oedes Documentation

Release 0.0.18

Marek Zdzislaw Szymanski

Apr 13, 2019

Contents:

CHAPTER 1

Introduction

`oedes` (Open Electronic DEvice Simulator) is a Python software package for modeling of electronic devices. It is primarily focused on emerging electronic devices. It is applicable to organic electronic, electrochemical, bioelectronic, and Peroskvite based devices.

oedes was written to take into account both the special aspects of non-conventional electronic devices, and modern trends in software development. The result a small but powerful package, which is tightly integrated with standard scientific software stack. This simplifies both running and sharing the simulations.

oedes mission is to:

- enable *open science* research in emerging electronic devices, by enabling full disclosure of numerical simulations;
- allow reuse of established device models, which were usually published without a runnanble implementation;
- accelerate research by providing a platform for parameter extraction from measurements and for device optimization driven by simulation;
- simplify development of new simulation models and tools, by providing a library of robust components;
- provide a standard way of running and sharing device simulations.

oedes is released as open-source under GNU Affero General Public License v3. It is free to use and distributed with complete source code.

1.1 Conventions

Code which is intended to execute in system command-line starts with `$`, for example:

```
$ python
```

All code which is indendent to be run in Python starts with >>> or `...`, for example:

```
>>> print('hello world')
```

Sequences of characters which can be copied into code in current context are emphasized as `print` or `1+2+3`. `oedes` formatted in that way refers to name of Python software package, or to Python symbol in current example.

General concepts are emphasized as *params*. *oedes* written in that way refers to the software project.

CHAPTER 2

Features

General

- Pure Python implementation
- Reliable fully implicit solver
- Flexible precision

Electric

- Cell centered finite-volume with arbitrary mesh spacing
- Scharfetter-Gummel discretization
- Conservation of charge guaranteed in a converged solution
- Arbitrary number of layers
- Ramo-Shockley calculation of terminal current
- Arbitrary temperature
- Support for multidimensional discretizations
- Arbitrary number of transported species

Optical

- Transfer matrix approach
- Arbitrary illumination

Analysis

- DC
- Transient
- AC small signal
- Sensitivity analysis

Models

- Gaussian DOS
- Extended Gaussian Disorder Model
- Generalized Einstein's relation
- Transient traps
- Shockley-Read-Hall recombination
- Langevin recombination
- Image-force barrier lowering
- Doping (also position dependent)
- Generation (also position dependent)
- Onsager-Braun model of exciton dissociation
- User supplied mobility model
- User supplied DOS

Outputs

- Current-voltage
- Current-voltage-light
- SCL
- Transient SCL
- Capacitance-voltage
- Impedance spectroscopy
- Efficiency (photovoltaic)
- Fill factor (photovoltaic)
- Dark current (photodetector)
- Responsivity (photodetector)
- Light output (light emitting diodes)

CHAPTER 3

Installation

3.1 Requirements

- Python 2.7 or Python 3.4+
- sparsegrad
- Python scientific stack

3.2 Installation from PyPI

It is recommended to use Python Package Index (PyPI) to install `oedes` package. This is done using command-line by `pip` program, which is normally installed together with Python. Using this method, all dependencies are resolved automatically.

Two variants of the installation are possible:

- system wide installation:

```
$ pip install oedes
```

- local installation not requiring administrator's rights:

```
$ pip install oedes --user
```

In the case of local installation, *oedes* is installed inside user's home directory. In Linux, this defaults to `$HOME/.local`.

3.3 Verifying the installation

After installing, it is advised to run the test suite to ensure that *oedes* works correctly on your system:

```
>>> import oedes
>>> oedes.test()
Running unit tests for oedes...
OK
<nose.result.TextTestResult run=15 errors=0 failures=0>
```

If any errors are found, *oedes* is not compatible with your system. Either your Python scientific stack is too old, or there is a bug.

oedes is evolving, and backward compatibility is not yet offered. It is recommended to check which version is in use by running:

```
>>> import oedes
>>> oedes.version
'0.0.18'
```

3.4 Upgrading/downgrading

By default, `pip` does not upgrade packages unless required. To change this behavior, option `--upgrade` should be used. For example, *oedes* is updated to the most recent version by running

```
$ pip install oedes --upgrade
```

It is also possible to upgrade/downgrade *oedes* by specifying an exact version to be installed. For example, to ensure that installed version is `0.0.18`, run

```
$ pip install oedes==0.0.18
```

Again, `--user` option can be given to restrict changes to users' home directory.

3.5 Development installation (advanced)

Current development version of sparsegrad can be installed from the development repository by running

```
$ git clone https://github.com/mzszym/oedes.git
$ cd oedes
$ pip install -e .
```

The option `-e` tells that `oedes` code should be loaded from `git` controlled directory, instead of being copied to the Python libraries directory. As with the regular installation, `--user` option should be appended for local installation.

3.6 Building documentation

Documentation of *oedes* is generated by sphinx. Its source files are placed in `doc` folder.

To produce HTML and PDF documentation, run

```
$ cd doc
$ make html
$ make latexpdf
```

CHAPTER 4

Tutorial: PN junction

In this tutorial a simple model of PN junction will be constructed. The same model is used in example included in oedes distribution in file `examples/interactive/pn.ipynb`.

