Sesame: a 2-dimensional solar cell modeling tool

Benoit Gaury
Center for Nanoscale Science and Technology, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA and Maryland NanoCenter, University of Maryland, College Park, MD 20742, USA

Yubo Sun and Peter Bermel
School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA

Paul M. Haney
Center for Nanoscale Science and Technology, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA

(Dated: August 6, 2019)

This work introduces a new software package "Sesame" for the numerical computation of classical semiconductor equations. It supports 1 and 2-dimensional systems and provides tools to easily implement extended defects such as grain boundaries or sample surfaces. Sesame is designed to facilitate fast exploration of the system parameter space and to visualize local charge transport properties. Sesame has been benchmarked against other software packages, and results for single crystal and polycrystalline CdS-CdTe heterojunctions are presented. Sesame is distributed as a Python package or as a standalone GUI application, and is available at `https://pages.nist.gov/sesame/`.

I. INTRODUCTION

Numerical simulations are an essential aspect of photovoltaic research and design. A number of free software packages have been developed and extensively used for solar cell modeling in 1-dimension, including AMPS [1], PC-1D [2], SCAPS [3], and wxAMPS [4]. Freely available 2-dimensional simulation tools are less common [5–7], but are necessary for describing systems with lateral inhomogeneity. A common class of such systems are polycrystalline thin film photovoltaics, such as CdTe [8], CIGS [9], and hybrid perovskites [10]. In these materials grain boundaries break the lateral symmetry of the p-n junction, leading to complex system geometries. Lateral inhomogeneity is also often encountered in nanoscale or mesoscopic measurements. The resolution of these measurements is typically achieved by using an excitation source or measurement probe with nanoscale spatial extent. Examples include electron beam induced current (EBIC) or scanning Kelvin probe microscopy, which are also often surface sensitive. An appropriate model for these measurements is therefore (at least) 2-dimensional and includes localized excitation/detection sources and relevant boundary conditions.

There are numerous examples of 2-dimensional solar cell modeling in the literature. For instance, the impact of grain boundaries in polycrystalline cells has been previously studied numerically [11–13] and analytically [14, 15]. Simulations have been used for interpreting experiments with localized excitations such as EBIC [16, 17], cathodoloLuminescence [18, 19], and two-photon photoluminescence [20]. Although these works are instructive, nonlinearities in the system response prevent a simple extrapolation of previous results to all possible system configurations of interest. Indeed there remain a number of unresolved questions of fundamental interest in polycrystalline photovoltaics, questions as basic as whether grain boundaries are harmful or beneficial to cell performance [8, 21]. It is therefore desirable for researchers to have widespread access to 2-d simulation software.

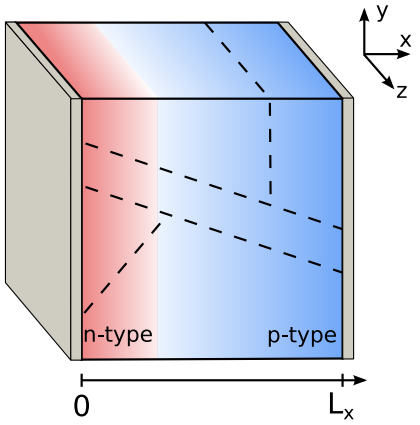

FIG. 1. Coordinate system of Sesame: rectilinear geometry and contacts located at $x = 0$ and $x = L$. pn junction doping is shown as an example.

In this work we introduce Sesame, a Python package developed by the authors (B. G. and P. M. H.) which solves the drift-diffusion-Poisson equations in 1 and 2 dimensions. Sesame is open source and distributed under the BSD license. Sesame is designed to easily construct systems with planar defects, such as grain boundaries or sample surfaces, which may contain both discrete or a continuum of gap state defects. While full-featured commercial packages allow simulations of complex device configurations together with multiple physical effects, the needs of research sometimes require access to the source code and licensing that enables usage on computing clus-

ters. Access to source code enables researchers to modify the program to suit their needs. The free licensing and Python implementation of Sesame allows batch processing of an arbitrary number of simultaneous simulations (limited only by cluster access). This provides the capacity for performing large scale parameter sweeps in multiple dimensions. The program and its code are publicly available at `https://pages.nist.gov/sesame/`.

The paper is organized as follows: In Sec. II we present a brief overview of the model and geometry. In Sec. III we compare the output of Sesame to established semiconductor modeling software, including SCAPS [3], Sentaurus [22], and COMSOL Semiconductor Module [23] [24]. In Sec. III we also present a hands-on tutorial script for solving a 2-dimensional system with a grain boundary, and briefly describe the functionality of the GUI. The mathematics underlying the model and technical details of the numerical implementation can be found in the Appendix.

II. OVERVIEW OF THE PHYSICAL MODEL

The system geometry consists of a semiconductor device connected to contacts at $x = 0$ and $x = L$ (see Fig. 1). Sesame describes the steady state behavior of this system, which is governed by the drift-diffusion-Poisson equations:

$$\vec{\nabla} \cdot \vec{J}_n = -q(G - R) \tag{1}$$

$$\vec{\nabla} \cdot \vec{J}_p = q(G - R) \tag{2}$$

$$\vec{\nabla} \cdot \left(\epsilon \vec{\nabla} \phi\right) = -\rho/\epsilon_0 \tag{3}$$

with the currents

$$\vec{J}_n = -q\mu_n n \vec{\nabla}\phi + qD_n \vec{\nabla} n \tag{4}$$

$$\vec{J}_p = -q\mu_p p \vec{\nabla}\phi - qD_p \vec{\nabla} p \ , \tag{5}$$

where n and p are the respective electron and hole number densities, and ϕ is the electrostatic potential. $\vec{J}_{n(p)}$ is the charge current density of electrons (holes). Here, q is the absolute value of the electron charge. ρ is the local charge density, ϵ is the dielectric constant of the material, and ϵ_0 is the permittivity of free space. $\mu_{n,p}$ is the electron/hole mobility, and is assumed to satisfy the Einstein relation: $D_{n,p} = k_B T \mu_{n,p}/q$. G is the electron/hole pair generation rate density and R is the recombination rate density.

