Contents

1. Constants

1.1 Program Constants

As defined in apd.h.

Name	Type	Value	Definition
NUMX	integer	3000	Number of sections into which to divide structure.
MAX_CHAR	integer	31	Maximum characters in a string.
MAXCHARS	integer	31	""
NUM_BIN	unsigned long	1100	Number of bins for time response histograms.
NUM_BIN1	unsigned long	1024	Number of bins used for statistics.
NUM_BIN2	unsigned long	32	Number of time bins within which autocorrelation is to be evaluated.
MAX_COUNT	integer	30000	Limits gain: used to prevent infinite loops.
STEP_LIMIT	integer	50	Limits the number of steps.
RANDOMBASE	float	RAND_MAX*	Resolution of random number.
MAXLAYERS	integer	70	Maximum number of layers allowed in the structure.

*As converted to float.

1.2 Physical Constants

As defined in apd.h. Note: All physical constants are of type double.

Name	Value	Units	Definition
q	1.60219e-19	Coulombs	Charge of an electron.
e	2.71828183	N/A	Euler's number: base of the natural logarithm.
eps0	8.85419e-12	Farads/meter	Dielectric constant in a vacuum.
ep0	8.85419e-12	Farads/meter	See above. Same values.
pi	3.141592653589793	N/A	Ratio of a circle's circumference to its diameter.
h	1.05459e-34	Double	H-bar: Reduced Planck (Dirac) Constant: Quantum of action divided by 2*pi.
am0	9.10953e-31	Kilograms	Electron mass (at rest)
bk	1.38066e-23	$m^2 * kg/(K * s^2)$	Boltzmann Constant

1.3 Other Constants

As defined in apd.h.

Name	Type	Value	Definition
tem	float	300.0	Temperature, presumably, in units of Kelvin.
de	float	0.035	Energy increment (eV?)
iemax	integer	100	Maximum energy steps.
ivalley	integer	2	Number of conduction band valleys.
iscat	integer	11	Number of scattering mechanisms.

2. Data Structures

2.1 Layer

As defined in apd.h. Contains information on the following:

Name	Type	Definition
Thickness	float	Thickness of the layer.
Doping	float	Doping of the layer
Material[MAXCHARS]	char	Name of the material.
Ae	float	"The parameters Alex used"
Ah	float	""
Eacte	float	""
Eacth	float	""
ce	float	""
ch	float	""
EacteH	float	Hard threshold for electron, above which ionization is certain.
EacthH	float	Hard threshold for hole.
Abs	float	Absorption coefficient, measured in meters^{-1}.
Ec[ivalley+1]	float	Information about band edge alignment. Ec[1] - Γ-Valley. Ec[2] = L-Valley.
Ev[ivalley+1]	float	""
eps	float	Relative dielectric constant at static frequency.
epf	float	Relative dielectric constant at high frequency.
ep	float	?
eg	float	Band gap.
me[ivalley+1]	float	Effective mass of electron.
mh[ivalley+1]	float	Effective mass of hole.
hs	float	Hole scattering rate scaling factor.
gm	float	Total scattering rate in this layer, as a function of energy.
swk[ivalley+1][iscat+1][iemax+1]	float	Individual scattering rate
af[ivalley+1]	float	Band parameters.
af2[ivalley+1]	float	""
af4[ivalley+1]	float	""
smh[ivalley+1]	float	Composite constants made of am, q, h ???
hhm[ivalley+1]	float	""
hm[ivalley+1]	float	""
rou	float	?
sv	float	?

Name	Type	Definition
hwo	float	?
hwij	float	?
hwe	float	?
hwaco	float	?
qd2	float	?
x	float	Aluminum fraction, used for alloy scattering estimate.
as	float	Alloy scattering coefficient
gmH	float	Total scattering rate for holes in this layer.
swkH[ivalley+1][iscat+1][iemax+1]	float	Individual scattering rate for holes.
afH[ivalley+1]	float	Band parameters for holes.
af2H[ivalley+1]	float	""
af4H[ivalley+1]	float	""
smhH[ivalley+1]	float	Composite constants (holes)
hhmH[ivalley+1]	float	""
hmH[ivalley+1]	float	""

2.2 Device

As defined in apd.h.

Name	Type	Definition
NumLayers	integer	Number of layers in device.
DeviceLayers[[MAXLAYERS]	layer	?
acu_thickness[MAXLAYERS]	float	?
TotalThickness	float	Total thickness of the device.
DepletionEnd	float	End of depletion region.
DepletionStart	float	Start of depletion region.
EFTable[NUMX+1]	float	Electric field as a function of position.
Potential[NUMX+1]	float	Potential energy as a function of position.
EFMax	float	Peak electric field
Vapp	float	Applied voltage.
Vbi	float	Bias voltage.
index[NUMX]	integer	Index as a function of position.

