TR-O-0088 34

# Software for Design of Semiconductor Multilayer Structures

Pablo O. Vaccaro

1995.12.28

ATR光電波通信研究所

# Software for design of semiconductor multilayer structures

Pablo O. Vaccaro

ATR
Optical and Radio
Communications Research
Laboratories

# Contents

# Abstract

Three computer programs for calculation of optical and electrical properties of semiconductor multilayer structures are presented. The first program calculates the reflectivity and transmissivity of multilayer structures made of AlAs and GaAs alloys. The second program calculates the energy levels and wave functions in arbitrary one-dimensional potential profiles that can represent either the conduction or valence band of a semiconductor multilayer structure. The third program solves selfconsistently the Schrödinger and Poisson equations in one dimension for a semiconductor multilayer structure with a given density of electron-hole pairs. This program also calculates the quasi-Fermi levels and the degree of overlap of electron and hole wave functions. The programs are written in C++ language.

KEYWORDS: semiconductors, multilayer structure, optical properties, transfer-matrix method.

# 1. Introduction

Multilayer structures are the key components of most of the modern optoelectronic devices made with semiconductor compounds. These multilayer structures are designed to obtain certain electric potential profiles, as in high-electron mobility transistors (HEMTs),[1] optical properties, as in the distributed Bragg reflectors (DBRs) used in vertical-cavity surface-emitting lasers (VCSELs)[2] and vertical-cavity optical modulators[3], or both electrical and optical properties, as in the active region of quantum-well lasers.[4]

The design of these devices requires on one hand a detailed knowledge of energy levels and wave functions and their dependence on carrier density. On the other hand it requires a knowledge of the optical properties like reflectivity and transmissivity for different light wavelengths.

In this report I present three computer programs that calculates these properties for arbitrary multilayer structures made of III-V semiconductor compounds. These programs were written firstly to meet the needs of my research on VCSELs[5] and piezoelectric effect[6,7] in quantum wells. The first program, named MIRROR, calculates reflectivity and transmissivity of multilayer structures made of $Al_xGa_{1-x}As$. The second program, named WAVY, calculates the energy levels and wave functions in an arbitrary potential profile. It also calculates the energy level's dependence on an applied electric field. The third program, named WAVYPOI, solves selfconsistently the Schrödinger and Poisson equations for arbitrary band edges profiles when there is a given concentration of electron-hole pairs in the structure. Application examples are included for each program. The source files are presented in the Appendix. All the programs are written in TurboC++ version 1.01 of Borland Intnl.

# 2. MIRROR: Reflectivity and transmissivity of multilayer structures

## *2.1 Theoretical background*

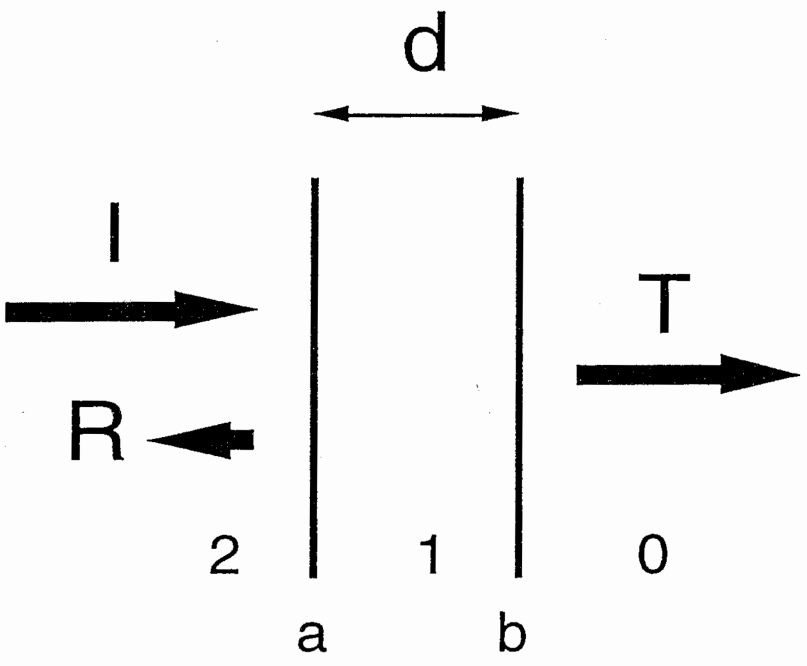

**Figure 1: Reflection and transmission of a dielectric slab**

This program calculates the reflectivity and transmissivity of a series of slabs of $Al_xGa_{1-x}As$ with x between 0 and 1, separated by abrupt plane boundaries, across which the electric and magnetic fields parallel to the boundary are continuous.[8] Figure 1 shows the incident, reflected and transmitted light in one slab of material "1" with thickness *d* between slabs of materials "0" and "2". The tangential components of the electric and magnetic fields are equated at each interface "a" and "b", and the characteristic matrix for each layer is obtained:

$$\begin{bmatrix} E_a \\ H_a \end{bmatrix} = \begin{bmatrix} \cos\delta_1 & \dfrac{i\cdot\sin\delta_1}{y_1} \\ \dfrac{i\cdot\sin\delta_1}{y_1} & \cos\delta_1 \end{bmatrix} \begin{bmatrix} E_b \\ H_b \end{bmatrix} \qquad \textbf{eq. 2-1}$$

where $y_1 = n_1 - i\cdot k_1$ is the optical admittance and $\delta_1 = 2\cdot\pi\cdot y_1\cdot d_1 / \lambda_{\text{light}}$. This result can be extended to *n*-layers and after rearranging the following equation is obtained:

$$\begin{bmatrix} B \\ C \end{bmatrix} = \prod_{j=1}^{n} \left\{ \begin{bmatrix} \cos\delta_j & \dfrac{i \cdot \sin\delta_j}{y_j} \\ \dfrac{i \cdot \sin\delta_j}{y_j} & \cos\delta_j \end{bmatrix} \right\} \begin{bmatrix} 1 \\ y_{sub} \end{bmatrix} \qquad \text{eq. 2-2}$$

where $B = E_a / E_b$, $C = H_a / H_b$ and $y_{sub}$ is the substrate's optical admittance. The reflection coefficient is calculated as:

$$R = \frac{(y_0 B - C)(y_0 B - C)^*}{(y_0 B + C)(y_0 B + C)^*} \qquad \text{eq. 2-3}$$

where $y_0$ is the optical admittance of the incidence media (usually, it is air).

The transmission coefficient is calculated as:

$$T = \frac{4 y_0 \cdot \mathrm{Re}(y_{sub})}{(y_0 B + C)(y_0 B + C)^*} \qquad \text{eq. 2-4}$$

For oblique incidence, the expressions remain the same except that $\delta_j$ and $y_j$ are replaced by their equivalents for oblique incidence:

$$\delta_j = (2\pi / \lambda) d_j (n_j^2 - k_j^2 - n_0^2 \sin^2\theta_0 - 2 i n_j k_j)^{1/2}$$

$$\eta_s = (n_j^2 - k_j^2 - n_0^2 \sin^2\theta_0 - 2 i n_j k_j)^{1/2} \qquad \text{eq. 2-5}$$

$$\eta_p = y_j^2 / \eta_s$$

where $\eta_s$ and $\eta_p$ correspond to *s*-polarization and *p*-polarization respectively. The electric vector is in the plane of incidence for *p*-

polarized light and normal to the plane of incidence for *s*-polarized light.

Absorption data of the GaAs[9] is included as a separated file called ABSOR.DAT. Absorption for different composition of AlGaAs is calculated by shifting adequately the band gap absorption edge energy. This approach is accurate for photon energies up to 2.4 eV.

## 2.2 *Results that can be obtained*

Structures including up to 900 slabs of different AlGaAs alloys can be calculated. The incidence media and the substrate can be chosen as an AlGaAs alloy or air. A complete VCSEL structure is easily computed.

Three different calculation of reflectance and transmittance can be realized:

- Dependence on photon energy or light wavelength.
- Dependence on incidence angle.
- Dependence on number of periods, in the case of a periodic structure.

Both *s*- and *p*-polarization are calculated in the three cases.

## 2.3 *User's manual*

### 2.3.1 Writing the input file

Write the input file following the model of Figure 2. The first six lines of the file are the parameters of the calculation:

- The first two define the photon energy range in the case of calculating the dependence on photon energy.
- The third line define the number of points to appear in the output file. More points means longer calculation time.
- The fourth line define the incidence angle. The incidence angle is scanned from 0° up to the value defined in the fourth line in the case of calculating the dependence on incidence angle.
- The fifth line allows the inclusion of a systematic change in the layers' thickness as would happen in the case of AlGaAs grown by

molecular beam epitaxy (MBE) when the effusion cells temperature drift slowly with time. The value indicates the percentile difference in thickness of the last layer of the structure as compared to the first layer. Both positive and negative values are allowed.

- The sixth line allows the inclusion of a random change in the layers' thickness as could happen in structure with very thin layers. The value indicates the percentile difference between the thinnest and the thickest layer.

These numerical values must be written starting after the 50*th* character of the line to be correctly read by the program.

```
E initial                                          1.1   eV
E final                                            1.5   eV
Number of points                                   200
Incidence angle                                    0.0    degrees
Systematic change in layer thickness               0.0
Random maximun change in layer thickness           0.0
1               0.0             0.0
10              1.0             0.08305
0               0.0             0.0696
1               1.0             0.08305
1               0.0             0.1392
10              1.0             0.08305
0               0.0             0.0696
1               9.9             0.0
9.9
```

**Figure 2: Model of the input file for MIRROR**

The following three columns describe the multilayer structure:

- The first column indicates the number of periods. Each period can be composed by many layers. These layers are indicated with a zero.
- The second column indicates the aluminum composition *x* of each layer, i.e., 0 means GaAs and 1 means AlAs.

- The third column is the thickness in microns of each layer.
- The first line is the substrate. The value of the third column for this line is not used.
- The last line is the incidence media. Also the value of the third column is not used. If the incidence media or the substrate are air, write 9.9 in the second column. Absorption in the incidence media always is taken as zero.
- The final 9.9 in the first column is the "end-of-file" code for the program.

### 2.3.2 Running MIRROR

After writing and saving the input file, run MIRROR. It will display a menu asking for the names of the input and output file and which *x*-axis do you want:

1. Energy: calculates the dependence on photon energy between the values indicated in the input file.
2. Incidence angle: calculates the dependence on incidence angle between 0° and the angle indicated in the input file (0° is perpendicular to the surface). The photon energy used in the calculation is the initial value (first line of input file).
3. Number of periods: calculates the dependence on the number of periods starting from one period up to the "number-of-points" value in the input file. Each period is the full structure described in the input file. The photon energy used in the calculation is the initial value (first line of input file) and the incidence angle is one indicated in the fourth line of the input file.

   The program will emit a short melody when the calculation is completed.

### 2.3.3 The output file

An example of output file is shown in Figure 3. It contains five columns. The first column is the *x*-axis chosen when running the program. The second and fourth columns are the reflectance for *s*- and *p*-polarization respectively, and the third and fifth columns are the transmittance for *s*- and *p*-polarization respectively.

```
                 s-polarization             p-polarization
             reflec.        transmis.     reflec.        transmis.
1.1000 6.900347e-01 3.099653e-01 6.900347e-01 3.099653e-01
1.1020 6.978206e-01 3.021794e-01 6.978206e-01 3.021794e-01
1.1040 7.017869e-01 2.982131e-01 7.017869e-01 2.982131e-01
1.1060 7.020046e-01 2.979954e-01 7.020046e-01 2.979954e-01
1.1080 6.984299e-01 3.015701e-01 6.984299e-01 3.015701e-01
1.1100 6.909005e-01 3.090995e-01 6.909005e-01 3.090995e-01
1.1120 6.791332e-01 3.208668e-01 6.791332e-01 3.208668e-01
1.1140 6.627264e-01 3.372736e-01 6.627264e-01 3.372736e-01
1.1160 6.411735e-01 3.588265e-01 6.411735e-01 3.588265e-01
1.1180 6.139007e-01 3.860993e-01 6.139007e-01 3.860993e-01
1.1200 5.803492e-01 4.196508e-01 5.803492e-01 4.196508e-01
1.1220 5.401290e-01 4.598710e-01 5.401290e-01 4.598710e-01
1.1240 4.932770e-01 5.067230e-01 4.932770e-01 5.067230e-01
1.1260 4.406232e-01 5.593768e-01 4.406232e-01 5.593768e-01
1.1280 3.842031e-01 6.157969e-01 3.842031e-01 6.157969e-01
1.1300 3.275148e-01 6.724852e-01 3.275148e-01 6.724852e-01
1.1320 2.752838e-01 7.247162e-01 2.752838e-01 7.247162e-01
1.1340 2.324595e-01 7.675405e-01 2.324595e-01 7.675405e-01
```

**Figure 3: Example of output file for MIRROR.**

## *2.4 Examples*

### 2.4.1 Example 1: A distributed Bragg reflector with chirped superlattices at the interfaces

A distributed Bragg reflector (DBR) is a multilayer structure of slabs of two alternating materials with different refraction index and one fourth of the light wavelength thick. The structure has a high reflectance in a certain range of energy that depends on the difference of refractive index between the two materials. The reflectance increases with the number of periods.

Many applications require a low electrical resistance perpendicular to the multilayer structure. The electrical resistance in the case of AlGaAs alloys is quite large due to the band gap differences and potential steps in the band edges. The electrical resistance is specially high in *p*-type doped DBRs. A solution to this problem is to include a gradual change of alloy composition at the slabs' interfaces. This approach is not practical in the case of DBRs grown by MBE, but a "digital" gradual change in composition can be obtained by making a chirped superlattice[10] as shown in Figure 4.

Figure 5 shows the results of the calculation for DBRs with and without the chirped alloy at the interfaces. These DBRs has twenty periods of alternating layers of AlAs and GaAs. The DBR with chirped superlattices does not show reduction of the reflectance but there is a small decrease in the energy width of the high reflection region.

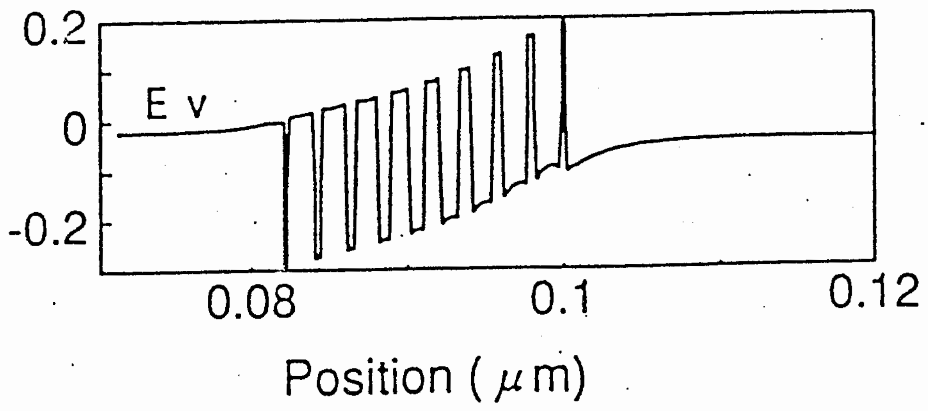

**Figure 4: Chirped superlattice at the DBR's interfaces**

This decrease can be understood qualitatively if we consider the effect of the superlattice on the refraction index of each slab. By choosing arbitrarily the boundary of the slabs at the center of the superlattice, it can be seen that some amount of GaAs is now in the AlAs slabs, and some amount of AlAs is in the GaAs slab. It means that the average composition of the slabs changed when including the superlattices.

This change in composition produces a decrease in the difference of reffraction index between slabs. Since the width of the high reflection region in a quarter-wavelength stack is given by the difference in reffraction index between the constituents materials, it is natural to observe a decrease of this width when the superlattices are included in the structure. This example also shows how sensitive is the reflecton spectra to small changes in the structure.

