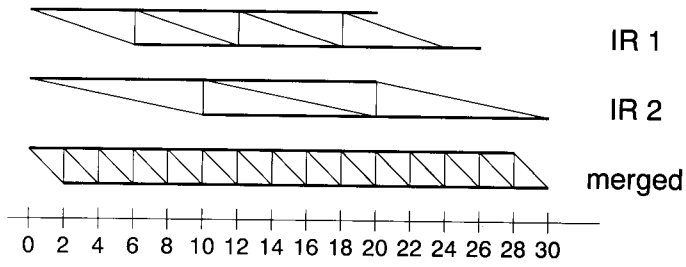

Fig. 3. Merging arithmetical interval relations.

2.4. *Analysis of the system of interval relations*

The crucial point in analyzing a geometric structure or the equivalent interval relation system is whether there is infinite grid line generation or not. On the one-dimensional coordinate axis, grid lines are represented by coordinates. If a grid line lies within one interval of an interval relation, another grid line is created inside the other interval. This grid line may lie in another interval and create another line and so on.

It cannot be ascertained from the original interval relation system whether all grid line generations are finite or not. Nevertheless, there is a class of interval relation systems which evidently [8] prevent infinite line generation. When all interval relations are normalized (see Section 2.2), and the right intervals of the relations do not overlap, all grid line generations are finite. On the other hand, if a geometric structure does not generate grid lines indefinitely, a one-dimensional representation without overlapping right intervals exists which is equivalent to the original representation described in Section 2.1. A proof requires only little effort and can be found in [8]. (Henceforth, the term 'relations overlap' means 'the right intervals of the relations overlap'.)

So after setting up the original system of interval relations, one just has to manipulate them as described in Section 2.3 to separate all right intervals. Whenever interval relations overlap, the overlapping part of one of them is split off and transferred. Arising arithmetical interval relations which overlap with a considerable part are merged. Basically, it would also be possible to merge geometrical interval relations in some cases. However, the probability of this occurring is rather small, so in our implementation all geometrical interval relations which overlap with other arithmetical or geometrical interval relations are considered forbidden interval relations.

There are three possibilities of finishing the separation:

- If the separation succeeds, a finite difference grid can be applied to the geometry. The coordinates of the grid lines can be calculated from the interval relation system. This gives the theoretical possibility of deciding how many additional grid lines would be generated by inserting a single line in a certain region and to plan the refinement of the grid by strategy. (This strategy has not been implemented, since BAMBI does not need it.)
- If the separation leads to a forbidden interval relation, there is no possibility of applying a finite difference grid to the structure. The geometrical reason can be found if the history of the relation is investigated. An example is the tilted square in Fig. 1.
- If the separation leads to overlapping arithmetical interval relations which have no reasonable common divisor length (see Section 2.3.3), the structure can be changed by shifting a boundary line to create such a common divisor. This enables grid generation.

3. Discussion and example

From the complexity of the above description and from the alternatives which are presently available, it can be seen that the new algorithm will not be the first choice when it comes to grid generation. It should be seen as a last resort if more flexible types of grids cannot be applied, and if boundary-fitted Cartesian finite difference grids are required by either the discretization scheme or the total structure of the program.

The advantages of the new method are as follows. There is no problem in recognizing infinite grid line generation loops because there are no such loops. The interval relation system provides exact information about the possibility of grid generation. It uses a compact representation of those elements of the geometry information that are important for the finite difference grid generation because no single parts of the geometry are analyzed but the whole geometrical structure is taken as one piece. There is no need for a special search for regions where infinite grid line generation could happen. In particular, there is no need to insert grid lines and look for consequences. If no grid can be generated, the algorithm provides a clear reason for it. If a slight change of the structure makes grid generation possible or easier, it is able to translate lines within user-applied maximum widths to reduce the number of grid points significantly. (To do this, one must evaluate the influence of the boundary lines upon the interval bounds before interval relations are merged [8]). The good numerical stability compared to the direct treatment of lines in the geometry is a further reason to use this algorithm.

There are also disadvantages that should be mentioned. The expense of code and data handling is high (more than 6000 lines of FORTRAN code in our implementation, special data structures for interval relations which consider the influence of the position of the boundary lines upon the interval bounds). There is a considerable amount of storage needed for the interval relations in comparison to direct grid generation methods.

Figure 4 shows a structure which has been constructed to show the effects of boundary line shifting. The structure consists of three regions I, II, and III, one of which has to be split into two parts (at line 7) which then are quasi-convex (subsections Ia, Ib). A split into convex subsections would also be allowed, although it would create more subsections. The numbered lines are used for the interval relation system.