Sesame includes Schockley-Read-Hall, radiative, and Auger recombination mechanisms. Sesame is currently limited to describing non-degenerate semiconductors with Boltzmann statistics, and does not include thermionic emission and quantum tunneling at interfaces. These can be important contributions to the transport in heterojunctions [25], so care should be exercised when using Sesame to simulate such systems. Sesame includes Ohmic and Schottky contact boundary conditions, and periodic or hardwall (infinite potential) transverse boundary conditions. See Table I for a list of software capabilities and data output options. Note that Sesame currently does not include a solver for Maxwell's equations. Sesame uses finite differences to solve Eqs. (1-3), and the standard Scharfetter-Gummel scheme for discretizing the current [26]. Details of the implementation can be found in the Appendix.

Model features

Heterojunction	✓	Schottky contact	✓
Radiative/Auger recombination	✓	Schockley-Read-Hall recombination	✓
Graded material parameters	✓	Planar defects	✓
Thermionic emission	✗	Tunneling	✗
Boltzmann statistics	✓	Fermi-Dirac statatistics	✗
Time-dependent	✗	Scripting/batch processing	✓

Model output

Spatially resolved electron/hole density	✓
Spatially resolved electron/hole current	✓
Surface (grain boundary) recombination	✓
Total recombination (defect, radiative, and Auger)	✓
$J - V$ (Dark and light conditions)	✓

TABLE I. List of features of Sesame (✓ indicates feature is available, ✗ indicates a feature is not available).

III. BENCHMARKS AND EXAMPLES

A. Benchmarks

We first verify the consistency between Sesame and other software packages. We have compared the output of Sesame with the well-established software packages Sentaurus, COMSOL, and SCAPS for many systems, and present two illustrative examples here. We first consider a 1-d heterojunction consisting of a thin n^+-doped layer of CdS and a p-type CdTe. The material parameters are shown in Table II. Fig 2(a) shows the computed J-V curve under a uniform generation rate density of $G = 3.3 \times 10^{20}$ cm^{-3} s^{-1}. We find close agreement between Sesame, Sentaurus, and COMSOL. To quantify the comparison, we define the relative difference between two computed currents J_1 and J_2 as $|J_1 - J_2| / \langle J_1 + J_2 \rangle$, where $\langle\rangle$ denotes the average. The maximum relative difference between Sesame and Sentaurus is 0.2 %, and between Sesame and COMSOL it is 2 %. We observe a more substantial difference between Sesame and SCAPS, with a maximum value of 7 %. In all cases, the maximum discrepancy occurs near $V_{\rm oc}$, where the current is minimized

Param.	CdS	CdTe
L [nm]	25	4000
ϵ	10	9.4
τ_n [ns]	10	5
τ_p [ns]	10^{-4}	5
N_C [cm^{-3}]	2.2×10^{18}	8×10^{17}
N_V [cm^{-3}]	1.8×10^{19}	1.8×10^{19}
E_g [eV]	2.4	1.5
χ [eV]	4.0	3.9
μ_n [cm^2/(V $\cdot$ s)]	100	320
μ_p [cm^2/(V $\cdot$ s)]	25	40
doping [cm^{-3}]	10^{17} (D)	10^{15} (A)

TABLE II. List of bulk parameters used for the 1-d and 2-d simulations. The label (D) and (A) for the doping value indicate donor and acceptor, respectively.

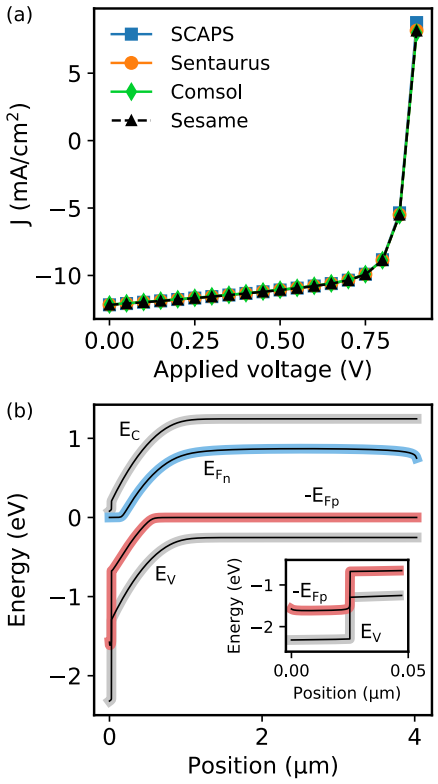

FIG. 2. Comparison between Sesame and SCAPS, Sentaurus, and COMSOL for a CdS-CdTe heterojunction. (a) Illuminated J-V curve. (b) Band diagram under short circuit conditions. Inset shows the valence band and hole quasi-Fermi level near the CdS layer. Black thin lines are the Sentaurus results, and thick colored lines are Sesame results.

so that *relative* differences are maximized. We attribute the larger difference between Sesame and SCAPS to the different interface recombination model used in SCAPS, in which the system variables are multi-valued at the interface and allow for recombination between layers [3].

We next consider a 2-d system comprised of the same bulk materials, with a vertical grain boundary in the CdTe layer (see inset of Fig. 3(a)). The system width is 3 μm and we use hard wall boundary conditions for the vertical system edges. The grain boundary is positioned in the middle of the CdTe layer, and terminates a distance 0.1 μm away from the p-type contact, and 25 nm away from the CdS interface. The grain boundary contains a discrete donor and acceptor defect, both positioned at 0.4 eV above midgap, with defect density $\rho_{GB} = 10^{14}$ cm^{-2} and equal hole and electron capture cross section $\sigma_{\mathrm{GB}} = 10^{-14}$ cm^2. For this simulation we again use a uniform generation rate $G = 3.3 \times 10^{21}$ cm^{-3} s^{-1}. Fig. 3(a) shows the illuminated J-V curve obtained with Sesame, COMSOL, and Sentaurus (SCAPS is not included, as it does not support 2-d geometries). We again find good agreement between Sesame and the other software packages: the largest relative difference between Sesame and Sentaurus is 1.8 %, and between Sesame and COMSOL it is 0.7 %. Fig. 3(b) shows the good agreement obtained for the band diagram along the grain boundary core under short-circuit conditions for the three software packages.

