



Bumblebee Manual

Amsterdam Modeling Suite 2024.2

www.scm.com

Oct 25, 2024

TABLE OF CONTENTS

1	General	1
1.1	Introduction	1
1.2	Features	1
1.3	Theory	2
1.3.1	Exciton Generation	2
	Energy Distribution Functions	3
1.3.2	Charge Hopping	3
1.3.3	Exciton Hopping	4
1.3.4	Exciton Decay	4
1.3.5	Intersystem Crossing	5
1.3.6	Exciton Annihilation	5
1.3.7	Interlayer Processes	5
1.3.8	Vibronic Coupling	6
1.3.9	Coulomb Interaction	6
1.3.10	Initial Charge Distribution	7
1.3.11	Dipole Distributions	7
1.3.12	Polymeric Materials	7
1.3.13	Alternating Voltage	7
1.3.14	Photoluminescence	8
1.3.15	Lifetime Simulations	8
1.3.16	OFET	8
1.3.17	Transient Responses	9
1.3.18	Simulation Volume	9
1.3.19	Morphology	9
1.3.20	Annealing	9
1.3.21	Pre-equilibration	10
1.3.22	Simulation Settings	10
	Timescale	10
	Parallelization	10
	Output	11
	Acceleration	11
	Memory Usage	11
	Reproducibility	11
2	Installation	13
2.1	Downloading Bumblebee	13
2.2	Web Interface	14
2.3	Bumblebee	17
2.3.1	Linux	17
2.4	Connecting to a Server	19

2.4.1	Server Setup	19
	Linux (cluster)	19
	Linux (local)	20
	Slurm configuration	20
2.4.2	Connection Setup	21
	Login Authorization	21
	Connection Settings	21
2.4.3	Testing the Installation	24
2.5	FAQ	25
3	Usage	27
3.1	Graphical User Interface	27
3.1.1	Materials	27
3.1.2	Compositions	28
3.1.3	Parameter Sets	28
3.2	Visualization	28
3.3	Input Files	29
3.4	Output Files	29
3.5	Python API	36
4	Bumblebee Tutorials	37
4.1	Getting Started	37
4.1.1	Basic Usage	37
	Project Setup	37
4.1.2	Bulk Simulation	39
	Create Materials	39
	Create a Stack	42
	Create a Parameter Set	42
	Starting the Simulation	47
	Monitoring the Simulation	49
	Simulation Output	50
4.1.3	Parameter Screening	50
	Create Materials	50
	Create Compositions	50
	Create a Stack	51
	Create a Parameter Set	51
	Starting the Simulation	54
	Simulation Output	54
4.1.4	Exciton Simulation	56
	Create Materials	56
	Create Compositions	60
	Create a Stack	60
	Create a Parameter Set	60
	Starting the Simulation	64
	Simulation Output	64
4.1.5	Acceleration	65
	Module Selection	65
	Configuring Output	66
	Cost Scaling	66
4.2	Advanced Features	66
4.2.1	Layer Morphology	66
	Basic Composition	67
	Layer Gradients	67
	Visualizing Gradients	76

	Layer Contacts	76
4.2.2	Polymeric OLED	81
	Create Materials	81
	Create Compositions	82
	Create a Stack	82
	Create a Parameter Set	82
	Starting the Simulation	82
	Simulation Output	84
4.2.3	Device Lifetime	85
	Create Materials	85
	Create Compositions	86
	Create a Stack	86
	Create a Parameter Set	87
	Running the Simulation	87
4.2.4	Photoluminescence	87
	Create Materials	87
	Create a Stack	89
	Create a Parameter Set	91
	Starting the Simulation	92
	Simulation Output	92
4.2.5	Transient Switching	92
	Create Compositions	92
	Create a Stack	92
	Create a Parameter Set	92
	Running the Simulation	94
4.2.6	OFET Simulation	94
	Create Materials	95
	Create a Stack	97
	Create a Parameter Set	97
	Starting the Simulation	99
	Simulation Output	99
4.2.7	Device Equilibration	99
	Circuit Closure	99
	Circuit Disconnect	99
4.2.8	Advanced Initialization	101
	Charge Carrier Initialization	101
	Consistency in HOMO, LUMO and Exciton Levels	103
	Dipole Orientation	103
4.3	API	105
4.3.1	API Access	105
	API Installation	105
	Establishing the API Connection	108
	Creating New Resources	108
	Modifying Existing Resources	109
	Deleting Resources	110
	Accessing Simulation Output	110
4.3.2	Job Submission	110
	Create Materials	111
	Create Compositions	112
	Create a Stack	113
	Create a Parameter Set	114
	Starting the Simulation	114
	Simulation Output	114
4.3.3	Customized Parameter Screening	115

	Create a Composition Template	115
	Create a Stack	115
	Create a Parameter Set	115
	Submitting Multiple Simulations	115
	Simulation Output	116
	Multi-dimensional Parameter Screenings	116
4.3.4	Advanced Morphology	116
	Create Materials	117
	Create a Composition	117
	Create a Stack	118
	Create a Parameter Set	118
	Starting the Simulation	118
	Simulation Output	118
5	Required Citations	121
5.1	General References	121
5.2	Key Publications	121
6	FAQ	123
6.1	How do I determine the required number of simulation steps?	123
6.2	How do I determine the maximum wall time for my workload manager?	123
6.3	Can I recover the results from a killed job?	124
6.4	How do I perform two-dimensional parameter screenings?	124
6.5	How do I include charge-generation layers in the stack?	124
6.6	How do I model a tandem stack?	124

1.1 Introduction

Bumblebee utilizes kinetic Monte Carlo simulations to model the opto-electronic properties of OLED stacks. The transport of the electrons through the device is described stochastically in order to determine the time-evolution and space-dependence of excitonic processes.

Input parameters can be obtained from

- the built-in materials database, or
- through the AMS **OLED workflow** (<https://www.scm.com/tools/oled-workflows/>), or
- through manual input.

Bumblebee may be used

- from the command line,
- from Python, or
- through a dedicated web interface.

1.2 Features

Bumblebee is capable of simulating OLED stacks containing numerous layers comprised of both pure and composite materials.

- An extensive array of excitonic interactions can be included to describe the opto-electronic character
- Device degradation can be modeled to estimate stack lifetimes
- Photoluminescent absorption allows replication of dark-field and PV studies
- Optical outcoupling simulations predict emission spectra at the device level
- Small-signal analysis with AC modulation

1.3 Theory

Kinetic Monte Carlo estimates the properties of the OLED stack through stochastic sampling. A model of the stack is formulated as a 3D grid, with each node representing a charge carrier site. Electrode contacts are placed at the edges of the device to allow for current flow. The probability for a particular opto-electronic process to occur, such as exciton generation or charge transfer, is taken to be proportional to the rate of that process. Sampling of the various mechanisms mimics the time-evolution of the OLED, allowing kMC to explore the relevant regimes encountered during device operation.

In order to accurately model real OLED devices, relevant processes must be included in the simulation. Various mechanisms are included in Bumblebee.

- Charge transfer
- Exciton generation and emission
- Exciton annihilation
- Charge injection/removal at the electrodes
- Intersystem crossing
- Exciplexation

1.3.1 Exciton Generation

The HOMO and LUMO energy levels are considered as the energy levels of the charge carriers. Due to the inhomogeneous nature of solid-state organic electronics, the HOMO and LUMO values tend to vary between sites. This leads to a layer-wide distribution of energy levels. Bumblebee includes various model distributions to represent this variance. A standard deviation can be specified for either level.

Excitons are restricted by Bumblebee to either be in a singlet or triplet state, as higher spin states are short-lived in typical OLED materials. The ratio of singlet to triplet excitons can be adjusted for each layer.

The singlet and triplet species each have a binding energy. This value is combined with the HOMO and LUMO levels to determine the singlet and triplet energy levels.

$$E_{ex} = E_{LUMO} - E_{HOMO} - E_b$$

With E_{ex} the exciton energy level and E_b the binding energy. Similar to the HOMO and LUMO energies, a distribution of exciton energy levels may also be specified.

By default, the distribution in the energy levels is assumed to be uncorrelated. A correlation between the HOMO and LUMO levels can be specified to preserve the band gap. Anti-correlation is also available, wherein the energy shift in the HOMO is opposite that of the LUMO.

The exciton distribution may in turn be correlated to the band gap:

$$\epsilon_{ex} = \sigma_{ex} \frac{\epsilon_{gap}}{\sqrt{\sigma_{HOMO}^2 + \sigma_{LUMO}^2}}$$

The binding energy of the excitons is, by default, unaffected by the energy level shifts. Automatic propagation of the exciton shifts to the binding energies may be enabled.

Exciton generation may occur either through charge transfer or thermal excitation. The latter mechanism, in the absence of photoluminescence, is described through a Boltzmann process. The associated prefactor is considered a static parameter, which is constant throughout the stack.

Energy Distribution Functions

Several model distributions are available when sampling the energetic disorder in energy levels.

$$E = E_0 + P(X)$$

- Dirac delta

$$P(X) = 0$$

- Gaussian

$$P(X) = \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}X^2\right)$$

- Exponential

$$P(X) = \frac{q}{|q|} k_b T \exp(-X)$$

- Mirrored exponential

$$P(X) = \frac{(X_1 - \frac{1}{2})}{|X_1 - \frac{1}{2}|} k_b T \exp(-X_2)$$

- Chi-squared-extended Gaussian

$$P(X) = P_{\text{Gaussian}}(X_1) + \frac{q}{|q|} W \frac{\sqrt{X_2} \exp(-\frac{1}{2}X_2)}{\Gamma(\frac{3}{2}) 2^{3/2}}$$

- Dipole-correlated Gaussian

Bumblebee supports the inclusion of explicit dipole fields associated with the molecular dipole moments in the stack. When a dipole-correlated energy-level distribution is requested, the average dipole potential is evaluated at each site. This field distribution is then normalized to obtain the site-specific energy shifts:

$$\sigma_A = \frac{\sum_j \frac{\bar{p}_j \cdot (\bar{p}_A - \bar{p}_j)}{[(\bar{p}_A - \bar{p}_j) \cdot (\bar{p}_A - \bar{p}_j)]^{3/2}}}{\sum_i \sum_j \frac{\bar{p}_j \cdot (\bar{p}_i - \bar{p}_j)}{[(\bar{p}_i - \bar{p}_j) \cdot (\bar{p}_i - \bar{p}_j)]^{3/2}}}$$

1.3.2 Charge Hopping

The transport of charge carriers through the layer is modeled as a hopping process between sites. The probability of electron hopping is described using a Boltzmann factor.

$$r = \nu \exp\left(\frac{-\Delta E}{k_b T}\right)$$

ΔE represents the change in energy following the hop, $k_b T$ is the thermal energy and ν is the hopping frequency. The frequency is assumed to decay exponentially with the inter-site distance. For hops between layers, a geometric average of the layer parameters is assumed:

$$\nu = \sqrt{(\nu_I \nu_F)} \exp\left(- (R - a) \left(\frac{1}{\lambda_I} + \frac{1}{\lambda_F}\right)\right)$$

ν_l represents the reference hopping frequency between adjacent sites at distance a in each layer. λ_l is the decay length characterizing the decrease in hopping frequency with distance.

Note: Since many opto-electronic processes occur at very high frequencies, a reference timescale is introduced to re-scale the rate constants. See the section on [timescales](#) (page 10) for more info.

Due to the exponential decay, hopping events will be localized near each charge carrier site. To accelerate calculations, the number of hopping events is restricted to a local environment. Per default, a cubic volume with an edge-length of $4a$ is employed.

For charge exchange with the electrodes, the electrode Fermi level is used to determine the change in energy. Charge transfer with the electrodes is restricted to the boundary sites. The rate of charge exchange with the electrodes may be modified in order to bias the sampling distribution. For typical simulations, frequent electrode exchange events may result in long computation times. Reducing exchange rates then enhances the sampling of events in the stack. Care should be taken that the exchange adjustments do not affect the physical output of the simulation.

In order to investigate the effect of trap states, dopants or contaminants, device-specific prefactors may be appended to the hopping frequencies. These prefactors allow modification of the hopping rates across the stack without having to modify individual material properties. For more detailed studies, trap states or dopants can also be explicitly included in the simulation.

Anisotropic charge transport may additionally be specified by introducing an anisotropy vector:

$$\nu_{ANI} = \left(\frac{|R_x| \mu_x^{ANI} + |R_y| \mu_y^{ANI} + |R_z| \mu_z^{ANI}}{|R_x| + |R_y| + |R_z|} \right) \nu$$

In excitonic simulations, charge hopping may result in formation of a new exciton following polaron collision. An additional collision prefactor can be introduced to alter the frequency of these events.

1.3.3 Exciton Hopping

Hopping of an exciton may occur through both Dexter and Förster mechanisms. The Dexter process involves charge transfer through an overlap of the molecular wavefunctions, and is therefore necessarily localized. Dexter transfer is described through the same hopping mechanisms as the charge carrier species, with an associated hopping frequency and decay length.

The Förster mechanism involves exciton transfer through dipolar coupling:

$$\nu = \left(\frac{F}{R} \right)^6$$

F denotes the Förster radius, characterizing the coupling strength.

Note: All hopping parameters are unique for singlet and triplet excitons.

The thermal dependence of the Förster rates can also be disabled, turning the Förster reactions into fully electronic processes.

1.3.4 Exciton Decay

Excitons may decay radiatively (resulting in photonic emission) or non-radiatively (generating heat). Rates are specified for both processes. These rates may differ between singlet and triplet species.

1.3.5 Intersystem Crossing

Intersystem crossing and reverse intersystem crossing processes can be included in the Bumblebee simulation to account for singlet-triplet interconversions. These crossing events are currently implemented with a static rate (independent of temperature or site environment).

1.3.6 Exciton Annihilation

Excitons may be converted through various mechanisms.

- Exciton-polaron quenching
- Singlet-singlet annihilation
- Singlet-triplet annihilation
- Triplet-triplet annihilation

Each of these reactions can be specified using both Dexter and Förster models. The Dexter parameters are assumed to be material-specific, therefore using the same values as for the exciton hopping. Förster reactions meanwhile have to be specified for individual processes.

Two reaction pathways can be defined for exciton-polaron quenching:

- Polaron transfer to the exciton site
- Polaron-induced exciton recombination

Bumblebee allows specification of distinct rates for these steps.

Triplet-triplet annihilation may lead to formation of both singlets and triplets. By default, a singlet-triplet ratio is specified for the annihilation product. This behavior may be altered to specify singlet-selective or triplet-selective products.

As with the polaron transport, a device-specific prefactor may be appended to the exciton hopping frequencies to investigate the effect of e.g. impurities. These prefactors can be defined for each individual mechanism to aid in the investigation of mechanistic sensitivity.

Since annihilation events involve electronic processes, they are considered athermal by default. Use of the temperature-dependent Boltzmann factor for annihilation reactions can be enforced in the settings.

1.3.7 Interlayer Processes

For Dexter processes that occur across layers, a geometric average is used to determine the transfer parameters. Förster processes, meanwhile, require the parameters to be specified for each unique pair of materials. To add a finer degree of control, it is possible to provide similar definitions for the Dexter prefactors as well.

Default constructors are available for the inter-layer Förster parameters to ease simulation setup and to aid in mechanistic screening studies. Material characteristics are used to determine the relevant processes and to estimate the transfer rates.