`oedes` contains models of typical devices. However, in this tutorial, the model will be constructed equation-by-equation to demonstrate basic functionality.

4.1 Importing *oedes*

Before starting, `oedes` package must be imported

```
>>> import oedes
```

4.2 The drift-diffusion system

For simulating transport, a model of temperature distribution must be assumed. In the simplest case of *isothermal simulation*, the same effective temperature is assumed equal everywhere inside the device. The model is created as follows:

```
>>> temperature = oedes.models.ConstTemperature()
```

The specification of model is separated from the specification of parameter values. The actual value of temperature, in Kelvins, will be given later.

The next step is to create Poisson's equation of electrostatics

```
>>> poisson = oedes.models.PoissonEquation()
```

Constructed objects `poisson` and `temperature` must are passed as arguments when creating transport equations:

```
>>> electron = oedes.models.equations.BandTransport(poisson=poisson, name='electron',␣
↪z=-1, thermal=temperature)
>>> hole = oedes.models.equations.BandTransport(poisson=poisson, name='hole', z=1,␣
↪thermal=temperature)
```

Above creates two conventional drift-diffusion equations for electrons and holes respectively. By default, the mobility is assumed constant and the DOS is modeled by using the Boltzmann approximation. `name` arguments are names which are used to identify parameters and outputs. `z` are charges of species expressed in units of elementary charge. *oedes* allows arbitrary number of species, and arbitrary values of `name` and `z`. This allows to construct complicated models with, for example, mixed ionic-electronic transport.

4.3 The doping profile

To model the PN junction, a doping profile must be defined. In the example, left half of the device is doped with ionized donor concentration given by parameter `Nd`, and right half of device is doped with doped with ionized acceptor concentration given by parameter `Na`.

The `pn_doping` function is called during model evaluation and is given as parameters the mesh, the evaluation context object `ctx`, and the discretized equation object `eq`. In the example, it uses `ctx` object to access parameters values of the dopant concentations (`'Na'`, `''Nd'`).

```
>>> def pn_doping(mesh, ctx, eq):
...     return oedes.ad.where(mesh.x<mesh.length*0.5,ctx.param(eq, 'Nd'),- ctx.
↪param(eq, 'Na'))
>>> doping = oedes.models.FixedCharge(poisson, density=pn_doping)
```

The dopants are assumed to be fully ionized and therefore it is modeled as fixed charge. The `FixedCharge` adds calculated doping profile to the previously created Poisson's equation `poisson`.

Above code uses a specialized version of function `where` is used instead of version from `numpy`. This is required for support of sensitivity analysis with respect to parameter values.

4.4 The Ohmic contacts

To keep the example simple, Ohmic contacts are assumed on both sides of the device. They are created as follows:

```
>>> semiconductor = oedes.models.Electroneutrality([electron, hole, doping],name=
↪'semiconductor')
>>> anode = oedes.models.OhmicContact(poisson, semiconductor, 'electrode0')
>>> cathode = oedes.models.OhmicContact(poisson, semiconductor, 'electrode1')
```

Ohmic contacts require knowledge of equilibrium charge carrier concentrations in semiconductor. This is calculated by `Electroneutrality`. Note that since concentrations in *doped* semiconductor are of interest, all charged species are passed to Electroneutrality. `'electrode0'` and `'electrode1'` refers to names of boundaries in the mesh.

4.5 Putting all together

To avoid divergence of the simulation due to infinitely large lifetime of electrons and holes, recombination should be added. Duirecrecombination model is created by

```
>>> recombination = oedes.models.DirectRecombination(semiconductor)
```

The calculation of terminal current is a non-trivial post-processing step. It is recommended to use Ramo-Shockley current calculation in most cases, which is created by

```
>>> current = oedes.models.RamoShockleyCurrentCalculation([poisson])
```

The discrete model is constructed and initialized by calling `oedes.fvm.discretize`. It takes two arguments: the system of equations and terms to solve, and the specification of domain. Below `oedes.fvm.mesh1d` creates a 1-D domain with length specified as argument.

```
>>> all_equations_and_terms = [ poisson, temperature, electron, hole, doping, current,
↪ semiconductor, anode, cathode, recombination ]
>>> domain = oedes.fvm.mesh1d(100e-9)
>>> model = oedes.fvm.discretize(all_equations_and_terms, domain)
```

4.6 Parameters

The physical parameters are provided as `dict`.

```
>>> params={
...     'T':300,
...     'epsilon_r':12,
...     'Na':1e24,
...     'Nd':1e24,
...     'hole.mu':1,
...     'electron.mu':1,
...     'hole.energy':-1.1,
...     'electron.energy':0,
...     'electrode0.voltage':0,
...     'electrode1.voltage':0,
...     'hole.N0':1e27,
...     'electron.N0':1e27,
...     'beta':1e-9
... }
```

Above, `'T'` key is used to specify temperature in Kelvins. It is used by `ConstTemperature` object. `'epsilon_r'` specifies the relative dielectric permittivity. It is used by discretized `PoissonEquation` object. `'Na'` and `'Nd'` are parameters accessed by `pn_doping` function, the concentrations of dopants. `'beta'` is used by `DirectRecombination`.