In Fig. 3(a) we also include the $J - V$ curve obtained with the 1-d simulation; the large difference between the 1-d and 2-d results reveals the crucial role the grain boundary plays in the system response. We've also checked that Sesame matches commercial software for systems with non-vertical, or "tilted" grain boundaries, and for system with multiple, intersecting grain boundaries [27]. Systems with complex grain boundary geometries are easily constructed in Sesame: the grain boundary endpoints are the only required input, and the software automatically embeds the planar defect in the real space mesh. The dependence of device behavior on grain boundary orientation and geometry was recently studied using Sesame in Ref. [28].

B. Scripting example

Sesame is run either through a self-contained GUI, or as a python package which is called in scripts. Running sesame with scripts is particularly convenient for running large-scale batch simulations on a computing cluster. Scripting also provides more flexibility in system definition (*e.g.* continuous grading of electronic parameters and doping). In the distribution, we provide several example scripts which describe standard PV simulations (e.g. J-V, IQE calculations), along with in-depth tutorials in the documentation. Here we give a description of a script to build and solve a simple 2-d simulation with a grain boundary.

We first import the numpy and sesame packages:

```
import sesame
import numpy as np
```

Next we define the grids for x and y. We use uniform grids for this example, but generally non-uniform grids are necessary to optimize the simulation accuracy

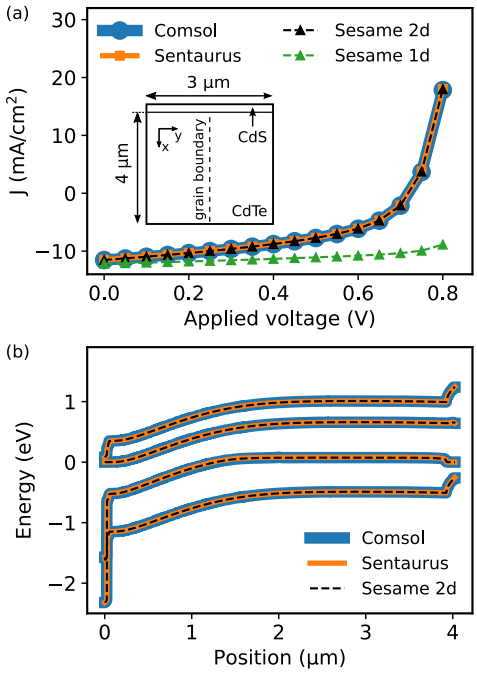

FIG. 3. Comparison between Sesame, COMSOL and Sentaurus for a 2-dimensional system. (a) Illuminated JV curve. Inset: schematic of the system, an n-p junction with a columnar grain boundary. (b) Band diagram along the grain boundary core under short-circuit conditions.

and speed (non-uniform grids are used in the simulation of Fig. 3). (Note: Sesame assumes all lengths are given in units of cm.)

```
x = np.linspace(0,3e-4,100)
y = np.linspace(0,3e-4,100)
```

We create the system with the `Builder` function. The input to `Builder` are the x and y grids. The output is an object `sys` which contains all the information needed to describe the simulation.

```
sys = sesame.Builder(x, y)
```

Additional simulation settings are set by calling various methods of `sys`, as we show below.

Next we define the material properties with a python dictionary object (called `mat` in this example). The dictionary key names correspond to standard definitions. (Note: Sesame assumes times are given in units of s, energies in units of eV, densities in units of cm^{-3}, mobility in units of $\mathrm{cm/V \cdot s}$)).

```
mat = {'Nc':8e17, 'Nv':1.8e19, 'Eg':1.5,
  'affinity':4.1, 'epsilon':9.4, 'Et':0,
  'mu_e':320, 'mu_h':40, 'tau_e':1e-8,
  'tau_h':1e-8}
```

The dictionary key `Et` represents the energetic position of bulk recombination centers, as measured from the intrinsic energy level, and `tau_e`/`tau_h` are the electron/hole lifetimes. The dependence of the Schockley-Read-Hall recombination on these parameters can be found in the Appendix. The material is added to the system using the `add_material` function, which takes the `mat` dictionary as input. Note that `add_material` is a method of the `sys` object, and is called with the command:

```
sys.add_material(mat)
```

To build a p-n junction we add a position-dependent doping profile to the system. We must define functions which describe the different doping regions; for this example, these functions are called `n_region` and `p_region`. They return True when the input variable `position` belongs to the region. For this example the two regions are delimited at the `junction` coordinate which corresponds to $x = 10^{-5}$ cm.

```
junction = 1e-5

def n_region(position):
  x, y = position
  return x < junction

def p_region(position):
  x, y = position
  return x >= junction
```

Having defined the different doping regions, we add the donors and acceptors with the `sys` methods `add_donor` and `add_acceptor`. The input for these methods are the doping magnitude and doping region functions we just defined. Sesame currently assumes that all bulk dopants are fully ionized. (Note: Sesame assumes the units of density is cm^{-3}):

```
donorDensity = 1e17
sys.add_donor(donorDensity, n_region)
acceptorDensity = 1e15
sys.add_acceptor(acceptorDensity, p_region)
```

Next we specify the contact boundary conditions. For this example, we specify Ohmic contacts with the function `contact_type`. Note the order of input arguments is left contact ($x = 0$) type first, right contact ($x = L$) type second:

```
sys.contact_type('Ohmic','Ohmic')
```

We next specify the value of recombination velocity for electrons and holes at both contacts (Note: Sesame assumes the units of velocity are cm/s). For this example, both contacts only collect majority carriers. This is accomplished with the function `contact_S`:

```
Sn_L, Sp_L, Sn_R, Sp_R = 1e7, 0, 0, 1e7
sys.contact_S(Sn_L, Sp_L, Sn_R, Sp_R)
```