1.3.8 Vibronic Coupling

The default expression for thermal activation assumes a ground-state occupation for both initial and final states.

$$r_{Boltzmann} = \nu \exp\left(\frac{-\Delta E}{k_b T}\right)$$

Marcus theory allows incorporation of a vibrational distortion of the molecular geometry following charge transfer, expressed in terms of a corresponding energy shift λ .

$$r_{Marcus} = \frac{\nu \exp\left[\frac{-(2\Delta E + \lambda_I + \lambda_F)^2}{8(\lambda_I + \lambda_F)k_b T}\right]}{\sqrt{2\pi(\lambda_I + \lambda_F)k_b T}}$$

The Levich-Jortner expression is used to incorporate tunneling.

$$r_{Jortner} = \frac{\nu \exp\left(\frac{-(\kappa_I + \kappa_F)}{\hbar(\omega_I + \omega_F)}\right)}{\sqrt{2\pi(\lambda_I + \lambda_F)k_b T}} \sum_{i=0}^n \frac{1}{i!} \left(\frac{\kappa_I + \kappa_F}{\hbar(\omega_I + \omega_F)}\right)^i \exp\left[\frac{-[2\Delta E + \lambda_I + \lambda_F + i\hbar(\omega_I + \omega_F)]^2}{8(\lambda_I + \lambda_F)k_b T}\right]$$

κ denotes the associated zero-point relaxation energy, n denotes the number of accessible vibrational energy levels and ω the normal mode frequency. A single, dominant vibration is assumed to define the vibronic coupling. A multi-modal formalism may similarly be enabled.

$$r_{multimode} = \frac{\nu \left[\prod_j^m \exp\left(\frac{-(\kappa_{I,j} + \kappa_{F,j})}{\hbar(\omega_{I,j} + \omega_{F,j})}\right) \right]}{\sqrt{2\pi(\lambda_I + \lambda_F)k_b T}} \sum_{\{i\}}^{\{n\}} \left(\prod_j^m \frac{1}{i_j!} \left(\frac{\kappa_{I,j} + \kappa_{F,j}}{\hbar(\omega_{I,j} + \omega_{F,j})}\right)^{i_j} \right) \exp\left[\frac{-[2\Delta E + \lambda_I + \lambda_F + \sum_j^m i_j \hbar(\omega_{I,j} + \omega_{F,j})]^2}{8(\lambda_I + \lambda_F)k_b T}\right]$$

For a set of m vibrational modes, each characterized by their own frequency and relaxation energy.

Because the vibronic coupling in the multimode form involves computing the interactions for each combination of harmonic overtones, this easily become a computational bottleneck in the simulation. An option is provided to compute the multimode rates using an iterative refinement approach, significantly reducing the overall computational cost, though some residual errors may remain in the system.

1.3.9 Coulomb Interaction

The voltage (V) applied to the stack creates an external field that affects the hopping rates:

$$\frac{\Delta E_{ext}}{\Delta x} = \frac{(V + E_{fa} - E_{fc})}{L}$$

With L the stack thickness and E_f the Fermi levels of the anode and cathode.

Coulomb interactions between charges are added to the external field contribution:

$$E_C = \frac{q}{4\pi\epsilon_r\epsilon_0} \frac{1}{R}$$

With q the carrier charge, ϵ_r the relative permittivity and ϵ_0 the vacuum permittivity. The pair-wise interactions are computed explicitly for nearby charges within a local environment, defined by a cutoff radius. Interactions outside the

cutoff radius are accounted for by considering a layer-uniform background charge density. To include interactions beyond the simulated region, accounting for the decay in Coulomb interactions with distance, the long-range contribution is truncated to a finite number of periodic images.

To accelerate calculations, particularly for homogeneous fields, an update interval may be specified for the long-range contribution. This assumption is valid when the change in the inter-layer charge distributions is slow.

1.3.10 Initial Charge Distribution

At the start of the simulation, no charge carriers are present in the layers. Charge carriers must be generated through excitonic processes or from the electrode current.

Initialization of charge carriers may be enabled to preserve carrier densities in e.g. bulk simulations. Charge carriers are then injected randomly throughout the stack. Carrier parameters are available to modify the material-specific injection probability. This allows initial charge distributions to be biased towards specific layers.

Defects may furthermore be specified to adjust the carrier densities in individual layers. Electrons, holes, singlet and triplet excitons can be selected.

1.3.11 Dipole Distributions

In order to compute the effect of dipole orientations on the electric field, molecular dipoles may be enabled. Random orientations of the dipoles are generated at each carrier site. A material-specific dipole magnitude is specified. Dipole orientations are randomly drawn from an ellipsoidal distribution, which can be modified to describe orientational preference.

Transition dipole fields may similarly be specified in order to account for orientational alignment in Förster processes. The transition dipole vector and the molecular dipole vector are assumed to be uncorrelated.

1.3.12 Polymeric Materials

Linear, polymeric organo-electronics typically exhibit different charge transfer characteristics in parallel or perpendicular directions to the backbone. Polymer chains can be defined in the layer morphology, assigning each site to a particular chain. Inter- and intra-chain charge transfer prefactors are then appended to the hopping frequencies:

$$\nu_{\parallel} = \mu_{\parallel} \nu \quad \wedge \quad \nu_{\perp} = \mu_{\perp} \nu$$

1.3.13 Alternating Voltage

By default, Bumblebee operates under a direct current. An alternating current may also be specified in terms of a modulated signal on top of the bias voltage. This signal is described by an amplitude, angular frequency, phase shift and a time offset relative to the start of the simulation. To improve sampling statistics for high-frequency signals, an AC update interval may be specified to dilate signal updates during the kMC sampling.

1.3.14 Photoluminescence

Photoluminescence simulations can be performed by enabling exciton generation following photo-absorption. For each material in the stack, an absorption coefficient is specified as an excitation probability. A static device exposure is defined in terms of the number of incident photons per second per cubic nanometer of material.

Excitations can be defined to give rise to various absorption products. Excitons, exciplexes or polarons may be specified. For exciton generation, a singlet-triplet ratio can be used to describe the spin statistics of the excitation.

1.3.15 Lifetime Simulations

In order to model the effect of molecular degradation on the lifetime device performance, sites may be deactivated following certain opto-electronic events. These include:

- Exciton generation
- Exciton annihilation
- Polaron quenching
- Polaron hopping
- Photo-absorption

Each event has a fixed probability to result in degradation of the material. When degradation occurs, the layer composition is altered by replacing the original material with a degraded material specified by the user. This mechanism can also be used to specify opto-electronically induced chemical reactions inside the stack.

1.3.16 OFET

The OFET model assumes that the electrode contacts act as gates. The source and drain are placed perpendicular to the gate field. A voltage can be applied between the source and the drain.

During OFET operation, a gate field is applied between the electrodes. Both single- and dual-gate devices can be modeled. A zero-potential reference is introduced in order to confine the gate field to the OFET interior. (By default, an insulator is placed at the cathode.) Corrections to the gate field are applied to preserve this potential. Minor field fluctuations are typically ignored by specifying a response threshold.

It is possible to select one of two control methods for preserving the potential:

- Modification of the external gate field
- Charge carrier injection

To inhibit rapid fluctuations in the device model due to these potential adjustments, gate updates are typically only performed at set intervals. The magnitude of the external field updates is restricted to a fixed field strength stepsize. Carrier injections are limited to one polaron per update.

1.3.17 Transient Responses

kMC can be used to observe the change in OLED behavior upon some external stimulus. It is possible to specify a perturbation to the system which takes place after some initial startup time.

- Modify the voltage, temperature, fluence of electrode Fermi levels
- Toggle degradation events
- Match the device voltage to a target current

Multiple perturbations can be defined (at different time offsets) to model more complex responses.

1.3.18 Simulation Volume

Bumblebee assumes a 1 nm distance between sites. The thickness of the stack is obtained as the sum of the layers. The (rectangular) surface area of the cell can be specified through the edge lengths.

Periodic boundary conditions may be specified to model bulk material behavior. This setting preserves the bias voltage of the electrodes, but disables external charge exchange.

1.3.19 Morphology

When the layer composition is made up of a mixture of different materials, a random distribution of components in the layer is assumed. A clustering modification may be defined to mimic self-aggregation.

Advanced compositions can be defined in order to generate more complex distributions. Linear and trapezoidal gradients are provided to specify one-dimensional material distributions. Multiple gradients can be combined to define the final morphology. A background material is introduced by default to assure compositional fractions always sum to unity.

The linear profile provides the average composition along the device stack. The distribution of components between channels remains random.

For polymeric materials, a polymer chain morphology can be defined. Chain growth is simulated using a self-avoiding walk. To define the morphology, an anisotropic propagation rate is defined along with the desired chain length. New chains are added to the layer until the polymer fraction is filled. Backbone rigidity can be specified to lock backwalking until a minimum number of growth steps has occurred.

1.3.20 Annealing

In order to simulate the voltage ramp that occurs following circuit closure, an annealing stage can be added to the simulation.

$$V(t) = V + \left(N_{anneal} - \frac{t}{T_{ramp}} \right) \left(\frac{V_{anneal} - V}{N_{anneal}} \right)$$

A stepped ramp function is used to increment the voltage within a timeframe of N_{anneal} simulation steps. The voltage is updated in T_{ramp} increments.

During OFET simulations, the voltage ramp is applied to the transistor field instead.

1.3.21 Pre-equilibration

Kinetic Monte Carlo analyses trajectory data to determine the properties of the OLED device. These trajectories depend on the initial state of the system. Typically, the properties of interest are obtained at equilibrium, while the initial state may be off-equilibrium. This introduces a bias in the trajectory statistics.

A pre-equilibration stage may be specified in the simulation settings to allow the system to de-correlate from the initial state, approaching the equilibrium regime. Sampling statistics will then only be generated for the samples obtained after pre-equilibration.

1.3.22 Simulation Settings

Timescale

A Bumblebee simulation typically includes a large number of processes, spanning various timescales. To simplify the input, a reference timescale is introduced. This allows many of the rate constants to be re-scaled to values close to unity.

For rates described using probabilistic distributions (such as charge hopping or exciton annihilation), prefactors are expressed in units of the reference timestep.

Note: Assuming a reference timestep of 10^{-11} s, a prefactor of 0.9 would correspond to a process with a frequency of $0.9 \cdot 10^{11} \text{ s}^{-1}$.

Fixed-rate processes may be provided in natural units, and are re-scaled internally:

- ISC/RISC
- Radiative and non-radiative exciton decay
- Photoluminescent fluence

Note: Assuming a reference timestep of 10^{-11} s, an ISC rate of 10^8 s^{-1} is converted to a probabilistic rate of 10^{-3} occurrences per simulation step.

By adjusting the reference timeframe of the simulation, this changes the magnitude of the rate values used internally by Bumblebee. Using rate values closer to 1 can improve the numerical stability of the simulation. For typical OLED devices, it is recommended to use the default settings.

Warning: Changing the reference timescale requires updating all the rate constants in the input. (The fixed rates remain unaltered.)

Parallelization

The accuracy of the statistical estimates provided by the kMC method depends on the number of samples. Typically, a very large number of samples is required to obtain reliable results. In order to improve performance, parallel sampling can be performed by selecting multiple simulation volumes. Samples from these parallel trajectories can be collated to obtain statistics for the aggregate dataset.

Output

As the kMC simulation progresses, the statistical estimate of the device performance is iteratively improved. In theory, the sampling could continue until all states have been exhausted. In order to define a practical termination condition for the simulation, a maximum number of sampling steps is defined.

Alternative conditions are also provided:

- The simulation can terminate once a certain lifetime has been evaluated. (The time progression for each sample in the simulation is obtained as the inverse of the process frequencies)
- The simulation can terminate once a homogeneous charge density has been achieved. This is expressed in terms of the normalized midrange of the 1D current profile:

$$\frac{I_{max} - I_{min}}{I_{avg}} < \tau_{conv}$$

- The simulation can terminate once all the excitons and/or charge carriers have been depleted. (As may be of interest for e.g. open circuits)

Simulation progress is written to the output files at fixed intervals. Default settings in the web interface automatically configure the relevant output. Users may modify the requested properties as desired. The simulation time increases as more output files are enabled, on account of the time needed to write them.

The output frequency may be configured separately for output regarding sampling statistics, or output related to device profiles.

For simulations that are dominated by high-frequency events, small time intervals will be used by the kMC simulation. A larger number of significant digits for the timestamps may therefore be enabled in the output settings.

Acceleration

A rate booster is available to promote anisotropic charge hopping along the electrode current. This accelerates the sampling of charge transfer through the stack. Note that these parameters modify the intrinsic rate distributions specified earlier. As the bias affects the sampling process, this invariably has an effect on the resulting statistics. Care should be taken that the boosting method preserved the relative rates encountered in the unbiased simulations.

Memory Usage

Bumblebee has to store all polaron and exciton species in the simulation volume. Since most OLED devices operate under sub-saturated carrier densities, a maximum number of carrier species may be specified to reduce memory consumption.

Dynamic memory allocation is performed whenever this limit is exceeded, such that simulations are never terminated prematurely. As additional allocations are quite slow compared to the normal initialization, it is recommended to set memory limits appropriately.

Reproducibility

The stochastic sampling process occurs through the use of random number generation. If reproducibility of the simulation trajectories is desired, a fixed RNG seed may be specified.

INSTALLATION

These pages document the installation process for Bumblebee. Note that Bumblebee consists of multiple modules:

- The Bumblebee executable, responsible for performing the calculations
- A web interface for creating, managing and analyzing your simulations
- Script utilities for connecting the web interface to a workstation or compute cluster

The web interface can be obtained from the [downloads](https://www.scm.com/downloads/bumblebee) (<https://www.scm.com/downloads/bumblebee>) page. Once installed, links to obtaining the executable and script utilities can be accessed from within the web interface.

2.1 Downloading Bumblebee

Installation files for Bumblebee are provided as a tar archive on the [downloads](https://www.scm.com/downloads/bumblebee) (<https://www.scm.com/downloads/bumblebee>) page.

- Select the correct version of the Bumblebee installer from the list and click the download button to access the files
- You will be prompted to verify your SCM username and password. These credentials were provided to you as part of the Bumblebee license. If you are having trouble accessing the downloads, please contact our [support team](https://www.scm.com/contact-us/) (<https://www.scm.com/contact-us/>)
- You should obtain a *bbweb.tar* archive containing the web interface installation. You can use your system file explorer to access the files in the archive
 - *bbweb.image* contains the web interface itself
 - *bbweb-kube.yaml* is the configuration file for the web interface
 - *Bumblebee.pdf* contains the offline version of the documentation

Note: Bumblebee is provided as a separate installation from AMS. AMS provides multi-scale OLED workflows for the determination of material parameters used in Bumblebee simulations. Instructions for the installation of these workflows can be found in the [AMS Manual](#).

2.2 Web Interface

The web interface is provided as a container application, allowing users to run their own Bumblebee instance on a local system. A container manager is necessary to launch the web interface. The following instructions use Podman.

- Download the [Podman Desktop installer](https://podman-desktop.io/downloads) (<https://podman-desktop.io/downloads>) and follow the OS-specific instructions on the website (<https://podman-desktop.io/docs/installation>) to install Podman Desktop
 - On Windows, Podman Setup may ask you to install WSL2. If WSL2 is not detected after installation, you may be required to restart your computer before resuming the setup
 - A Windows Firewall message may pop up, asking whether to allow Podman access to the network. You only need to enable this if you want access to the web interface from other computers
 - Optional: It is recommended to enable the Autostart setting for Podman. This skips having to launch the Podman engine every time you open Podman Desktop. If you want to change this setting later:
 - * Go to the Extensions tab
 - * Click on the Podman extension
 - * Click on the file icon labeled *Edit properties of Podman extension*
 - * Enable/disable autostart
 - On Windows, you will see the message: *We could not find any Podman machine*. Podman needs to setup a virtual machine in WSL before launching the container:
 - * The name does not matter, you may use the default
 - * The image path can be left empty
 - * Bumblebee does not require a machine with root privileges. It is recommended to keep this setting disabled to improve security
 - * User mode networking is disabled by default. This setting is only required when you are connecting through a third-party VPN
- Open Podman Desktop. On the Dashboard page, there should be a Podman widget with the *Running* status. If this is not the case, follow the instructions in the previous step to set up Podman
- Go to the Images tab in Podman Desktop
- Click the Load button and select the *bbweb.image* file obtained from the *bbweb.tar* [download](https://www.scm.com/downloads/bumblebee) (<https://www.scm.com/downloads/bumblebee>)

Note: Podman has both Load and Import options. The Load option has to be selected when adding the container.