Other parameters are in form `name.parameter`. `name` is passed to the equation, and they can be nested. For example, if a transport equation were created as

```
something = oedes.models.BandTransport(name='zzz',...)
```

then the corresponding mobility parameter would be identified by key `'zzz.mu'`.

The mobilities `electron.mu` and `hole.mu` are given in $\mathrm{m^2V^{-1}s^{-1}}$, therefore are equal to 1000 $\mathrm{cm^2V^{-1}s^{-1}}$ each. In the example above, instead of specifying electron affinity and band-gap, the energies of both bands are specified directly by *energy* parameters, in eV. The voltages are applied to Ohmic contacts are specified by `'electrode0.voltage'` and `'electrode1.voltage'`, in Volts. `N0` denotes the total density of states, in $\mathrm{m^{-3}}$.

4.6.1 params

By convention, values of physical parameters are specified in `dict` object named `params`, with string keys, and float values. All values of given in SI base units, except for small energies which are specified in eV. *oedes* currently does not assume default values of parameters. If any necessary parameter is not specified in `params`, exception `KeyError` is raised.

4.7 Solving

`oedes.context` objects binds models with their parameters and solutions. It also provides convenience functions for solving, post-processing and plotting the data.

The following calculates soltuion for parameters specified in dict `params`

```
>>> c = oedes.context(model)
>>> c.solve(params)
```

4.8 Examining output

The solution can be investigated by calling *output* function, which returns a dict of available outputs:

```
>>> out=c.output()
>>> print(sorted(out.keys()))
['.meta', 'D', 'Dt', 'E', 'Et', 'J', 'R', 'c', 'charge', 'electrode0.J', 'electrode1.J
↪', 'electron.Eband', 'electron.Ef', 'electron.J', 'electron.Jdiff', 'electron.Jdrift
↪', 'electron.c', 'electron.charge', 'electron.ct', 'electron.j', 'electron.jdiff',
↪'electron.jdrift', 'electron.phi_band', 'electron.phi_f', 'hole.Eband', 'hole.Ef',
↪'hole.J', 'hole.Jdiff', 'hole.Jdrift', 'hole.c', 'hole.charge', 'hole.ct', 'hole.j',
↪ 'hole.jdiff', 'hole.jdrift', 'hole.phi_band', 'hole.phi_f', 'potential',
↪'semiconductor.Ef', 'semiconductor.electron.c', 'semiconductor.hole.c',
↪'semiconductor.phi', 'total_charge_density']
```

The outputs are *numpy* arrays. For example, the electrostatic potential is

```
>>> print(out['potential'])
[-0.17857923 -0.17858006 -0.17858115 -0.17858258 -0.17858445 -0.17858693
  -0.17859023 -0.17859468 -0.1785994  -0.17860442 -0.17860982 -0.17861567
  ...
  -0.92141696 -0.92141772]
```

To access additional information about output (such as its mesh), use `.meta` subdictionary.

```
>>> out['.meta']['potential']
OutputMeta(mesh=<oedes.fvm.mesh.mesh1d object at ...>, face=False, unit='V')
```

4.8.1 outputs

Most useful outputs are given below. Just as for *params*, all values are in SI base units, except for small energies in eV. * denotes prefix identifying the equation, such as `electron` or `hole`.

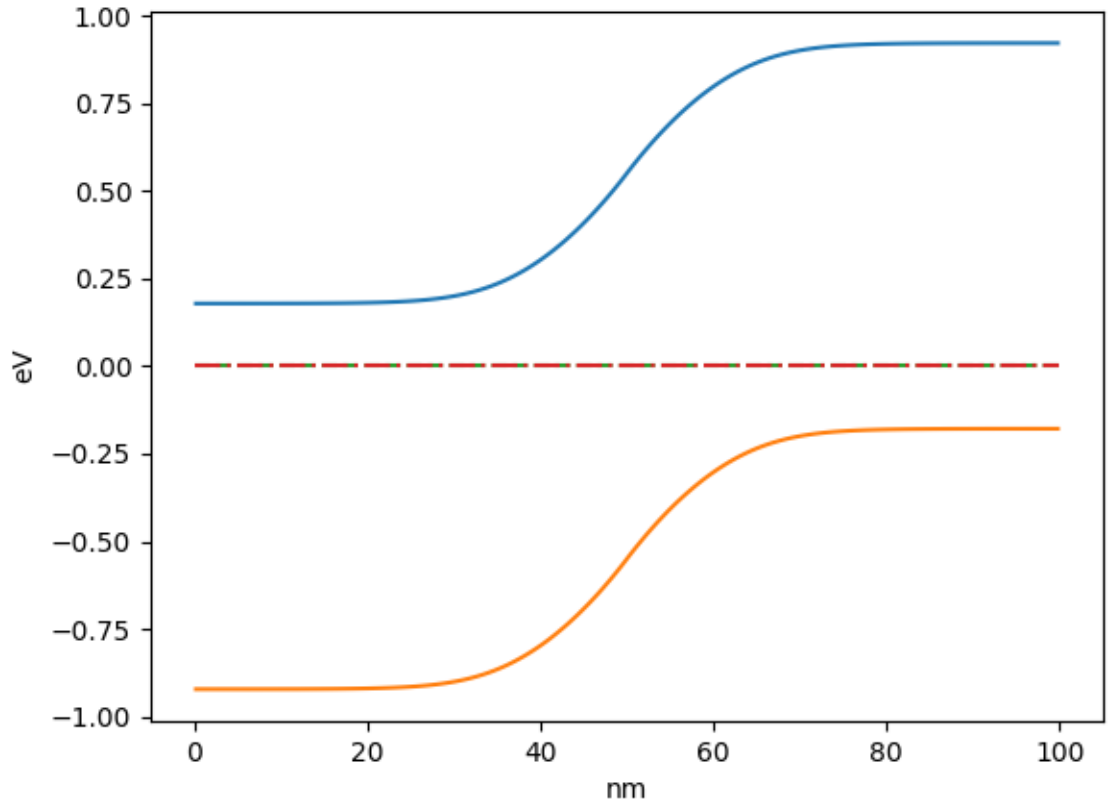
- `*.c`: concentration of particles, in m^{-3}
- `*.j`: flux of particles, in $\mathrm{m}^{-2}\mathrm{s}^{-1}$