The coordinate intervals of the subsections are transformed onto a linear axis according to Table 2 by adding an offset. Afterwards, the necessary interval relations are constructed for all structure lines. Table 3 shows the original interval relations named with capital letters and gives information about the structure lines from which they originate. On creating these relations, they are also checked for

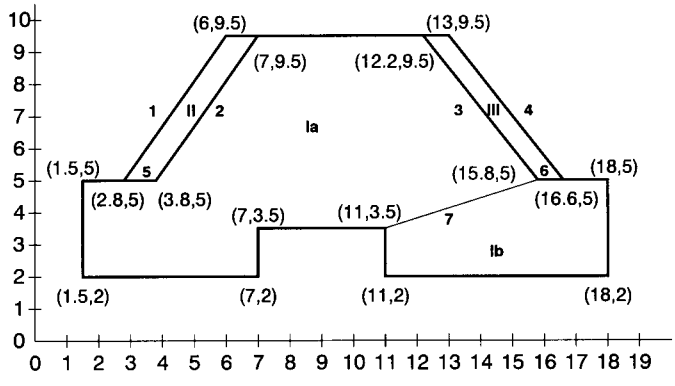

Fig. 4. The trapezoidal example structure.

Table 2
Construction of the one-dimensional linear axis

Region	Coordinate	Offset	Original interval	One-dimensional interval
Ia	x	0	[1.5, 15.8]	[1.5, 15.8]
Ia	y	20	[2.0, 9.5]	[22.0, 29.5]
Ib	x	20	[11.2, 18.2]	[31.2, 38.2]
Ib	y	40	[2,0, 5,0]	[42.0, 45.0]
II	x	50	[2.8, 7.0]	[52.8, 57.0]
II	y	60	[5.0, 9.5]	[65.0, 69.2]
III	x	60	[12.2, 16.6]	[72.2, 76.6]
III	y	80	[5.0, 9.5]	[85.0, 89.5]

Table 3
The interval relations of the example structure

Name	Origin of interval relation	Normalized internal relation	Type
A	x_{Ia}: [3.8, 7.0] – x_{II}: [3.8, 7.0], line 2	[3.8, 7.0]–[53.8, 57.0]*	Noninverting
B	x_{Ia}: [12.2, 15.8] – x_{III}: [12.2, 15.8], line 3	[12.2, 15.8]–[72.2, 75.8]*	Noninverting
C	x_{Ia}: [2.8, 3.8]–x_{II}: [2.8, 3.8], line 5	[2.8, 3.8]–[52.8, 53.8]*	Noninverting
D	x_{Ib}: [15.8, 16.6]–x_{III}: [15.8, 16.6], line 6	[35.8, 36.6]–[75.8, 76.6]*	Noninverting
E	x_{Ia}: [11.0, 15.8]–x_{Ib}: [11.0, 15.8], line 7	[11.0, 15.8] – [31.0, 35.8]*	Noninverting
F	y_{Ia}: [5.0, 9.5]–y_{II}: [5.0, 9.5], line 2	[25.0, 29.5]–[65.0, 69.5]*	Noninverting
G	y_{Ia}: [5.0, 9.5]–y_{III}: [5.0, 9.5], line 3	[25.0, 29.5]–[85.0, 89.5]*	Noninverting
H	Y_{Ia}: [3.5, 5.0]–y_{Ib}: [3.5, 5.0], line 7	[23.5, 25.0]–[43.5, 45.0]*	Noninverting
I	x_{Ia}: [3.8, 7.0]–y_{Ia}: [5.0, 9.5], line 2	[3.8, 7.0]–[25.0, 29.5]*	Noninverting
J	X_{II}: [2.8, 6.0]–y_{II}: [5.0, 9.5], line 1	[52.8, 56.0]–[65.0, 69.5]	Noninverting
J′	Transfer J at F	[25.0, 29.5]–[52.8, 56.0]	Noninverting
J_1	Split J′ at 53.8	[25.0, $26\frac{13}{32}$]–[52.8, 53.8]	Noninverting
J_1'	Transfer J_1 at C	[2.8, 3.8]–[25.0, $26\frac{13}{32}$]	Noninverting
J_1''	Transfer J_1' at I	[2.8, 3.8]–[3.8, 4.8]*	Noninverting
J_2	Split J′ at 53.8	[$26\frac{13}{32}$, 29.5]–[53.8, 56.0]	Noninverting
J_2'	Transfer J_2 at A	[3.8, 6.0]–[$26\frac{13}{32}$, 29.5]	Noninverting
J_2''	Transfer J_2' at I	[3.8, 6.0]–[4.8, 7.0]*	Arithmetic
K	x_{Ia}: [15.8, 12.2]–y_{Ia}: [5.0, 9.5], line 3	[15.8, 12.2]–[25.0, 29.5]	Inverting
K′	Transfer K at I	[7.0, 3.8]–[12.2, 15.8]*	Inverting
L	X_{III}: [16.6, 13.0]–y_{III}: [5.0, 9.5], line 4	[76.6, 73.0]–[85.0, 89.5]	Inverting
L′	Transfer L at G	[29.5, 25.0]–[73.0, 76.6]	Inverting
L_1	Split L′ at 75.8	[29.5, 26.0]–[73.0, 75.8]	Inverting
L_1'	Transfer L_1 at B	[15.8, 13.0]–[26.0, 29.5]	Inverting
L_1''	Transfer L_1' at I	[7.0, $4\frac{23}{45}$] – [13.0, 15.8]	Inverting
L_1'''	Transfer L_1'' at K′	[3.8, $6\frac{13}{45}$]–[$4\frac{23}{45}$, 7.0]*	Arithmetic
L_2	Split L′ at 75.8	[26.0, 25.0]–[75.8, 76.6]	Inverting
L_2'	Transfer L_2 at D	[26.0, 25.0]–[35.8, 36.6]*	Inverting

overlaps, and only nonoverlapping relations are inserted into the system. Note that relations A–H, which originate plainly from linking the subsections, are noninverting relations and do not overlap with their right intervals.