- Go to the Pods tab in Podman Desktop
- Click the button labeled *Play Kubernetes YAML* and select the *bbweb-kube.yaml* file obtained from the *bbweb.tar* [download](https://www.scm.com/downloads/bumblebee) (<https://www.scm.com/downloads/bumblebee>)
 - If you encounter an HTTP or server error when playing the YAML file, you may have a firewall enabled, blocking connections to the container registry. Contact your IT department or SCM support to resolve this issue
- Wait until all indicator lights turn green
 - When restarting Podman Desktop, you may need to click on the *Start Pod* button (indicated by the play icon) before the web interface will load
- Click on the name of the pod (podbbweb)

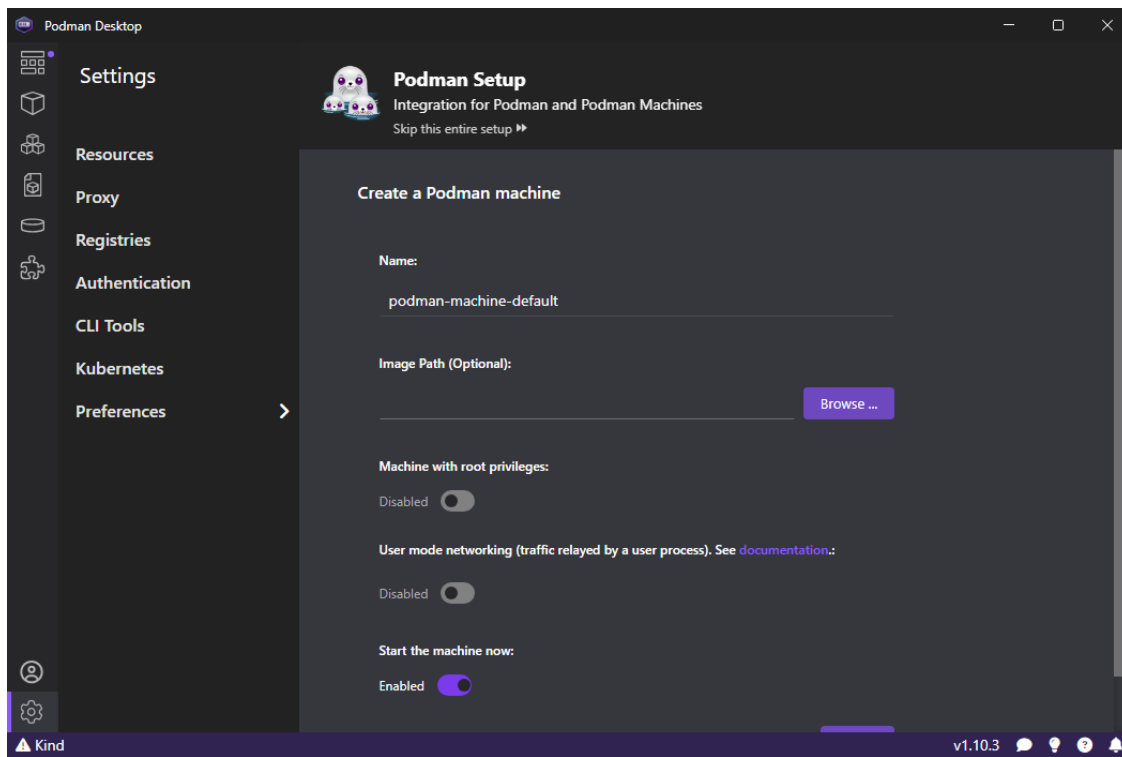


Fig. 2.1: Podman machine settings for WSL

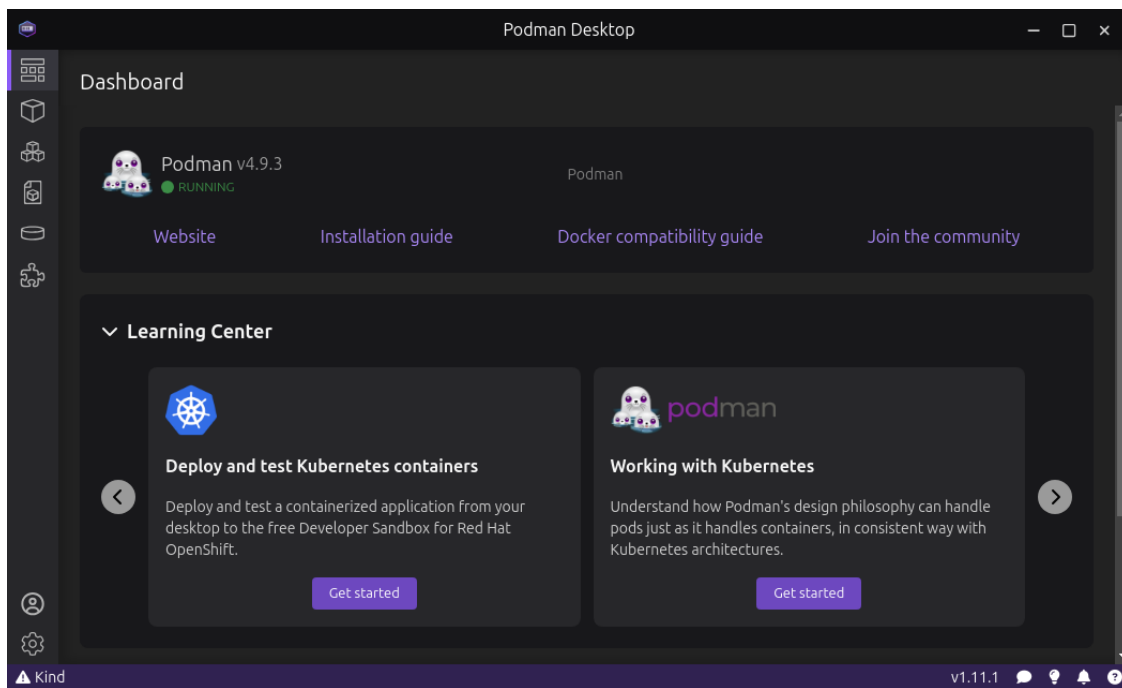


Fig. 2.2: Configured Podman

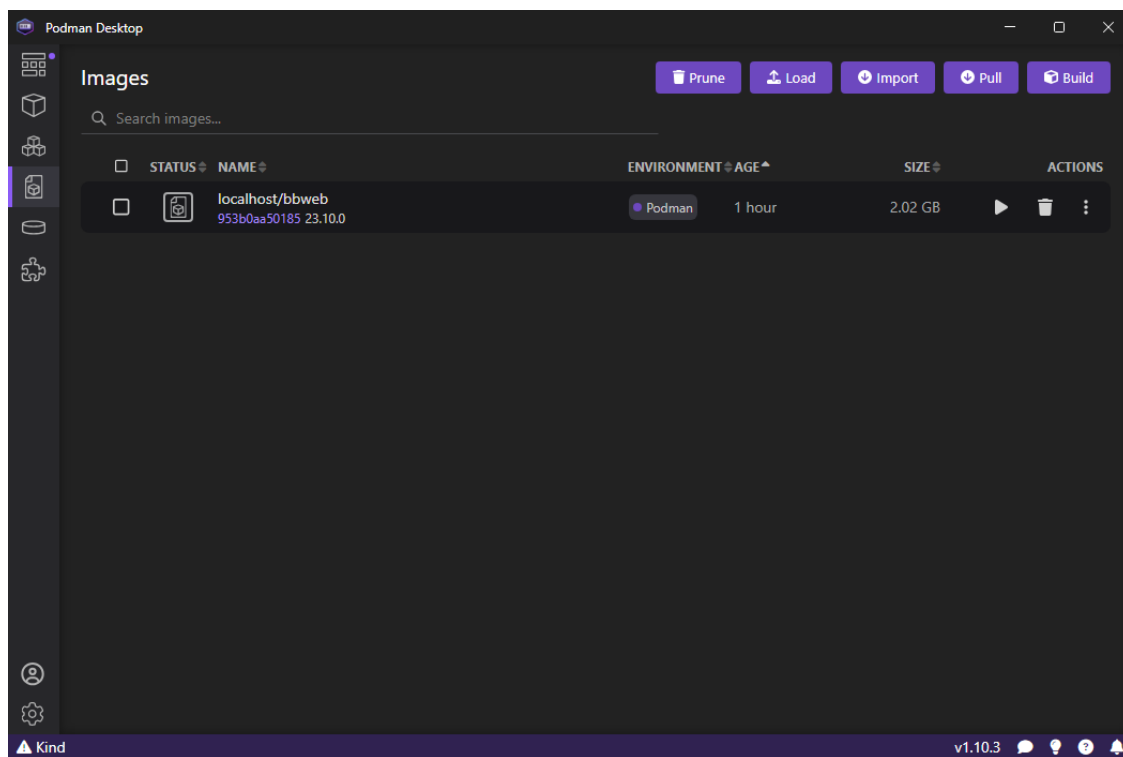


Fig. 2.3: After loading the image

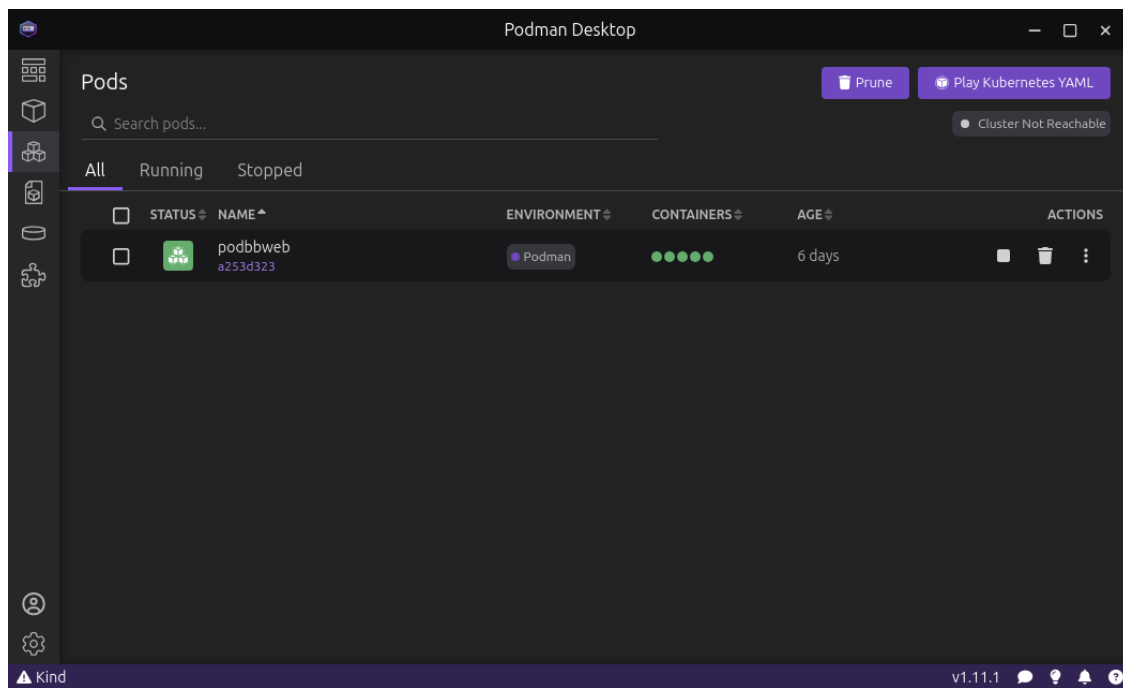


Fig. 2.4: Web interface is running

- Click on the Open button (maximize icon) to launch the web interface
- If this is the first time opening the web interface, you will be prompted to create an admin account. This account is stored locally on your computer and is not shared with SCM

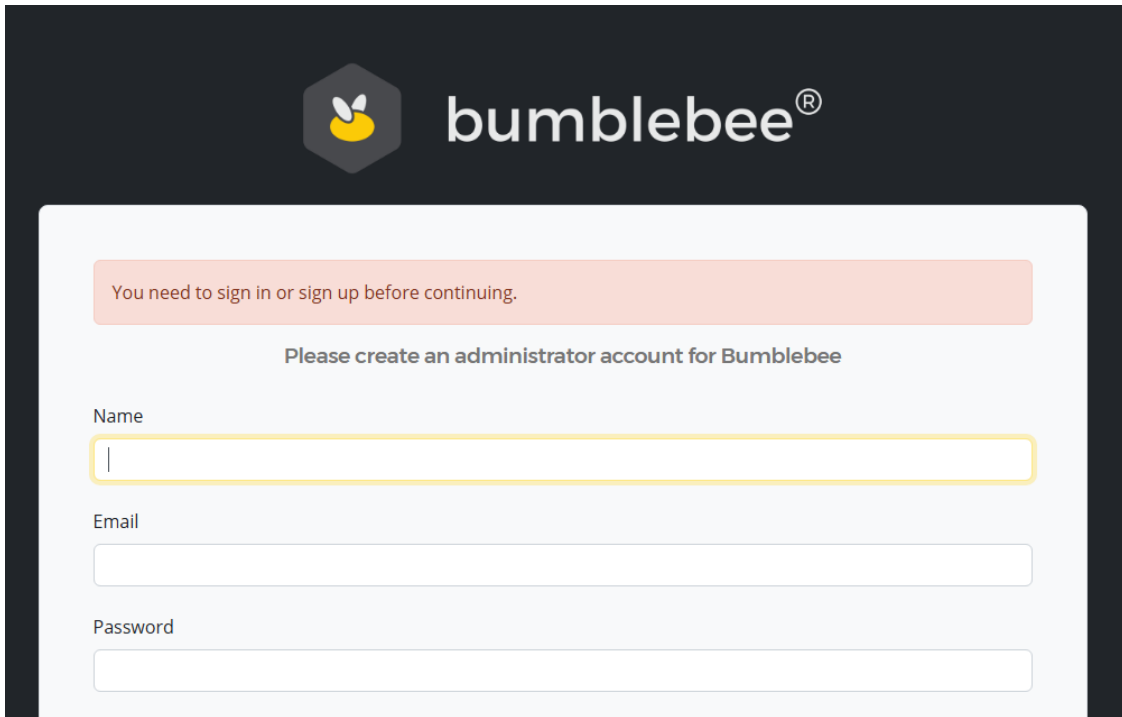
The screenshot shows the Bumblebee web interface. At the top, there is a dark header with the Bumblebee logo (a yellow bee inside a hexagon) and the text "bumblebee®". Below the header, a light gray box contains a pink message: "You need to sign in or sign up before continuing." Below this message, the text "Please create an administrator account for Bumblebee" is displayed. There are three input fields: "Name" (with a yellow border), "Email", and "Password".

Fig. 2.5: Create an admin account on your first login to the web interface

2.3 Bumblebee

The Bumblebee executable can be obtained from the Downloads tab in the web interface. Clicking on your username will open a menu, from which you can access Downloads.

The Downloads page contains a button to download the binary.