2.3 Stats

As defined in apd.h.

Name	Type	Definition
ImpactDist_e[NUMX+1]	integer	Spatial distribution of electron impacts.
ImpactDist_h[NUMX+1]	integer	Spatial distribution of hole impacts.
AbsDist[NUMX+1]	integer	Distribution of absorption events.
GainDist[MAX_COUNT+1]	integer	Gain distribution.
max_steps	integer	Number of biases estimated.
stat_bin	integer	Bin interval for statistics of single carrier gain (m).
Ip	integer	?
gain[STEP_LIMIT]	float	Variables for gain and bias
Vapp[STEP_LIMIT]	float	max_step isn't constant, so use huge arrays.
EFMax[STEP_LIMIT]	float	Track the peak electric field.
sq_gain[STEP_LIMIT]	float	To calculate the noise.
bw[STEP_LIMIT]	float	To record the bandwidth.
max_time	float	Longest time in device.
TimeDist[NUM_BIN+1]	integer	Current defined as electron arrival number vs. time.
Current[NUM_BIN+1]	float	Conduction current defined as instantaneous velocity x electron charge.
Fp[NUM_BIN+1]	float	Power spectrum.
Auto[NUM_BIN+1]	float	Auto correlation function of the pulse.
Sf[NUM_BIN2]	float	Shot noise power.
Sf0[STEP_LIMIT]	float	Low frequency noise (2eI term, times Fano factor).
TotI2	float	$Itotal^2$ mean from a series of pulses.
AvgI	float	Average current over max_time.
TotI2s[STEP_LIMIT]	float	?
AvgIs[STEP_LIMIT]	float	?
total_count	long	?
total_sq_count	long	?
EnergyDist_e[NUMX+1]	float	Total energy of electron in this position.
VelocityDist_e[NUMX+1]	float	Total velocity of electron in this position.
CountDist_e[NUMX+1]	integer	Counted number of times electron has passed this position.
EnergyDist_h[NUMX+1]	float	Total energy of hole in this position
VelocityDist_h[NUMX+1]	float	Total velocity of hole in this position
CountDist_h[NUMX+1]	integer	Counted number of times hole has passed this position.
GDist_e[NUMX+1]	integer	Γ-valley electron.
LDist_e[NUMX+1]	integer	L-valley electron.
GDist_h[NUMX+1]	integer	Heavy hole.
LDist_h[NUMX+1]	integer	Light hole.

2.4 Params

As defined in apd.h.

Name	Type	Definition
tem	float	Temperature.
photonev	float	Incident photon energy.
impfac	float	Impurity scattering rate scaling factor.
DeltaEg	float	Bandgap energy temperature coefficient.
I0	integer	Number of initial electron-hole pairs.
avg	integer	Number of simulations over which final output is averaged.
max_time	double	Maximum time checked (?)
min_time	float	Minimum time scale (?)
Vapp	float	Starting bias (applied).
Vinc	float	Increment by which to increase the
bias. Vbi	float	Built-in potential.
max_gain	float	Maximum gain before terminating and saving.
max_steps	integer	Maximum number of electric fields to try.
stat_bin	integer	Bin interval for statistics of single carrier gain (m).
structfilename[MAX_CHAR]	char	Filename for "structure" information.
gainfilename[MAX_CHAR]	char	Filename for "gain" information.
distrfilename[MAX_CHAR]	char	Filename for "distr" information.
freqfilename[MAX_CHAR]	char	Filename for "freq" information.
infilename[MAX_CHAR]	char	Filename for input file.
carrierfilename[MAX_CHAR]	char	Filename for "carrier" information.
swk0filename[MAX_CHAR]	char	Filename for "swk0" information.
swk1filename[MAX_CHAR]	char	Filename for "swk1" information.
swk2filename[MAX_CHAR]	char	Filename for "swk2" information.
swHfilename[MAX_CHAR]	char	Filename for "swkH" information.

2.5 Electron

As defined in apd.h.

Name	Type	Definition
kx	float	X-Component of wave vector.
ky	float	Y-Component of wave vector.
kz	float	Z-Component of wave vector.
iv	integer	Valley index. iv = 1 for Γ-valley ; iv = 2 for L-valley.
ve	float	Electron average velocity during a flight time.
e	float	Electron initial energy dring each free flight.
position	float	Position of electron.
ft	float	?
impact	integer	?
count	integer	?
sign	integer	?
EnergyDist[NUMX+1]	float	Total energy of electron in this position
VelocityDist[NUMX+1]	float	Total velocity of electron in this position
CountDist[NUMX+1]	integer	Counted number of times electron has passed this position.
GDist[NUMX+1]	integer	Γ-valley electron.
LDist[NUMX+1]	integer	L-valley electron.