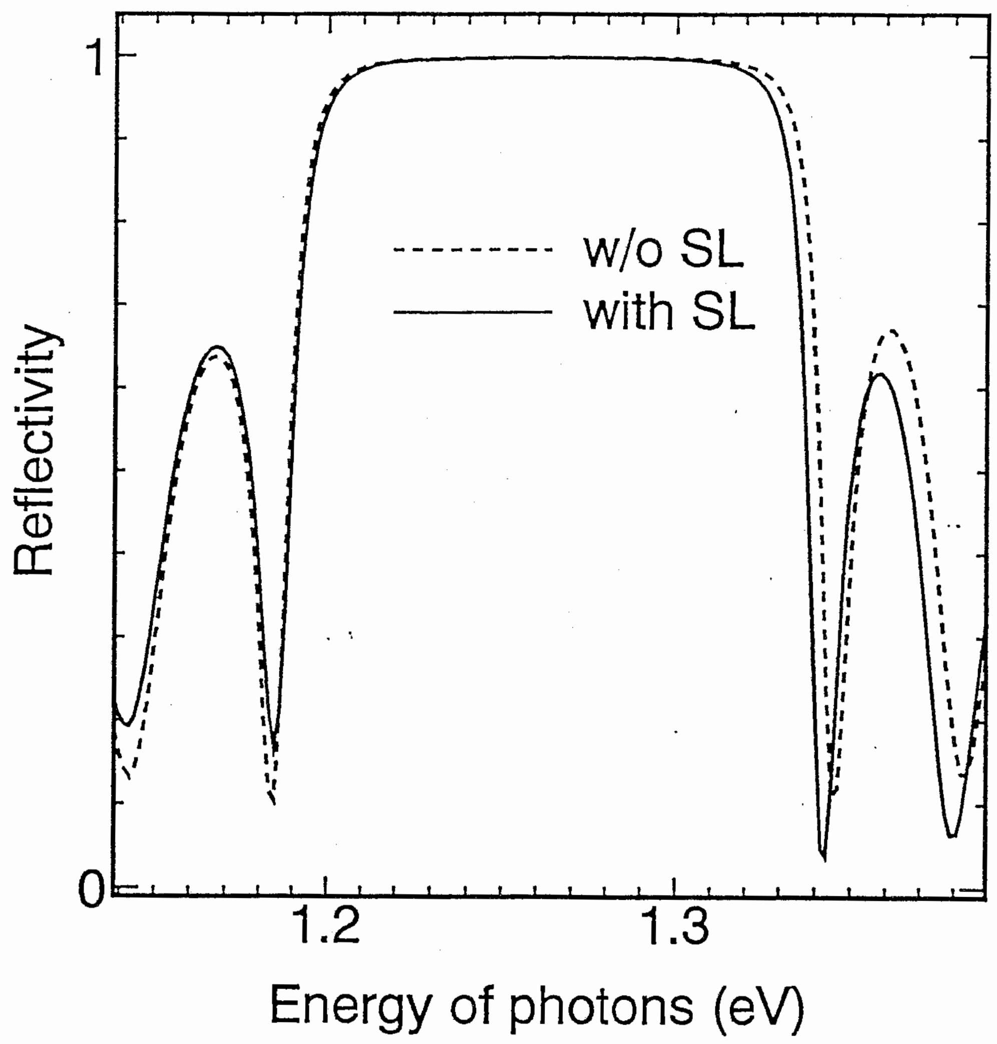

**Figure 5: Influence of the chirped superlattice on the DBR's reflectivity.**

### 2.4.2 Example 2: Influence of the variations in layer thickness

Variations in layer thickness can be systematic or random. Systematic variation in the layers thickness for structures grown by MBE appear because a drift of the effusion cell temperature or because depletion of source material. Random variations are expected in structures with very thin layers because the time error in the opening and closing of the cells' shutters. Figure 6 and Figure 7 show the reflectance spectra of an AlGaAs/GaAs DBR with twenty periods when systematic and random variations in the layers thickness are included. Systematic variations have a much stronger influence on the reflectance spectra than random variations.

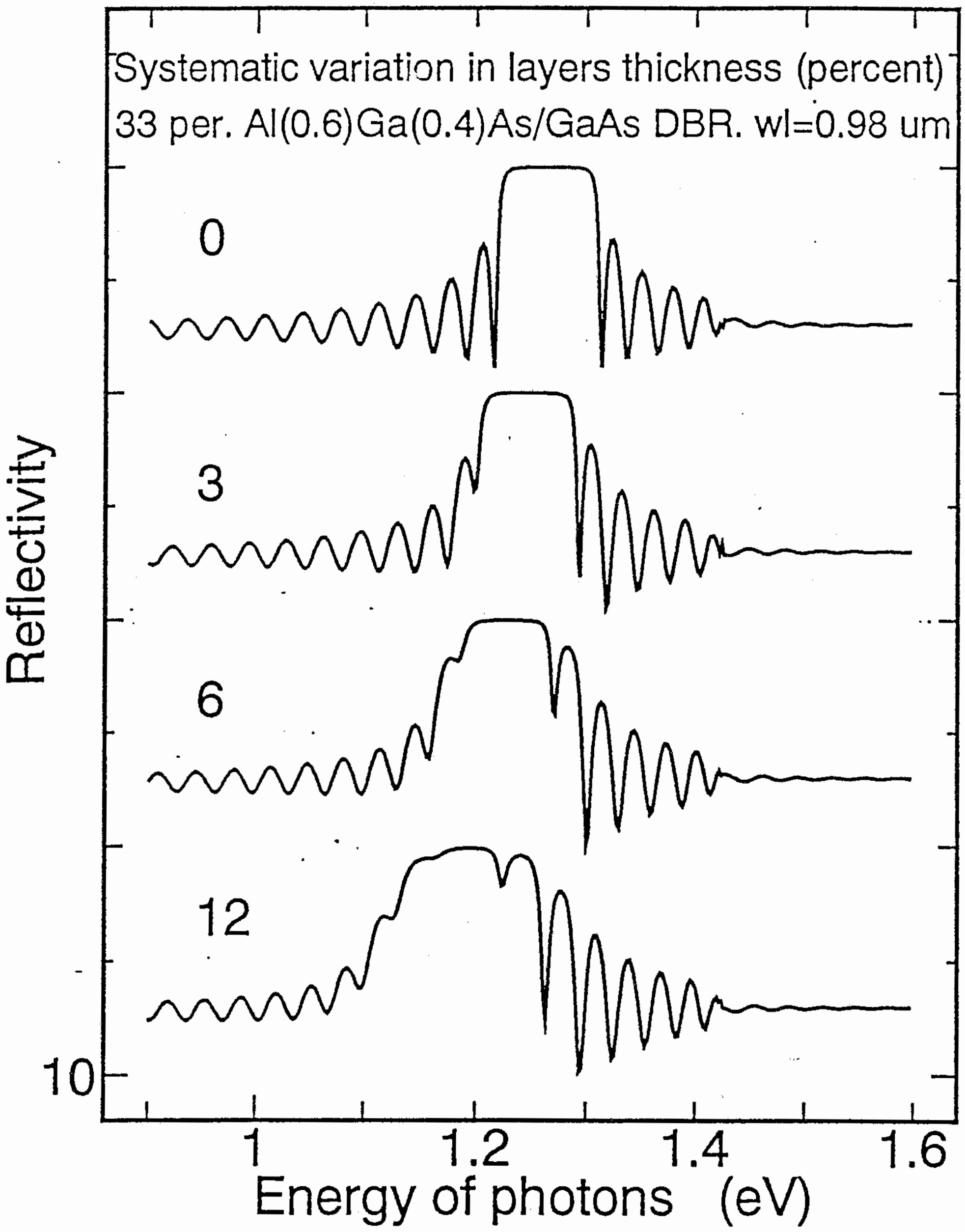

**Figure 6: DBR with a systematic variation in the layer thickness.**

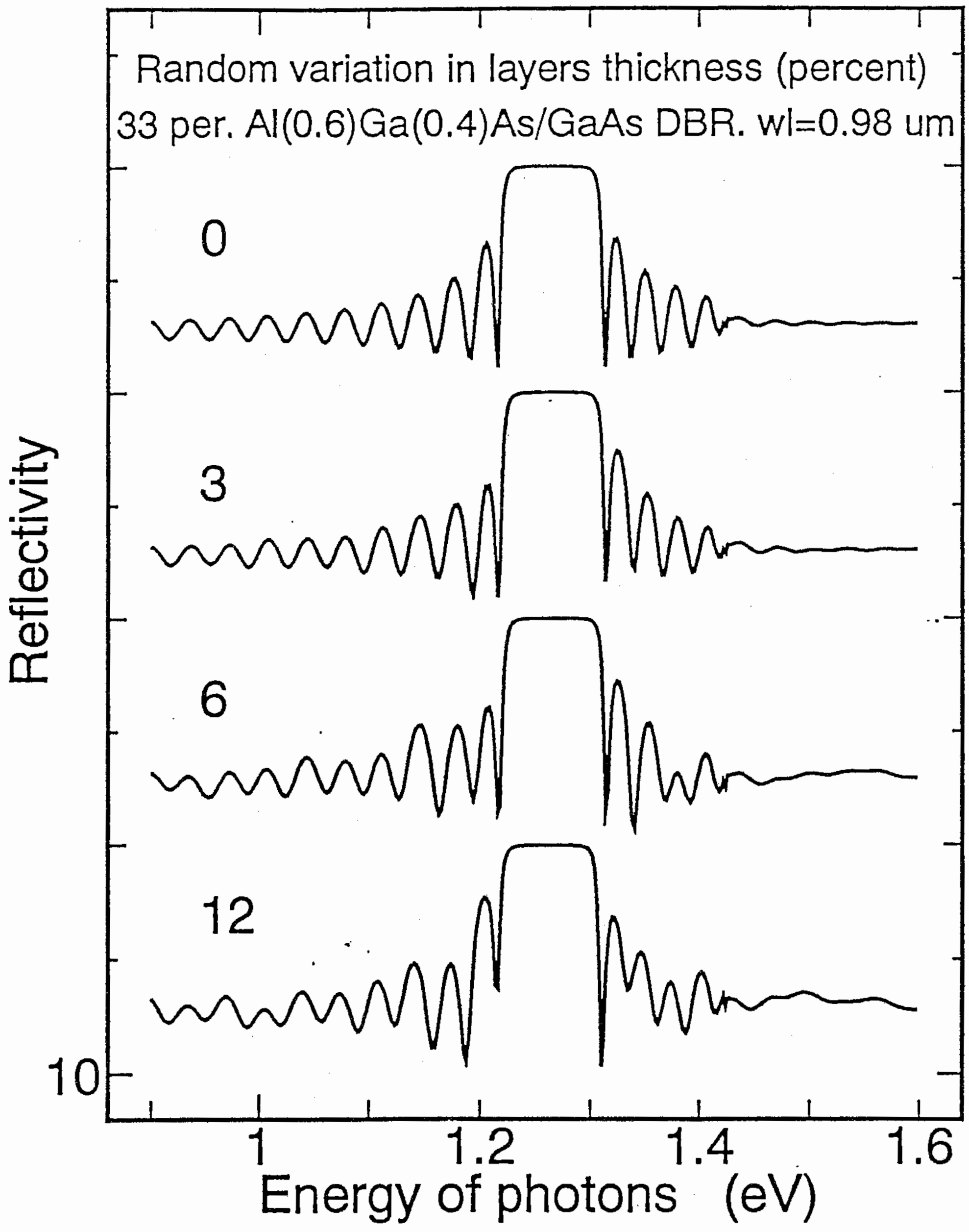

**Figure 7: DBR with a random variation in the layer thickness.**

### 2.4.3 Example 3: Reflectance dependence on angle of incidence for a VCSEL structure

A typical VCSEL structure consists of two DBRs with an optical cavity one wavelength thick in between that conforms a Fabry-Perot resonator. Light generated in the cavity is reflected back if the photon energy is in the high reflection region of the DBRs unless that it has just the energy of the cavity resonance. In this case it can escape from the cavity and contribute to the laser radiation from the device. But the reflectance of the DBRs has also an strong dependence on the angle of incidence. Figure 8 shows this dependence for a VCSEL structure composed of two 33-periods DBRs and one-wavelength long cavity illuminated with light at the resonant wavelength (0.98 micron). At 0° degree, the reflectivity is very small because the presence of the resonant mode. The DBRs show a high reflectance up to 17°. The *p*-polarized light reflectance decreases at a smaller angle than the *s*-polarized light. Between 17° and 70° degrees there is a low reflectance window where most of the light can escape from the VCSEL structure. For angles larger than 70°, there is total internal reflection and the light is waveguided in the plane of the structure.

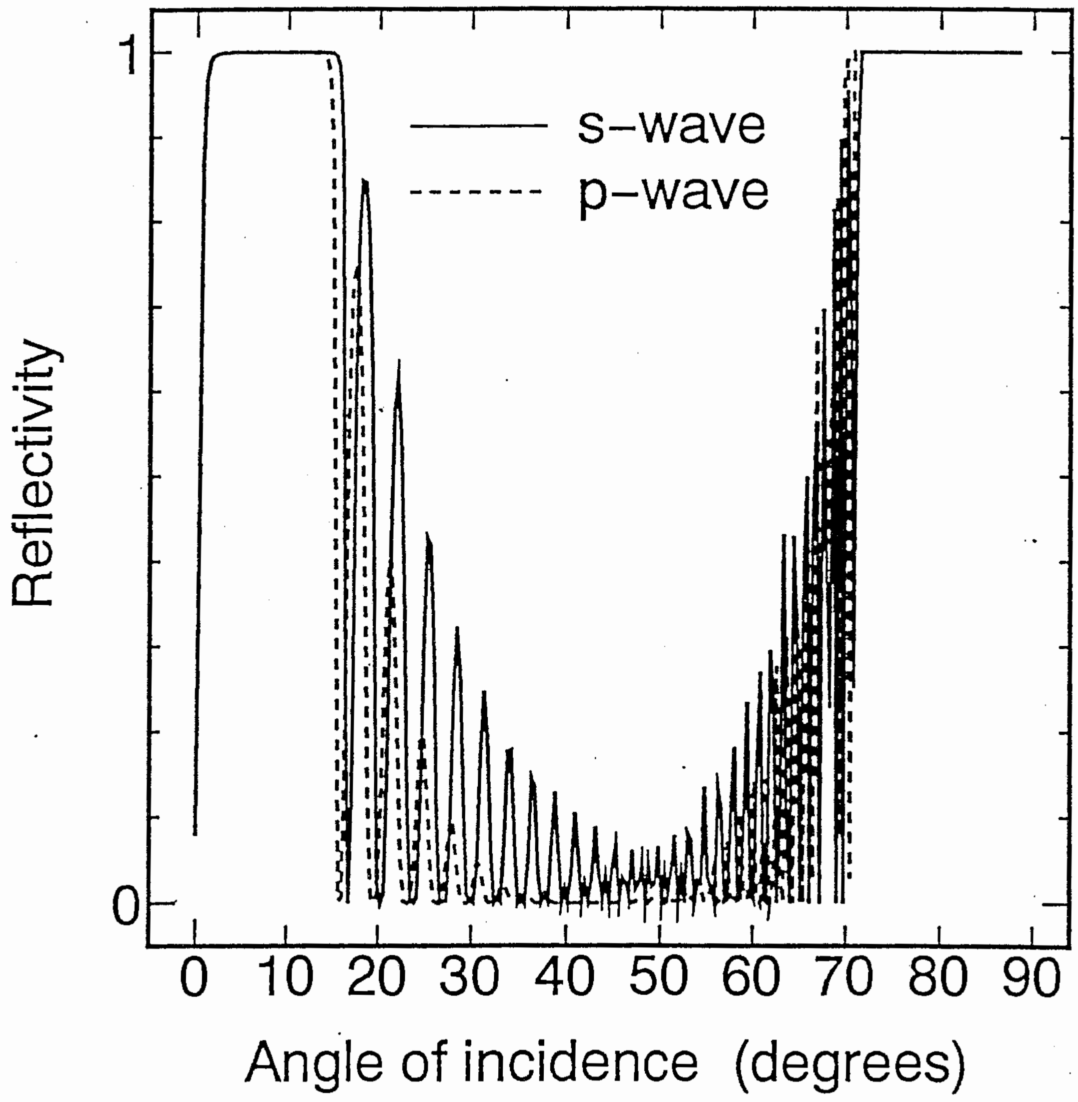

**Figure 8: Reflectivity of a Fabry-Perot resonant cavity. The light wavelength is equal to the cavity resonance.**

# 3. WAVY: Energy levels and wave functions in a arbitrary potential profile

## *3.1 Calculation method*

The properties of charged particles in a multilayer semiconductor material can be found by solving the one-dimensional Schrödinger equation

$$\frac{\hbar^2}{2}\frac{d}{dz}\left(\frac{1}{m^*(z)}\frac{d}{dz}\right)\chi_i + V_0(z)\chi_i = E_i\chi_i \qquad \text{eq. 3-1}$$

where $V_0(z)$ is the potential profile of the structure in the *z* direction, $E_i$ are the energy levels and $\chi_I$ are the wave functions. Here, I follow the envelope-function approximation by Bastard[11]. In this approximation all rapidly varying phenomena, on the scale of atomic cells, are discarded.