2.3.1 Linux

The provided executable is compiled for Linux systems and should work out-of-the-box. It may be necessary to enable executable permissions:

```
chmod +x bumblebee
```

To confirm that the executable is compatible with your system, simply run the following command from the terminal:

```
./bumblebee --version
```

You may get a message indicating that the license file is missing. Bumblebee uses the AMS license by default. Please follow the instructions on the [AMS page](#) to configure the general license.

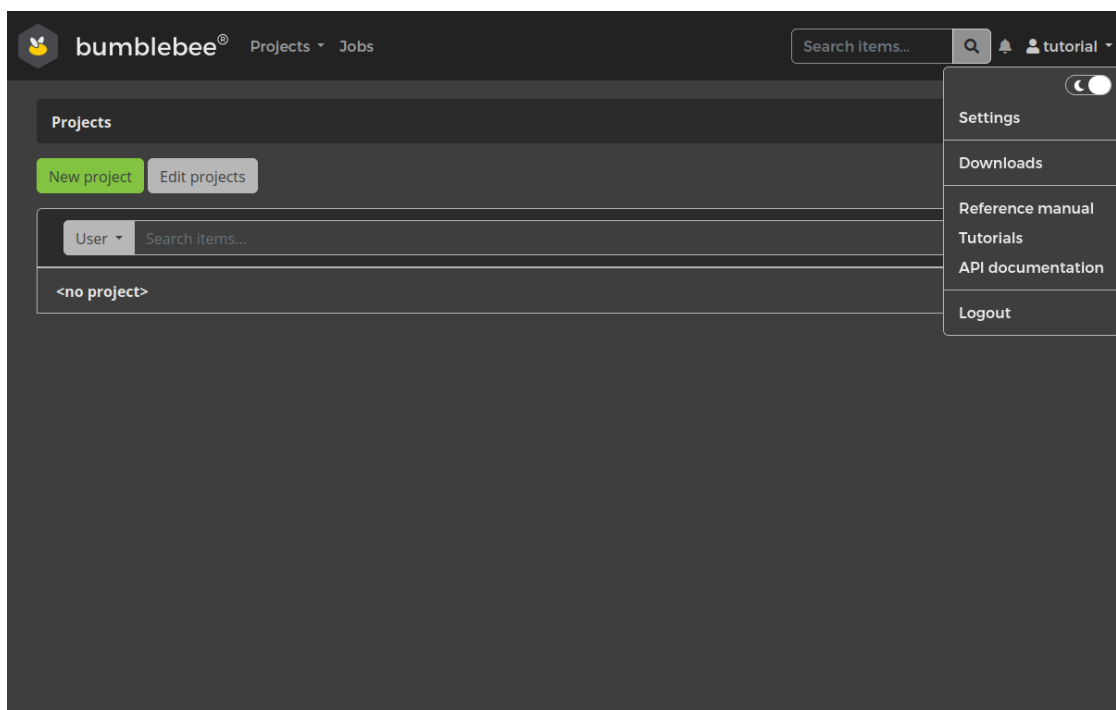


Fig. 2.6: Access Downloads in the web interface

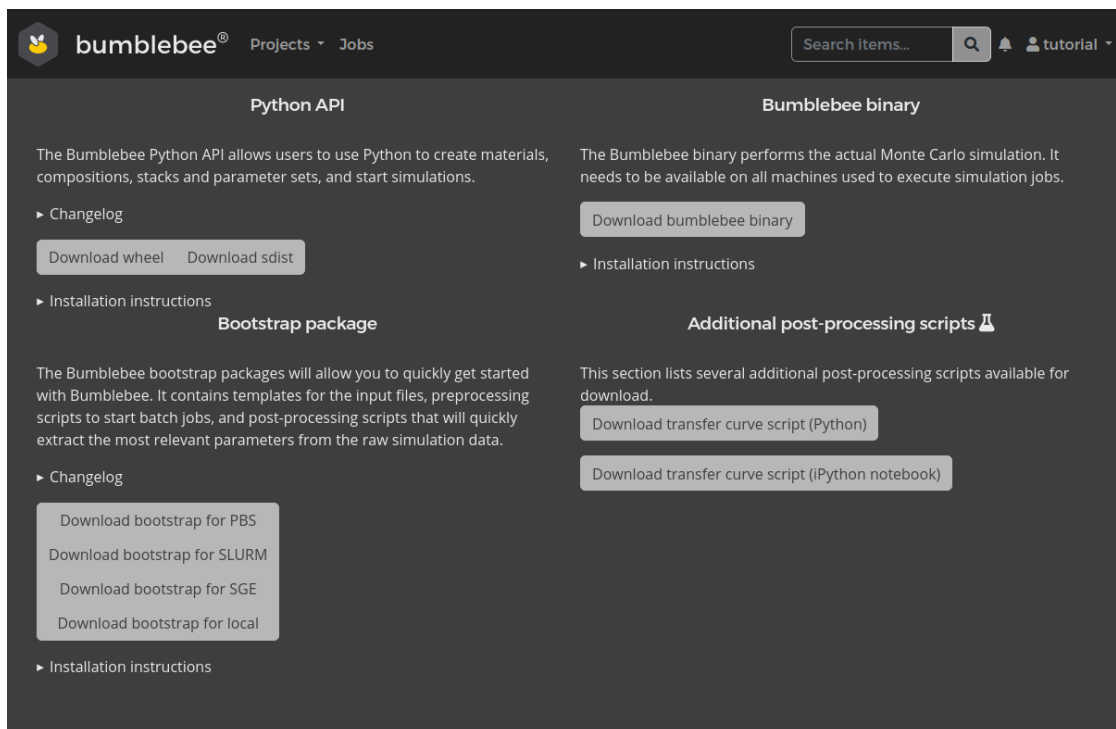


Fig. 2.7: Downloads page containing the Bumblebee binary

If your license only includes Bumblebee, you will receive an email with instructions for obtaining a dedicated license file. Bumblebee assumes that this file can be accessed from the SCMLICENSE location. You may configure the license location by adding the following line to the `~/.bashrc` file (first, make sure that the path is correct):

```
export SCMLICENSE=$HOME/license.txt
```

You can add it directly to the `~/.bashrc` file using this command:

```
echo 'export SCMLICENSE=$HOME/license.txt' >> ~/.bashrc
```

Note that you may need to adjust the path to the location where you installed the license file. Make sure to source the configuration file for these changes to go into effect.

```
source ~/.bashrc
```

Warning: Do not change the name of the *license.txt* file itself.

2.4 Connecting to a Server

In order to submit calculations automatically through the web interface, a connection must be established with the machine running Bumblebee. This process requires 2 steps:

- Install Bumblebee Bootstrap on the server
- Setup server connection in the web interface

2.4.1 Server Setup

The Bumblebee executable has been installed on the server. In order for the web interface to use your Bumblebee installation, the Bootstrap script utilities must now be installed.

Linux (cluster)

The Bootstrap scripts are currently designed to work with the Slurm workload manager (WLM). If you are using a different WLM, please adapt the scripts accordingly.

- Download the Bootstrap scripts from the Downloads page in the web interface
- Copy the *bootstrap.zip* file to the server
- Unzip the *bootstrap.zip* contents into a new folder. This will act as the working directory for the web interface, and will also contain the output files from your simulations

Bootstrap requires Python3 to be installed on the cluster, along with the numpy, scipy, yaml and numexpr packages. If you have administrative permissions, you can install these through a package manager such as apt:

```
sudo apt install python3 python3-yaml python3-scipy python3-numpy python3-numexpr
```

If you are not allowed to install packages yourself, contact your server admin.

If the queuing system on your cluster requires additional Slurm settings (such as account information) to be included during job submission, you may create a *qsub.conf* file in the preprocess directory.

```
cp preprocess/qsub.conf.example preprocess/qsub.conf
```

Any SBATCH directives specified in this file will be included in the default job scripts generated from the web interface.

Finally, the Bumblebee executable must be accessible for your jobs. The executable should be part of the server PATH. If you are not allowed to modify the PATH yourself, a default install location is assumed at `~/.local/bin/`. To check that Bumblebee is part of the PATH, you should be able to call the executable from any location on the cluster:

```
bumblebee --version
```

Linux (local)

It is possible to run Bumblebee on the same machine that is hosting the web interface. Follow the [Slurm installation instructions](https://slurm.schedmd.com/quickstart_admin.html) (https://slurm.schedmd.com/quickstart_admin.html) to configure the WLM on your computer. Then proceed through the same steps as described above.

Slurm configuration

- Install slurm

```
sudo apt install openssh-server slurm-wlm

sudo gzip -cd /usr/share/doc/slurmd/examples/slurm.conf.simple.gz > /etc/slurm/slurm.
↪conf
```

- Configure the Slurm settings

- Make sure the SlurmctldHost matches the hostname
- Make sure the Nodename matches the hostname
- Correctly set the CPU and Memory configurations of the Node

```
HN=$(hostname -s) && sudo sed -i "s|=workstation|=${HN}|g" /etc/slurm/slurm.conf

HN=$(hostname -s) && sudo sed -i "s|=server|=${HN}|g" /etc/slurm/slurm.conf

SC=$(slurmd -C | head -1) && sudo sed -i "s|^NodeName=.*|${SC}|g" /etc/slurm/slurm.
↪conf
```

- Start slurm

```
sudo systemctl enable slurmd slurmctld

sudo systemctl start slurmd slurmctld

sudo systemctl status slurmd slurmctld
```

- The status of both slurmd and slurmctld should be *active*. If this is not the case, check your configuration or consult the [Slurm manual](https://slurm.schedmd.com/quickstart_admin.html) (https://slurm.schedmd.com/quickstart_admin.html)

2.4.2 Connection Setup

Login Authorization

In order to allow Bumblebee to connect to your account, a key-based login must be enabled. On the Server page, click the button *Generate New Key Pair*. The key that is generated must now be added to the `~/.ssh/authorized_keys` file on the server. For example (adjust this to include your own key):

```
echo 'ssh-rsa AAAAB3NzaC1yc2= 3d808818-477a-45de-9f66-1e8eb5abd061' >> $HOME/.ssh/.
↪authorized_keys
```

If the `.authorized_keys` file did not previously exist, make sure that the file permissions are set to mode 600.

```
chmod 600 ~/.ssh/authorized_keys
```

Warning: Both the public and private keys generated by Bumblebee are stored in a user-accessible folder inside the container.

For WSL users: note that files stored under WSL are exposed to the Windows user account. While access protection still applies by default, administrative privileges on the Windows account-level can be used to override these restrictions.

If you believe your private key may have been compromised:

- Remove the old key from the `authorized_keys` on the server
- Generate a new key in the web interface
- Authorize the new key on the server to re-establish the connection. You can confirm the connection status in the server settings of the web interface

Connection Settings

In the Bumblebee web interface, go to the Settings menu and select the Server tab.

Click on the *Add Server* button to configure a new connection.

- Choose any server name. This is the name that will be shown in the web interface when selecting the server for job submission
- Enter the hostname of the login node of your server
 - If you are running Bumblebee on the same computer as the web interface, you may use `host.containers.internal` instead
- Select an open port for ssh connections to the server. (The default is port 22. If your server login requires a specific port, adjust as needed.)
- The username must be set to your login name on the server
- *Folder* should be the path to the folder containing the Bootstrap scripts (relative to the Home directory on the login node)
- The backend should be set to remote
- Set the name of the job submission queue on the cluster
- Click the *Check Connection* button to test your connection. A check mark should appear to indicate a successful setup

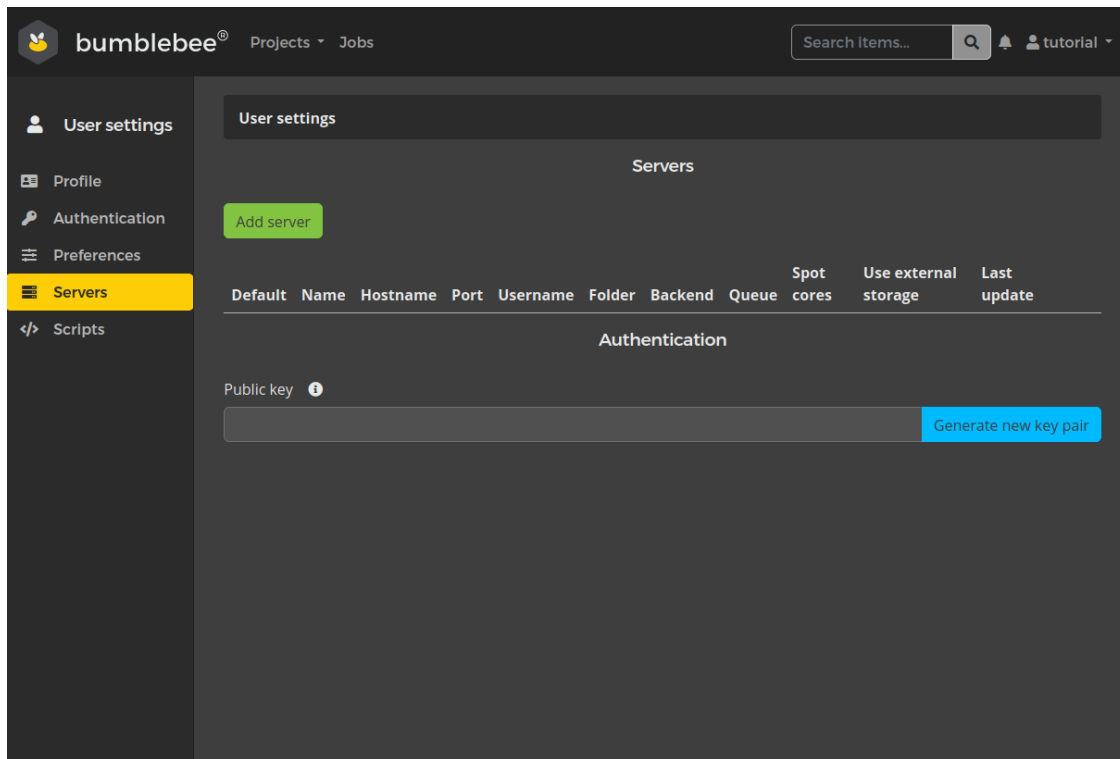


Fig. 2.8: Server settings page

If there is a problem with the connection, a warning symbol will appear. Hover over the symbol to view the error message.