2.6 Hole

As defined in apd.h.

Name	Type	Definition
kx	float	X-Component of wave vector.
ky	float	Y-Component of wave vector.
kz	float	Z-Component of wave vector.
iv	integer	Valley index. iv = 1 for heavy hole ; iv = 2 for light hole.
ve	float	Hole average velocity during a flight time.
e	float	Hole initial energy dring each free flight.
position	float	Position of hole.
ft	float	?
impact	integer	?
count	integer	?
sign	integer	?
EnergyDist[NUMX+1]	float	Total energy of hole in this position
VelocityDist[NUMX+1]	float	Total velocity of hole in this position
CountDist[NUMX+1]	integer	Counted number of times hole has passed this position.
GDist[NUMX+1]	integer	Heavy hole.
LDist[NUMX+1]	integer	Light hole.

3. Function Declarations

3.1 For Setting Up Device Parameters

As defined in apd.h.

Function Name	Inputs	Outputs	Description
SetBias	device float DesiredBias float JunctionLocation1	void	Sets bias of given junction location to given input value.
CalcBias	device float JunctionLocation1	float	Calculates the bias for a given junction location.
BuildEFTable	device float JunctionLocation1	void	Builds electric field table for a given junction location.
FindJunction	device	float	Finds junction in given device.
GetDoping*	device float position	float	Gets the doping value for a given position.
GetAbs	device float position	float	Gets the absorption coefficient value for a given position.
GetParams	device float position	layer	Gets the parameter values of the layer at a given position.

*Obsolete. GetDoping is a function which is not defined in the most updated version of the APD code. We should be able to remove this without consequence.

3.2 For Generating Electrons and Holes

As defined in apd.h.

Function Name	Inputs	Outputs	Description
StartElectron	device params electron hole stats	void	
StartHole	device params hole electron stats	void	
AbsorbPhoton	device stats	float	Generates an electron-hole pair based on the absorption coefficients of the different material layers; returns the position of the electron-hole pair.

3.3 For Data Manipulation

As defined in apd.h.

Function Name	Inputs	Outputs	Description
InitArrays	stats	void	Initializes values for stats.
LoadParams	device params	void	Imports parameters from input datafile.
FreqResponse	stats params	void	Computes the Fourier transform of the time response.
DataToFile	device stats params	void	Outputs the data calculated its respective files.

3.4 For Program Flow

As defined in apd.h.

Function Name	Inputs	Outputs	Description
BiasLoop	device params stats electron hole float JunctionLocation1	void	
CarrierLoop	device params stats electron hole	void	

3.5 For Physical Processes

As defined in apd.h.

Function Name	Inputs	Outputs	Description
ElectronDrift	electron device params float tau	void	
HoleDrift	hole device params float tau	void	
ElectronScat	device params electron	void	
HoleScat	device params hole	void	
realft	float data[] unsigned long n int isign	void	Uses numerical recipes to find the FFT for a real function (as opposed to a complex function with four1.C).
four1	float data[] unsigned long n int isign	void	Uses numerical recipes to find the FFT for a complex function.

4. Code Explanations

4.1 apd.h

Inputs (none)
Outputs (none)
Header File. Defines: constants, physical constants, structures, & function declarations.

4.2 AbsorbPhoton.C

Includes header file apd.h.
Inputs device APDdevice, stats APDResults
Outputs float position

Overview

Generates an electron-hole pair based on the absorption coefficients of the different material layers; returns the position of the electron-hole pair.

Explanation

Steps through each of the sections of the device (defined using NUMX). For each step, generates a random number, and compares it to the absorption level in that layer. If the absorption level is higher than the random number for any position, this position is noted, the loop is terminated, and the position returned as a float.

Uses GetAbs.C
See Also: CarrierLoop

4.3 BiasLoop.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, stats APDresults, electron NewElectron, hole NewHole, float JunctionLocation1
Outputs (none)

Overview

Sets up biases for the device.

Explanation

Loops through several steps, provided gain is not too high. For each bias, sets the following stats.* variables to their respective values:

Name	Value
gain[i]	0.0
sq_gain[i]	0.0
AvgIs[i]	0.0
TotI2s[i]	0.0
Ip	0

Using InitArrays.C	
TimeDist[i]	0
Current[i]	0.0
Fp[i]	0.0
Auto[i]	0.0
Sf[i]	0.0
GainDist[i]	0
ImpactDist_h[i]	0
ImpactDist_e[i]	0
AbsDistr[i]	0
EnergyDistr_e[i]	0.0
EnergyDistr_h[i]	0.0
VelocityDistr_e[i]	0.0
VelocityDistr_h[i]	0.0
CountDistr_e[i]	1
CountDistr_h[i]	1
LDist_e[i]	0
GDist_e[i]	0
LDist_h[i]	0
GDist_h[i]	0

Sets the bias of the device to the applied voltage, then calls CarrierLoop.C. Calculates the primary current based on gain distribution, then uses that value to calculate the gain and sq_gain for this step. Takes the fourier transform of the time response using FreqResponse.C.