The actual manipulations start when it comes to the slanted interfaces and boundaries. There, the interval relations usually overlap, so they have to be transferred and split at the bounds of other relations, until they reach their final state, which is marked with an asterisk in the table.

The table is incomplete insofar as the manipulation stops at a state where relations have to be merged. In the example, the interval relations J_2'', L_1''' and J_1'' have to be merged to proceed with the analysis. At this state of progress, the algorithm has to decide about finding the greatest common divisor of the translation widths 1.0 and 32/45. In this example, a translation width of 1/45 would solve the problem and give a grid with 33 216 points. However, in realistic examples, the ratio of the translation widths is usually quite arbitrarily placed, which makes the greatest common divisor much smaller and increases the number of grid points drastically. Even here the number might to too large, so we decided to allow a deviation of a maximum of 1% structure line shift. This results in an approximation of the ratio by 5/7, with 3546 grid points. The shift of the thickness of region II is about 0.4%, line 1 is moved to (2.8004. . . , 5)–(6.004. . . , 9.5). Figure 5 shows the grid in this situation. Of course, further reduction of the number of grid points is possible at the expense of further changes; for instance, an approximation of the ratio by 2/3 gives 222 grid points with 6.7% shift.

The example is of course an artefact which shows the limit actions of the method; most of today's common input structures are gridded without changes, and some (like the tilted square) do not allow grid generation at all. The example shows a dense grid in some areas and a sparse grid just beneath.

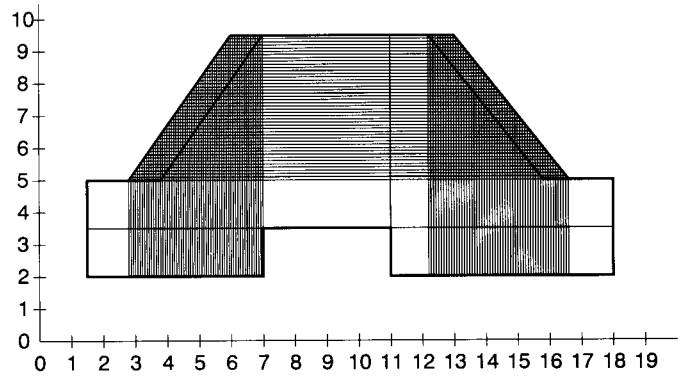

Fig. 5. The example structure with grid.

The grid has then to be completed according to some rules concerning quasi-uniformity etc., which is a fairly straightforward task.

4. Conclusion

An advanced method of improved grid generation under restrictive boundary-matching conditions has been proposed, which approaches the limits of finite difference grid generation at the cost of some effort in program development.

Acknowledgment

This work has been supported by Digital Equipment Corporation at Hudson, USA, and by the Fonds zur Förderung der wissenschaftlichen Forschung, Project Nr. P 7485-PHY.

References

[1] G. Nanz, Numerische Methoden in der zweidimensionalen Halbleiterbauelementsimulation, Technische Universität Wien, Dissertation, 1989.
[2] S. Selberherr, Analysis and Simulation of Semiconductor Devices (Springer, New York, 1984).
[3] J.F. Thompson, Z.U.A. Warsi and C.W. Mastin, Numerical Grid Generation (Elsevier, New York, 1985).
[4] K. Wimmer, R. Bauer and S. Selberherr, Body-fitting coordinate generation for two-dimensional process simulation, in: Proc. Internat. AMSE Conf. on Signals and Systems, Chengdu (China) (AMSE Press, Tassin, France, 1990).
[5] A. Franz, Numerische Simulation von Leistungsfeldeffekthalbleiterbauelementen, Technische Universität Wien, Dissertation, 1984.
[6] G. Franz, Numerische Simulation von bipolaren Leistungshalbleiterbauelementen, Technische Universität Wien, Dissertation, 1984.
[7] A.F. Franz, G.A. Franz, S. Selberherr, C. Ringhofer and P. Markowich, Finite boxes – A generalisation of the finite-difference method suitable for semiconductor device simulation, IEEE Trans. Electron Devices 30 (1983) 1070–1082.
[8] C. Fischer, Gittererzeugung in der zweidimensionalen Halbleiter-Bauelemente-Simulation, Technische Universität Wien, Diplomarbeit, 1989.