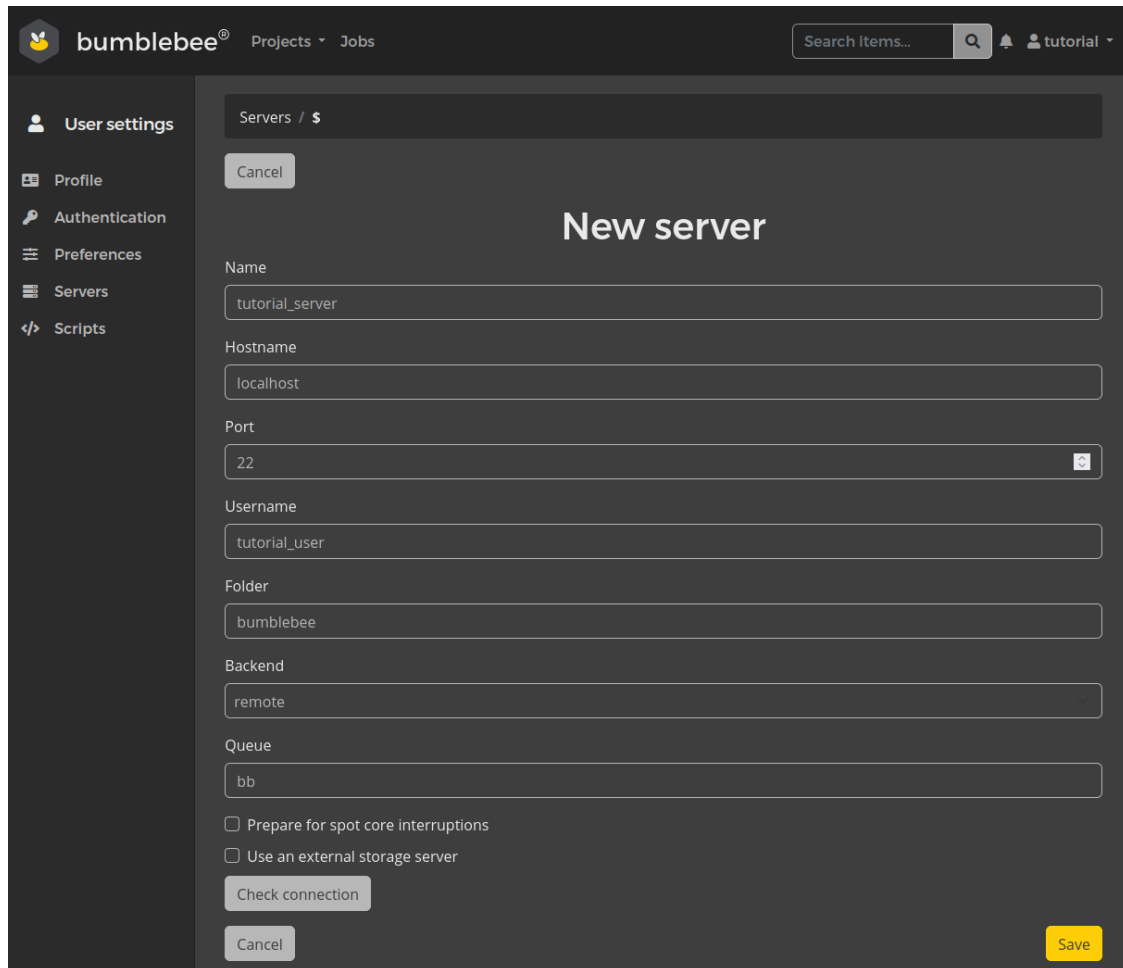
- If a connection could not be made:
 - Check that your computer is connected to the network
 - Check that the server is connected to the network
 - Check that the hostname is correct

```
ping hostname
```

- If the connection was refused:
 - Check that your username is correct
 - Check that incoming ssh connections on the server are allowed on the selected port

```
ssh -p 22 username@hostname
```

- If the Bootstrap version could not be verified:
 - Check that the *Folder* has been set correctly
 - * Navigate to the Bootstrap folder
 - * Run the *pwd* command to obtain the absolute path
 - * Copy the absolute path to the *Folder*
 - Check that the public key has been added to the *authorized_keys*
 - The *authorized_keys* file must have its permissions set to 600



The screenshot shows the Bumblebee application interface with a dark theme. On the left is a sidebar with 'User settings' selected, containing links for Profile, Authentication, Preferences, Servers, and Scripts. The main window has a top bar with the Bumblebee logo, 'Projects' and 'Jobs' dropdowns, a search bar, and a user profile 'tutorial'. Below the top bar, a breadcrumb 'Servers / \$' is shown. The 'New server' dialog is open, featuring a 'Cancel' button at the top left. The dialog contains the following fields: 'Name' (tutorial_server), 'Hostname' (localhost), 'Port' (22 with a copy icon), 'Username' (tutorial_user), 'Folder' (bumblebee), 'Backend' (remote), and 'Queue' (bb). At the bottom, there are two unchecked checkboxes: 'Prepare for spot core interruptions' and 'Use an external storage server', followed by a 'Check connection' button, another 'Cancel' button, and a yellow 'Save' button.

bumblebee® Projects Jobs Search Items... Q tutorial ▾

Servers / \$

Cancel

New server

Name
tutorial_server

Hostname
localhost

Port
22

Username
tutorial_user

Folder
bumblebee

Backend
remote

Queue
bb

☐ Prepare for spot core interruptions

☐ Use an external storage server

Check connection

Cancel Save

Fig. 2.9: Configuring server settings

- The `~/ssh` folder must have its permissions set to 700
- Check that you have executable permissions on the login node
- If Bootstrap is installed on NFS, make sure that the NSF is mounted with executable permissions
- To confirm that you have the correct path and permissions, navigate to the *Folder* on the cluster. You should be able to run the following command:

```
./preprocess/bumblebee-bootstrap-version -l
```

- If there are dependency issues:
 - Make sure that Python is installed on the cluster
 - Check that all of the required Python packages are installed
 - Your Python version may be incompatible with Bootstrap
 - * Install a different Python version
 - * Install the required packages for the new Python version
 - * Make the new Python version the default when using Bumblebee
 - If Bootstrap files have been deleted, re-install Bootstrap by downloading the files from the web interface

2.4.3 Testing the Installation

See the Tutorials page in the web interface to setup your first simulation. The simulation status should change to *submitted* to indicate that your job is now running on the server.

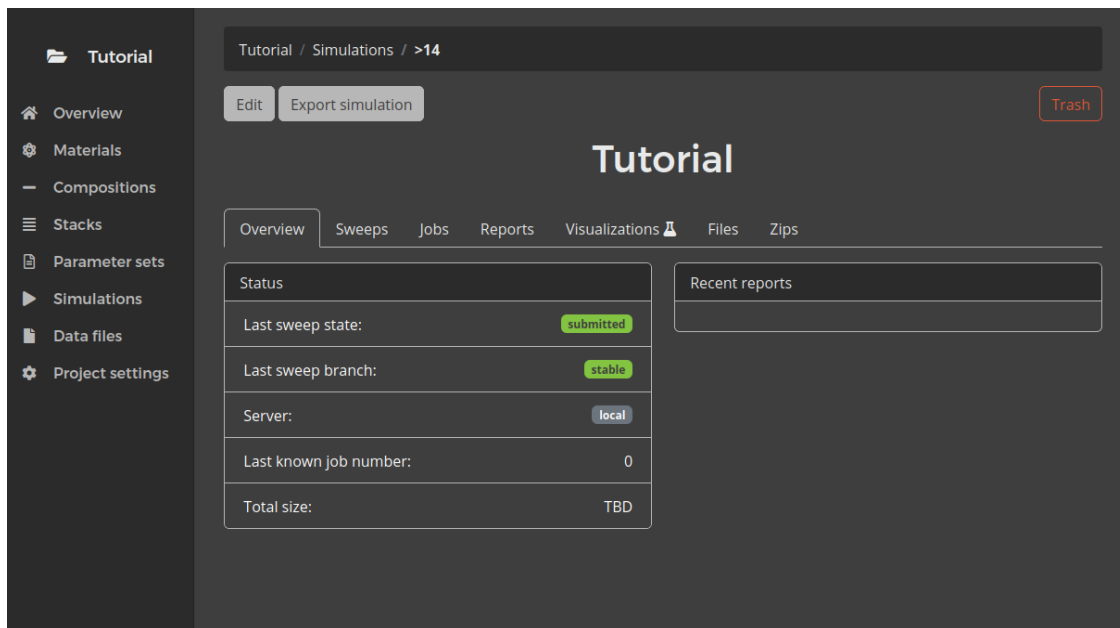


Fig. 2.10: Successful job submission

You can go to the Jobs page in the web interface to see the status of your ongoing simulations.



Fig. 2.11: Overview of active simulations

2.5 FAQ

If job submissions fail to run, this indicates a server-side issue.

- First verify that the server connection is still active by checking the server status on the Settings page of the web interface
- The pod may not have network access. On some operating systems, Podman disables pod connections if your machine does not have network access when starting Podman. This error can occur even when running Bumblebee locally. Re-start the pod to resolve this issue
- The partition selected in the web interface server settings may not exist, or is currently DOWN. Login to the server and run the *sinfo* command to check the partition status
- Key-based login may be disabled on the server. Contact your server administrator if this is the case
- Check the simulation output for error information. You can access these files by logging in to the server and navigating to the job folder, or through the web interface by selecting the Simulation and navigating to the Files tab. You may need to click the Refresh button to load the files
 - *error.log* files document error messages from the server
 - *std.err* files report on error messages from Bumblebee. These errors include issues with your input settings or SCM License
 - *std.out* reports the output of the Bumblebee simulation

If your jobs are successfully submitted, then immediately exit:

- Check the simulation output for error information, as detailed above
- You may have incorrectly set the SCMLICENSE location in the *preprocess/bumblebee.qsub* file. Check the *output.log* file to see the license configuration
- Your license may have expired. On the server, run the following command to check for license errors:

```
./bumblebee --version
```

- Necessary SBATCH settings may be missing from the *qsub.conf* file. Check the *std.err* file for Slurm messages
- Bumblebee may not be in the PATH on the server. Check the *output.log* file to see if the executable could be found
- *.bashrc* settings may not be loaded automatically when submitting jobs through the web interface, as the session is non-interactive. Either adjust the *.bashrc* file to enable settings for non-interactive login, or add necessary calls to the *preprocess/bumblebee.qsub* file

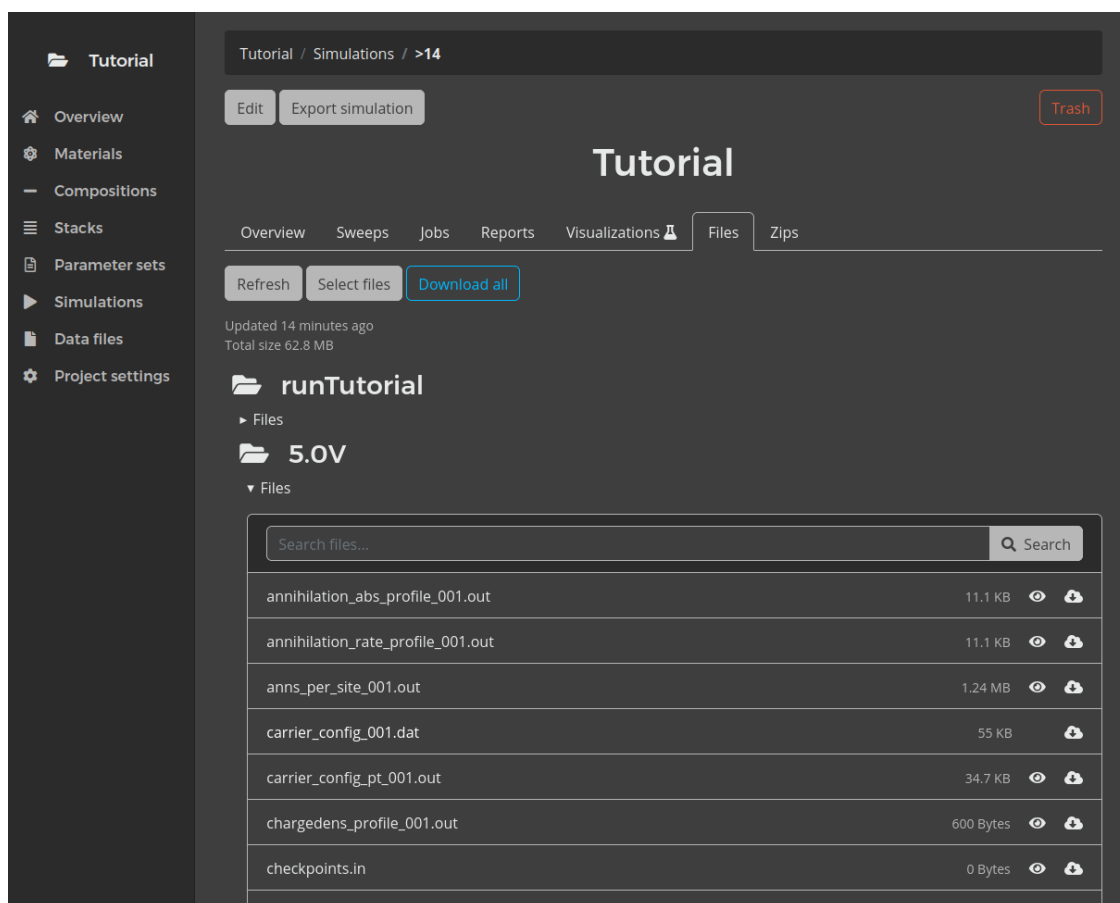


Fig. 2.12: View simulation output files in the web interface

3.1 Graphical User Interface

The web interface manages the creation, submission and analysis of Bumblebee simulations.

The materials tab allows definition of molecular properties for the various materials in the OLED stack layers. Composite layers can also be defined by using nanoscale distributions of multiple materials.

Materials and compositions may be used to construct OLED stacks by combining various layers. Inter-layer transfer processes can be defined based on the molecular layer properties.

Simulation settings are configured as parameter sets. Job submissions and output visualization are handled in the simulations tab.

Various presets are provided to aid the simulation setup.

3.1.1 Materials

Several template settings are available to expedite the configuration of new materials.

- Host layers allow for radiative decay of both singlet and triplet species
- Transport layers do not emit light
- Fluorescent dyes assume all singlet decay processes to be radiative
- Phosphorescent dyes assume all excitons to be triplets. All decay processes are assumed to be radiative
- TADF dyes assume triplet decay to occur through ISC
- Optical layers do not participate in charge carrier processes and are included only to enable calculation of optical device parameters
- Advanced materials disable all presets

3.1.2 Compositions

- Basic compositions are defined as a mixture of materials with a specific ratio. Components are distributed randomly within the layer. Mesoscopic layer properties are computed as a stoichiometric average of the material parameters
- Advanced compositions allow usage of gradients to construct more complex distributions, as well as the creation of polymeric morphologies

3.1.3 Parameter Sets

Template parameter sets are provided to automatically configure modules and commonly used settings for the various simulation types facilitated by Bumblebee.

- Single voltage points allow simulation of excitonic processes at DC
- Voltage sweeps automatically configure the voltage as the simulation sweep parameter. Hole-only and electron-only versions allow specification of a fixed carrier density
- Periodic box simulations enable PBC but disable excitonics
- Lifetime simulations include device degradation processes
- Photoluminescence simulations model current generation upon light exposure. PBC are enabled. Grid densities are doubled compared to other defaults to account for polaron interactions
- Photovoltaic and photodetector simulations model photoluminescence processes under closed circuit conditions
- Transistor simulations are used to model OFETs
- Advanced simulations disable presets

Photoluminescent, photovoltaic and photodetector presets enable transient output files by default.

3.2 Visualization

After a simulation has completed, reports can be generated in the web interface.

- Box reports displays the output of individual kMC trajectories
 - Morphology shows the distribution of materials in the stack
 - Convergence shows the transient current profile
 - Box overview shows the transient carrier densities, photo-electronic process frequencies and device potential
- Multibox reports collect output of multiple kMC trajectories
 - Transient photoluminescence reports on the radiative excitation rates and exciton densities
 - Material profiles report on carrier densities and photo-electronic process frequencies for individual layer materials
 - Optical reports on outcoupling efficiencies for individual layer materials
 - Transient OLED responses report aggregate statistics on transient current profiles, carrier densities, photo-electronic process frequencies, voltages and device degradation
 - OLED reports show the average carrier densities, photo-electronic process frequencies and device potential
 - OLED emission shows the computed emission spectrum
 - Convergence reports aggregate statistics on the transient current profiles and integrated quantum efficiency

- Sweep reports display the device variations when varying external parameters (such as the voltage)
 - Profiles report on the carrier densities and photo-electronic process frequencies
 - OLED reports yield the voltage-current profiles, integrated quantum efficiency, external quantum efficiency, voltage-dependent process frequencies and carrier densities
 - OLED luminance reports the effective device luminance for the CIE 1931 and CIE 1978 standard eye responses, in addition to the power efficiency
 - Excitonic events report on the relative process frequencies
 - Effective mobility reports on the voltage-dependent polaron mobility
- Optical reports display the optical output from the device
 - Purcell displays the anisotropic Purcell factors
 - Output intensity shows the polarized outcoupling intensity
 - Outcoupling efficiency reports on the optical effectiveness

The available reports are shown depending on the output generated by the simulation. Parameter set templates automatically configure output for relevant processes.