Sets the following stats.* values accordingly:

Name	Value
Vapp[i]	stats.Vapp
EFMax[i]	stats.EFMax
AvgIs[i]	stats.AvgI
TotI2s[i]	stats.TotI2
Sf[i]	Sf[10]

Prints "Vapp[i] — gain[i] — sq_gain[i] — gain[i]/gain[i] (???) — AvgIs[i] — AvgIs[i]//sqrt(TotI2s[i]-AvgIs[i]*AvgIs[i]) — Sf0[i] — Sf0[i]/2*q*AvgIs[i]*gain[i] — (Sf0[i]/2*q*AvgIs[i]*gain[i])/sq_gain[i]/gain[i]/gain[i] (???).

Increments Vapp by Vinc, then the loop repeats, as long as the number of steps is less than the maximum number of steps.

Uses InitArrays.C, CarrierLoop.C, FreqResponse.C, & SetBias.C
See Also: main.C

4.4 BuildEFTable.C

Includes header file apd.h.
Inputs device MyDevice, float JunctionLocation1
Outputs (none)

Overview

Builds the table of electric fields using the value for the max electric field stored with the structure, as set by other routines. Starts at the junctions and works outward in both directions to find the field over the full device.

Explanation

Given the device and the junction location, defines a dx as the thickness of the device as split into NUMX sections. Sets DepletionStart and DepletionEnd to 0.0. Values for the electric field table are indexed from 0 to NUMX.

Up to and including the junction index, values for the electric field table are set to the EFMAX, which has already been defined for the device by other procedures.

After the junction index, the value for EFTable[i] is calculated as follows:

$$EFTable[i] = (q * dx)/(\epsilon_0 * \epsilon_{layer} * Doping_{layer} + EFTable[i-1])$$

For any index with an EFTable value less than 1e5, the value is set to 1e5. Otherwise, the value for DepletionEnd is set to i*dx.

Next, the table is build backwards from the junction, stepping from the junction index down to zero. These values had previously been set to EFMAX for the purposes of calculating the EFTable values after the junction. Now, the EFTable values for this section are calculated as follows:

$$EFTable[j] = EFTable[j+1] - (q * dx)/(\epsilon_0 * \epsilon_{layer} * Doping_{layer}))$$

For any index with an EFTable value less than 1e5, the value is set to 1e5. Otherwise, the value for DepletionStart is set to j*dx.

Uses FindJunction.C
See Also: CalcBias.C, GetParams.C, & SetBias.C

4.5 CalcBias.C

Includes header file apd.h.
Inputs device MyDevice, float JunctionLocation1
Outputs float MyDevice.Vapp - MyDevice.Vbi

Overview

Calculates the bias across the device given the set maximum electric field.

Explanation

Calculates the table of electric fields for the device. Divides the device into NUMX sections of width dx, then builds a table of potential values based on the electric field values at these positions. Calculated as follows:

$$Potential(x) = Potential(x-1) + ElectricField(x) * dx$$

Then sets the applied voltage to the potential at the final position on the device, subtracts the built-in bias, and returns this difference as a float.

Uses BuildEFTable.C
See Also: SetBias.C

4.6 CarrierLoop.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, stats APDresults, electron NewElectron, hole NewHole
Outputs (none)

Overview

Loops through all the carriers injected into the device.

Explanation

Defines initial values as follows:

Name	Type	Value
position	float	0.0
ft	float*	malloc((APDparams.I0+100)*sizeof(float))
time	float	0.0
p	double	1.0/e
i	integer	0
randomnumber	double	undefined
randomnumber1	double	undefined
stats.total_count	?	0
stats.total_sq_count	?	0

Prints the value of the index, and defines a random number for the double (randomnumber). For each index value, through an integer n starting at 0. Prints the value of n, then calculates the poisson probability as the previous probability divided by this integer n. Defines a second random number for the double (randomnumber1). Calculates a value for ft[i+n-1] as follows:

$$ft[i+n-1] = (i * time^2_{max})/(I_0 * (I_0 + randomnumber1 - 0.5))$$

Breaks through the loop once randomnumber is greater than or equal to p; otherwise, increments index i by n.

A second loop runs while index i is less than I_0, the number of injected carriers, as defined by APDparams.
For each index i, sets parameters as follows:

Name	Value
time	0
NewElectron.count	0
NewHole.count	0
NewElectron.sign	i+1
NewElectron.ft	ft[i]*

*As calculated in previous part.