Solving the Schrödinger equation in arbitrary potential profiles requires using a numerical method. The transfer-matrix method can be used for a wide range of problems dealing with second-order differential equations. In this algorithm, the potential profile is approximated with a piece wise linear potential profile. The solution of the Schrödinger equation in this potential profile can be expressed as a linear combination of Airy functions.[12] However, evaluating correctly the Airy functions is a difficult task, especially if the slope of the potential profile is close to zero. A more reliable approach is to subdivide each layer with slope different from zero into several thinner layers with constant potential, and to write the wave functions as a complex exponential functions in each layer *j*:[13]

$$\chi^j(z) = A_j \cdot e^{p_j(z)} + B_j \cdot e^{-p_j(z)} \qquad \text{eq. 3-2}$$

where $p_j(z)$ is a complex function, given by

$$p_j(z) = \begin{cases} \Gamma_0 \cdot x & (j=0) \\ \Gamma_j \cdot (x - x_{j-1}) & (j>0) \end{cases} \qquad \text{eq. 3-3}$$

The unknown complex constants $A_j$ and $B_j$ are to be determined by the boundary conditions for the wave functions. The variable $\Gamma_j(E)$ is related to the eigenvalue (energy level) $E$ by the expression

$$\Gamma_j(E) = i \cdot \sqrt{\frac{2 \cdot m_j^*}{\hbar^2} \cdot (E - V)_j} \,. \qquad \text{eq. 3-4}$$

The boundary conditions for the wave functions at the border between two layers can be written as

$$\chi_{j-1}(x_{j-1}) = \chi_j(x_{j-1}) \qquad \text{eq. 3-5}$$

$$\frac{1}{m_{j-1}^*} \cdot \frac{d}{dx}\left[\chi_{j-1}(x_{j-1})\right] = \frac{1}{m_j^*} \cdot \frac{d}{dx}\left[\chi_j(x_{j-1})\right] \qquad \text{eq. 3-6}$$

Applying these boundary conditions eq. 3.5 and eq. 3.6 to eq. 3.3, it is possible to derive an expression that relates the $A$ and $B$ constants in layer $j+1$ in the following way:

$$\begin{bmatrix} A_{j+1} \\ B_{j+1} \end{bmatrix} = M_j \cdot \begin{bmatrix} A_j \\ B_j \end{bmatrix} \qquad \text{eq. 3-7}$$

where $M_j$ is a 2x2 matrix. By repeatedly applying eq. 3.7 we can find a relation between the $A$ and $B$ coefficients in the outermost layers

$$\begin{aligned} \begin{bmatrix} A_{N-1} \\ B_{N-1} \end{bmatrix} &= M_{N-2} \cdot M_{N-3} \cdots M_1 \cdot M_0 \cdot \begin{bmatrix} A_0 \\ B \end{bmatrix} \\ &= \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ B \end{bmatrix} \end{aligned} \qquad \text{eq. 3-8}$$

One more constraint is necessary to obtain the solution. The wave functions should be bound in space, that is, they should be zero in amplitude when $x$ tends to plus or minus infinity. This constraint demands that $A_0$ and $B_{N-1}=0$, which is satisfied if the coefficient $\alpha_{22}$ equals zero in eq. 3.8. In other words, a solution to eq. 3.1 is found whenever an eigenvalue $E$ fulfills the requirement

$$\alpha_{22}(E) = 0. \qquad \text{eq. 3-9}$$

In general, $\alpha_{22}$ is complex and both the real and imaginary parts should be zero to satisfy the eigenvalue condition.

The algorithm used in WAVY scans *E* and finds all the zeros of eq. 3.9. Coefficients *A* and *B* for each zero are stored and used to construct the wave functions.

## *3.2 Proposed applications*

Designing quantum well structures without being able to predict the energy levels and wave functions would limit us to copy structures already published adding some minor changes in dimensions or materials. When we face the task of creating devices with novel structures that involve discrete energy levels, a software able to calculate energy levels and wave functions in arbitrary potential profiles becomes indispensable.

WAVY calculates single and multiple quantum wells, asymmetric quantum wells and superlattices. In the case of superlattices, the discrete levels of the finite superlattice structure and its wave functions can be obtained. The miniband structure can be obtained later by calculating the energy dependence of the density of levels. A very useful feature of this program is that it calculates also the dependence of the energy levels on an external applied bias. This feature is very useful when looking for the spatial localization of carriers in the structure under applied bias.

## *3.3 User's manual*

### 3.3.1 Writing the input file

Write the input file following the model of Figure 9. The seven first lines of the file describe the calculation to be performed:

- Initial energy step: is the step used for scanning energy when looking for bound levels. It should be smaller than the smallest energy separation between successive energy levels, otherwise it could happen that energies levels are not found. A smaller step means also longer calculation time.
- Number of energy levels you want: is useful for reducing the calculation time when you are interested only in the lower energy levels.

- Starting and ending energies: is useful for scanning a small energy range with high resolution, and allows to resolve energy levels that have very little energy separation. If you want to scan all the bound-levels region of the structure, set values large enough to cover this region. The program will scan only the bound-levels region.
- If you want to obtain the energy levels when an external field is applied to the structure, write the Minimum, Maximum and Step fields in kV/cm units. The program will calculate the energy levels dependence on the electric field and will omit the wave function calculation. If you want to obtain the wave function, set both Minimum and Maximum fields to the same value.

```
Initial energy step (meV)                          3
Number of energy levels you want (1-7)             5
Starting energy (eV)                               -.5
Ending energy (eV)                                 1.0
Minimum external applied field (kV/cm)             00
Maximum external applied field (kV/cm)             40
Field step (kV/cm)                                 5
10.0      .067      0.0       0.0                          1
5.0       .067      0.0       0.0                          5
4.5       .067      -.124     0.0                          9
3.0       .067      0.0       0.0                          6
6.0       .067      -.124     0.0                          12
5.0       .067      0.0       0.0                          5
0.0       .067      0.0       0.0                          1

Lz        m*/m0     V        Field                      steps
(nm)               (eV)      (kV/cm)
```

**Figure 9: Model for the input file of WAVY.**

These numerical values must be written starting after the 50*th* character of the line to be correctly read by the program.

The following five columns describe the multilayer structure:

- Data for the material to the left (right) of the quantum structure is written in the first (last) line of the structure's table.

- The first column is the thickness of each layer in nanometers. The first value is not used but must be different of zero.
- The second column is the carrier's effective mass in free-electron-mass units.
- The third column is the potential in electron volts at each layer measured when there is not electric field in the structure. The first value must be zero.
- The fourth column is the electric field in each layer in kV/cm units. Positive values mean that the potential increases toward the right side. The first and last value must be zero.
- The fifth column is the number of steps in which each layer is divided to approximate the slope given by the electric field. If the electric field in a particular layer is zero, one step is enough. Four to eight steps are enough for most of the usual QW structures. Try different number of steps to estimate the error.

  If an external field is going to be applied, remember to write enough steps in each layer.

  The design of an structure is quite straightforward. You only need to draw in paper the structure without any electric field (include only the band edge alignment) to obtain the right potential values.

### 3.3.2 Running WAVY

After writing and saving the input file, run WAVY. It will display a menu asking for the names of the input and output files. After introducing the file names, the evolution of the calculation is displayed: the range of the energy scan, the applied electric field, the energy scan position and the number of energy levels found can be followed. The program will emit a short melody when the calculation is completed.

### 3.3.3 The output file

Two kinds of output file are obtained, accordingly to the performed calculation. The first kind is obtained when there is not scan of the applied electric field (see Figure 10).

```
levels:   -0.07952    -0.05151    0.00000     0.00000     0.00000
Wavefunctions
X(nm)        e1          e2          e3          e4          e5
0.000    3.492e-04   4.127e-03   0.000e+00   0.000e+00   0.000e+00
0.117    3.813e-04   4.429e-03   0.000e+00   0.000e+00   0.000e+00
0.235    4.163e-04   4.754e-03   0.000e+00   0.000e+00   0.000e+00
0.352    4.545e-04   5.102e-03   0.000e+00   0.000e+00   0.000e+00
0.470    4.963e-04   5.476e-03   0.000e+00   0.000e+00   0.000e+00
0.587    5.418e-04   5.878e-03   0.000e+00   0.000e+00   0.000e+00
0.705    5.916e-04   6.308e-03   0.000e+00   0.000e+00   0.000e+00
0.823    6.459e-04   6.771e-03   0.000e+00   0.000e+00   0.000e+00
0.940    7.053e-04   7.267e-03   0.000e+00   0.000e+00   0.000e+00
1.057    7.701e-04   7.799e-03   0.000e+00   0.000e+00   0.000e+00
1.175    8.408e-04   8.371e-03   0.000e+00   0.000e+00   0.000e+00
1.292    9.180e-04   8.984e-03   0.000e+00   0.000e+00   0.000e+00
1.410    1.002e-03   9.643e-03   0.000e+00   0.000e+00   0.000e+00
1.527    1.094e-03   1.035e-02   0.000e+00   0.000e+00   0.000e+00
1.645    1.195e-03   1.111e-02   0.000e+00   0.000e+00   0.000e+00
1.762    1.305e-03   1.192e-02   0.000e+00   0.000e+00   0.000e+00
1.880    1.425e-03   1.280e-02   0.000e+00   0.000e+00   0.000e+00
1.997    1.555e-03   1.373e-02   0.000e+00   0.000e+00   0.000e+00
2.115    1.698e-03   1.474e-02   0.000e+00   0.000e+00   0.000e+00
2.232    1.854e-03   1.582e-02   0.000e+00   0.000e+00   0.000e+00
```

**Figure 10: Output file of WAVY. Energy levels and wave functions.**

In this case, the first line of the output file shows the energy levels in electron-Volt units. The energy level's values are relatives to the potential profile zero-of-energy. Every value should be negative because they are bound levels. If an energy level has not the negative sign, it means that the program has set a flag for this level's calculation. The energy value is correct (changing the sign from positive to negative) but there is some possibility that the wave function is not correct. This problem appears when the wave function is so small in the last layer of the structure that it cannot be properly evaluated due to numerical error. You have to cut from the input file the regions where the wave function is negligible and run WAVY again. The remaining of the file contains the wave functions' information. The first column is the spatial position in nanometer units. From the second to the eight column are the wave functions. The ninth column shows the actual potential profile including the electric field and used in the calculation, in electron-Volt units. Each column has 200 datum ready to plot.

The second kind of file is obtained when the external applied field is scanned (see Figure 11). The first column is the external applied field and the remaining columns are the energy levels' values.

```
E (kV/cm)    e1        e2        e3        e4        e5        e6        e7
0.00    -0.07952 -0.05151 0.00000 0.00000 0.00000 0.00000 0.00000 0.000
5.00    -0.07285 -0.04676 0.00000 0.00000 0.00000 0.00000 0.00000 0.000
10.00    -0.06649 -0.04180 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
15.00    -0.06047 -0.03656 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
20.00    -0.05484 -0.03103 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
25.00    -0.04959 -0.02521 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
30.00    -0.04471 -0.01912 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
35.00    -0.04015 -0.01283 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
40.00    -0.03585 -0.00645 0.00000 0.00000 0.00000 0.00000 0.00000 0.00
```

**Figure 11: Output file of WAVY. Dependence of energy levels on applied electric field.**

## *3.4 Examples*

### 3.4.1 Example 1: Two asymmetrical InGaAs/GaAs quantum wells

The structure consists of two InGaAs/GaAs single quantum wells 4.5 nm and 6.0 nm thick separated by a GaAs barrier 3.0 nm thick. The structure is supposed to be embedded in the *i*-section of a *p-i-n* diode. The InGaAs layers are compressively strained. Figure 12 (a) shows the potential profile, energy levels and wave functions for this structure calculated by using WAVY. The conduction and valence band were calculated independently. The energy gap is not at scale with the band edge discontinuities. There are only two bound levels in the conduction band. The wave function of the lower level is mainly in the larger well. Conversely, the wave function of the upper level is mainly in the narrow well. However, both wave functions have an appreciable amplitude in both wells, showing that the electron states are delocalized. In the valence band there are more than the two levels drawn, but I omitted the others for clarity.