3.3 Input Files

The web interface automatically generates the input files necessary to run Bumblebee simulations. Relevant data can be found in the following files:

- *params* contains the simulation settings (materials, compositions, stack, parameters)
- *morphology* is used to store the component distributions in the layers
- *dipoles* is used to store the dipole field
- *checkpoints* documents the transient responses

3.4 Output Files

Bumblebee output is split across various files.

- Human-readable *.out* files detailing the computed device parameters or simulation statistics
- Binary *.dat* files contain trajectory snapshots that can be analyzed to investigate the individual kMC samples. 64 bit floating point precision is used

Visualization and analysis of these files is handled automatically by the web interface. An overview of the files is provided here if custom visualization, analysis or post-processing is desired.

General output:

- *output.log* documents the progress of the Bumblebee simulation
- *disorder_result.out* stores the simulation summary for the final Monte Carlo step
- *time_progress.out* summarizes the simulation progress results
- *param_overview.out* records the input parameters after Bumblebee pre-processing and input sanitation

Time-averaged device profiles:

- *elec_profile.out* and *hole_profile.out* report the time-average cross-sectional polaron charge. The combined charge profile is stored in *chargedens_profile.out*
- *mat_elec_profile.out* and *mat_hole_profile.out* contain the time-average cross-sectional polaron charge per material. Separate files are generated for each material
- *singlet_profile.out* and *triplet_profile.out* report the time-average cross-sectional exciton counts
- *mat_singlet_profile.out* and *mat_triplet_profile.out* contain the time-average cross-sectional exciton counts per material. Separate files are generated for each material
- *potential_profile_normal.out* contains the time-average cross-sectional voltage potential. *potential_profile_disccutout.out* reports the voltage potential after including the short-range Coulomb corrections. *potential_profile_incl_impot.out* reports the voltage potential after including periodic image charges. *potential_profile_disccutout_incl_impot.out* reports the voltage potential after including both the short-range Coulomb corrections and periodic image charges

Transient device profiles:

- *electroncurrent.out* and *holecurrent.out* report the polaron fluxes for each cross-section. The first entry on each line gives the timestamp. The net charge flux is reported in *layercurrent.out*
- *electroncurrentx.out* and *holecurrentx.out* report the polaron fluxes along the OFET gate for each cross-section. The first entry on each line gives the timestamp. The net charge flux is reported in *layercurrentx.out*
- *recombination_abs_profile.out* reports the cross-sectional radiative exciton recombination event count. The first entry on each line gives the timestamp. *recombination_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *generation_abs_profile.out* reports the cross-sectional exciton generation event count. The first entry on each line gives the timestamp. *generation_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *dissociation_abs_profile.out* reports the cross-sectional exciton dissociation event count. The first entry on each line gives the timestamp. *dissociation_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *thermal_generation_abs_profile.out* reports the cross-sectional thermal exciton generation event count. The first entry on each line gives the timestamp. *thermal_generation_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *electron_quenching_abs_profile.out* and *hole_quenching_abs_profile.out* report the cross-sectional electron and hole quenching event counts. The first entry on each line gives the timestamp. *quenching_abs_profile.out* reports the total quenching event count. *electron_quenching_rate_profile.out*, *hole_quenching_rate_profile.out* and *quenching_rate_profile.out* report the respective per-gridpoint, per-timestep averages inside each cross-section.
- *annihilation_abs_profile.out* reports the cross-sectional exciton annihilation event count. The first entry on each line gives the timestamp. *annihilation_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *nonrad_decay_abs_profile.out* reports the cross-sectional non-radiative exciton decay event count. The first entry on each line gives the timestamp. *nonrad_decay_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *absorption_abs_profile.out* reports the cross-sectional photonic absorption event count. The first entry on each line gives the timestamp. *absorption_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *degradation_abs_profile.out* reports the cross-sectional degradation event count. The first entry on each line gives the timestamp. *degradation_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section

- *exciplex_decay_abs_profile.out* reports the cross-sectional radiative exciplex decay event count. The first entry on each line gives the timestamp. *exciplex_decay_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *exciplex_nonrad_decay_abs_profile.out* reports the cross-sectional non-radiative exciplex decay event count. The first entry on each line gives the timestamp. *exciplex_nonrad_decay_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section
- *exciplex_thermal_generation_abs_profile.out* reports the cross-sectional thermal exciplex generation event count. The first entry on each line gives the timestamp. *exciplex_thermal_generation_rate_profile.out* reports the per-gridpoint, per-timestep average inside each cross-section

Transient trajectories: (Note that these generate significant amounts of data)

- *transient_recombination_abs_profile.out* reports the cross-sectional radiative exciton recombination event count during each individual iteration
- *transient_quenching_abs_profile.out* reports the total cross-sectional polaron quenching event count during each individual iteration
- *transient_annihilation_abs_profile.out* reports the cross-sectional exciton annihilation event count during each individual iteration
- *transient_nonrad_decay_abs_profile.out* reports the cross-sectional non-radiative exciton decay event count during each individual iteration
- *transient_exciplex_decay_abs_profile.out* reports the cross-sectional radiative exciplex decay event count during each individual iteration
- *transient_exciplex_nonrad_decay_abs_profile.out* reports the cross-sectional non-radiative exciplex decay event count during each individual iteration
- *transient_exciplex_tgens_abs_profile.out* reports the cross-sectional thermal exciplex generation event count during each individual iteration

Locations of electronic species:

- *carrier_config.dat* and *carrier_config_pt.out* report on the charge carrier positions. For each species, the carrier coordinates are followed by the polaron type (0 indicates an electron, 1 indicates a hole)
- *exciton_config.dat* and *exciton_config_pt.out* report on the charge carrier positions. For each species, the carrier coordinates are followed by the exciton type (1 indicates a singlet, 3 indicates a triplet)

Properties per gridpoint:

- *composition.dat* and *composition_plaintext.out* record the material ID for each gridpoint inside the stack
- *disorder.dat* and *disorder_plaintext.out* report on the energetic disorder in the HOMO and LUMO levels at each gridpoint
- *excitonic_disorder.dat* and *excitonic_disorder_plaintext.out* report on the energetic disorder in the singlet and triplet levels at each gridpoint. *exciton_binding_energies.out* reports on the singlet and triplet binding energies at each gridpoint
- *kappa.dat* and *kappa_plaintext.out* report on the transition multipole orientations at each gridpoint
- *electrons_per_site.out* and *holes_per_site.out* report the time-average polaron charge for each gridpoint
- *singlets_per_site.out* and *triplets_per_site.out* report the time-average exciton counts for each gridpoint

A one-dimensional gridpoint indexing system is used. The gridpoint coordinates can be obtained from the (0-based)

index k as:

$$\begin{aligned}x &= k \bmod N_X \\y &= \frac{k}{N_X} \bmod N_Y \\z &= \frac{k}{N_X N_Y} \bmod N_Z\end{aligned}$$

Event log per gridpoint:

- *singlet_recomb_per_site.out* and *triplet_recomb_per_site.out* report the number of singlet and triplet recombination events that occurred at each gridpoint. *recomb_per_site.out* reports on the total number of recombination events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_gens_per_site.out* and *triplet_gens_per_site.out* report the number of singlet and triplet generation events that occurred at each gridpoint. *gen_per_site.out* reports on the total number of generation events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_dis_per_site.out* and *triplet_dis_per_site.out* report the number of singlet and triplet dissociation events that occurred at each gridpoint. *dis_per_site.out* reports on the total number of dissociation events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_quenchings_per_site.out* and *triplet_quenchings_per_site.out* report the number of singlet and triplet quenching events that occurred at each gridpoint. *quenchings_per_site.out* reports on the total number of quenching events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_anns_per_site.out* and *triplet_anns_per_site.out* report the number of singlet and triplet annihilation events that occurred at each gridpoint. *anns_per_site.out* reports on the total number of annihilation events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_nonrads_per_site.out* and *triplet_nonrads_per_site.out* report the number of non-radiative singlet and triplet decay events that occurred at each gridpoint. *nonrads_per_site.out* reports on the total number of non-radiative decay events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_iscs_per_site.out* and *triplet_iscs_per_site.out* report the number of intersystem and reverse intersystem crossing events that occurred at each gridpoint. *iscs_per_site.out* reports on the total number of crossing events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *singlet_tgens_per_site.out* and *triplet_tgens_per_site.out* report the number of thermal singlet and triplet generation events that occurred at each gridpoint. *tgens_per_site.out* reports on the total number of thermal generation events. Each line in these files contains the gridpoint coordinates, followed by the event count
- *abs_per_site.out* records the number of photonic absorption events that occurred at each gridpoint
- *degradations_per_site.out* records the number of degradation events that occurred at each gridpoint
- *exciplex_decay_per_site.out* records the number of radiative exciplex decay events that occurred at each gridpoint
- *exciplex_nonrads_per_site.out* records the number of non-radiative exciplex decay events that occurred at each gridpoint
- *exciplex_tgens_per_site.out* records the number of thermal exciplex generation events that occurred at each gridpoint

Transient event log:

- *eventlog.out* records all photo-electronic events. Switches are available to select (0) polaron hopping, (1) polaron injection at the electrodes, (2) polaron collection at the electrodes, (3) radiative exciton recombination, (4) exciton generation, (6) thermal exciton generation, (7) exciton hopping, (8) non-radiative exciton decay, (9) exciton annihilation, (10) polaron quenching, (11) exciton dissociation, (12) intersystem crossing, (13) photonic absorption, (14) degradation, (15) radiative exciton decay, (16) non-radiative exciton decay, (17) thermal exciplex generation, (18) polaron-induced exciton recombination and/or (19) transfer-induced polaron quenching. For each event, the

gridpoint, reactant and product are reported (0 for electrons, 1 for holes, 2 for singlets, 3 for triplets). For degradation events, the degradation mechanism and product material are reported instead (0 for exciton generation, 1 for polaron quenching, 2 for exciton annihilation, 3 for hole quenching, 4 for electron quenching, 5 for photonic absorption, 6 for polaron hopping)

- *emissionlog.out* records the location of individual exciton recombination events. For each event, the exciton type and energy are reported
- *degradationlog.out* records the location of individual degradation events. For each event, the degradation mechanism and degradation products are reported. The carrier is indicated as 0 for electrons, 1 for holes, 2 for a singlet, 3 for a triplet
- *avlog.dat* and *avlog.out* record the voltage and current under AC operation

Binary storage:

- *profiles.dat* is a binary storage for the properties of the stack. This file contains:
 - The Bumblebee version number
 - The charge carrier current
 - The electron current
 - The hole current
 - The OFET carrier current
 - The OFET electron current
 - The OFET hole current
 - The cross-sectional exciton recombination event counts
 - The cross-sectional exciton generation event counts
 - The cross-sectional exciton dissociation event counts
 - The cross-sectional thermal exciton generation event counts
 - The cross-sectional polaron quenching event counts
 - The cross-sectional electron quenching event counts
 - The cross-sectional hole quenching event counts
 - The cross-sectional exciton annihilation event counts
 - The cross-sectional non-radiative exciton decay event counts
 - The cross-sectional photonic absorption event counts
 - The cross-sectional degradation event counts
 - The cross-sectional radiative exciplex decay event counts
 - The cross-sectional non-radiative exciplex decay event counts
 - The cross-sectional thermal exciplex generation event counts
 - The time-averaged cross-sectional electron counts
 - The time-averaged cross-sectional hole counts
 - The time-averaged cross-sectional singlet counts
 - The time-averaged cross-sectional triplet counts
 - The time-averaged cross-sectional electron counts per material

- The time-averaged cross-sectional hole counts per material
 - The time-averaged cross-sectional singlet counts per material
 - The time-averaged cross-sectional triplet counts per material
 - The time-averaged electron counts at each gridpoint
 - The time-averaged hole counts at each gridpoint
 - The time-averaged singlet counts at each gridpoint
 - The time-averaged triplet counts at each gridpoint
- *rng.dat* records the final state of the random number generator. This file is used for continuation runs
- *stats.dat* records the event statistics. This file contains:
 - The Bumblebee version number
 - The number of elapsed Monte Carlo steps
 - The current simulation time
 - The last stepsize
 - The number of electron injections at the anode
 - The number of electron injections at the cathode
 - The number of hole injections at the anode
 - The number of hole injections at the cathode
 - The number of electron abstractions at the anode
 - The number of electron abstractions at the cathode
 - The number of hole abstractions at the anode
 - The number of hole abstractions at the cathode
 - The instantaneous device current
 - The instantaneous device current at the previous iteration
 - The number of exciton recombination events
 - The number of exciton recombination events at the previous iteration
 - The number of exciton generation events
 - The number of exciton generation events at the previous iteration
 - The number of photonic absorption events
 - The number of photonic absorption events at the previous iteration
 - The number of thermal exciton generation events
 - The number of exciton dissociation events
 - The number of polaron quenching events
 - The number of degradation events
 - The external field strength
 - The number of exciton annihilation events
 - The number of non-radiative exciton decay events

- The number of intersystem crossings
- The number of reverse intersystem crossings
- The normalized minimum cross-sectional current
- The normalized maximum cross-sectional current
- The total number of exciton recombination events per gridpoint
- The number of singlet recombination events per gridpoint
- The number of triplet recombination events per gridpoint
- The number of photonic absorption events per gridpoint
- The number of singlet-generating absorption events per gridpoint
- The number of triplet-generating absorption events per gridpoint
- The total number of exciton generation events per gridpoint
- The number of singlet generation events per gridpoint
- The number of triplet generation events per gridpoint
- The total number of thermal exciton generation events per gridpoint
- The number of thermal singlet generation events per gridpoint
- The number of thermal triplet generation events per gridpoint
- The total number of exciton dissociation events per gridpoint
- The number of singlet dissociation events per gridpoint
- The number of triplet dissociation events per gridpoint
- The total number of polaron quenching events per gridpoint
- The number of singlet quenching events per gridpoint
- The number of triplet quenching events per gridpoint
- The total number of exciton annihilation events per gridpoint
- The number of singlet annihilation events per gridpoint
- The number of triplet annihilation events per gridpoint
- The total number of non-radiative exciton decay events per gridpoint
- The number of non-radiative singlet decay events per gridpoint
- The number of non-radiative triplet decay events per gridpoint
- The total number of (reverse) intersystem crossing events per gridpoint
- The number of intersystem crossing events per gridpoint
- The number of reverse intersystem crossing events per gridpoint
- The number of degradation events per gridpoint
- The number of radiative exciplex decay events per gridpoint
- The number of non-radiative exciplex decay events per gridpoint
- The number of thermal exciplex generation events per gridpoint
- The device voltage

- The final output timestamp
- The timestamp of the last voltage update
- The number of radiative exciplex decay events
- The number of non-radiative exciplex decay events
- The number of thermal exciplex generation events

Remaining files serve as utilities:

- *bumblebee.qsub* contains the job submission script for the workload manager
- *jobid.out* records the job ID number assigned by the workload manager
- *std.out* reports messages from the workload manager
- *std.err* reports errors from the workload manager
- *error.log* contains error messages from Bumblebee
- *pipeline_status.log* records messages from the web interface
- *started.out* records the UNIX timestamp for the start of the simulation
- *finished.out* records the UNIX timestamp for the end of the simulation

3.5 Python API

A Python API is available to allow for automated submission of Bumblebee simulations.