Within this loop, an index j is stepped from 0 to NUMX. For each index j, parameters are set as follows:

Name	Value
NewElectron.EnergyDist[j]	0.0
NewElectron.VelocityDist[j]	0.0
NewElectron.CountDist[j]	0
NewElectron.GDist[j]	0
NewElectron.LDist[j]	0
NewHole.EnergyDist[j]	0.0
NewHole.VelocityDist[j]	0.0
NewHole.CountDist[j]	0
NewHole.GDist[j]	0
NewHole.LDist[j]	0

Prints the value for APDresults.total_count, then calls AbsorbPhoton.C to set the value for NewElectron.position and NewHole.position. Defines a dx as the thickness of the device as split into NUMX sections. The PosIndex integer is calculated by dividing NewElectron.position by the dx thickness, as converted to an integer. This PosIndex value is printed.

If NewElectron.ft is above the maximum time or below the minimum time, as defined by APDparams, sets it to the maximum or minimum time, respectively. Sets the following values accordingly:

Name	Value
bktq	$bk * Temp/q$
NewElectron.ve	0
NewHole.ve	0
NewElectron.iv	1
NewHole.iv	1

$$NewElectron.e = \frac{(photoneV - Ec[NewElectron.iv]_{layer} - Ev[NewHole.iv]_{layer}) * mh[NewHole.iv]_{layer}}{mh[NewHole.iv]_{layer} + me[NewElectron.iv]_{layer}}$$

$$NewHole.e = photoneV - Ec[NewElectron.iv]_{layer} - Ev[NewHole.iv]_{layer} - NewElectron.e$$

$$ki = smh[NewElectron.iv]_{layer} * sqrtNewElectron.e * (1.0 + af[NewElectron.iv]_{layer}) * NewElectron.e)$$

Now, a random value is assigned to randomnumber. A float cs is defined as $0.5 * randomnumber$, and a second float sn is defined as $\sqrt{1.0 - cs^2}$. A new random number is now assigned to randomnumber, and the following values are calculated and assigned:

Name	Value
float fai	$2 * \pi * randomnumber$
NewElectron.kx	$ki * cs$
NewElectron.ky	$ki * sn * cos(fai)$
NewElectron.kz	$ki * sn * sin(fai)$

The value for ki is now reassigned as follows:

$$ki = smhH[1]_{layer} * \sqrt{NewHole.e * (1.0 + afH[1]_{layer} * NewHole.e)}$$

A new random number is generated and assigned to randomnumber, and the value for cs is reassigned to $cs = 1.0 - 2.0 * randomnumber$. The value for sn is reassigned to $\sqrt{1.0 - cs^2}$, where cs is the new value. Values for the following variables are reassigned as indicated in the table below.

Name	Value
float fai	$2 * \pi * randomnumber$
NewElectron.kx	$ki * cs$
NewElectron.ky	$ki * sn * cos(fai)$
NewElectron.kz	$ki * sn * sin(fai)$

The procedure then calls the StartElectron.C and StartHole.C routines. As long as NewElectron.count is less than MAX_COUNT, APDresults.GainDist[NewElectron.count] is incremented. Otherwise, gain overflow occurs, and APDresults.GainDist[MAX_COUNT] is incremented. If this value is greater than or equal to three, the loop is terminated, as there have been too many overcounts.

NewElectron.count is added to the total count in the states, and the square of NewElectron.count is added to the total_sq_count. The following data is incremented from j = 0 to j = NUMX with a step of 1:

Name	Value
APDresults.EnergyDist_e[j]	+= NewElectron.EnergyDist[j]
APDresults.VelocityDist_e[j]	+= NewElectron.VelocityDist[j]
APDresults.CountDist_e[j]	+= NewElectron.CountDist[j]
APDresults.GDist_e[j]	+= NewElectron.GDist[j]
APDresults.LDist_e[j]	+= NewElectron.LDist[j]
APDresults.EnergyDist_h[j]	+= NewHole.EnergyDist[j]
APDresults.VelocityDist_h[j]	+= NewHole.VelocityDist[j]
APDresults.CountDist_h[j]	+= NewHole.CountDist[j]
APDresults.GDist_h[j]	+= NewHole.GDist[j]
APDresults.LDist_h[j]	+= NewHole.LDist[j]

This ends the loop for the injected carriers.

Next, opens the current file, and exports the data as follows, with i stepping from 0 to NUM_BIN in increments of 1:

$\frac{MAX\_TIME*i}{NUM\_BIN}$ — Current[i]

Next, the fano factor is calculated. (Add more here... currently too confused.)*

Uses AbsorbPhoton.C, StartElectron.C, & StartHole.C
See Also: BiasLoop.C

4.7 DataToFile.C

Includes header file apd.h.
Inputs device APDdevice, stats APDresults, params APDparms
Outputs (none)

Overview

Outputs the data calculated to its respective files.