Next we add a grain boundary. We must specify the grain boundary defect energy level `EGB` (note the defect energy level is measured from the intrinsic energy level), the electron and hole capture cross sections `sigmaeGB` and `sigmahGB`, the defect density `rhoGB`, and the endpoints of the line defining the grain boundary `p1, p2`. These are input arguments to the function `add_defects` which creates a grain boundary. We also specify the charge transition states of the defect with the function input `transition`. In this case the specified charge states are (+1,-1), corresponding to having a donor and acceptor at the same energy level.

```
EGB = 0.4
sigmaeGB = 1e-15
sigmahGB = 1e-15
rhoGB = 1e14
p1 = (.1e-4, 1.5e-4)
p2 = (2.9e-4, 1.5e-4)
sys.add_defects([p1, p2], rhoGB,
  sigmaeGB, sigmahGB, EGB, transition=(+1,-
1))
```

We add illumination by defining a function `illumination` which returns the position-dependent intensity as a function of the input coordinate `x,y`

```
def illumination(x,y):
  return 2.3e21 * np.exp(-2.3e4 * x)
sys.generation(illumination)
```

With the system now fully defined, we specify the list of applied voltages used to compute the current-voltage relation with the `IVcurve` function:

```
voltages = np.linspace(0,1,11)
jset = sesame.IVcurve(sys, voltages, 'GB_JV')
```

The function `IVcurve` returns an array `jset` containing the computed current density for each applied voltage. The `IVcurve` function also saves output files with seedname "GB_JV" concatenated with a suffix labeling the applied voltage index. These output files contain objects describing the simulation settings and the solution arrays. By default these files are compressed data files containing python Pickle objects (.gzip files). There is also an option to output the data in Matlab format (.mat files). Sesame includes an `Analyzer` object which contains several functions for computing quantities of interest from the solution, such as current densities, total recombination, carrier densities, and others. We refer the reader to the online documentation for a detailed list of all these functions.

C. GUI

Use of the standalone GUI as an alternative to scripting can be more convenient for small-scale calculations, or for those without access to a python distribution. Simulation settings can be saved and loaded, and the GUI also provides an interactive python prompt. The GUI is divided into three tabs, as shown in Fig. 4:

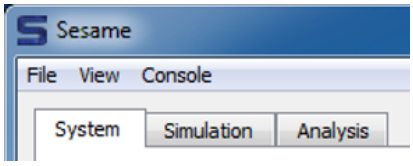

FIG. 4. Menu and 3 main tabs of the Sesame GUI. See text for a description of the functionality of each tab.

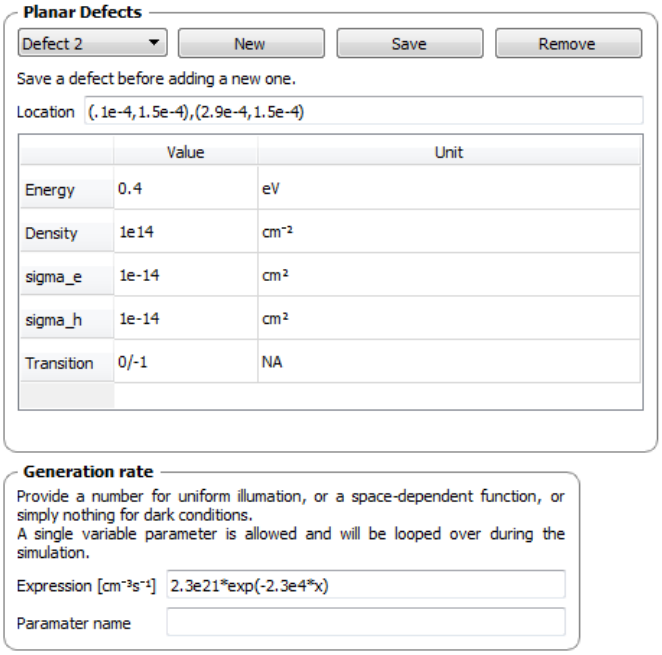

FIG. 5. Panels from System tab of the GUI. In the planar defects panel, an arbitrary number of planar defects can be defined by specifying the endpoints of the boundary (a 1-d line for a 2-dimensional simulation), the defect energy, density, electron and hole capture cross sections, and charge states. The Generation rate panel allows for one user-defined parameter to be varied.

1. The System tab contains fields to define the system geometry and material parameters (see Fig. 5).
2. The Simulation tab lets the user specify which parameter is varied: either the voltage is swept, or a user-defined variable related to the generation rate density is swept. The boundary conditions and output file information is also set here, and the simulation is launched from this tab. The program output is provided so that the user can follow the progress of the calculations.
3. The Analysis tab enables the user to plot the

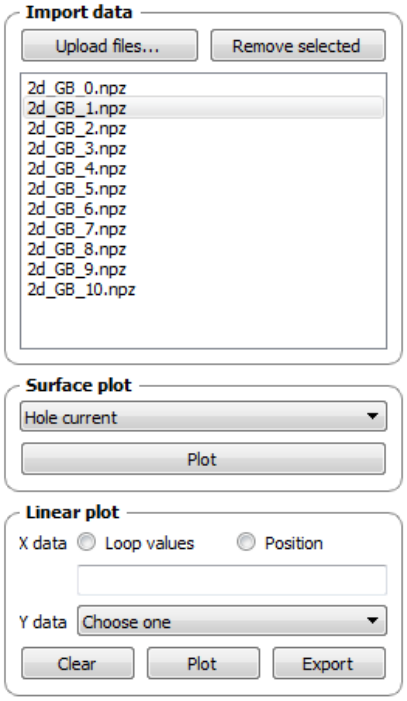

FIG. 6. Panel from Analysis tab of the GUI. Data files can be selected for analysis, and surface plots of system variables and observables can be generated for 2-dimensional systems. A linear plot with two modes is available. In "loop values" mode, a scalar (such as total current) is plotted versus the looped parameters. In the "position" mode, a system variable or observable from a single solution is plotted versus spatial coordinate.