The API acts as a portal to the web interface. All materials, compositions, stacks, parameters and simulations are stored in the web database. By connecting through the API, it is possible to access these materials, create and upload new materials and even submit simulations through a registered server.

The API provides object representations of the materials, compositions, stacks, parameter sets and simulations. These objects serve as local copies of the database entries. Scripts can then be created to manipulate the values inside the objects, without modifying the original data. This approach allows for the setup of custom parameter screening studies and data visualisation routines.

The API package can be obtained from the Downloads page inside the Bumblebee web interface. Installation instructions are provided for adding the API to your local Python environment. Connections between the web interface and the Python API are secured through an authentication token, which can be obtained through the user profile settings.

Details of the API usage are provided in the [Bumblebee API Tutorials](#).

BUMBLEBEE TUTORIALS

This set of tutorials provides examples for the most common simulation scenarios when modeling OLED devices using Bumblebee.

Tutorials for the prediction of material parameters using the multi-scale [OLED Workflows](#) can be found in the [AMS Manual](#).

4.1 Getting Started

These tutorials are intended to get you acquainted with the web interface and job submission.

4.1.1 Basic Usage

Bumblebee provides a web interface to manage the OLED simulations, which will be used in these tutorials. Consult the manual for details on the installation procedure. To start using the web interface:

- Open the container manager (e.g. Podman)
- From the Pods tab, select podbbweb
- Open the web interface in your browser
- Log in using your credentials

Project Setup

Bumblebee simulations can be stored in separate projects to aid the organization of your research data.

You can create a new project by selecting the *New Project* option from the home page in the web interface.

By default, the web interface will only show materials that are part of your current project. To access materials outside the current project, select *All* as the search location in the navigation menu.

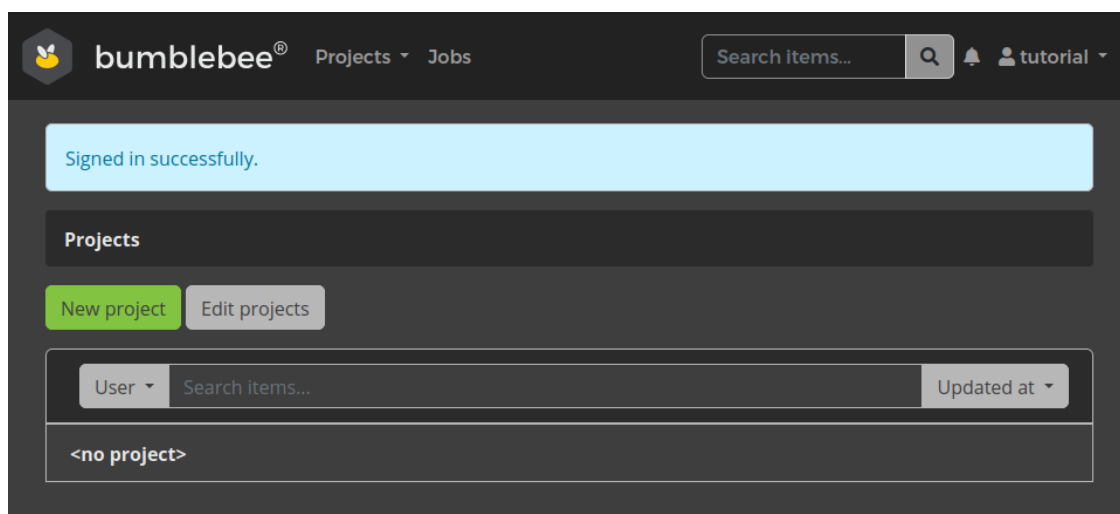


Fig. 4.1: The web interface homepage for managing your projects

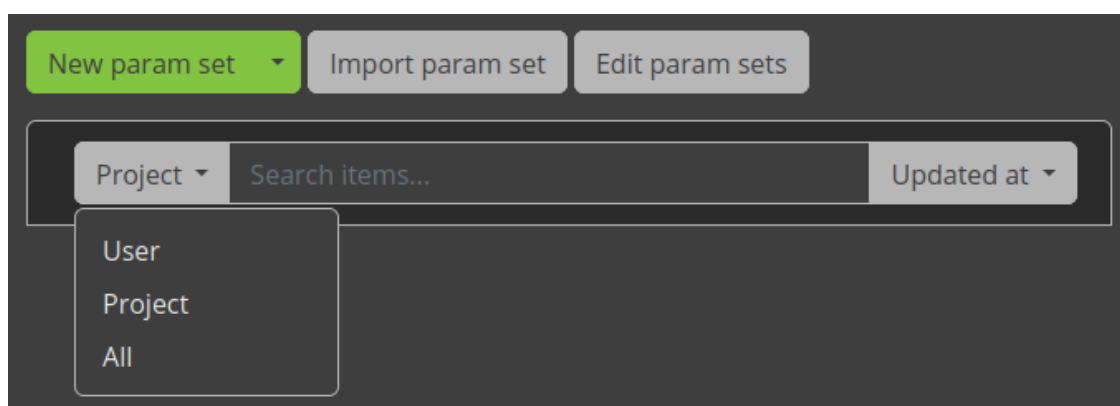


Fig. 4.2: File browser for accessing materials, stacks and parameter sets

4.1.2 Bulk Simulation

Periodic boundary conditions can be enabled in order to study the behavior of the bulk layer material under an active current.

Create Materials

We start by creating the materials that make up the OLED layers. For this tutorial, we will investigate a bulk TAPC material.

Navigate to the *Materials* tab in the GUI and select the *New Material* option. Choose the *Transport* template to set up a material without excitation processes.

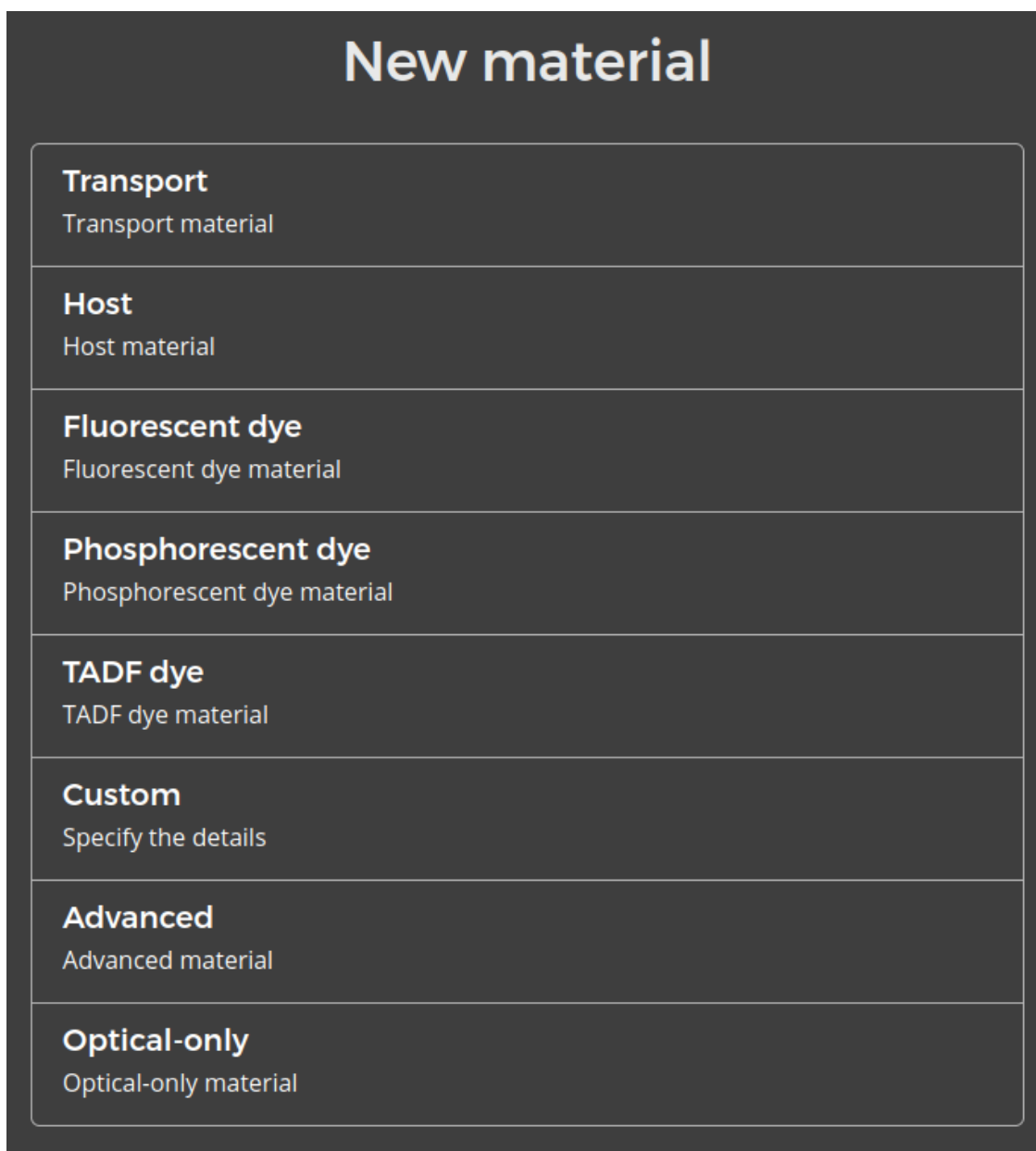


Fig. 4.3: Materials template prompt. Alternatively, a dropdown menu is available next to the *New Material* button

You can provide a name for the material on the *Main* tab. When creating a new material, the *Create a pure composition* option will be enabled by default. Compositions are used to create blends of multiple materials for use as layers in the OLED. Pure compositions only contain a single material, and allow you to directly use the new materials to create OLED stacks.

Blank / Materials / %156

Cancel Delete

Editing material

Main Electronic Excitonic Advanced Optical

Name

TAPC

Description

Transport material

Optional

Template

transport

Project

Blank

Group

Project

☒ Create a pure composition for this material

Cancel Save as Save

Fig. 4.4: Main settings page for new materials

The *Electronic* tab specifies the parameters for the polarons (electrons and holes). For this tutorial, we will use a HOMO level of -5.5 eV and a LUMO level of -0.96 eV.

The DOS type is used to introduce variations in the energy levels between gridpoints. Due to the amorphous nature of typical OLED materials, the molecular environment differs throughout the layers. These environmental differences affect the inter-molecular interactions, resulting in a distribution of energy levels.

Blank / Materials / %156

CancelDelete

Editing material

MainElectronicExcitonicAdvancedOptical

Energy levels

HOMO energy (ionization potential) [eV] ⓘ

-5.5

LUMO energy (electron affinity) [eV] ⓘ

-0.96

DOS type ⓘ

Gaussian

σ_{HOMO} [eV] ⓘ

0.1

σ_{LUMO} [eV] ⓘ

0.1

Fig. 4.5: Polaron settings page for new materials

Here, a Gaussian distribution will be used to model this effect. Set both the standard deviations to 0.1 eV. We will leave the transport parameters at the default for now.

The *Excitonic* tab specifies parameters for the excitons (singlets, triplets) and the excitation processes. As we will not be modeling excitons in this tutorial yet, simply set the singlet and triplet energy levels to 0. You can now save the material.

The material should now be visible in the *Materials* tab. A pure composition has also been created in the *Compositions* tab. If you select the pure TAPC composition, you will see that it has a mole fraction of 1 for the TAPC material that we just created.

Create a Stack

After defining the materials, we can now create an OLED stack by defining the layers. Here, we will be using only a single layer in order to model the bulk material behavior.

Navigate to the *Stacks* tab in the GUI and select the *New Stack* option. As before, provide a name for the stack and select *Save*. This will bring you to the stack editor.

In the stack editor, we can create a new layer. We will use a layer thickness of 50 nm. Select the Pure TAPC composition created earlier and select the *Create layer* button to add the new layer to the stack. A diagram of the current stack layout will be shown. Hovering over the diagram will show a summary of the material properties.

The remaining sections of the stack editor relate to the excitonic processes, so we can leave these alone for now.

Create a Parameter Set

Having created an OLED stack, we will now configure the simulation settings. Navigate to the *Parameter Sets* tab and select the *New Parameter Set* option.

You will be prompted to select a template. As we are modeling a bulk material, select the *Periodic Box*. This will automatically configure some of the presets required for the bulk simulation.

Device Settings

In the *Main* tab, provide a name for this parameter set. Select the stack that we created in the previous step to add it to the parameter set.

In the Physical Parameters section of the *Main* tab, we can set the device voltage. We will use 1 V for this simulation. Keep the temperature at 300 K. The relative permittivity is used to set the effective permittivity of the device (i.e. for all the layers in series). For now, we will use a default value of 3.

Simulation Settings

Initialization settings are provided in the *Box and Energy Landscape* tab. The periodic boundary conditions should have been enabled by default when using the *Periodic Box* template.

The number of sites determines the simulation grid and sets the size of the modeled surface area of the cell. We will use the default of 50 nm in both directions.

Because we are modeling the bulk of the device using periodic boundary conditions, this means that the electrode contacts are not included in the simulation grid. A fixed number of polarons will therefore be used to investigate charge transport. We will use a single polaron type for this tutorial by setting the number of electrons to 30 and the number of holes to 0.