Explanation

Exports data which varies as function of position to structfilename.
Includes: Position — Abs (Doping) — Electric Field — Electron Impact — Hole Impact — Absorption — E_C — E_V

Exports impulse response data to freqfilename.
Includes: Time — Current-Ramo — Frequency (GHz) — Sf

Exports data which varies with bias to gainfilename.
Includes: Bias — Gain — Excess Noise Factor — Average Current — Signal to Noise Ratio — Sf[0] — yF(M) — y

Exports the histogram of the distribution of the gain to distrfilename.
Includes: Gain — Number of Carriers — Log(Number of Carriers)

Exports data for the average carrier at a given position.
Includes: Position — Average Kinetic Energy of Electron — Average Velocity of Electron — Average Kinetic Energy of Hole — Average Velocity of Hole — Γ Dist — L Dist — X Dist — Heavy Hole Dist — Light Hole Dist — Split Dist

Uses (none)
See Also: main.C

4.8 ElectronDrift.C

Includes header file apd.h.
Inputs electron DriftElectron, device APDdevice, params APDparams, float tau
Outputs (none)

Overview

Simulates electron drift under the electric field.

Explanation

Creates variables accordingly:

Name	Type	Value	Comments
qh	float	q/h	Constant
dx	float	APDdevice.TotalThickness/NUMX	Divides the device into NUMX sections, each of thickness dx.
PosIndex	integer	DriftElectron.position/dx	Value as converted to an integer.
fx	float	APDdevice.EFTable[PosIndex]	Electric field at this point.
ei	float	DriftElectron.e	Initial energy of the drift electron
dkx	float	qh*fx*tau	Change in the value for kx, based on the field fx.

Declares the x-component of the wave vector (kx) to modify by dkx, as defined above. Creates a new set of floats skx, sky, and skz as the squares of kx, ky, and kz components, respectively. Creates another float sk, the sum of skx, sky, and skz. Defines floats sq and ef as noted below.

$$sq = \sqrt{1.0 + APDdevice.af4[ValleyIndex]_{layer} * APDdevice.hmm[ValleyIndex]_{layer} * sk}$$

$$ef = \frac{sq - 1.0}{APDdevice.af2_{layer}}$$

Sets the initial energy of the DriftElectron to ef, and the value for ElectronDrift.impact to 0. Sets the value of DriftElectron's average carrier velocity (ve) to

$$ve = \frac{(ef - ei) * tau}{fx}$$

Sets the maximum and minimum of the DriftElectron velocity to $\pm 3 \times 10^7 eV$. Creates an integer variable PosIndexL, equal to the position of the electron as divided by the section thickness dx. If the PosIndexL is between 1 and NUMX, incrememnts the position value of the DriftElectron by $velocity \times \tau$. Ensures that the position value s between 0 and the TotalThickness of the device, then recalculates PosIndex, now that the position has been changed. Increments the DriftElectron EnergyDist at this position index by the energy of the DriftElectron.

If the valley index of the DriftElectron indicates the Γ-valley, increment GDist at this position index by 1. If the valley index indicates the L-valley, increment LDist at this position index by 1. Increment VelocityDist at this position index by this DriftElectron's average carrier velocity, and the CountDist at this position index by 1.
If this position index is between 1 and NUMX, and the effective mass in the layer indicated by PosIndex is not equal to the effective mass in the layer indicated by PosIndexL, then proceed as follows:

Create float EL, with a value equal to the energy of the DriftElectron. Create a float ER, with a value as follows: ER = (DriftElecton's Energy) - (E_c of layer at PosIndex, with this valley index) + (E_c of layer at PosIndexL, with this valley index). Creates another float kL, equal to the square root of 2 times the effective mass at PosIndexL times am0 times EL, divided by h. Declares another float kR, the same as kL but at PosIndex. Generates a random number to assign to randomnumber. Declares yet another variable, Reflection, as follows:

$$Reflection = \left(\frac{\frac{kL}{m_{eff,PosIndexL}} - \frac{kR}{m_{eff,PosIndex}}}{\frac{kL}{m_{eff,PosIndexL}} + \frac{kR}{m_{eff,PosIndex}}} \right)^2$$

If this randomnumber is less than the Reflection, sets the DriftElectron parameters as follows:

Name	**Set Value**
DriftElectron.e	EL
DriftElectron.kx	-DriftElectron.kx
DriftElectron.position	PosIndexL*dx
DriftElectron.ve	-DriftElectron.ve