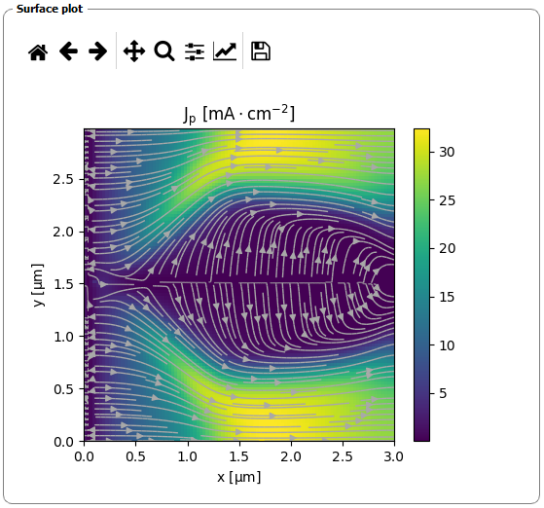

FIG. 7. The surface plot panel of the Analysis tab of the GUI. This shows the hole currents flowing in a homojunction with a single grain boundary.

output of the simulation, and to save and export plotted data (see Figs. 6 and 7).

Sesame is distributed with a number of sample input files for setting up standard PV simulations in the GUI. More detailed documentation for the GUI is included in the distribution.

IV. CONCLUSION

Modeling tools are essential for describing and understanding polycrystalline materials and nanoscale measurements. System behavior for complex, 2-dimensional geometries can be drastically different than the textbook 1-dimensional p-n junction model. Numerical simulations provide the capability to explore and develop intuition about this rather unchartered territory. Our aim in releasing Sesame is to provide the research community with a free, easy-to-use resource which will enable broader use of simulation in complex photovoltaic systems. There are opportunities for additional functionalities (e.g. time-dependence, small-signal analysis, more advanced interface transport models) and further optimizations (e.g. use of Cython) of the code. Our intent in releasing the fully documented source code is to provide users the option to make these and other additions as their research needs require. An additional feature not discussed here is 3-dimensional modeling, which is included in the distribution as an untested feature which will be investigated further in future work.

ACKNOWLEDGMENTS

B. G. acknowledges support under the Cooperative Research Agreement between the University of Maryland and the National Institute of Standards and Technology Center for Nanoscale Science and Technology, Award 70NANB14H209, through the University of Maryland. Y. S. and P. B. acknowledge Support provided by the Department of Energy, under DOE Cooperative Agreement No. DE-EE0004946 (PVMI Bay Area PV Consortium), and the National Science Foundation Award EEC 1454315 – CAREER: Thermophotonics for Efficient Harvesting of Waste Heat as Electricity. We thank Marc Burgelman for helpful correspondence, and thank Mike Scarpulla and Heayoung Yoon for helpful feedback on the software design. We thank Nick Rohrman for assistance in code development.

[1] H. Zhu, A. K. Kalkan, J. Hou, and S. J. Fonash, in *AIP Conference Proceedings*, Vol. 462 (AIP, 1999) pp. 309–314.

[2] P. A. Basore and D. A. Clugston, in *Photovoltaic Specialists Conference, 1996., Conference Record of the Twenty Fifth IEEE* (IEEE, 1996) pp. 377–381.
[3] M. Burgelman, P. Nollet, and S. Degrave, Thin Solid Films **361**, 527 (2000).
[4] Y. Liu, Y. Sun, and A. Rockett, Solar Energy Materials and Solar Cells **98**, 124 (2012).
[5] J. L. Gray, in *Photovoltaic Specialists Conference, 1991., Conference Record of the Twenty Second IEEE* (IEEE, 1991) pp. 436–438.
[6] P. P. Altermatt, Journal of computational electronics **10**, 314 (2011).
[7] P. A. Basore and K. Cabanas-Holmen, IEEE Journal of Photovoltaics **1**, 72 (2011).
[8] J. D. Major, Semiconductor Science and Technology **31**, 093001 (2016).
[9] Y. Yan, R. Noufi, and M. Al-Jassim, Physical review letters **96**, 205501 (2006).
[10] J. S. Yun, A. Ho-Baillie, S. Huang, S. H. Woo, Y. Heo, J. Seidel, F. Huang, Y.-B. Cheng, and M. A. Green, The journal of physical chemistry letters **6**, 875 (2015).
[11] S. Edmiston, G. Heiser, A. Sproul, and M. Green, Journal of applied physics **80**, 6783 (1996).
[12] M. Gloeckler, J. R. Sites, and W. K. Metzger, Journal of applied physics **98**, 113704 (2005).
[13] U. Rau, K. Taretto, and S. Siebentritt, Applied Physics A **96**, 221 (2009).
[14] B. Gaury and P. M. Haney, Journal of applied physics **120**, 234503 (2016).
[15] B. Gaury and P. M. Haney, Physical Review Applied **8**, 054026 (2017).
[16] P. M. Haney, H. P. Yoon, B. Gaury, and N. B. Zhitenev, Journal of applied physics **120**, 095702 (2016).
[17] Y. Jin and S. T. Dunham, IEEE Journal of Photovoltaics **7**, 329 (2017).
[18] A. Kanevce, J. Moseley, M. Al-Jassim, and W. K. Metzger, IEEE Journal of Photovoltaics **5**, 1722 (2015).
[19] B. Mendis, A. Howkins, D. Stowe, J. Major, and K. Durose, Ultramicroscopy **167**, 31 (2016).
[20] A. Kanevce, D. Levi, and D. Kuciauskas, Progress in Photovoltaics: Research and Applications **22**, 1138 (2014).
[21] S. G. Kumar and K. K. Rao, Energy & Environmental Science **7**, 45 (2014).
[22] S. D. U. Guide and E. Version, Mountain View, CA, USA (2013).
[23] C. Multiphysics, COMSOL AB, Stockholm, Sweden (2017).
[24] The full description of the procedures used in this paper requires the identification of certain commercial products. The inclusion of such information should in no way be construed as indicating that such products are endorsed by NIST or are recommended by NIST or that they are necessarily the best software for the purposes described.
[25] K. Horio and H. Yanai, IEEE transactions on electron devices **37**, 1093 (1990).
[26] H. K. Gummel, IEEE Trans. Electron Devices **11**, 455 (1964).
[27] B. Gaury, Y. Sun, P. Bermel, and P. Haney, in *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC)(A Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU PVSEC)* (IEEE, 2018) pp. 1882–1885.
[28] B. Gaury and P. M. Haney, ACS Applied Energy Materials **2**, 144 (2018).
[29] R. F. Pierret, *Semiconductor device fundamentals* (Pearson Education India, 1996).
[30] S. J. Fonash, *Solar cell device physics* (Academic Press, 1981).
[31] S. Selberherr, *Analysis and simulation of semiconductor devices* (Springer Science & Business Media, 2012).
[32] D. Vasileska, S. M. Goodnick, and G. Klimeck, *Computational Electronics: semiclassical and quantum device modeling and simulation* (CRC press, 2017).
[33] K. M. Van Vliet and A. H. Marshak, Solid-State Electronics **23**, 49 (1980).
[34] W. Shockley and W. T. Read, Phys. Rev. **87**, 835 (1952).
[35] G. Brown and B. Lindsay, Solid-State Electronics **19**, 991 (1976).