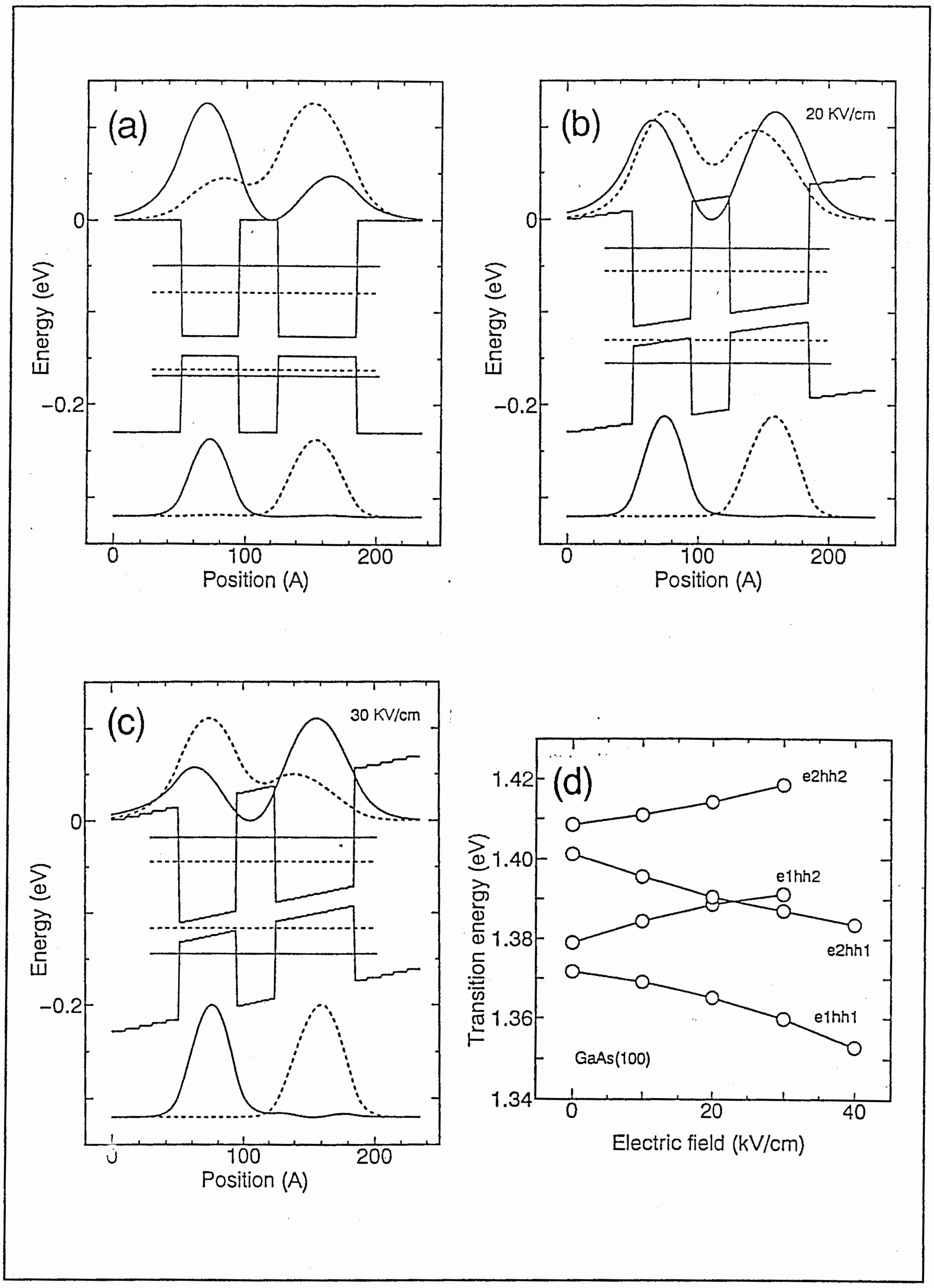

**Figure 12: Two asymmetrical InGaAs/GaAs quantum wells.**

The wave function of the level that lies deepest in the well is localized in the larger well and the other wave function is in the narrow well. This localization of the heavy holes reflects the larger effective mass of this valence sub-band. Figure 12 (b) and Figure 12 (c) show the effect of applying an external electric field on the structure. The electron wave functions interchange their positions in the wells. The wave function of the lowest lying level becomes stronger in the narrow well and the wave function of the higher level becomes stronger in the larger well. On the other hand, the hole's wave functions do not change their positions, but the energy difference between the hole's levels becomes larger. Figure 12 (d) shows the transition energy of the four possible combinations of levels. The lowest energy transition, labeled e1hh1, decreases its energy as applied bias increases. However, the wave function overlap of the e1hh1 transition also decreases and the optical matrix element becomes smaller. The e1hh2 transition that initially had a small wave function overlap, shifts toward higher energies and the wave function overlap becomes larger as the electron wave function is transferred to the narrow well. The net effect of this changes is a blueshift of the absorption spectra of this structure.

### 3.4.2 Example 2: Wannier-Stark effect in a five periods GaAs/AlAs superlattice

Another illustrative example of the capabilities of the WAVY program is the design of a GaAs/AlAs superlattice where Wannier-Stark ladders are observable. The structure consists of five 3.4 nm GaAs quantum wells separated by two monolayers (about 0.57 nm) AlAs barriers. The potential profiles and wave functions without and with applied bias are shown in Figure 13. The wave functions shown correspond to the first subband of electrons and holes. Each sub-band is composed by the five energy levels whose wave functions are shown. These sub-bands are s-like, i.e., it means that the wave functions have only one maximum at each well. The wave functions of the next higher sub-band (not shown in the figure) are p-like, i.e., they have two maximums at each well. When the external bias is not applied, the wave functions are symmetrical about the central well.

The characteristics of the larger "quantum well" formed by the five quantum well structure can be distinguished because the barriers are very thin. The lowest wave functions (e1 and hh1) have an envolvent with one lobe that resembles the fundamental level of the larger quantum well. The next wave functions (e2 and hh2) have an envolvent with two lobes resembling first excited state of the large "quantum well". The same happens for the higher levels.

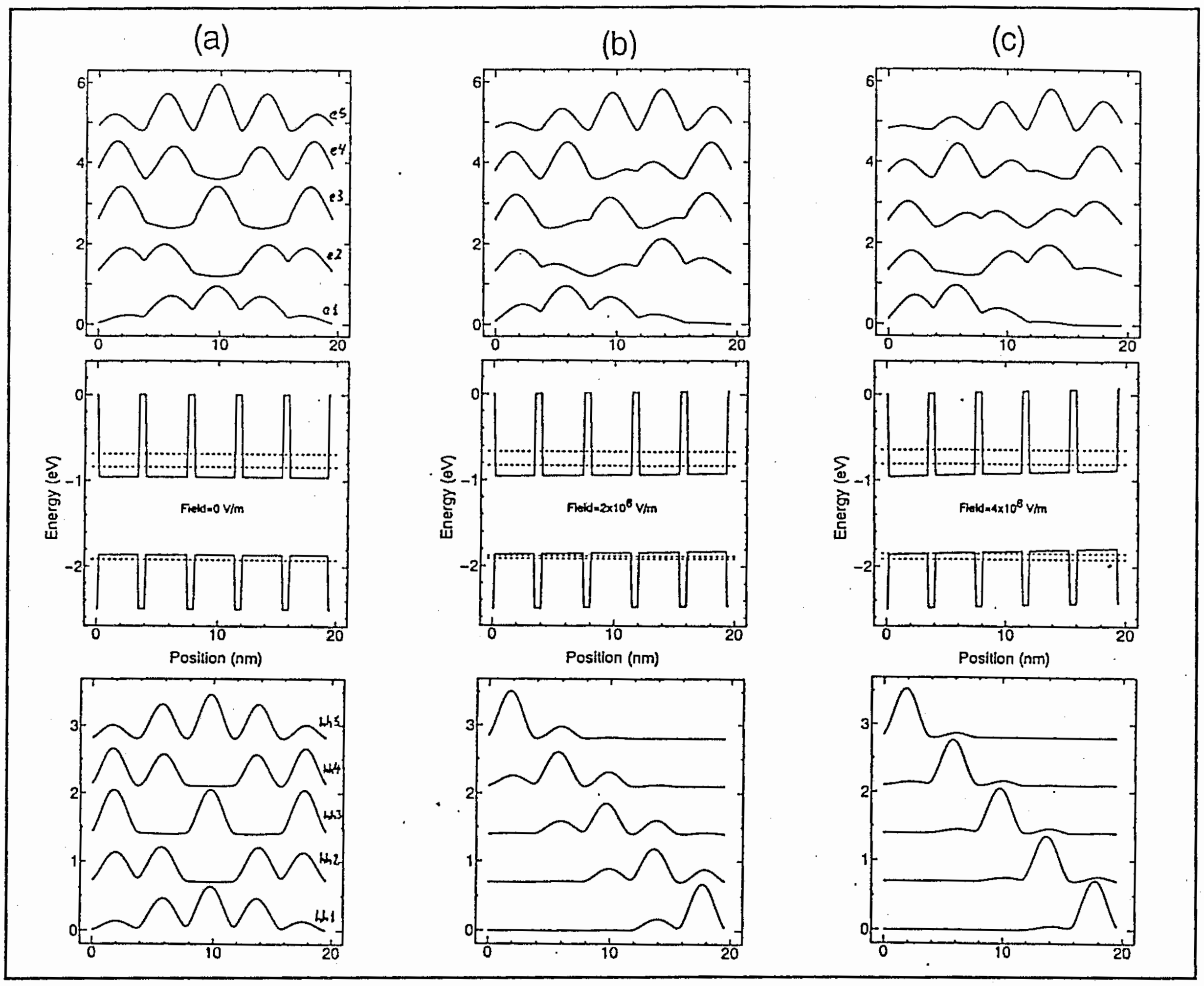

**Figure 13: Wannier-Stark localization in a five-periods superlattice.**

Anyway, the wave functions show that both electron and hole states are delocalized. Only the highest and lowest energy levels for the electrons and holes subbands are drawn on the potential profile for clarity. The energy difference between levels in the valence band is so small that it look like only one line in Figure 13 (a).

When an electric bias is applied (Figure 13 (b) and (c)), the states of the valence band become strongly localized in each well. On the other hand, the states of the conduction band show a weak localization. The strong localization of the holes is due to the larger effective mass in this heavy-hole subband. Figure 14 shows the transition energies among the five levels of the hole subband and the four lowest levels of the first electron subband. The energy level's separation increases with applied bias showing the Wannier-Stark ladder (i.e., the "miniband" formed by the five closely spaced levels disappears, and individual energy levels are observed).

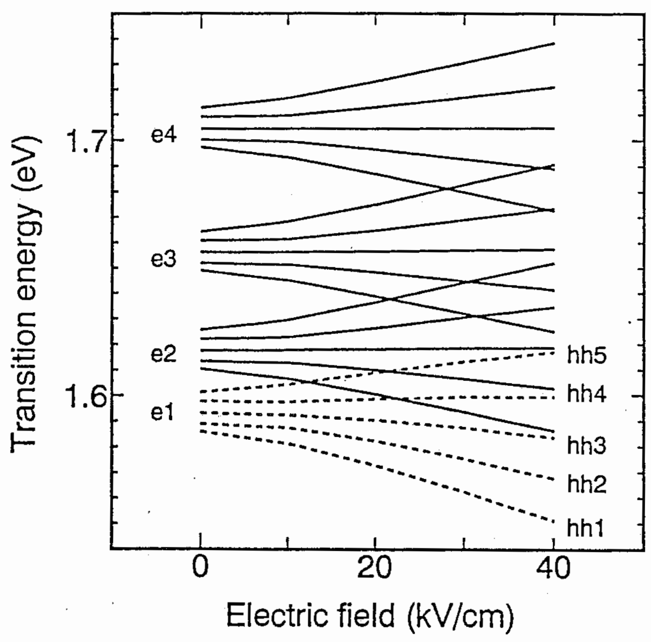

**Figure 14: Dependence of transition energies on applied electric field.**

There are not anticrossing behaviour between different energy levels when they become close. The anticrossing behavior can be observed between the levels of the first subband and the second subband of the same band (conduction or valence). Figure 15 shows the anticrossing behavior between states of the first and second electron subbands. The energy gap between the levels is about 20 meV. The electric field required to reach the anticrossing is three order of magnitude larger than the field in Figure 14

Figure 16 shows absorption spectra simulated by using the calculated transition energies and wave functions' overlap integrals. I chose arbitrarily a 3 meV width for the exciton peaks and a height relationship between the exciton peak and the absorption continuum equal to three for all the transitions. Decrease of the absorption is observed between 1.58 eV and 1.60 eV. The spectra resemble closely the usual experimental data.

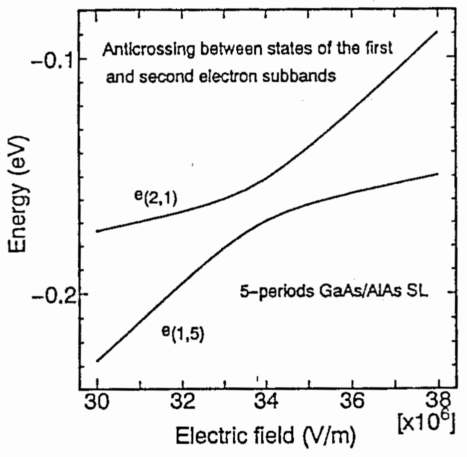

Figure 15: Anticrossing between states in the conduction band.

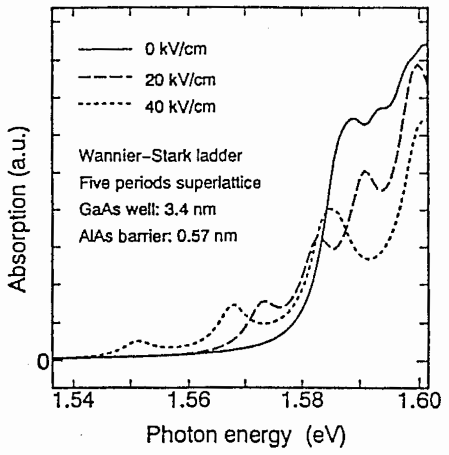

Figure 16: Simulated absorption spectra for the superlattice.

# 4. WAVYPOI: Selfconsistent solution of Schrödinger and Poisson equations

## *4.1 Theoretical background*

The previous program, WAVY, is useful when studying structures with a low carrier density. If the structure is under high optical excitation, there is a high density of photoexcited electrons-holes pairs that are redistributed accordingly to the potential profile. But electrons and holes also move apart of each other, accordingly to the forces in the conduction and valence band. This, in turn, generates strong electrostatic forces that distorts the potential profile. Therefore, it is necessary to calculate selfconsistently the photoexcited electron and hole distributions and the potential profile.

The quantized energy levels and their wave functions are determined from the self-consistent calculation using the effective mass Schrödinger equation within the envelope function framework and the Poisson equation.[14] The Schrödinger equation has been already introduced when discussing WAVY in the previous chapter. The only modification is that the potential must be replaced with a potential including the Hartree potential $V_H(z)$:

$$V(z) = V_0(z) + V_H(z) \qquad \textbf{eq. 4-1}$$

The Poisson equation takes the form (in SI units)

$$\frac{d}{dz}\left(\kappa(z)\frac{dV_H}{dz}\right) = -\frac{e}{\varepsilon_0}\left(n(z) - p(z)\right) \qquad \textbf{eq. 4-2}$$

where $\kappa(z)$ is the local dielectric constant, and $n(z)$ and $p(z)$ are the electron and hole densities respectively given by

$$\left.\begin{matrix} n(z) \\ p(z) \end{matrix}\right\} = \frac{k_B T}{\pi\hbar^2}\sum_i m_i^* \ln\left[1 + \left(\frac{E_F - E_i}{k_B T}\right)\right]\chi_i^2(z) \qquad \textbf{eq. 4-3}$$

where $E_F$ is the quasi-Fermi level in the conduction or valence bands, $E_i$ are the energy levels of the structure and

$$\frac{1}{m_i^*} = \int_{-\infty}^{\infty} \chi_i^2(z)\frac{1}{m_i^*(z)}dz.. \qquad \textbf{eq. 4-4}$$

I replaced this last equation with a constant value of $m_i^*$ because the error introduced by doing this is negligible and the calculation time is considerably reduced.

The program WAVIPOI calculates first the energy levels and wave functions in the structure without photocarriers. Next, it calculates the Hartree potential and recalculates the energy levels and wave function including it. The program continues iterating until self-consistency is reached. Finally, the intersubband transition energy (from a state *j* into *i*), $E_{ij} = E_i - E_j$ and its oscillator strength can be determined. The oscillator strength $f_{ij}$ is defined as

$$f_{ij} = \frac{2m_0 E_{ij}}{\hbar^2}\left|\langle i|z|j\rangle\right|^2 \qquad \textbf{eq. 4-5}$$

where $m_0$ is the rest mass of the electrons and $\left|\langle i|z|j\rangle\right|$ is the dipole matrix element of the transition.

## *4.2 User's manual*

### 4.2.1 Writing the input file

Write the input file following the model of Figure 17. The calculation to be performed is described from the second to the seventh line of the file.