Otherwise, if the randomnumber is greater than or equal to the Reflection, executes the following:
If the DriftElectron average carrier velocity is negative, sets the position to $(PosIndex - 0.5) \times dx$. If the DriftElectron velocity is not negative, sets the position to $(PosIndex + 0.5) \times dx$, and the energy to ER. Either way, if the energy is less than zero, generates a new value for randomnumber, calculates a new float $bktq = bk * tem/q$, and sets the value for the DriftElectron energy to $-bktq \times \log randomnumber \times 1.5$. Now, regardless of the energy sign, sets values as follows:

Name	**Set Value**
ki	$smh_{PosIndex} \times \sqrt{Energy \times (1 + af_{PosIndex} \times Energy)}$
kyz	$\sqrt{\lvert ki^2 - kx^2 \rvert}$
randomnumber	new random number
fai	$2 * \pi * randomnumber$
DriftElectron.ky	$kyz \times cos(fai)$
DriftElectron.kz	$kyz \times sin(fai)$

Note: Doesn't Loop!

Uses (none)
See Also: StartElectron.C

4.9 ElectronScat.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, electron ScatElectron
Outputs (none)

Overview

Calculates scattering of electroncs according to scattering rates.

Explanation

Uses StartElectron.C, StartHole.C
See Also: StartElectron.C

4.10 FindJunction.C

Includes header file apd.h.
Inputs device MyDevice
Outputs (none)

Overview

Finds the position of a doping junction for a given device.

Explanation

Steps through the layers of a device, and compares the dopings of each layer to the first layer. If the doping on these two layers don't agree, returns the position value of the interface between them as a float.

Uses (none)
See Also: BuildEFTable.C, main.C

4.11 four1.C

Includes header file apd.h.
Inputs float data[], unsigned long nn, int isign
Outputs (none)

Overview

Uses numerical recipes to find the FFT for a complex function.

Explanation

Numerical recipe. Not really important to go into for the sake of the MC simulation.

Uses (none)
See Also: realft.C

4.12 FreqResponse.C

Includes header file apd.h.
Inputs stats APDresults, params APDparams
Outputs (none)

Overview

Computes the Fourier transform of the time response.

Explanation

Numerical recipe. Not really important to go into for the sake of the MC simulation.

Uses realft.C
See Also: BiasLoop.C

4.13 GetAbs.C

Includes header file apd.h.
Inputs device MyDevice, float position
Outputs (none)

Overview

Steps through the structure until the layer containing the desired position is found, and returns the absorption in that layer.

Explanation

As long as the current thickness of the device is less than the input position, adds the thickness of the layer to the current thickness. Once the position is less than the current thickness, returns the absorption value of the layer before it, and returns it as a float.

Uses (none)
See Also: AbsorbPhoton.C

4.14 GetParams.C

Includes header file apd.h.
Inputs device MyDevice, float position
Outputs (none)

Overview

Steps through the structure until the layer containing the desired position is found, and returns the doping in that layer.

Explanation

As long as the current thickness of the device is less than the input position, adds the thickness of the layer to the current thickness. Once the position is less than the current thickness, returns the layer before it.

Uses BuildEFTable.C
See Also: (none)

4.15 HoleDrift.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, float tau
Outputs (none)

Overview

Simulates hole drift under the electric field.

Explanation

Uses (none)
See Also: StartHole.C

4.16 HoleScat.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, hole ScatHole
Outputs (none)

Overview

Simulates hole scattering according to calculated hole scattering rates.

Explanation

Uses StartElectron.C
See Also: StartHole.C

4.17 InitArrays.C

Includes header file apd.h.
Inputs stats APDresults
Outputs (none)

Overview

Initializes values for stats.

Explanation

Sets values for stats.* as follows:

Name	Value
TimeDist[i]	0
Current[i]	0.0
Fp[i]	0.0
Auto[i]	0.0
Sf[i]	0.0
GainDist[i]	0
ImpactDist_h[i]	0
ImpactDist_e[i]	0
AbsDistr[i]	0
EnergyDistr_e[i]	0.0
EnergyDistr_h[i]	0.0
VelocityDistr_e[i]	0.0
VelocityDistr_h[i]	0.0
CountDistr_e[i]	1
CountDistr_h[i]	1
LDist_e[i]	0
GDist_e[i]	0
LDist_h[i]	0
GDist_h[i]	0

Uses (none)
See Also: BiasLoop.C

4.18 LoadParams.C

Includes header file apd.h.
Inputs device APDDevice, params APDparams
Outputs (none)

Overview

Imports parameters from input datafile.

Explanation

Imports parameters from input datafile.

Device Input File

Input datafile with descriptions.