Appendix A MODEL DETAILS

A Mathematical Description

In this section we provide a full description of the equations solved by Sesame. These are fairly standard and can be found in textbooks [29–32], but we include them here for the sake of completeness and to specify notation and conventions used in the code. We first write densities in terms of quasi-Fermi levels, denoted by E_{F_n} and E_{F_p} for electrons and holes, respectively. Since we assume Boltzmann statistics (*i.e.* a non-degenerate semiconductor), the carrier densities are related to quasi-Fermi levels by:

$$n = N_C \exp\left(\frac{E_{F_n} + \chi + q\phi}{k_B T}\right) \tag{6}$$

$$p = N_V \exp\left(\frac{-E_{F_p} - \chi - E_g - q\phi}{k_B T}\right). \tag{7}$$

where E_g is the material band gap, χ is the electron affinity, and $N_{C,V}$ are the conduction, valence band effective density of states, respectively. All quantities except temperature can vary with position.

The electron and hole current can be expressed in terms of the spatial gradient of the quasi-Fermi levels [33]:

$$\vec{J}_n = q\mu_n n \vec{\nabla} E_{F_n} \tag{8}$$

$$\vec{J}_p = q\mu_p p \vec{\nabla} E_{F_p}. \tag{9}$$

1 Recombination

Sesame includes Shockley-Read-Hall, radiative and Auger recombination. The steady-state Shockley-Read-Hall recombination rate density is given by:

$$R_{\text{SRH}} = \frac{np - n_i^2}{\tau_p(n + n_1) + \tau_n(p + p_1)} \tag{10}$$

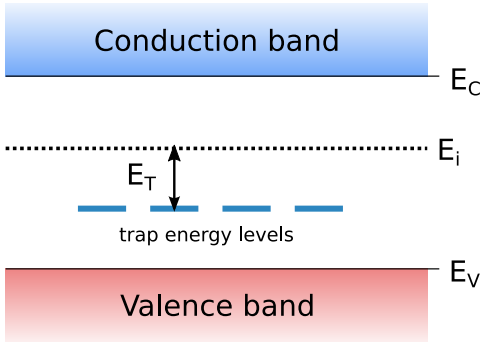

FIG. 8. Depiction of energy levels of defect states. The intrinsic energy level is $E_i = E_g/2 - k_B T/q \ln(N_C/N_V)$, as measured from the valence band edge.

where n_i is the material intrinsic carrier density, given by $n_i = \sqrt{N_C N_V} \exp(-E_g/(2k_B T))$. The equilibrium Fermi energy at which $n = p = n_i$ is the instrinsic energy level E_i. We specify the defect energy level E_T relative to E_i (see Fig. 8), so that the expressions for n_1 and p_1 in Eq. 10 are given by:

$$n_1 = n_i \exp\left(\frac{E_T}{k_B T}\right), \tag{11}$$

$$p_1 = n_i \exp\left(-\frac{E_T}{k_B T}\right) \tag{12}$$

$\tau_{n,(p)}$ is the bulk lifetime for electrons, holes. It is given by

$$\tau_{n,p} = \frac{1}{N_T v^{\text{th}}_{n,p} \sigma_{n,p}} \tag{13}$$

where N_T is the three-dimensional trap density, $v^{\text{th}}_{n,p}$ is the thermal velocity of carriers ($v^{\text{th}}_{n,p} = \sqrt{3k_B T/m_{n,p}}$ with $m_{n,p}$ the electron/hole effective mass), and $\sigma_{n,p}$ is the capture cross-section for electrons, holes.

The radiative recombination has the form

$$R_{\text{rad}} = B(np - n_i^2) \tag{14}$$

where B is the radiative recombination coefficient of the material. The Auger mechanism has the form

$$R_{\text{A}} = (C_n n + C_p p)(np - n_i^2) \tag{15}$$

where C_n (C_p) is the electron (hole) Auger coefficient.

2 *Planar defects*

Sesame has been created with the intent of studying extended defects in solar cells, such as grain boundaries and sample surfaces. These extended planar defects are represented by a point in a 1-d model, a line in a 2-d model, and a plane in a 3-d model. The extended defect energy level spectrum can be discrete or continuous. For a discrete spectrum, we label the defect with the subscript d. The occupancy of the defect level f_d is given by [34]

$$f_d = \frac{S_n n + S_p p_d}{S_n(n + n_d) + S_p(p + p_d)} \tag{16}$$

where n (p) is the electron (hole) density at the spatial location of the defect, S_n, S_p are recombination velocity parameters for electrons and holes respectively. n_d and p_d are

$$n_d = n_i \exp\left(\frac{E_d}{k_B T}\right) \tag{17}$$

$$p_d = n_i \exp\left(-\frac{E_d}{k_B T}\right) \tag{18}$$

where E_d is calculated from the intrinsic level E_i.