- `*.Ef`: quasi Fermi level, in eV
- `*.Eband`: band energy, in eV
- `R`: recombination density, in $\mathrm{m}^{-3}\mathrm{s}^{-1}$
- `J`: total electric current density, in Am^{-2}
- `E`: electric field, in V/m
- `potential`: electrostatic potential, V

4.9 Plotting

`oedes.context` object simplifies plotting results using `matplotlib`. For example, bands and quasi Fermi levels are plotted as

```
>>> import matplotlib.pylab as plt
>>> fig,ax = plt.subplots()
>>> p=c.mpl(fig, ax)
>>> p.plot(['electron.Eband'],label='$E_c$')
>>> p.plot(['hole.Eband'],label='$E_v$')
>>> p.plot(['electron.Ef'],linestyle='--',label='$E_{Fn}$')
>>> p.plot(['hole.Ef'],linestyle='-.',label='$E_{Fp}$')
>>> p.apply_settings({'xunit':'n','xlabel':'nm'})
```



CHAPTER 5

Physical models

5.1 Concentrations of charges in thermal equilibrium

Probability f_{FD} that an electronic state with energy E is occupied is given by the Fermi-Dirac distribution:

$$f_{FD}\left(E - E_F\right) = \frac{1}{1 + \exp\frac{E - E_F}{k_B T}}$$

with E_F denoting the Fermi energy, k_B denoting the Boltzmann constant and T denoting the temperature.

The states in the conduction band are distributed in energy according to the density of states function $DOS_n\left(\epsilon = E - E_c\right)$. The total concentration n of electrons is given by integral

$$n\left(E_F\right) = \int_{-\infty}^{+\infty} DOS_n\left(E - E_c\right) f_{FD}(E - E_F) dE$$

In the valence band, almost all states are occupied by the electrons. It is therefore useful to track unoccupied states, *holes*, instead of the occupied states. The concentration of holes is given by:

$$p\left(E_F\right) = \int_{-\infty}^{+\infty} DOS_p\left(E_v - E\right)\left[1 - f_{FD}(E - E_F)\right] dE$$

Noting that

$$1 - f_{FD}(E - E_F) = f_{FD}(E_F - E)$$

and changing the integration variable, both concentrations can be written in a common form as:

$$\begin{aligned} c_i\left(\eta\right) &= \int_{-\infty}^{+\infty} DOS_i\left(\epsilon\right) f_{FD}\left(\epsilon - \eta\right) d\epsilon \\ n &= c_n\left(\eta = E_F - E_c\right) \\ p &= c_p\left(\eta = E_v - E_F\right) \end{aligned} \tag{5.1}$$

Practical note: The SI base unit of energy is Joule (J), however the energies such as E_F are very small and should be expressed in electronvolts (eV). The SI basic unit of concentration is meter^{-3}, although cm^{-3} is often encountered. The value of $k_B T$ at the room temperature is approximately 26 meV. The SI unit of temperature is Kelvin, $27^{\circ}\mathrm{C} \approx 300\mathrm{K}$.

5.2 Band energies

In an idealized case, the energies E_c and E_v of the conduction and valence bands are

$$\begin{aligned} E_c &= -q\psi - \chi \\ E_v &= E_c - E_g \end{aligned} \tag{5.2}$$

where $\chi > 0$ is the electron affinity energy, and $E_g > 0$ is the bandgap energy. ψ is the electrostatic potential. q is the elementary charge.

5.3 Electrostatic potential

The electric field $\mathbf{E}$ is related to the elestrostatic potential ψ as

$$\mathbf{E} = -\nabla\psi \tag{5.3}$$

In linear, isotropic, homogeneous medium the electric displacement field is

$$\mathbf{D} = \varepsilon\mathbf{E}$$

with permittivity

$$\varepsilon = \varepsilon_0\varepsilon_r$$

where ε_0 is the vacuum permittivity, and ε_r is the relative permittivity of the material.