For overall device:

Line No.	Sample Name	Comments
1	pin.prm	File name
2	tem	-
3	300	Temperature, in Kelvin
4	photonev	-
5	0.8	Photon energy, in eV
6	impfac	-
7	1	Unused - leave at ”1”
8	DeltaEg	-
9	0.05	Temperature coefficient of bandgap
10	I0	-
11	10000	Number of injected carriers
12	maxtime	-
13	1e-8	Maximum time, in seconds
14	mintime	-
15	1e-17	Minimum time, in seconds
16	Vapp	-
17	10	Applied voltage, in V
18	Vinc	-
19	0.5	Increment of increasing voltage, in V
20	Vbi	-
21	1.1	Built-in voltage, in V
22	maxgain	-
23	1000	Limiting gain
24	maxsteps	-
25	10	Maximum number of steps (voltage increments)
26	statbin	-
27	1	Bin size for statistics calculations

Line No.	Sample Name	Comments
28	structfilename	-
29	structpin1.txt	Output file name for structure outputs
30	gainfilename	-
31	gainpin1.txt	Output file name for gain outputs
32	distrfilename	-
33	distrpin1.txt	Output file name for gain distribution outputs
34	freqfilename	-
35	freqpin1.txt	Output file name for frequency calculation outputs
36	carrierfilename	-
37	carrierpin1.txt	Output file name for carrier distribution outputs
38	swk0filename	-
39	swk0.out	File name for exporting scattering parameters for X-valley
40	swk1filename	-
41	swk1.out	File name for exporting scattering parameters for Gamma-valley
42	swk2filename	-
43	swk2.out	Scattering parameters for L-valley
44	swkHfilename	-
45	swkH.out	File name for exporting scattering parameters for heavy-hole
46	freqfilename	-
47	freqpin1.txt	Output file name for frequency calculation outputs
48	carrierfilename	-
49	carrierpin1.txt	Output file name for carrier distribution outputs

Uses (none)
See Also: main.C

4.19 main.C

Includes header file apd.h.
Inputs (N/A)
Outputs (N/A)

Overview

This function is the entry point, where everything really happens. Takes in the data, sets up everything, loops through biases, and saves the data.

Explanation

Imports the input file, and declares:

Type	Name
electron	NewElectron
hole	NewHole
params	APDparams
device	APDdevice
stats	APDresults

Loads parameters into APDdevice and APDparams using LoadParams.C. Creates a float JunctionLocation1, which is calculated using FindJunction.C.

Then, loops through several biases using BiasLoop.C, and finally saves the data generated using DataToFile.C.

Uses LoadParams.C, BiasLoop.C, FindJunction.C & DataToFile.C
See Also: (none)

4.20 realft.C

Includes header file apd.h.
Inputs float data[], unsigned long n, int isign
Outputs (none)

Overview

Uses numerical recipes to find the FFT for a real function (as opposed to a complex function with four1.C).

Explanation

Computed using numerical recipes. Not especially useful to define with respect to Monte Carlo simulation.

Uses four1.C
See Also: FreqResponse.C

4.21 SetBias.C

Includes header file apd.h.
Inputs device MyDevice, float DesiredBias, float JunctionLocation1
Outputs (none)

Overview

Sets the bias for the device, then determines the electric field profile and tabulates it in the electric field table.

Explanation

While the error greater than the tolerance, works toward decreasing the error through the following procedure. The error is found by subtracting the desired bias from the calculated bias across the device. Then, the maximum electric field is recalculated, subtracting from itself the error divided by the total thickness of the device. The applied bias is recalculated accordingly, and the electric field table is rebuilt. This technique eventually converges on setting the bias for the device appropriately and rebuilding its table of electric fields.

Uses CalcBias.C, BuildEFTable.C
See Also: BiasLoop.C

4.22 StartElectron.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, electron NewElectron, hole NewHole, stats APDresults
Outputs (none)

Overview

Generates a new electron through either photo-generation or impact ionization event. Executes drift and scattering of the electron.

Explanation

Uses ElectronDrift.C, ElectronScat.C, & StartHole.C
See Also: CarrierLoop.C, ElectronScat.C, HoleScat.C, & StartElectron.C

4.23 StartHole.C

Includes header file apd.h.
Inputs device APDdevice, params APDparams, hole NewHole, electron NewElectron, stats APDresults
Outputs (none)

Overview

Generates a new electron through either photo-generation or impact ionization event. Executes drift and scattering of the electron.

Explanation

Gets a position and a time at which an electron starts to move after an event. If there is another impact ionization, it starts a new electron and hole at the new time and place. If the count exceeds a maximum value, it won't start any more carriers. If the carrier is collected, it adds to the count and updates the time-response histogram, then passes the position and time as referenced to call the function (??).
Uses HoleDrift.C, HoleScat.C, & StartElectron.C
See Also: CarrierLoop.C, ElectronScat.C, HoleScat.C, & StartElectron.C