The electron/hole recombination velocity are related to the electron/hole capture cross section and the defect density ρ_d according to:

$$S_{n,p} = \rho_d \sigma_{n,p} v^{\text{th}}_{n,p}. \tag{19}$$

The defect recombination is of Shockley-Read-Hall form:

$$R_d = \frac{S_n S_p (np - n_i^2)}{S_n(n + n_d) + S_p(p + p_d)}. \tag{20}$$

The charge density given of a single defect depends on the defect type (acceptor vs. donor)

$$Q = q\rho_d \times \begin{cases} (1 - f_d) & \text{donor} \\ (-f_d) & \text{acceptor} \end{cases} \tag{21}$$

where ρ_d is the defect density of state at energy E_d. Multiple defects are described by summing over defect label d, or performing an integral over a continuous defect spectrum.

B Boundary conditions at the contacts

For a given system definition, Sesame first solves the equilibrium problem. In equilibrium, the quasi-Fermi level of electrons and holes levels are equal and spatially constant. We choose an energy reference such that in equilibrium, $E_{F_p} = E_{F_n} = 0$. The equilibrium problem is therefore reduced to a single variable $\phi^{\text{eq}}(\mathbf{r})$. Sesame employs both Dirichlet and von Neumann equilibrium boundary conditions for ϕ^{eq}, which we discuss next.

1 *System in thermal equilibrium*

Sesame uses Dirichlet boundary conditions as the default. This is the appropriate choice apply when the equilibrium charge density at the contacts is known *a priori*.

This applies for Ohmic and ideal Schottky contacts. For Ohmic boundary conditions, the carrier density is assumed to be equal and opposite to the ionized dopant density at the contact. For an n-type contact with N_D ionized donors at the $x = 0$ contact (*i.e.* no free excess carriers at the contact), Eq. 6 yields the expression for $\phi^{\mathrm{eq}}(x = 0)$:

$$q\phi^{\mathrm{eq}}(0, y, z) = k_B T \ln\left(\frac{N_D}{N_C}\right) - \chi(0, y, z) \tag{22}$$

Similar reasoning yields expressions for $q\phi^{\mathrm{eq}}$ for p-type doping and at the $x = L$ contact.

For Schottky contacts, we assume that the Fermi level at the contact is equal to the Fermi level of the metal. This implies that the equilibrium electron density is $N_C \exp\left[-\left(\Phi_M - \chi\right)/k_B T\right]$, where Φ_M is the work function of the metal contact. Eq. 6 then yields the expression for ϕ^{eq} (shown here for the $x = 0$ contact):

$$q\phi^{\mathrm{eq}}(0, y, z) = -\Phi_M|_{x=0 \text{ contact}} \tag{23}$$

An identical expression applies for the $x = L$ contact.

Sesame also has an option for von Neumann boundary conditions, where it's assumed that the electrostatic field at the contact vanishes:

$$\frac{\partial \phi^{\mathrm{eq}}}{\partial x}(0, y, z) = \frac{\partial \phi^{\mathrm{eq}}}{\partial x}(L, y, z) = 0. \tag{24}$$

The equilibrium potential ϕ^{eq} determines the equilibrium densities n^{eq}, p^{eq} according to Eqs. 6 and 7 with $E_{F_n} = E_{F_p} = 0$.

2 *System out of thermal equilibrium*

Out of thermal equilibrium, Dirichlet boundary conditions are imposed on the electrostatic potential. For example, in the presence of an applied bias V at $x = L$, the boundary conditions are

$$\phi(0, y, z) = \phi^{\mathrm{eq}}(0, y, z) \tag{25}$$
$$\phi(L, y, z) = \phi^{\mathrm{eq}}(L, y, z) + qV \tag{26}$$

where ϕ^{eq} is the equilibrium electrostatic potential.

For the drift-diffusion equations, the boundary conditions for carriers at charge-collecting contacts are parameterized with the surface recombination velocities for electrons and holes at the contacts, denoted respectively by S_{c_n} and S_{c_p}:

$$J_n^x(0, y, z) = qS_{c_n}^0\left(n(0, y, z) - n^{\mathrm{eq}}(0, y, z)\right) \tag{27}$$
$$J_p^x(0, y, z) = -qS_{c_p}^0\left(p(0, y, z) - p^{\mathrm{eq}}(0, y, z)\right) \tag{28}$$
$$J_n^x(L, y, z) = -qS_{c_n}^L\left(n(L, y, z) - n^{\mathrm{eq}}(L, y, z)\right) \tag{29}$$
$$J_p^x(L, y, z) = qS_{c_p}^L\left(p(L, y, z) - p^{\mathrm{eq}}(L, y, z)\right) \tag{30}$$

C Numerical implementation

In this section we review the set of equations solved by Sesame and provide some details of their implementation in the one-dimensional case.

1 *Scharfetter-Gummel scheme*

Sesame uses finite differences to solve the drift-diffusion-Poisson equations on a nonuniform grid. Fig. 9 shows our index-labeling convention for sites and links: link i connects site i and site $i+1$. Site-defined quantities (such as density and electrostatic potential) are labeled with a subscript denoting the site number. Link-defined quantities (such as electrical current and electric field) are labeled with a superscript denoting the link number.