Initial energy step: is the step used for scanning energy when

```
PARAMETERS OF THE STRUCTURE              electrons      holes
Initial energy step (meV)                   8.0           3.0
Number of energy levels you want (1-7)      3             2
Damping factor (>1)                         3
Energy bandgap of first layer (eV)          2.019
Thickness of each layer (nm)                1.0
Photocarrier density (car/m2)               2e16
1       .067     0.0             .9         0.0               0.0
20      .067     -.3             .9         -.2               0.0
10      .058     -.3613          .9         -.2854            150.0
20      .067     -.3             .9         -.2               0.0
0       .067     0.0             .9         0.0               0.0

        <---electrons---->   <----holes----->
steps   m*/m0     V           m*/m0     V                     Field
```

**Figure 17: Model for the input file of WAVYPOI.**

- looking for bound levels. It should be smaller than the smallest energy separation between successive energy levels, otherwise it could happen that energies levels are not found. A smaller step means also longer calculation time. Different step values can be chosen for electrons and holes. The step for holes usually must be smaller than for electrons because energy levels are more closely spaced.
- Number of energy levels you want: is useful for reducing the calculation time when you are interested only in the lower energy levels. Different number of energy levels can be chosen for electrons and holes.
- The damping factor is useful for avoiding oscillations in the calculated Hartree potential when high carrier densities are introduced. Try first with a small value (2 or 3) and if there is oscillations in the potential after 10 or 20 iterations, increase it by 1.

- The energy bandgap of the first layer, expressed in eV units, is used to calculate the interband transition energies.
- The thickness of each layer, expressed in nm units, is the spatial discretization step. Ten to twenty steps in each slab of the structure are usually enough to obtain precise results. Thinner layers means more steps and a longer calculation time.
- The photocarrier density is expressed in units of electron-hole pair per square meter (car/m$^2$).

These numerical values must be written starting after the 50*th* character of the line to be correctly read by the program.

The following six columns describe the multilayer structure. Data is arranged in the following way:

- Data for the material to the left (right) of the quantum structure is written in the first (last) line of the structure's table.
- The first column is the number of steps in each material slab. The thickness of the slab is given by the number of steps multiplied by the thickness of each layer written in the sixth line of the file.
- The second column is the electron's effective mass in free-electron-mass units.
- The third column is the conduction band potential in electron volts at each slab measured when there is not electric field in the structure. The first value must be zero.
- The fourth column is the hole's effective mass in free-electron-mass units.
- The fifth column is the valence band potential in electron volts at each slab measured when there is not electric field in the structure. The first value must be zero.
- The sixth column is the electric field in each layer in kV/cm units. Positive values mean that the potential increases toward the right side. The first and last value must be zero.

  The design of an structure is quite straightforward. You only need to draw in paper the structure without any electric field (include only the band edge alignment) to obtain the right potential values.

### 4.2.2 Running WAVIPOI

After writing and saving the input file, run WAVYPOI. It will display a menu asking for the names of the input and output files. After introducing the file names, the evolution of the calculation is displayed: the range of energy scan, the energy scan position, the number of energy levels found, the number of iterations and the remaining error in the electric field can be followed. The calculation is finished manually by pressing ESC and waiting until the current iteration ends. The criteria to decide when interrupt the program is to observe that the values of the energy scan do not change more after each iteration and that the error in the electric field if a value small enough to no influence appreciably the potential profile. The program will emit a short melody when the calculation is completed.

### 4.2.3 The output file

```
Input file name: pof.dat
Photocarrier density (1/m2): 2.0000e+16

Ener. levels   -0.3569   -0.3005   -0.2831   0.0000   0.0000   0.0000   0.000
Fermi level (eV): -0.2929

Wavefunctions
0.000e+00    2.883e-07    1.351e-03    1.138e-02    0.000e+00    0.000e+00
1.000e+00    7.035e-07    2.731e-03    2.159e-02    0.000e+00    0.000e+00
2.000e+00    1.501e-06    4.597e-03    3.343e-02    0.000e+00    0.000e+00
3.000e+00    3.009e-06    6.955e-03    4.549e-02    0.000e+00    0.000e+00
4.000e+00    5.850e-06    9.806e-03    5.632e-02    0.000e+00    0.000e+00
5.000e+00    1.119e-05    1.315e-02    6.462e-02    0.000e+00    0.000e+00
6.000e+00    2.122e-05    1.699e-02    6.934e-02    0.000e+00    0.000e+00
```

**Figure 18: Output file of WAVYPOI.**

An example of output file is shown in Figure 18. It includes the input file name and the photocarrier density. The next line shows the electron energy levels as in the output file of WAVY, and the electron quasi-Fermi level. The wave functions and potential profile are also presented as in the output file of WAVY. The analogous data for holes is below, and this is followed by the squared wave function overlap integral and transition energies between electron and hole energy levels.

## *4.3 Examples*

### 4.3.1 Example 1: A piezoelectric quantum well with high optical excitation

This structure consists of one InGaAs single quantum well with 10 nm of thickness grown on a (111)-oriented substrate. There are 20 nm thick spacer at each side of the well followed by AlGaAs barrier to keep the photoexcited carriers near the well (see Figure 19). The strain in the InGaAs grown on this polar substrate produces an strong electric field in the well that tilts the band edges. The electric field used in this example is an experimental value of 154 kV/cm.

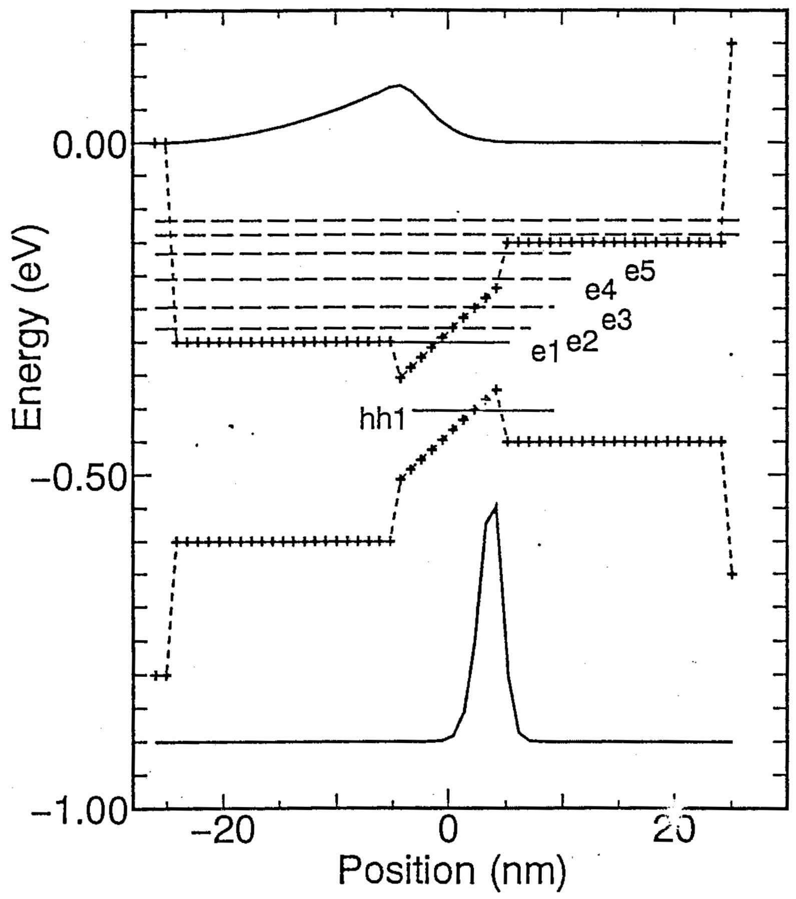

**Figure 19: Potential profile of the piezoelectric InGaAs/GaAs quantum well with a low electron-hole pair density.**

Figure 19 shows the quantum well with a low electron-hole pair density. The piezoelectric field is not screened and the first energy level in the conduction band is very near the GaAs band edge. The

AlGaAs barriers create other energy levels in the structure. There are levels confined between the GaAs spacer to the right and the AlGaAs barrier to the left in a 30-nm-thick potential well, and there are also levels confined between the AlGaAs barriers in a 50-nm-thick potential well. The wave function overlap of e1 and hh1 levels is very small, and most of the electron's wave function is spread throughout the GaAs spacer.

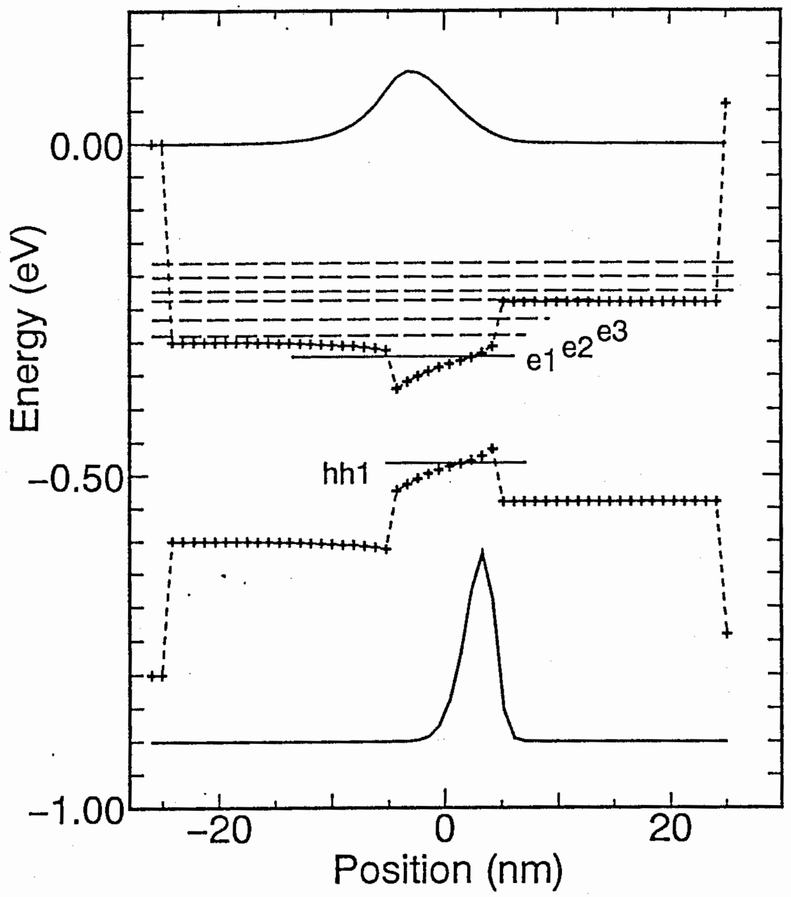

**Figure 20: Potential profile of the InGaAs/GaAs piezoelectric quantum well with a high electron-hole pair density.**

Figure 20 shows the quantum well with an electron-hole pair density of $1 \times 10^{12}$ cm$^{-2}$. The piezoelectric field is partially screened, and the redshift due to quantum-confined Stark-effect (QCSE) is reduced. There is also a large increase in the degree of overlap electron and hole wave functions. The electron quasi-Fermi level moved upward until reaching the second electron energy level. The hole quasi-Fermi level does not change greatly because of the large heavy-hole effective mass on this crystal orientation (0.9 $m_0$).

### 4.3.2 Example 2: Dependence of the transition energy and wave function overlap on electron-hole pair density

The strength of the piezoelectric field does not change with well thickness, but the potential profile and wave function overlap show a large change. This example is an extension of the previous one for quantum wells 2.5 nm, 5 nm and 10 nm thick, and for different electron-hole pair densities.

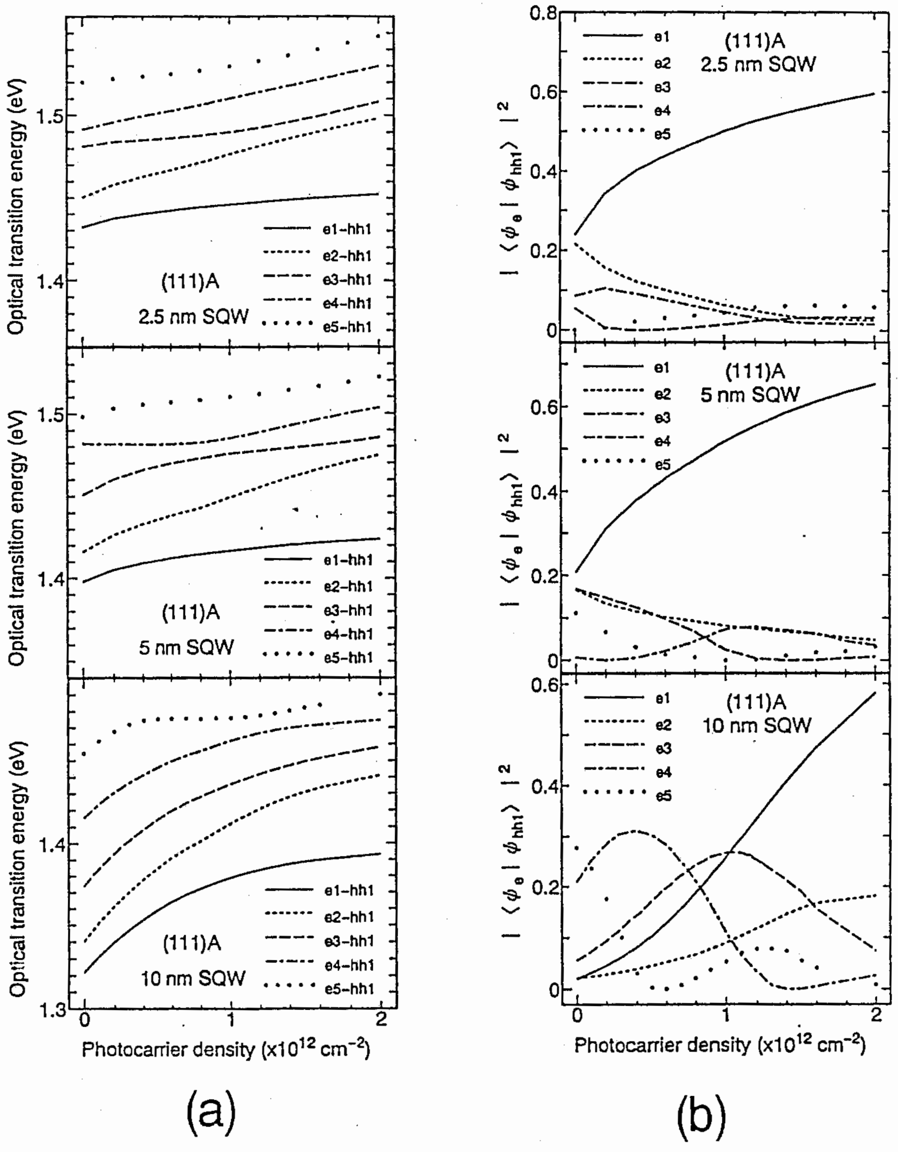

Figure 21: Dependence of the optical transition energy and degree of electron-hole wave function overlap on electron-hole pair density.

Figure 21 (a) and Figure 21 (b) respectively show the dependence of the calculated optical transition energy and degree of electron-hole wave function on electron-hole pair density. In addition to the fundamental transition e1-hh1, we include results for “forbidden” transitions between higher electron levels and the fundamental heavy-hole level hh1. These transitions are allowed in these quantum wells because the symmetry of the wave function is broken by the piezoelectric field. As electron-hole pair density in the wells increases, the field is screened, symmetry is recovered, and the oscillator strength (which is proportional to the degree of wave function overlap) of the forbidden transitions becomes small. The largest variation of transition energy and oscillator strength of the fundamental transition as electron-hole pair density increases is obtained for the thickest QW, i.e., the 10 nm quantum well.