The electric displacement field satisfies the electric the Gauss's equation

$$\nabla \cdot \mathbf{D} = \rho_f \tag{5.4}$$

where ρ_f is the density of free charge

$$\rho_f = q\left(p - n + \dots\right)$$

with q denoting the elementary charge. Above, $\dots$ denotes other charges, such as ionized dopants.

Combining the above equations gives the usual Poisson's equation for electrostatics:

$$\nabla^2\psi = -\frac{q}{\varepsilon}\left(p - n + \dots\right) \tag{5.5}$$

The SI unit of electrostatic potential is Volt, and the unit of electric field is Volt/meter. The unit of permittivity is Farad/meter. The unit of charge density is $\mathrm{Coulomb}/\mathrm{meter}^3$.

5.4 Approximation for low concentrations

If the concentration of charge carriers is low enough, only states on the edge of band gap are important. In such case, the density of states can be assumed as a sharp energetic level,

$$DOS_n\left(\epsilon\right) = N_c\delta\left(\epsilon\right)$$

in case of electrons and

$$DOS_p\left(\epsilon\right) = N_v\delta\left(\epsilon\right)$$

in case of holes. Substiting into (5.1) gives

$$n = N_c \, f_{FD} \left(E_c - E_F \right)$$
$$p = N_v \, f_{FD} \left(E_F - E_v \right)$$

At low charge carrier concentrations, Fermi-Dirac distribution f_{FD} is simplified as

$$f_{FD}(x) = \frac{1}{1 + \exp \frac{x}{k_B T}} \approx \exp - \frac{x}{k_B T}$$

The approximation is considered valid when $x > 4 k_B T$.

Approximate charge carrier concentrations are

$$\begin{aligned} n &= N_c \exp \frac{E_F - E_c}{k_B T} \\ p &= N_v \exp \frac{E_v - E_F}{k_B T} \end{aligned} \tag{5.6}$$

5.5 Gaussian density of states

In the case of Gaussian DOS, the density of states shape function DOS_i is the Gaussian distribution function scaled by total density of states N_i:

$$DOS_i \left(\epsilon \right) = N_i \frac{1}{\sqrt{2 \sigma^2 \pi}} \exp \frac{-\epsilon^2}{2 \sigma^2}$$

Concentrations of species are given by integral

$$c_i \left(\eta \right) = N_i \frac{1}{\sqrt{2 \sigma^2 \pi}} \int_{-\infty}^{+\infty} \exp \frac{-\epsilon^2}{2 \sigma^2} f_{FD}(\epsilon - \eta) d\epsilon$$

5.6 Conservation equation

The *conservation equation* is:

$$\frac{\partial c_i}{\partial t} + \nabla \cdot \mathbf{j_i} = S_i$$

where c_i denotes the concentration, t is time, and j_i is the *flux density*. S_i denotes source term, which is positive for generating particles, and negative for sinking particles of type *i*. The SI unit of source term is $1/\left(\mathrm{meter}^3 \mathrm{second}\right)$.

The conservation equation must be satisfied for each species separately. In the case of transport of electrons and holes, this gives

$$\begin{aligned} \frac{\partial n}{\partial t} + \nabla \cdot \mathbf{j_n} &= S_n \\ \frac{\partial p}{\partial t} + \nabla \cdot \mathbf{j_p} &= S_p \end{aligned} \tag{5.7}$$

where the source *S* term contains for example generation G and recombination R terms

$$S_{n,p} = G - R$$

The conservation of electric charge must be satisfied everywhere. Therefore, the source terms acting at given point must not create a net electric charge. In the case of system of electron and holes, this requires

$$S_n = S_p$$

5.7 Current density

Current density $\mathbf{J_i}$ is related to the density flux $\mathbf{j_i}$ by the *charge of single particle* $z_i q$. Obviously, for electrons $z = -1$ and for holes $z = 1$, therefore

$$\begin{aligned} \mathbf{J_n} &= -q\mathbf{j_n} \\ \mathbf{J_p} &= q\mathbf{j_p} \end{aligned} \tag{5.8}$$

Note that a convention is adopted to denote the electric current with uppercase letter $\mathbf{J}$, and the flux density with lowercase letter $\mathbf{j}$. The SI unit of density flux $\mathbf{j_i}$ is $1/(\mathrm{meter}^2\,\mathrm{second})$, while the unit of electric current density $\mathbf{J_i}$ is $\mathrm{Amper}/\mathrm{meter}^2$.

5.8 Equilibrium conditions

In the equilibrium conditions, Fermi level energy E_F has the same value everywhere. The electrostatic potential ψ can vary, and the density of free charge ρ_f does not need to be zero. Equations (5.1), (5.2), (5.4) are satisfied simultaneously. The current flux, the source terms, and the time dependence are all zeros, so conservation (5.7) is trivially satisfied.