We consider a one-dimensional system to illustrate the model discretization. First, we rewrite the current on link i in semi-discretized form:

$$J_n^i = q\mu_{n,i} n_i \left.\frac{\mathrm{d}E_{F_n}}{\mathrm{d}x}\right|_i \tag{31}$$
$$J_p^i = q\mu_{p,i} p_i \left.\frac{\mathrm{d}E_{F_p}}{\mathrm{d}x}\right|_i \tag{32}$$

A key step to ensure numerical stability is to integrate Eqs. (31) and (32) in order to get a completely discretized version of the current $J_{n,p}^i$. This discretization is known as the Scharfetter-Gummel scheme [26]. Here we give the final expressions for the hole current J_p^i between sites i and $i+1$:

$$J_p^i = \frac{q}{\Delta x^i}\left(\frac{\psi_{p,i+1} - \psi_{p,i}}{\exp\left(\frac{\psi_{p,i+1}}{k_B T}\right) - \exp\left(\frac{\psi_{p,i}}{k_B T}\right)}\right) \times \mu_{p,i}\left[\exp\left(\frac{-E_{F_p,i+1}}{k_B T}\right) - \exp\left(\frac{-E_{F_p,i}}{k_B T}\right)\right]. \tag{33}$$

where $\psi_p = q\phi + \chi + E_g - k_B T \ln(N_V)$ is the effective potential. The electron current J_p^i is given by:

$$J_n^i = -\frac{q}{\Delta x^i}\left(\frac{\psi_{n,i+1} - \psi_{n,i}}{\exp\left(\frac{-q\psi_{n,i+1}}{k_B T}\right) - \exp\left(\frac{-q\psi_{n,i}}{k_B T}\right)}\right) \times \mu_{n,i}\left[\exp\left(\frac{E_{F_n,i+1}}{k_B T}\right) - \exp\left(\frac{E_{F_n,i}}{k_B T}\right)\right]. \tag{34}$$

where $\psi_n = q\phi + \chi + k_B T \ln(N_C)$.

In the limit where either $\delta\psi_{n(p)} \equiv -q\left(\psi_{n(p),i+1} - \psi_{n(p),i}\right)/k_B T$ or $\delta E_{F_{n(p)}} \equiv \left(E_{F_{n(p)},i+1} - E_{F_{n(p)},i+1}\right)/k_B T$ are smaller than 10^{-5}

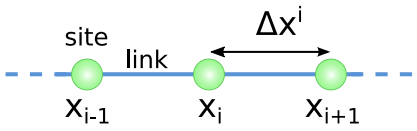

FIG. 9. Sites and links of the grid used in the discretization of the drift diffusion and Poisson equations.

Quantity	Expression	Value
Density	N_0	10^{19} cm^{-3}
Mobility	μ_0	1 cm^2/(V $\cdot$ s)
Temperature	T_0	300 K
Energy	$k_B T_0$	0.0258 eV
Length (x_0)	$\sqrt{\epsilon_0 k_B T/(q^2 N_0)}$	3.78×10^{-8} cm
Time	$\epsilon_0/(q\mu_0 N_0)$	5.5×10^{-14} s
Gen. rate density	$N\mu_0 E_0/(qx_0^2)$	1.81×10^{32} 1/(cm$^3 \cdot$ s)
Current	$\mu_0 N_0 k_B T/x_0$	1.10×10^6 A/cm^2

TABLE III. Quantities used to scale variables to dimensionless form.

and 10^{-9}, respectively, we replace the expressions for the current with a Taylor series expansion of the small parameter. In the expansion, we evaluate the current up to second order in $\delta\psi_{n(p)}$, and up to first order in $\delta E_{F_{n(p)}}$.

Embedding a two-dimensional density into the three-dimensional model is formally accomplished with the use of a delta function. Numerically, the two-dimensional defect densities of states and the surface recombination velocities are divided by the size of the discretized grid at the position of the plane, and along the direction normal to the plane.

2 *Newton-Raphson algorithm*

The discretization of Eqs. (1)-(3) leads to the system of three equations for all sites of the discretized space (except boundary sites):

$$0 = \frac{2}{\Delta x^i + \Delta x^{i-1}} \left(J_p^i - J_p^{i-1}\right) + G_i - R_i \tag{35}$$

$$0 = \frac{2}{\Delta x^i + \Delta x^{i-1}} \left(J_n^i - J_n^{i-1}\right) - G_i + R_i \tag{36}$$

$$0 = \rho_i + \frac{2}{\Delta x^i + \Delta x^{i-1}} \times \left[\left(\frac{\epsilon_{i+1} + \epsilon_i}{2}\right)\left(\frac{\phi_{i+1} - \phi_i}{\Delta x^i}\right) - \left(\frac{\epsilon_i + \epsilon_{i-1}}{2}\right)\left(\frac{\phi_i - \phi_{i-1}}{\Delta x^{i-1}}\right)\right] \tag{37}$$

Because we exchanged the carrier densities for the quasi-Fermi levels as the unknowns of the problem, we are therefore looking for the sets E_{F_n}, E_{F_p}, ϕ at every grid point.

We use the Newton-Raphson method to solve the above set of equations: Given a general nonlinear function $f(x)$, we want to find its root $\bar{x} : f(\bar{x}) = 0$. Given an initial guess x_1, one can estimate the error δx in this guess, assuming that the function varies linearly all the way to its root

$$\delta x = \left(\frac{\mathrm{d}f}{\mathrm{d}x}(x_1)\right)^{-1} f(x_1)\,. \tag{38}$$

An updated guess is provided by $x_2 = x_1 - \delta x$. The assumption of linear variation is key here, as if the guess x_1 is too far from the root, the convergence of the algorithm is very uncertain.

In multiple dimensions the derivative in Eq. (38) is replaced by the Jacobian. In this case, Eq. (38) is a matrix equation of the form

$$\delta \mathbf{x} = A^{-1}\mathbf{F}(\mathbf{x}) \tag{39}$$

where $\mathbf{F}$ is a vector function of the unknowns of the problem on all sites of the discretized space, and A is the Jacobian matrix given by

$$A_{ij} = \frac{\partial F_i}{\partial x_j}. \tag{40}$$

We find that convergence of the Newton-Raphson algorithm for this problem requires exact (analytically computed) values for the Jacobian.

In case the guess is far from the root we are looking for, the correction given by Eq. 38 can overshoot the solution. A simple way to improve the convergence is to damp the corrections $\delta\mathbf{x}$ given by Eq. (39). Inspired by an earlier work [35], we found that the following procedure gives good results. For $\delta\mathbf{x_i} > 1$, we replace $\delta\mathbf{x_i}$ by

$$\delta\bar{\mathbf{x}}_\mathbf{i} = \mathrm{sgn}(\delta\mathbf{x_i}) \log\left(1 + 1.72|\delta\mathbf{x_i}|\right). \tag{41}$$