## References

[1] C. Weisbuch and B. Vinter, *"Quantum Semiconductor Structures"*, Academic Press, Boston, (1991) ch. II.
[2],K. Iga and F. Koyama, *"Surface Emitting Semiconductor Lasers and Arrays"*, edited by G. A. Evans and J. M. Hammer, Academic Press, San Diego, (1993) p. 71.
[3] J. Trezza, M. Larson, S. Lord and J. Harris, J. Appl. Phys. **74** (1993) 6495.
[4] W. Chow, S. Koch and M. Sargent, *"Semiconductor-Laser Physics"*, Springer-Verlag, Berlin, (1994).
[5] M. Takahashi, P. Vaccaro, K. Fujita and T. Watanabe, Appl. Phys. Lett. **66** (1995) 93.
[6] P. Vaccaro, K. Tominaga, H. Hosoda, K. Fujita and T. Watanabe, Jpn. J. Appl. Phys. **34** (1995) 1362.
[7] P. Vaccaro, M. Takahashi, K. Fujita and T. Watanabe, J. Appl. Phys. **76** (1994) 8037.
[8] H. A. MacLeod, Applied Optics and Optical Engineering, vol. X, (1985) 12.
[9] E. D. Palik, *"Handbook of Optical Constants of Solids"*, edited by E. D. Palik, Academic Press, San Diego, (1985) p. 435.
[10] R. S. Geels, S. W. Corzine, J. W. Scott and L. A. Coldren, Photonics Technol. Lett., **2** (1990) 234.
[11] G. Bastard, *"Wave Mechanics Applied to Semiconductor Heterostructures"*, Halsted Press, New York, (1988) p. 63.
[12] D. Ahn and S. L. Chuang, Phys. Rev. **B33** (1986)8758.
[13] B. Jonsson and S. T. Eng, J. Quantum Electronics, **26** (1990) 2025.
[14] W. Q. Chen and T. G. Andersson, J. Appl. Phys. **73** (1993) 4484.