5.9 Nonequilibrium conditions

In the non-equilibrium conditions, the transport is introduced as a perturbation from equilibrium. The Fermi energy level is replaced with *quasi Fermi level*, which is different for each species. In (5.1), the equilibrium Fermi level for electrons E_F is replaced with a quasi Fermi level E_{Fn}. Similarly,, the equilibrium Fermi level for holes is replaced wuth quasi Fermi level for holes E_{Fp}, giving

$$\begin{aligned} n &= c_n \left(E_{Fn} - E_c\right) \\ p &= c_p \left(E_v - E_{Fp}\right) \end{aligned} \tag{5.9}$$

Quasi Fermi levels have associated quasi Fermi potential according to the formula for energy of an electron in electrostatic field $E_F = -q\phi$:

$$\begin{aligned} E_{Fn} &= -q\phi_n \\ E_{Fp} &= -q\phi_v \end{aligned} \tag{5.10}$$

The transport is modeled by approximating electric current density as

$$\begin{aligned} \mathbf{J_n} &= \mu_n n \nabla E_{Fn} \\ \mathbf{J_p} &= \mu_p p \nabla E_{Fp} \end{aligned} \tag{5.11}$$

where μ denotes the respective mobilities. The SI unit of mobility is $\mathrm{meter}^2/(\mathrm{Volt}\,\mathrm{second})$, although $\mathrm{cm}^2/(\mathrm{V\,s})$ is often used.

Equations (5.2), (5.5), (5.9), (5.11), (5.7) are simultaneously satisfied in non-equilibrium conditions.

5.10 Drift-diffusion system

Standard form of density fluxes in the drift-diffusion system is

$$\begin{aligned} \mathbf{j_n} &= -\mu_n n \mathbf{E} - D_n \nabla n \\ \mathbf{j_p} &= \mu_p p \mathbf{E} - D_p \nabla p \end{aligned} \tag{5.12}$$

or more generally, allowing arbitrary charge $z_i q$ per particle

$$\mathbf{j_i} = z_i \mu_i c_i \mathbf{E} - D_i \nabla c_i \tag{5.13}$$

D_i is the *diffusion coefficient*, with SI unit $\mathrm{meter}^2\mathrm{second}^{-1}$.

5.11 Drift-diffusion system: low concentration limit

To obtain the conventional drift-diffusion formulation (5.12), the the low concentration approximation (5.6) should be used. After introducing quasi Fermi levels, as it is done in (5.9), one obtains

$$n = N_c \exp \frac{E_{Fn} - E_c}{k_B T}$$
$$p = N_v \exp \frac{E_v - E_{Fp}}{k_B T}$$

From that, the quasi Fermi energies are calculated as

$$E_{Fn} = E_c + k_B T \log \frac{n}{N_c}$$
$$E_{Fp} = E_v - k_B T \log \frac{p}{N_v}$$

Using (5.2), and assuming constant ionization potential $\nabla \chi = 0$, bandgap $\nabla E_g = 0$, constant total densities of states $\nabla N_c = \nabla N_V = 0$, and constant temperature $\nabla T = 0$, substituting into (5.11), and using (5.3)

$$\mathbf{J_n} = q\mu_n n \mathbf{E} + \mu_n k_B T \nabla n$$
$$\mathbf{J_p} = q\mu_p p \mathbf{E} - \mu_p k_B T \nabla p$$

In terms of density flux (5.8), this reads

$$\begin{aligned} \mathbf{j_n} &= -\mu_n n \mathbf{E} - \mu_n V_T \nabla n \\ \mathbf{j_p} &= \mu_n n \mathbf{E} - \mu_p V_T \nabla p \end{aligned} \tag{5.14}$$

where *thermal voltage*

$$V_T = \frac{k_B T}{q}$$

5.12 Einstein's relation

Equation (5.14) is written in the standard drift-diffusion form (5.12) when the diffusion coefficient satisfies

$$\frac{D_{n,p}}{\mu_{n,p}} = V_T \tag{5.15}$$

This is called *Einstein's relation*.

5.13 Drift-diffusion system: general case

Using functions defined in (5.1), bands (5.2) and approximation (5.9)

$$E_{Fn} = -q\phi - \chi + c_n^{-1}(n)$$
$$E_{Fp} = -q\phi - \chi - E_g - c_p^{-1}(p)$$

current densities under assumptions $\nabla\chi = \nabla E_g = 0$ are

$$\begin{aligned} \mathbf{J_n} &= \mu_n n\mathbf{E} + \mu_n n\nabla c_n^{-1}(n) \\ \mathbf{J_p} &= \mu_p p\mathbf{E} - \mu_p p\nabla c_p^{-1}(p) \end{aligned} \tag{5.16}$$

5.14 Generalized Einstein's relation

In equation (5.16), assuming $\nabla T = \nabla N_c = \nabla N_v = 0$

$$\begin{aligned} n\nabla c_n^{-1}(n) &= \frac{n}{\frac{\partial c_n}{\partial \eta_n}}\nabla n \\ p\nabla c_p^{-1}(p) &= \frac{p}{\frac{\partial c_p}{\partial \eta_p}}\nabla p \end{aligned}$$

In order to express equation (5.16) in the standard drift-diffusion form (5.12), the diffusion coefficient must satisfy

$$\frac{D_i}{\mu_i} = \frac{1}{q}\frac{c_i}{\frac{\partial c_i}{\partial \eta_i}}$$

This is so called *generalized Einstein's relation* .

5.15 Intrinsic concentrations

Intrinsic concentrations n_i, p_i, and intrinsic Fermi level E_{Fi} satisfy electric neutrality conditions

$$\begin{aligned} n_i &= c_n(E_{Fi} - E_c) \\ p_i &= c_p(E_v - E_{Fi}) \\ n_i &= p_i \end{aligned}$$

5.16 Direct recombination

Direct recombination introduces source term

$$R = \beta(np - n_i p_i)$$

where β can be chosen freely.

5.17 Unidimensional form

By substituting $\nabla \to \frac{\partial}{\partial x}$ and $\nabla^2 \to \frac{\partial^2}{\partial x^2}$, the equations (5.5), (5.7), (5.12) of the basic drift-diffusion device model are

$$\begin{aligned}
\frac{\partial^2 \psi}{\partial x^2} &= -\frac{q}{\varepsilon}\left(p - n + \dots\right) \\
E &= -\frac{\partial \psi}{\partial x} \\
j_n &= -\mu_n n E - D_n \nabla \frac{\partial n}{\partial x} \\
j_p &= \mu_p p E - D_p \nabla \frac{\partial p}{\partial x} \\
\frac{\partial n}{\partial t} + \frac{\partial j_n}{\partial x} &= G - R \\
\frac{\partial p}{\partial t} + \frac{\partial j_p}{\partial x} &= G - R
\end{aligned}$$

5.18 Total electric current density

Total electric current $\mathbf{J}$ is a sum of currents due to transport of each species and the displacement current $\mathbf{J_d}$

$$\begin{aligned}
\mathbf{J} &= \mathbf{J_n} + \mathbf{J_p} + \mathbf{J_d} + \dots \\
\mathbf{J_d} &= \frac{\partial \mathbf{D}}{\partial t}
\end{aligned}$$

Total electric current satisfies the conservation law

$$\nabla \cdot \mathbf{J} = 0$$

This can be verified by taking time derivative (5.4), using (5.7) and considering that the sum of all charge created by the source terms must be zero.

5.19 Electrode current

Current I_α passing through a surface Γ_α of electrode α is

$$I_\alpha = \int_{\Gamma_\alpha} \boldsymbol{J} \cdot d\mathbf{S} \tag{5.17}$$

5.20 Metal

In metal, the relation between the electrostatic potential ψ, the workfunction energy $W_F > 0$ and the Fermi level E_F is

$$E_F = -q\psi - W_F$$

On the other hand, the Fermi potential corresponds to the applied voltage V_{appl}

$$E_F = -qV_{appl}$$

This leads to electrostatic potential at metal surface

$$\psi = V_{appl} - W_F/q$$

5.21 Ohmic contact

Ohmic contact is an idealization assuming that there is no charge accumulation at the contact, and the applied voltage V_{appl} is equal to quasi Fermi potentials (5.10) of charged species

$$\begin{aligned} \phi_n &= V_{appl} \\ \phi_p &= V_{appl} \\ n + N_A^- &= p + N_D^+ \end{aligned}$$

Above three conditions uniquely determine the charge concentrations n, p, and the electrostatic potential ψ at the contact.

5.22 Electrochemical transport

Electrochemical potential for ionic species is

$$\mu_i^{el} = z_i q\psi + k_B T \log c_i + \dots$$

It should be noticed that so defined "potential" has the unit of energy, unlike the electrostatic potential and quasi Fermi potentials. Above $\dots$ denote corrections, for example due to steric interactions. Electrochemical potential μ_i^{el} should not be confused with mobility μ_i.

Density flux is approximated as

$$\mathbf{j_i} = -\frac{1}{q}\mu_i c_i \nabla \mu_i^{el}$$

yielding the standard form (5.13) using Einstein's relation (5.15).

Electrochemical species should be included in Poisson's equation, by including proper source terms of form qz_ic_i. A variant of Poisson's equation (5.5) where are free charges are ions can be written as

$$\nabla^2 \psi = -\frac{q}{\varepsilon} \sum z_i c_i$$

5.23 Steric corrections

To account for finite size of ions, the electrochemical potential in the form introduced in [LE13] is useful

$$\mu_i^{el} = z_i q\psi + k_B T \log \frac{v_i c_i}{\Gamma}$$

where v_i denotes volume of particle of type i. Γ is the unoccupied fraction of space

$$\Gamma = 1 - \sum_k v_k c_k$$

where summing is taken over all species occupying space, including solvent.

CHAPTER 6

Optical models

6.1 Photon energy

$$E_p = h\nu$$

with h Planck constant and ν is photon frequency.

6.2 Coherent transfer matrix method

Transfer matrix method a convenient way of modeling thin film stacks. It is assumed that layers are stacked along x axis, with $x_{i,i+1}$ being interface between layer i and layer $i+1$. Optical properties of each layer are specified by wavelength dependent complex refraction coefficient $\tilde{n}_i(\lambda)$.

Optical field inside layer i at given point along x axis is specified by column vector $\left[E_i^+(x), E_i^-(x)\right]^T$, with $E_i^+(x)$ being complex amplitude of forward traveling wave, and $E_i^-(x)$ being complex amplitude of backward traveling wave.