# Appendix: Source programs

```
/***********************************************************************
 *                         MIRROR                                      *
 *                           by Pablo O. Vaccaro, ATR-ORCL, 1993       *
 *                                                                     *
 * This program calculates the reflectivity and transmitivity of      *
 * multilayer structures made of AlAs and GaAs alloys. The dependence *
 * on polarization, angle of incidence and wavelength of incident     *
 * light are also calculated. Random and systematic variations of     *
 * the layer thickness is optionally included.                        *
 *                                                                     *
 *  Written in TurboC++ version 1.01 (Borland Intnl., 1990)           *
 ***********************************************************************/

/***********************************************************************
 *       Include files                                                 *
 ***********************************************************************/

#include<math.h>
#include<stdio.h>
#include <conio.h>
#include<stdlib.h>
#include<complex.h>
#include<dos.h>

/***********************************************************************
 *       Define                                                        *
 ***********************************************************************/

#define MAXLAYER       900     /* maximum number of layers in the structure */
#define PUNTOS         500     /* maximun number of points in the energy */
#define WAVE           1.2398 /* conversion between energy and wavelength */
#define PI             3.141592654
#define DIVER          0.000001  /* Limit of resonance value near bandgap */

/***********************************************************************
 *       Global Variables                                              *
 ***********************************************************************/

int Layer,Pun;
double Mir[MAXLAYER][2],E,Reflec[PUNTOS][5],Eini,Efin,DelE,Tp,Ts,Rp,theta;
double Abs[32][2];
char a,nom[80],nomsal[80];
```

```
/************************************************************************
 *       Prototypes                                                    *
 ************************************************************************/

int LeoMir(void);
void GuardoReflec(void);
double CalcReflec(void);
double Refin(double n);
double Absor(double n);
void Energia(void);
void Angulo(void);
void Periodo(void);
void MenuEntrada(void);
void MenuSalida(void);

/************************************************************************
 *       Main                                                          *
 ************************************************************************/

main()
{
int loop;
MenuEntrada();
do
        {
        loop=1;
        a=getch();
        switch(a)
                {
                case '1':
                        clrscr();
                        puts("     Calculating dependence on energy. Wait...");
                        Energia();
                        break;
                case '2':
                        clrscr();
                        puts("     Calculating dependence on incidence angle. Wa
it...");
                        Angulo();
                        break;
                case '3':
                        clrscr();
                        puts("     Calculating dependence on Nb. of periods. Wai
t...");
                        Periodo();
                        break;
                default: loop=0;
                }
        } while(loop==0);
clrscr();
MenuSalida();
return(0);
}

/***********************************************************************
 *     Energia                                                         *
 * Calculates the depedence on energy                                  *
 ***********************************************************************/

void Energia(void)
{
int i;
Layer=LeoMir();
DelE=(Efin-Eini)/Pun;
for( i=0 ; i<Pun ; i++)
        {
        E=Eini+i*DelE;
        Reflec[i][0]=E;
        Reflec[i][1]=CalcReflec();
        Reflec[i][2]=Ts;
        Reflec[i][3]=Rp;
        Reflec[i][4]=Tp;
        }
GuardoReflec();
}

/***********************************************************************
 *      Angulo                                                         *
 * Calculates the dependence on incidence angle                        *
 ***********************************************************************/

void Angulo(void)
{
int i;
Layer=LeoMir();
DelE=theta/Pun;
for( i=0 ; i<Pun ; i++)
        {
        E=Eini;
        theta=i*DelE;
        Reflec[i][0]=theta;
        Reflec[i][1]=CalcReflec();
        Reflec[i][2]=Ts;
        Reflec[i][3]=Rp;
        Reflec[i][4]=Tp;
        }
GuardoReflec();
}

/**********************************************************************
 *    Periodo                                                         *
 * Calculates the dependence on number of periods                     *
 **********************************************************************/

void Periodo(void)
{
int i;
double Dum;
Pun=LeoMir();
for( Layer=4 ; Layer<=Pun ; Layer=Layer+2)
        {
        E=Eini;
        Dum=Mir[Layer-1][0];
        Mir[Layer-1][0]=Mir[Pun-1][0];
        Reflec[Layer/2-2][0]=(double)Layer/2-1;
        Reflec[Layer/2-2][1]=CalcReflec();
        Reflec[Layer/2-2][2]=Ts;
        Reflec[Layer/2-2][3]=Rp;
        Reflec[Layer/2-2][4]=Tp;
        Mir[Layer-1][0]=Dum;
        }
Pun=Pun/2-1;
GuardoReflec();
}

/**********************************************************************
 *       LeoMir                                                       *
 * Read data from input file, make an array with one layer per        *
 * element, includes random or systematic thickness variations.
 **********************************************************************/

int LeoMir(void)
{
int t,q=0,m,i=0;
double Mirin[50][3],sist,alea;
```

```
FILE *in;
  in=fopen("absor.dat","r");
  while(fscanf(in,"%lf %lf¥n",&Abs[i][0],&Abs[i][1]),i<32)i++;
  fclose(in);
  i=0;
  in=fopen(nom,"r");
  fscanf(in,"%*50c%le¥n%*50c%le¥n%*50c%i¥n%*50c%le¥n%*50c%le¥n%*50c%le¥n",
                &Eini,&Efin,&Pun,&theta,&sist,&alea);
  while(fscanf(in,"%lf %lf %lf¥n",&Mirin[i][0],&Mirin[i][1],&Mirin[i][2]),Mirin
[i][0]!=9.9)i++;
  fclose(in);
  for(i=0; Mirin[i][0]!=9.9; i++)
        {
        for(t=0; t<(int)Mirin[i][0]; t++)
                {
                m=0;
                do
                        {
                        Mir[q][0]=Mirin[i+m][1];
                        Mir[q][1]=Mirin[i+m][2];
                        m++;
                        q++;
                        }
                while(Mirin[i+m][0]==0);
                };
        }
  randomize();
  for(i=0; i<q-1; i++)
        {
        Mir[i][1]=Mir[i][1]*(1+sist*i/(100*(q-2)));
        Mir[i][1]=Mir[i][1]*(1+alea*(random(201)-100)/20000);
        }
  return(q);     /* return the number of layers in the structure */
}

/**********************************************************************
 *       CalcReflec                                                   *
 * Calculates the reflectivity and transmisivity of the multilayer    *
 * using the transfer-matrix method                                   *
 **********************************************************************/

double CalcReflec()
{
int j;
double n,ns,k,ks,d,Rs,ni;
```

```
complex ii(0.0,1.0),Xs(1.0,0.0),Xp(1.0,0.0),Bs,Cs,Bp,Cp,a,yp,ys,Yp,Ys,ysubs,ysub
p;
if(Mir[0][0]==9.9) {ns=1; ks=0;}
else    {
        ns=Refin(Mir[0][0]);
        ks=Absor(Mir[0][0]);
        }
if(Mir[Layer-1][0]==9.9) ni=1;    /* incidence media's reffraction index */
else ni=Refin(Mir[Layer-1][0]);
complex Y(ns,-ks);      /* substrate's optical admittance */
Ys=sqrt(ns*ns-ks*ks-ni*ni*(sin(theta*PI/180))*(sin(theta*PI/180))-2*ii*ns*ks);
Yp=Y*Y/Ys;
ysubs=Ys;
ysubp=Yp;
for( j=1 ; j<Layer-1 ; j++)
        {
        n=Refin(Mir[j][0]);
        d=Mir[j][1];
        k=Absor(Mir[j][0]);
        complex y(n,-k);
        ys=sqrt(n*n-k*k-ni*ni*(sin(theta*PI/180))*(sin(theta*PI/180))-2*ii*n*k);
        yp=y*y/ys;
        a=2*PI*d*(E/WAVE)*sqrt(n*n-k*k-ni*ni*(sin(theta*PI/180))*
                (sin(theta*PI/180))-2*ii*n*k);  /* delta de la teoria */
        Bs=Xs*cos(a)+Ys*ii*sin(a)/ys;
        Cs=Xs*ii*ys*sin(a)+Ys*cos(a);
        Xs=Bs;
        Ys=Cs;
        Bp=Xp*cos(a)+Yp*ii*sin(a)/yp;
        Cp=Xp*ii*yp*sin(a)+Yp*cos(a);
        Xp=Bp;
        Yp=Cp;
        }
Bs=ni*Bs*cos(theta*PI/180);
Rs=real((Bs-Cs)*conj(Bs-Cs)/((Bs+Cs)*conj(Bs+Cs)));
Ts=real(4*ni*cos(theta*PI/180)*real(ysubs)/((Bs+Cs)*conj(Bs+Cs)));
Bp=ni*Bp/cos(theta*PI/180);
Rp=real((Bp-Cp)*conj(Bp-Cp)/((Bp+Cp)*conj(Bp+Cp)));
Tp=real(4*ni*ysubp/(((Bp+Cp)*conj(Bp+Cp))*cos(theta*PI/180)));
return(Rs);
}
```

```
/**********************************************************************
 *       Refin                                                        *
 * Calculates the AlGaAs reffraction index according to               *
 * Sol. State Commun. 15, -59-, (1974)                                *
 **********************************************************************/

double Refin(double x)
{
double E0,Ed,Ep,diver;
double Xe,Ef,eta,M1,M3;

E0=3.65+0.87*x+0.179*x*x;
Ed=36.1-2.45*x;
Ep=1.424+1.266*x+0.26*x*x;

Ef=sqrt(2*E0*E0-Ep*Ep);
eta=PI*Ed/(2*E0*E0*E0*(E0*E0-Ep*Ep));

M1=eta*(Ef*Ef*Ef*Ef-Ep*Ep*Ep*Ep)/(2*PI);
M3=eta*(Ef*Ef-Ep*Ep)/(PI);

diver=sqrt((Ep*Ep-E*E)*(Ep*Ep-E*E));
if(diver<DIVER) diver=DIVER;
Xe=M1+M3*E*E+eta*E*E*E*E*log((Ef*Ef-E*E)/diver)/PI;
Xe=sqrt(Xe+1);
return(Xe);
}


/**********************************************************************
 *       GuardoReflec                                                 *
 * Save results in the output file                                    *
 **********************************************************************/

void GuardoReflec(void)
{
FILE *out;
int i=0;
  out=fopen(nomsal,"w");
  fprintf(out,"             s-polarization             p-polarization¥n");
  fprintf(out,"         reflec.       transmis.     reflec.       transmis.¥n");
  while(fprintf(out,"%.4lf %le %le %le %le¥n",Reflec[i][0],Reflec[i][1],Reflec[i]
[2],Reflec[i][3],Reflec[i][4]),i++<=(Pun-2));
  fclose(out);
}
```

```
/*********************************************************************
 *       Absor                                                       *
 * Calculates the absorption dependence on energy and composition    *
 * using data in ABSOR.DAT                                           *
 *********************************************************************/

double Absor(double x)
{
double k;
int i=0;
while((Abs[i][0]+1.266*x+0.26*x*x)<E)i++; /* shift de la abs con la compos. */
if(i==0)return(0.0);
k=Abs[i-1][1]+(Abs[i][1]-Abs[i-1][1])*(E-Abs[i-1][0])/(Abs[i][0]-Abs[i-1][0]);
return(k);
}

/*********************************************************************
 *       MenuEntrada                                                 *
 * Input menu                                                        *
 *********************************************************************/

void MenuEntrada(void)
{
clrscr();
window(5,2,80,23);
textattr(LIGHTGREEN);
cprintf("MIRROR calculates the reflectivity of AlGaAs multilayers.¥n");
gotoxy(1,22);
cprintf("Read the manual for preparation of input file.");
gotoxy(37,3);
textattr(YELLOW);
cprintf("by Pablo O. Vaccaro, ATR-ORCL (1993)");
window(10,10,45,13);
textattr(LIGHTCYAN);
cprintf("         Input file is: ");
gotoxy(1,3);
cprintf("        Output file is: ");
gotoxy(25,1);
gets(nom);
gotoxy(25,3);
gets(nomsal);
window(1,15,80,17);
gotoxy(1,1);
textattr(YELLOW);
cprintf("Choose the x-axis: 1. Energy  2. Incidence angle  3. Nb. of periods ");
}

/*************************************************************************
 *       MenuSalida                                                      *
 * Output menu                                                           *
 *************************************************************************/

void MenuSalida(void)
{
sound(1000);
delay(300);
sound(3000);
delay(300);
sound(2000);
delay(300);
nosound();
clrscr();
window(1,2,70,3);
textattr(LIGHTGREEN);
cprintf("MIRROR completed the calculation.");
}
```

```
/***********************************************************************
 *                              WAVY                                   *
 *                                by Pablo O. Vaccaro, ATR-ORCL, 1994  *
 *                                                                     *
 *  Solves the Schrodinger equation for arbitrary potential profiles. *
 *  Calculates:                                                        *
 *  (a) the bound energy levels and wave functions or                  *
 *  (b) the bound energy levels dependence on applied bias             *
 *  by using the transfer-matrix method.                               *
 *                                                                     *
 *  Written in TurboC++ version 1.01 (Borland Intnl., 1990)            *
 ***********************************************************************/

/***********************************************************************
 *       Include files                                                 *
 ***********************************************************************/

#include<stdio.h>
#include<math.h>
#include<complex.h>
#include<conio.h>
#include<dos.h>

/***********************************************************************
 *       Define                                                        *
 ***********************************************************************/

#define ELE                 1.6021917e-19     /* electron charge */
#define HB                  1.0545919e-34     /* reduced Planck constant */
#define M                   9.109558e-31      /* electron mass */
#define PI                  3.141592654
#define LAY                 50                /* max. lines in input file */
#define SUBLAY              100               /* max. layers in structure */
#define PASO                .00101246764      /* initial energy step (meV) */
#define PRECISION           0.000001          /* target value for c22=0 */
#define PUNTOS              200               /* numero de puntos en wavefunction */
#define ESC             0x1b
```

```
/************************************************************************
 *       Global Variables                                               *
 ************************************************************************/

double V[LAY],Vorig[LAY],L[LAY],m[LAY],step,field[LAY],fiorig[LAY],Vij[SUBLAY];
double Ener[10],Vmin,wave[PUNTOS][9],iniene,bandgap,emin,emax,apfield,
                minfi,maxfi,stefi,En,limite;
int frog,numstep[LAY],t,s,u,cuantos,maxlevel,nufi,c;
complex Aj[SUBLAY][8],Bj[SUBLAY][8],Rj[SUBLAY][8];

/************************************************************************
 *       Prototypes                                                     *
 ************************************************************************/

void LeeDatos(void);
void Calculo(void);
void Waveform(void);
void PresentaDatos(void);
void PresentaSegundo(void);
void DisplayCondition(int j);
void MenuSalida(void);
void MenuEntrada(void);
void MenuRun(void);

/************************************************************************
 *       Main                                                           *
 ************************************************************************/

main()
{
nufi=0;
c='';
  LeeDatos();
  clrscr();
/*************** (a) energy levels and wave functions *****************/
  if(minfi==maxfi)
        {
        apfield=minfi;
        Calculo();
        if(c!=ESC)
                {
                Waveform();
                PresentaDatos();
                }
        }
```

```
/*************** (b) energy levels dependence on bias *****************/
  else
        {
        for(apfield=minfi;apfield<=maxfi;apfield+=stefi)
                {
                Calculo();
                if(c==ESC) break;
                wave[nufi][0]=apfield;
                for(s=0;s<cuantos;s++)
                        wave[nufi][s+1]=Ener[s];
                nufi++;
                }
        if(c!=ESC) PresentaSegundo();
        }
window(1,1,80,25);
clrscr();
_setcursortype(_SOLIDCURSOR);
}

/**********************************************************************
 *   LeeDatos                                                         *
 * Read data from input file. Prepare array with all the layers in the*
 * structure. Total number of layers is the global variable "t".      *
 **********************************************************************/

void LeeDatos(void)
{
FILE *in;
        t=0;
char nom[80];
MenuEntrada();
  gets(nom);
  window(1,1,80,25);
  clrscr();
  in=fopen(nom,"r");
  fscanf(in,"%*40c%le¥n%*40c%d¥n%*40c%le¥n%*40c%le¥n%*40c%le¥n%*40c%le¥n%*40c%le
¥n%*40c%le¥n",
        &iniene,&maxlevel,&bandgap,&emin,&emax,&minfi,&maxfi,&stefi);
  while(fscanf(in,"%le %le %le %le %d¥n",
        &L[t],&m[t],&Vorig[t],&fiorig[t],&numstep[t]),L[t]!=0.0)t++;
  fclose(in);
  for(s=0;s<=t;s++)
        {
        m[s]=m[s]*M;
        Vorig[s]=Vorig[s]*ELE;
```

```
          }
emin*=ELE;
emax*=ELE;

}

/***********************************************************************
 *       Calculo                                                       *
 * Solves the Schrodinger equation in the structure by using the       *
 * transfer-matrix method.                                             *
 ***********************************************************************/

void Calculo(void)
{
double Lz,Zero,Antes=0.0,Vf,fitep,mara,Vj,pared;
complex gjn,gj,alfa,eplus,eminus,ii(0.0,1.0),
          uno(1.0,0.0),a11,a12,a21,a22,b11,b12,b21,b22,c11,c12,c21,c22;
int j=0,h,v;
Vmin=0.0;
limite=0.0;
step=PASO*ELE*iniene;
for(s=1;s<=(t-1);s++)field[s]=(fiorig[s]+apfield)*ELE*100000;
for(s=0;s<=t;s++)V[s]=Vorig[s];
for(s=0;s<=t;s++)
          {
          for(u=t;u>s;u--)
                    V[u]=V[u]+field[s]*L[s]*1e-9;
          if(Vmin>V[s])Vmin=V[s];
          if(Vmin>(V[s]+field[s]*L[s]*1e-9))Vmin=V[s]+field[s]*L[s]*1e-9;
          }
if(Vmin<emin)Vmin=emin;
pared=step;
En=Vmin;
if(V[t]<0.0)limite=V[t];
if(limite>emax)limite=emax;
MenuRun();
do
   {
   DisplayCondition(j);
   if(kbhit()!=0) c=getch();
   if(c==ESC) break;
   En=En+pared;
   Lz=L[1]*1e-9;
   Vf=field[1]*Lz;
   fitep=Vf/numstep[1];
```

```
Vj=V[1]+fitep/2;
gjn=-sqrt(2*m[0]*(-En)/(HB*HB));
if(En>Vj)
   gj=ii*sqrt(2*m[1]*(En-Vj)/(HB*HB));
else
   gj=-sqrt(2*m[1]*(-En+Vj)/(HB*HB));
alfa=gjn*m[1]/(gj*m[0]);
Aj[0][j]=0; Bj[0][j]=1; Rj[0][j]=gjn;
v=1;
a11=(uno+alfa)/2;
a12=(uno-alfa)/2;
a21=(uno-alfa)/2;
a22=(uno+alfa)/2;
Aj[v][j]=a12; Bj[v][j]=a22; Rj[v][j]=gj;
for(s=1;s<t;s++)
   {
   Lz=L[s]*1e-9;
   Vf=field[s]*Lz;
   fitep=Vf/numstep[s];
   Vj=V[s]+fitep/2;
   for(h=1;h<=numstep[s];h++)
      {
      gjn=gj;
      Vij[v]=Vj/ELE;
      Vj=Vj+fitep;
      if(h==numstep[s])
         {
         mara=m[s+1]/m[s];
         Vj=V[s+1]+field[s+1]*L[s+1]*1e-9/(2*numstep[s+1]);
         }
      else mara=1;
      if(En>Vj)
         gj=ii*sqrt(2*m[s]*mara*(En-Vj)/(HB*HB));
      else
         gj=-sqrt(2*m[s]*mara*(-En+Vj)/(HB*HB));
      alfa=gjn*mara/gj;
      v++;
      eplus=exp(gjn*Lz/numstep[s]);
      eminus=exp(-gjn*Lz/numstep[s]);
      b11=eplus*(uno+alfa)/2;
      b12=eminus*(uno-alfa)/2;
      b21=eplus*(uno-alfa)/2;
      b22=eminus*(uno+alfa)/2;
      c11=b11*a11+b12*a21;
      c12=b11*a12+b12*a22;
```

```
            c21=b21*a11+b22*a21;
            c22=b21*a12+b22*a22;
            Aj[v][j]=c12; Bj[v][j]=c22; Rj[v][j]=gj;
            a11=c11; a12=c12; a21=c21; a22=c22;
            }
        }
    Zero=real(c22);
    if((Zero*Antes)<0.0)    /*when an energy level is found, the energy*/
        pared=-pared/10;   /*step is reduced to increase precision*/
    if(Zero==Antes)         /*if the limit given by the precision of "double"*/
        {                  /*is reached, the energy value is stored*/
        pared=step;
        Zero=0.0;
        Ener[j]=-En/ELE;
        j++;
        }
    if(norm(c22)<PRECISION)  /*if the zero of c22 is reached, */
        {                   /* the energy value is stored */
        pared=step;
        Zero=0.0;
        Ener[j]=En/ELE;
        j++;
        }
    Antes=Zero;
    }while((En<(-step+limite))&&(j!=maxlevel));
cuantos=j;
}

/*************************************************************************
 *        PresentaDatos                                                  *
 * Write to the output file the energy levels and wave functions        *
 *************************************************************************/

void PresentaDatos(void)
{
int i=0;
FILE *out;
char nom[80];
MenuSalida();
  gets(nom);
  out=fopen(nom,"w");
  fprintf(out,"levels:  %.5lf     %.5lf     %.5lf     %.5lf     %.5lf     %.5lf     %.
5lf     %.5lf     %.5lf¥n",
        Ener[0],Ener[1],Ener[2],Ener[3],Ener[4],Ener[5],
        Ener[6],Ener[7],Ener[8]);
  fprintf(out,"Wavefunctions¥n");
  fprintf(out,"X(nm)        e1          e2          e3          e4          e5
       e6          e7      potential¥n");
  while(fprintf(out,"%.3lf   %.3le   %.3le   %.3le   %.3le   %.3le   %.3le   %.3
le   %.3le¥n",
        wave[i][0],wave[i][1],wave[i][2],wave[i][3],wave[i][4],wave[i][5],
        wave[i][6],wave[i][7],wave[i][8]),i++<(PUNTOS-1));
  fclose(out);
  clrscr();
}

/*************************************************************************
 *       PresentaSegundo                                                 *
 * Write to the output file the energy levels dependence on bias         *
 *************************************************************************/

void PresentaSegundo(void)
{
int i=0;
FILE *out;
char nom[80];
MenuSalida();
  gets(nom);
  out=fopen(nom,"w");
  fprintf(out,"E (kV/cm)   e1        e2        e3        e4        e5        e6
 e7 ¥n");
  while(fprintf(out,"%.2lf   %.5lf %.5lf %.5lf %.5lf %.5lf %.5lf %.5lf %.5lf¥n",
        wave[i][0],wave[i][1],wave[i][2],wave[i][3],wave[i][4],wave[i][5],
        wave[i][6],wave[i][7],wave[i][8]),i++<(nufi-1));
  fclose(out);
  clrscr();
}

/*************************************************************************
 *       Waveform                                                        *
 * Calculates the wave functions                                         *