Snell law is determines angles of propagation in each layer

$$n_0 \sin\theta_0 = n_i \sin\theta_i$$

where index 0 refers to medium before first layer. θ_0 is angle of illuminating wave. All angles can be complex numbers. Since $\arcsin$ is multivalued function, angle of forward traveling wave is found from conditions that forward wave has forward pointing Poynting vector, or alternatively, that the amplitude of forward wave decays in absorbing medium.

In this convention, interface between layers is described by matrix $\mathbf{M}$ as

$$\begin{bmatrix} E_i^+(x) \\ E_i^-(x) \end{bmatrix} = \mathbf{M_{i,i+1}} \begin{bmatrix} E_{i+1}^+(x) \\ E_{i+1}^-(x) \end{bmatrix}$$

with entries of matrix $\mathbf{M}$ specified as

$$\mathbf{M_{i,i+1}} = \frac{1}{t_{i,i+1}} \begin{bmatrix} 1 & r_{i,i+1} \\ r_{i,i+1} & 1 \end{bmatrix}$$

where transmission coefficient $t_{i,i+1}$ and reflection coefficient $r_{i,i+1}$ are given by Fresnel equations for complex amplitudes of light passing from layer i to layer i+1. Coefficients for backward propagating wave $t_{i+1,i}$ and $r_{i+1,i}$ are eliminated using Stokes relations.

For s-polarized wave:

$$r_{i,i+1} = \frac{n_i \cos\theta_i - n_{i+1} \cos\theta_{i+1}}{n_i \cos\theta_i + n_{i+1} \cos\theta_{i+1}}$$
$$t_{i,i+1} = \frac{2 n_i \cos\theta_i}{n_i \cos\theta_i + n_{i+1} \cos\theta_{i+1}}$$

For p-polarized wave:

$$r_{i,i+1} = \frac{n_{i+1} \cos\theta_i - n_i \cos\theta_{i+1}}{n_{i+1} \cos\theta_i + n_i \cos\theta_{i+1}}$$
$$t_{i,i+1} = \frac{2 n_i \cos\theta_i}{n_{i+1} \cos\theta_i + n_i \cos\theta_{i+1}}$$

Propagation inside layer is described by matrix $\mathbf{P}$ as

$$\begin{bmatrix} E_i^+(x) \\ E_i^-(x) \end{bmatrix} = \mathbf{P_i}(\mathbf{x}) \begin{bmatrix} E_i^+(x = x_{i,i+1}) \\ E_i^-(x = x_{i,i+1}) \end{bmatrix}$$

$$\mathbf{P_i}(\mathbf{x}) = \begin{bmatrix} \exp -i\delta_i(x) & 0 \\ 0 & \exp i\delta_i(x) \end{bmatrix}$$

with phase-shift

$$\delta_i(x) = \left(\frac{2\pi}{\lambda_0} \tilde{n}_i \cos\theta_i\right)(x_{i,i+1} - x)$$

Light entering layer i, on side of layer $i-1$ has vector of complex amplitudes

$$\mathbf{v_k}\left(x = x_{k-1,k}\right) = \left(\Pi_{k \le i \le n} \mathbf{M_{i-1,i}} \mathbf{P_i}(x_{i-1,i})\right) \mathbf{M_{n,n+1}} \begin{bmatrix} t \\ 0 \end{bmatrix}$$

with vector $[t, 0]$ denoting light leaving the device on the side opposite to illumination, with t being complex amplitude of transmitted wave.

Applying above to whole device gives

$$\begin{bmatrix} 1 \\ r \end{bmatrix} = \mathbf{v_1}$$

with amplitude of illuminating wave set arbitrarily to 1 and r being complex amplitude of reflected wave.

When analyzing stack, firstly, solution t, r is found. Then intensity of light anywhere inside the device is calculated using found vectors $\mathbf{v_i}$ and propagation matrices $\mathbf{P_i}$. Total intensity is found by applying Poynting formula. Absorbed energy is found by differentiating with respect to x.

6.3 Incoherent light

Incoherent light is described by spectrum $S(\lambda)$. Absorption of incoherent light is calculated as

$$A = \int A_{\text{coherent}}(\lambda)\, S(\lambda)\, d\lambda$$

where $A_{\text{coherent}}(\lambda)$ is calculated using coherent transfer matrix method.

CHAPTER 7

Examples

oedes comes with examples demonstrating typical use. Each example shows usage of oedes and is used as a test suite to check for correctness. Many examples additionally partially or completely reproduce published papers. They therefore allow to experiment with published models.

The examples provided as files with extension `.ipynb` run in Jupyter Notebook. Before running the examples, *oedes* should be installed.

The examples are included in *oedes* source distribution, and can also be downloaded from oedes repository on github. Also, they can be browsed on *oedes* website.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Bibliography

[LE13] Jinn-Liang Liu and Bob Eisenberg. Correlated Ions in a Calcium Channel Model: A Poisson–Fermi Theory. *The Journal of Physical Chemistry B*, 117(40):12051–12058, 2013. PMID: 24024558. URL: https://doi.org/10.1021/jp408330f, arXiv:https://doi.org/10.1021/jp408330f, doi:10.1021/jp408330f.