 *************************************************************************/

void Waveform(void)
{
double xj,largo=0.0,paso,x,wamax[8]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
double area[8]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
int q=1,v,h,j;
complex aj,bj,rj;
for(s=1;s<=t;s++)largo=largo+L[s];
```

```
s=1; h=0;
xj=0.0;
paso=largo/PUNTOS;
                for(v=0;v<PUNTOS;v++)
                        {
                        x=v*paso;
                        while(x>(xj+L[s]/numstep[s]))
                                {
                                q++;
                                h++;
                                xj+=L[s]/numstep[s];
                                if(h==numstep[s])
                                        {
                                        s++;
                                        h=0;
                                        }
                                }
                        wave[v][0]=x;
                        wave[v][8]=Vij[q];
                        for(j=1;j<=cuantos;j++)
                                {
                                aj=Aj[q][j-1];
                                bj=Bj[q][j-1];
                                rj=Rj[q][j-1]*(x-xj)*1e-9;
                                wave[v][j]=norm(aj*exp(rj)+bj*exp(-rj));
                                area[j]=area[j]+wave[v][j];
                                if(wamax[j]<wave[v][j])wamax[j]=wave[v][j];
                                }
                        }
                for(v=0;v<PUNTOS;v++)
                        {
                        for(j=1;j<=cuantos;j++)
                                {
                                wave[v][j]=wave[v][j]*(-Vmin)*area[1]/(area[j]*w
amax[1]*ELE);

                                }
                        }
}

/************************************************************************
 *      DisplayCondition                                                *
 * Display the evolution of the calculation                             *
 ************************************************************************/

void DisplayCondition(int j)
```

```
{
double porcen;
porcen=100*(En-limite)/(Vmin-limite);
gotoxy(35,8);
cprintf("%4.1f",apfield);
gotoxy(27,10);
cprintf("%3.0f",porcen);
gotoxy(41,12);
cprintf("%1d",j);
}

/**********************************************************************
 *       MenuSalida                                                   *
 * Sound a bell and end the program                                   *
 **********************************************************************/

void MenuSalida(void)
{
sound(1000);
delay(300);
sound(3000);
delay(300);
sound(2000);
delay(300);
nosound();
clrscr();
window(1,2,70,3);
textattr(LIGHTGREEN);
cprintf("WAVY1.0 completed the calculation.");
window(10,10,40,11);
textattr(LIGHTCYAN);
_setcursortype(_SOLIDCURSOR);
cprintf("Save output file as: ");
}

/**********************************************************************
 *       MenuEntrada                                                  *
 * Menu for input of data and results file names                      *
 **********************************************************************/

void MenuEntrada(void)
{
clrscr();
window(5,2,80,23);
textattr(LIGHTGREEN);
cprintf("WAVY1.0 calculates energy levels of quantum structures.¥n");
gotoxy(1,22);
cprintf("Read the manual for preparation of input file.");
gotoxy(37,3);
textattr(YELLOW);
cprintf("by Pablo Vaccaro, ATR-ORCL (1994)");
window(10,10,40,11);
textattr(LIGHTCYAN);
cprintf("Input file is: ");
}

/**********************************************************************
 *       MenuRun                                                      *
 * Menu displayed during the calculation                              *
 **********************************************************************/

void MenuRun(void)
{
window(1,1,80,24);
clrscr();
textattr(YELLOW);
gotoxy(2,2);
cprintf("Now WAVY1.0 is calculating. Press ESC to end.          ");
textattr(LIGHTGREEN);
gotoxy(10,8);
cprintf("Applied electric field:       kV/cm");
gotoxy(10,10);
cprintf("Energy remaining:    percent");
gotoxy(10,12);
cprintf("Number of energy levels found: ");
textattr(LIGHTMAGENTA);
gotoxy(5,5);
cprintf("Energy scan from %1.4f eV to %1.4f eV",Vmin/ELE;limite/ELE);
textattr(YELLOW);
_setcursortype(_NOCURSOR);
}
```

```
/*********************************************************************
 *                          WAVYPOI                                  *
 *                              by Pablo O. Vaccaro, ATR-ORCL, 1995  *
 *                                                                   *
 *  Solves the Schrodinger and Poisson equations autoconsistently for *
 *  undoped heterostructures under optical excitation. Calculates the *
 *  bound energy levels, wavefunctions and quasi-Fermi levels for     *
 *  conduction and valence band. Calculates the wave function overlap *
 *  and transition energy between electron and hole bound levels.     *
 *                                                                   *
 *  Written in TurboC++ version 1.01 (Borland Intnl., 1990)          *
 *********************************************************************/

/*********************************************************************
 *      Include files                                                *
 *********************************************************************/

#include<stdio.h>
#include<math.h>
#include<complex.h>
#include<conio.h>
#include<dos.h>
#include<string.h>

/*********************************************************************
 *      Define                                                       *
 *********************************************************************/

#define RO              4.1768e18       /* states/eV*m2  m0/pi*h2 */
#define ELE             1.6021917e-19   /* electron charge */
#define HB              1.0545919e-34   /* reduced Planck constant */
#define M               9.109558e-31    /* electron mass */
#define PI              3.141592654
#define KAP             12.91           /* GaAs dielectric constant */
#define EZERO           8.854e-12       /* (sec2/m2, permitiv. vacuum, SI */
#define LAY             100             /* max. lines in input file */
#define PASO            .00101246764    /* initial energy step (meV) */
#define PRECISION       0.0001          /* target value for c22=0 */
#define ESC             0x1b
```

```
/**********************************************************************
 *       Global Variables                                             *
 **********************************************************************/

float wave[LAY][9][2];
double V[LAY],field[LAY],Vij[LAY],Vorig[LAY][2],m[LAY][2],fiorig[LAY];
double L,iniene[2],step,bandgap,Tran[9][9];
double Ener[10][2],Vmin,En,limite,screen[LAY],photoc,fermiele[2],diferen;
double scrbef,amorti,overint[9][9],scrog[LAY];
int frog,t,u,cuantos,maxlevel[2],c,itera;
complex Aj[LAY][8],Bj[LAY][8],Rj[LAY][8];
char nom[80],nomsal[80];
FILE *out;

/**********************************************************************
 *       Prototypes                                                   *
 **********************************************************************/

void LeeDatos(void);
void Calculo(int n);
void Waveform(int n);
void Screening(void);
void Overlap(void);
void PresentaDatos(void);
void DisplayCondition(int j);
void MenuSalida(void);
void MenuEntrada(void);
void MenuRun(int n);
```

```
/*********************************************************************
 *       Main                                                        *
 *********************************************************************/

main()
{
int n;
MenuEntrada();                  /* input menu */
LeeDatos();                     /* read data from input file */

/*** Iterates until the user is satisfied with the autoconsistency ***/

for(itera=0;c!=ESC;itera++)
        {
        for(n=0;n<2;n++)        /* n=0 for electrons, n=1 for holes */
        {
        c='';
        clrscr();
        Calculo(n);          /* transfer matrix routine */
        Waveform(n);         /* wave function calculation */
        window(1,1,80,25);
        }
        if(kbhit()!=0) c=getch(); /* to stop the iteration loop */
        Screening();              /* potential due to photocarriers */
        }

/*********************************************************************/

Overlap();                       /* wave functions overlap */
PresentaDatos();             /* write results to output file */
clrscr();
_setcursortype(_SOLIDCURSOR);
MenuSalida();                     /* end-of-program routine */
}
```

```
/***********************************************************************
 *   LeeDatos                                                          *
 * Read data from input file. Prepare array with all the layers in the*
 * structure. Total number of layers is the global variable "t".       *
 ***********************************************************************/

void LeeDatos(void)
{
FILE *in;
int numstep[LAY],j;
t=0;
in=fopen(nom,"r");
fscanf(in,"%*95c%le %le¥n%*40c%d %d¥n",
        &iniene[0],&iniene[1],&maxlevel[0],&maxlevel[1]);
fscanf(in,"%*40c%le¥n%*40c%le¥n%*40c%le¥n%*40c%le¥n",
        &amorti,&bandgap,&L,&photoc);
while(fscanf(in,"%d %le %le %le %le %le¥n",&numstep[t],&m[t][0],
        &Vorig[t][0],&m[t][1],&Vorig[t][1],&fiorig[t]),numstep[t]!=0)
        {
        for(j=t;j<t+numstep[t];j++)
                {
                m[j][0]=m[t][0];        Vorig[j][0]=Vorig[t][0];  fiorig[j]=fior
ig[t];
                m[j][1]=m[t][1];        Vorig[j][1]=Vorig[t][1];
                }
        t=j;
        }
fclose(in);
}

/***********************************************************************
 *      Calculo                                                        *
 * Solves the Schrodinger equation in the structure by using the       *
 * transfer-matrix method.                                             *
 ***********************************************************************/

void Calculo(int signo)
{
double Lz,Zero,Antes=0.0,Vf,fitep,mara,Vj,pared;
complex gjn,gj,alfa,eplus,eminus,ii(0.0,1.0),
        uno(1.0,0.0),a11,a12,a21,a22,b11,b12,b21,b22,c11,c12,c21,c22;
int j=0,h,v,s;
Vmin=0.0;
limite=0.0;
step=PASO*ELE*iniene[signo];
```

```
for(s=1;s<=(t-1);s++)
        field[s]=fiorig[s]*(1-2*signo)*ELE*100000+(2*signo-1)*screen[s];
for(s=0;s<=t;s++)
        V[s]=Vorig[s][signo]*ELE;
for(s=0;s<=t;s++)
        {
        for(u=t;u>s;u--)
                V[u]=V[u]+field[s]*L*1e-9;
        if(Vmin>V[s])
                Vmin=V[s];
        if(Vmin>(V[s]+field[s]*L*1e-9))
                Vmin=V[s]+field[s]*L*1e-9;
        }
pared=step;
En=Vmin;
if(V[t]<0.0)
        limite=V[t];
MenuRun(signo);
do{
        DisplayCondition(j);
        En=En+pared;
        Lz=L*1e-9;
        Vf=field[1]*Lz;
        fitep=Vf;
        Vj=V[1]+fitep/2;
        gjn=-sqrt(2*m[0][signo]*M*(-En)/(HB*HB));
        if(En>Vj)
                gj=ii*sqrt(2*m[1][signo]*M*(En-Vj)/(HB*HB));
        else
                gj=-sqrt(2*m[1][signo]*M*(-En+Vj)/(HB*HB));
        alfa=gjn*m[1][signo]/(gj*m[0][signo]);
        Aj[0][j]=0; Bj[0][j]=1; Rj[0][j]=gjn;
        v=1;
        a11=(uno+alfa)/2;
        a12=(uno-alfa)/2;
        a21=(uno-alfa)/2;
        a22=(uno+alfa)/2;
        Aj[v][j]=a12; Bj[v][j]=a22; Rj[v][j]=gj;
        for(s=1;s<t;s++)
                {
                Lz=L*1e-9;
                Vf=field[s]*Lz;
                fitep=Vf;
                Vj=V[s]+fitep/2;
                gjn=gj;
```

```
            Vij[v]=Vj/ELE;
            Vj=Vj+fitep;
            mara=m[s+1][signo]/m[s][signo];
            Vj=V[s+1]+field[s+1]*L*1e-9/2;
            if(En>Vj)
                    gj=ii*sqrt(2*m[s][signo]*M*mara*(En-Vj)/(HB*HB));
            else
                    gj=-sqrt(2*m[s][signo]*M*mara*(-En+Vj)/(HB*HB));
            alfa=gjn*mara/gj;
            v++;
            eplus=exp(gjn*Lz);
            eminus=exp(-gjn*Lz);
            b11=eplus*(uno+alfa)/2;
            b12=eminus*(uno-alfa)/2;
            b21=eplus*(uno-alfa)/2;
            b22=eminus*(uno+alfa)/2;
            c11=b11*a11+b12*a21;
            c12=b11*a12+b12*a22;
            c21=b21*a11+b22*a21;
            c22=b21*a12+b22*a22;
            Aj[v][j]=c12; Bj[v][j]=c22; Rj[v][j]=gj;
            a11=c11; a12=c12; a21=c21; a22=c22;
            }
        Zero=real(c22);
        if((Zero*Antes)<0.0)  /*when an energy level is found, the energy*/
            pared=-pared/10;   /*step is reduced to increase precision*/
        if(Zero==Antes)       /*if the limit given by the precision of "double"*
/
            {                   /*is reached, the energy value is stored*/
            pared=step;
            Zero=0.0;
            Ener[j][signo]=En/ELE;
            j++;
            }
        if(norm(c22)<PRECISION)    /*if the zero of c22 is reached, */
            {                       /* the energy value is stored */
            pared=step;
            Zero=0.0;
            Ener[j][signo]=En/ELE;
            j++;
            }
        Antes=Zero;
        }while((En<(-step+limite))&&(j!=maxlevel[signo]));
cuantos=j;
}
```

```
/***********************************************************************
 *       Overlap                                                       *
 * Calculates the wave function overlap and the transition energies   *
 * for electron-hole transitions                                       *
 ***********************************************************************/

void Overlap(void)
{
int v,j,k;
double s1,s2,mini=1.0,mihi=1.0;
for(k=1;k<=maxlevel[1];k++)
        {
        for(j=1;j<=maxlevel[0];j++)
                {
                overint[j][k]=0.0;
                for(v=1;v<(t-1);v++)
                        {
                        s1=wave[v][j][0]-wave[v-1][j][0];  /*reconstruct the sha
pe of*/
                        s2=wave[v+1][j][0]-wave[v][j][0];  /*original wave funct
ions*/
                        if(((s1*s2)<0.0)&&(s1<s2))
                                mini=-mini;
                        s1=wave[v][j][1]-wave[v-1][j][1];
                        s2=wave[v+1][j][1]-wave[v][j][1];
                        if(((s1*s2)<0.0)&&(s1<s2))
                                mihi=-mihi;
                        overint[j][k]+=mini*mihi*sqrt(wave[v][j][0]*wave[v][k][1])
;
                        }
                overint[j][k]*=overint[j][k];
                Tran[j-1][k-1]=bandgap+Ener[j-1][0]+Ener[k-1][1];
                }
        }
}

/***********************************************************************
 *       Waveform                                                      *
 * Calculates the wave functions of electron and hole levels          *
 ***********************************************************************/

void Waveform(int r)
{
```

```
float area[8]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
double renorm;
int v,j,h;
complex aj,bj,rj;
for(v=0;v<(t-1);v++)
        {
        wave[v][0][r]=v*L;
        wave[v][8][r]=Vij[v+1];
        for(j=1;j<=cuantos;j++)
                {
                aj=Aj[v+1][j-1];
                bj=Bj[v+1][j-1];
                rj=Rj[v+1][j-1]*L*1e-9;
                wave[v][j][r]=(float)norm(aj*exp(rj)+bj*exp(-rj))*renorm;
                area[j]=area[j]+wave[v][j][r];
                if(wave[v][j][r]>1e30)
                        {           /*keep the wave function values in the "floa
t" range*/
                        v=0;
                        renorm*=1e-30;
                        for(h=1;h<=cuantos;h++) area[h]=0.0;
                        j=cuantos;
                        }
                }
        }
for(v=0;v<(t-1);v++)
        {
        for(j=1;j<=cuantos;j++)
                wave[v][j][r]=wave[v][j][r]/area[j];
        }
}

/*********************************************************************
 *       Screening                                                   *
 * Calculates the potential due to the photocarriers, the quasi-Fermi *
 * levels and the occupation of the energy levels.                   *
 *********************************************************************/

void Screening(void)
{
int v,j,cuen,nivel,nivho;
double remain,elenivel[8]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
double holnivel[8]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

remain=photoc;
```

```
for(nivel=0;remain>0.0;nivel++)
        remain-=RO*m[0][0]*(Ener[nivel+1][0]-Ener[nivel][0])*(nivel+1);
remain+=RO*m[0][0]*(Ener[nivel][0]-Ener[nivel-1][0])*nivel;
fermiele[0]=Ener[nivel-1][0]+remain/(nivel*RO*m[0][0]);
for(cuen=0;cuen<nivel;cuen++)
        elenivel[cuen]=RO*m[0][0]*(Ener[nivel-1][0]-Ener[cuen][0])+remain/nivel;

remain=photoc;
for(nivho=0;remain>0.0;nivho++)
        remain-=RO*m[0][1]*(Ener[nivho+1][1]-Ener[nivho][1])*(nivho+1);
remain+=RO*m[0][1]*(Ener[nivho][1]-Ener[nivho-1][1])*nivho;
fermiele[1]=Ener[nivho-1][1]+remain/(nivho*RO*m[0][1]);
for(cuen=0;cuen<nivho;cuen++)
        holnivel[cuen]=RO*m[0][1]*(Ener[nivho-1][1]-Ener[cuen][1])+remain/nivho;

for(v=0;v<(t-1);v++)
        {
        screen[v]=0.0;
        for(cuen=0;cuen<7;cuen++)
                {
                for(j=0;j<=v;j++)
                        {
                        screen[v]+=elenivel[cuen]*wave[j][cuen+1][0]
                                -holnivel[cuen]*wave[j][cuen+1][1];
                        }
                }
        screen[v]*=ELE*ELE/(KAP*EZERO);
        screen[v]=(screen[v]+(amorti-1)*scrog[v])/amorti;
        scrog[v]=screen[v];
        }
diferen=(scrbef-screen[(t-1)/2])/ELE;
scrbef=screen[(t-1)/2];
}

/***********************************************************************
 *      PresentaDatos                                                  *
 * Write to the output file the results                                *
 ***********************************************************************/

void PresentaDatos(void)
{
int i,r;
out=fopen(nomsal,"w");
fprintf(out,"Input file name: %s¥n",nom);
fprintf(out,"Photocarrier density (1/m2): %.4le¥n",photoc);
```

```
for(r=0;r<2;r++)
        {
        i=0;
        fprintf(out,"¥n");
        fprintf(out,"Ener. levels %.4lf  %.4lf  %.4lf  %.4lf  %.4lf  %.4lf  %.41
f %.4lf  %.4lf¥n",
                Ener[0][r],Ener[1][r],Ener[2][r],Ener[3][r],Ener[4][r],Ener[5][r],
                        Ener[6][r],Ener[7][r],Ener[8][r]);
        fprintf(out,"Fermi level (eV): %.4lf¥n",fermiele[r]);
        fprintf(out,"¥n");
        fprintf(out,"Wavefunctions¥n");
        while(fprintf(out,"%.3le   %.3le   %.3le   %.3le   %.3le   %.3le   %.3le
   %.3le   %.3le¥n",
                wave[i][0][r],wave[i][1][r],wave[i][2][r],wave[i][3][r],wave[i]
[4][r],
                        wave[i][5][r],wave[i][6][r],wave[i][7][r],wave[i][8][r]),
i++<(t-2));
        }
fprintf(out,"¥n");
fprintf(out,"Squared overlap integral. First line is for first hole level and so
 on¥n");
for(r=1;r<=maxlevel[1];r++)
        {
        fprintf(out,"%d  %.4le  %.4le  %.4le  %.4le  %.4le  %.4le  %.4le  %.4le¥
n",
                r,overint[1][r],overint[2][r],overint[3][r],overint[4][r],overin
t[5][r],
                        overint[6][r],overint[7][r],overint[8][r]);
        }
fprintf(out,"¥n");
fprintf(out,"Transition energies (eV). First line is for first hole level and so
 on¥n");
for(r=0;r<maxlevel[1];r++)
        {
        fprintf(out,"%d  %.4lf  %.4lf  %.4lf  %.4lf  %.4lf  %.4lf  %.4lf  %.4lf
 %.4lf¥n",
                r+1,Tran[0][r],Tran[1][r],Tran[2][r],Tran[3][r],Tran[4][r],Tran
[5][r],
                        Tran[6][r],Tran[7][r],Tran[8][r]);
        }
fclose(out);
}


/*********************************************************************
```

```
 *      DisplayCondition                                              *
 * Display the evolution of the calculation                          *
 *********************************************************************/

void DisplayCondition(int j)
{
double porcen;
porcen=100*(En-limite)/(Vmin-limite);
gotoxy(39,10);
cprintf("%3.0f",porcen);
gotoxy(41,12);
cprintf("%1d",j);
gotoxy(41,14);
cprintf("%1d",itera);
gotoxy(32,16);
cprintf("%1.4f",diferen);
}

/*********************************************************************
 *       MenuSalida                                                  *
 * Sound a bell and end the program                                  *
 *********************************************************************/

void MenuSalida(void)
{
sound(1000);
delay(300);
sound(3000);
delay(300);
sound(2000);
delay(300);
nosound();
clrscr();
window(1,2,70,3);
textattr(LIGHTGREEN);
cprintf("WAVYPOI completed the calculation.");
}

/*********************************************************************
 *       MenuEntrada                                                 *
 * Menu for input of data and results file names                     *
 *********************************************************************/

void MenuEntrada(void)
{
```

```
clrscr();
window(5,2,80,23);
textattr(LIGHTGREEN);
cprintf("WAVYPOI solves Schrodinger and Poisson eq. autoconsistently.¥n");
gotoxy(1,22);
cprintf("Read the manual for preparation of input file.");
gotoxy(37,3);
textattr(YELLOW);
cprintf("by Pablo O. Vaccaro, ATR-ORCL (1995)");
window(10,10,45,13);
textattr(LIGHTCYAN);
cprintf("         Input file is: ");
gotoxy(1,3);
cprintf("        Output file is: ");
gotoxy(25,1);
gets(nom);
gotoxy(25,3);
gets(nomsal);
window(1,1,80,25);
clrscr();
}

/*********************************************************************
 *      MenuRun                                                      *
 * Menu displayed during the calculation                             *
 *********************************************************************/

void MenuRun(int signo)
{
window(1,1,80,24);
clrscr();
textattr(YELLOW);
gotoxy(2,2);
cprintf("WAVYPOI is calculating. Press ESC when the error is small enough.");
textattr(CYAN);
gotoxy(10,8);
if(signo==0)
        cprintf("Electrons");
else
        cprintf("Holes");
textattr(LIGHTGREEN);
gotoxy(10,10);
cprintf("Energy remaining:                percent");
gotoxy(10,12);
cprintf("Number of energy levels found: ");
```

```
gotoxy(10,14);
cprintf("Number of iterations: ");
gotoxy(10,16);
cprintf("Error of elec. field:                  V/m");
textattr(LIGHTMAGENTA);
gotoxy(5,5);
cprintf("Energy scan from %1.4f eV to %1.4f eV",Vmin/ELE,limite/ELE);
textattr(YELLOW);
_setcursortype(_NOCURSOR);
}
```