Blank / Materials / %156

CancelDelete

Editing material

MainElectronicExcitonicAdvancedAdvancedOptical

Energy levels

Singlet binding energy [eV] ⓘ

4.54

Triplet binding energy [eV] ⓘ

4.54

☒ Link singlet and triplet binding energies ⓘ

Singlet energy [eV] ⓘ

0.0

Triplet energy [eV] ⓘ

0.0

Excitonic DOS type ⓘ

Gaussian

σ_S [eV] ⓘ

0.05

σ_T [eV] ⓘ

0.05

Fig. 4.6: Exciton settings page for new materials

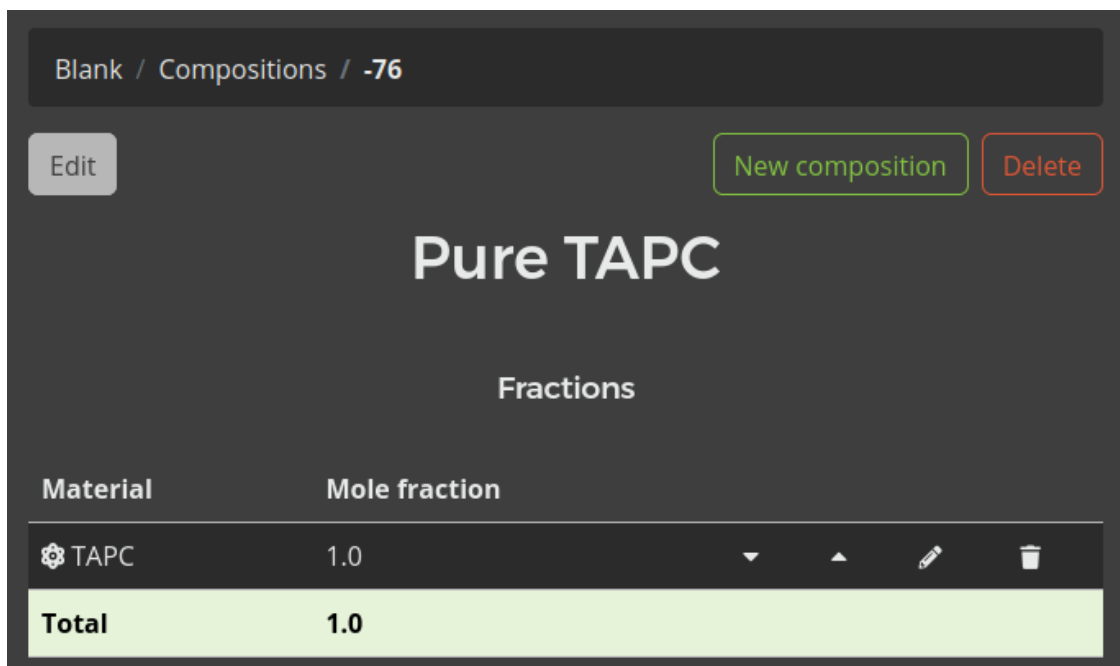


Fig. 4.7: Pure compositions are automatically created for new materials

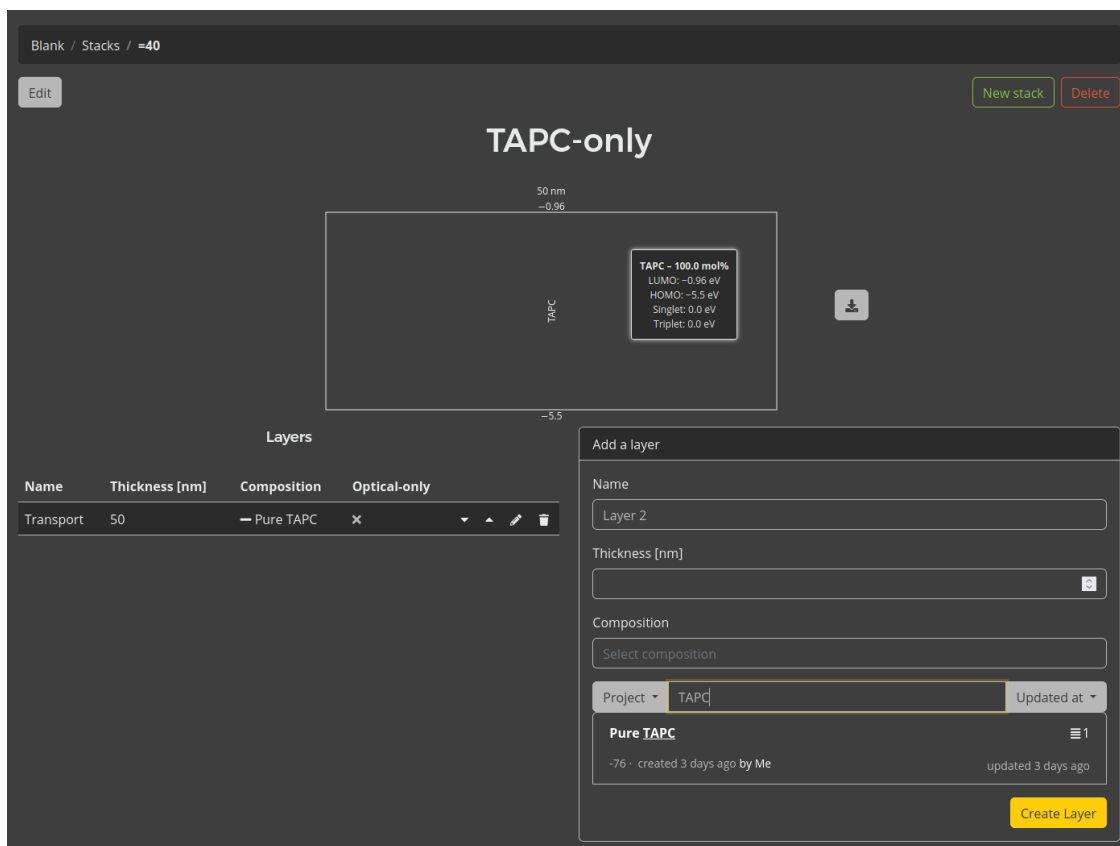


Fig. 4.8: Stack editor layout. Hovering over the stack diagram shows the material properties

New parameter set


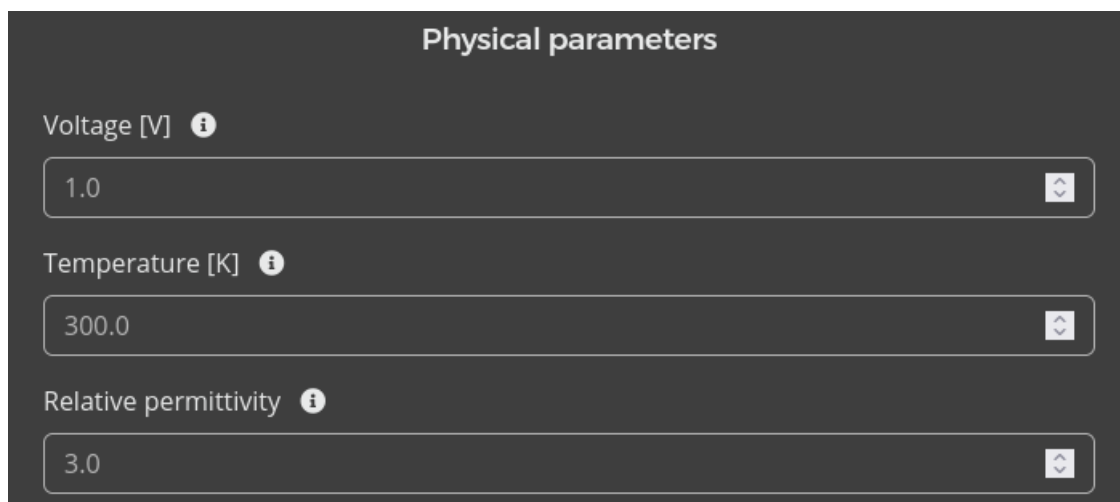
Single voltage point
Electrical characteristics at a single voltage
Voltage sweep (bipolar)
Perform a voltage sweep on a bipolar device
Voltage sweep (hole-only)
Perform a voltage sweep on a hole-only device
Voltage sweep (electron-only)
Perform a voltage sweep on an electron-only device
Periodic box
Fully periodic boundary conditions
Lifetime simulation
Simulates degradation scenarios
Photoluminescence
Steady-state or transient PL in a box
PV and photodetector
Simulate photoresponse
Transistor 
Simulate a small section of a dual gate transistor
Advanced
Advanced simulation

Fig. 4.9: Parameter template prompt



The image shows a dark-themed dialog box titled "Physical parameters". It contains three input fields, each with an information icon (i) to its right. The first field is labeled "Voltage [V]" and contains the value "1.0". The second field is labeled "Temperature [K]" and contains the value "300.0". The third field is labeled "Relative permittivity" and contains the value "3.0". Each input field has a small up/down arrow icon on its right side.

Physical parameters

Voltage [V] ⓘ

1.0

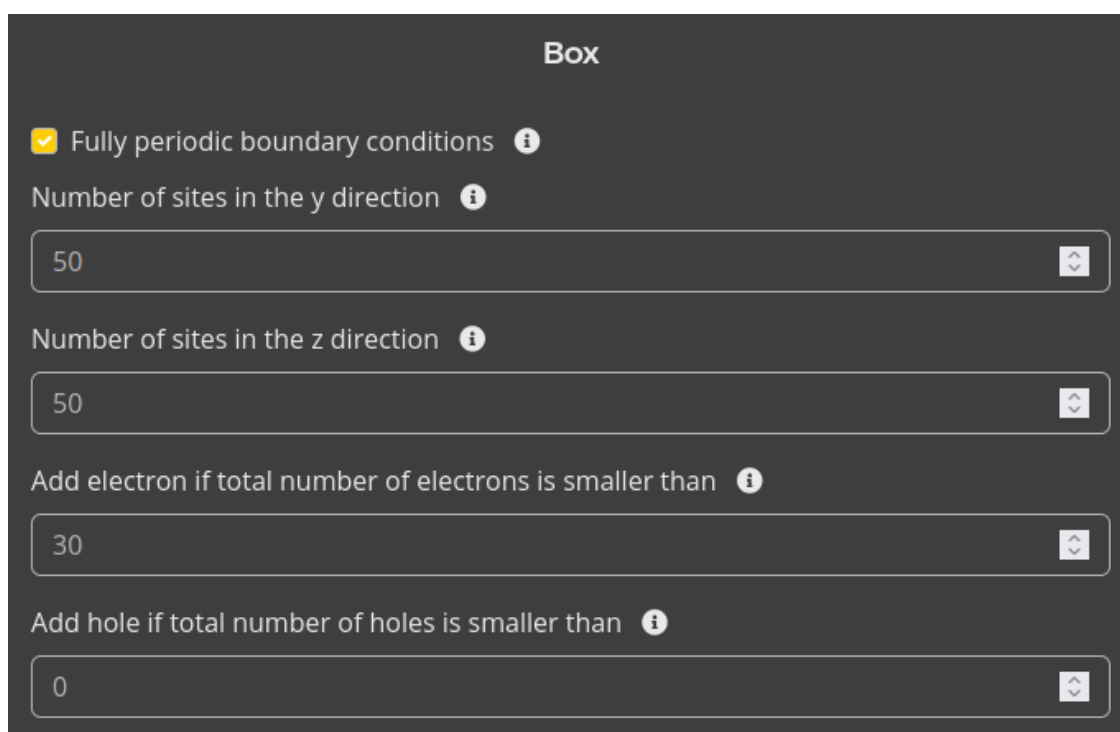
Temperature [K] ⓘ

300.0

Relative permittivity ⓘ

3.0

Fig. 4.10: Physical parameter settings for a new parameter set



The image shows a dark-themed dialog box titled "Box". It contains five settings. The first is a checked checkbox labeled "Fully periodic boundary conditions" with an information icon (i). The second is a text input labeled "Number of sites in the y direction" with an information icon (i) and the value "50". The third is a text input labeled "Number of sites in the z direction" with an information icon (i) and the value "50". The fourth is a text input labeled "Add electron if total number of electrons is smaller than" with an information icon (i) and the value "30". The fifth is a text input labeled "Add hole if total number of holes is smaller than" with an information icon (i) and the value "0". Each text input field has a small up/down arrow icon on its right side.

Box

☒ Fully periodic boundary conditions ⓘ

Number of sites in the y direction ⓘ

50

Number of sites in the z direction ⓘ

50

Add electron if total number of electrons is smaller than ⓘ

30

Add hole if total number of holes is smaller than ⓘ

0

Fig. 4.11: Initialization settings for a new parameter set

Simulation Duration

The *Termination Criteria* tab specifies the duration of the simulation.

As Bumblebee determines the device performance through stochastic sampling, it is important that a sufficient number of steps is allowed in order to provide accurate statistics. This can be achieved in 2 ways:

- Increase the number of steps in the simulation
- Perform multiple simulations in parallel and collect the results (this can be enabled during the job submission step)

For this tutorial, we will set the number of simulation steps to 1.000.000.000. This determines the maximum number of steps that the simulation will go through. Additional convergence criteria can be specified in this tab to allow early termination. These can be left off for now.

Output Settings

The *Output* tab allows specifying the frequency with which simulation results are reported. This frequency is set separately for 2 types of files:

- The report interval determines how often the simulation summary and log files are updated
- The output interval determines updates to the remaining output files

Writing output to a large number of files can slow down the simulation significantly. For this reason, the output interval is often taken to be larger than the report interval. Individual output files can also be enabled or disabled manually in the *Output* tab to reduce the cost of writing files.

For this tutorial, we will set a report interval of 5000 steps and an output interval of 1.000.000 steps.

Finally, select the *Save* option to create the parameter set.

Starting the Simulation

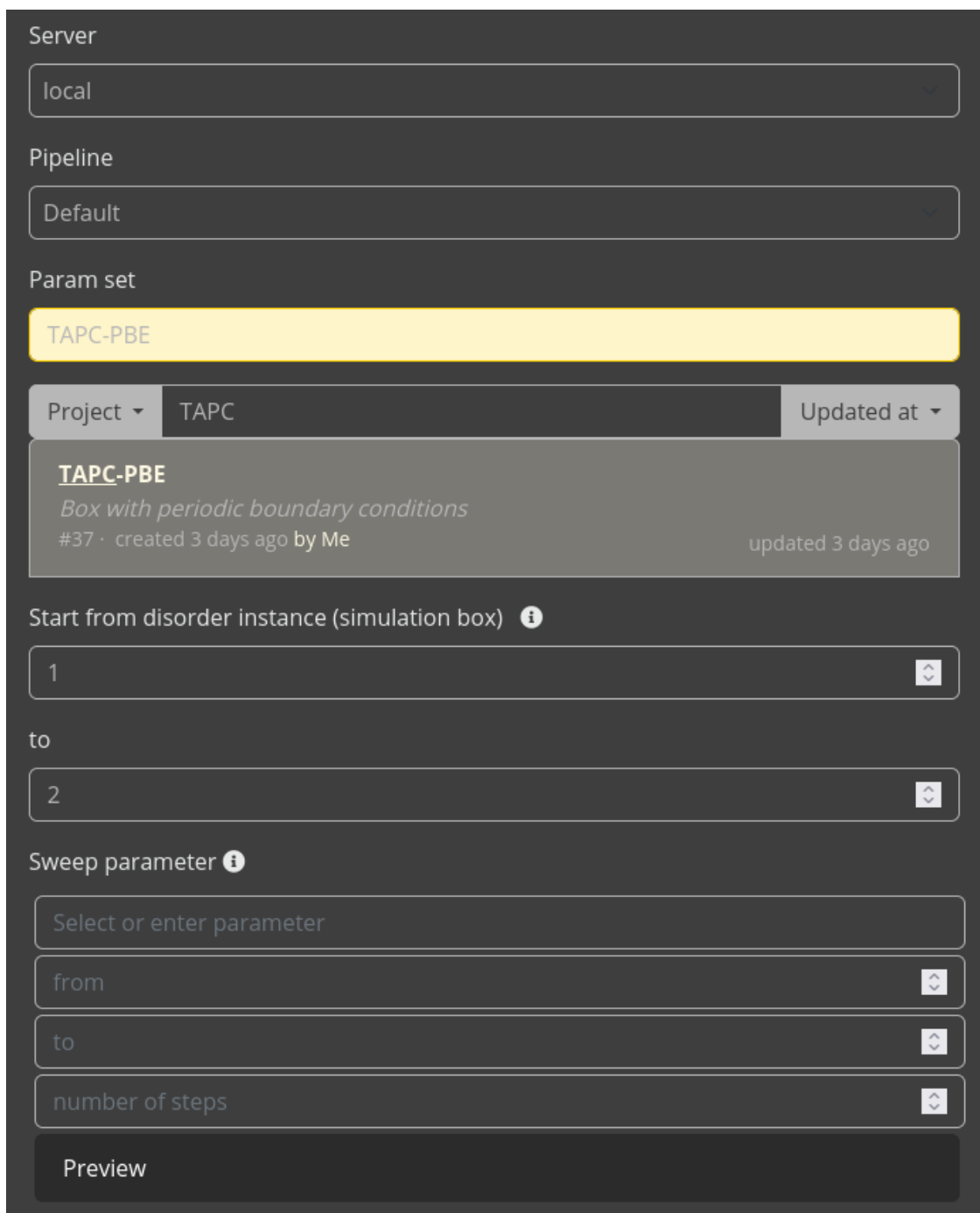
The *Simulations* tab allows submitting simulation jobs using the previously specified parameter set.

Select *New Simulation* to set up a submission. Specify a name for the simulation and check that the correct server is selected for running the job. Then, select the parameter set for the TAPC periodic box.

The disorder instance can be used to determine the number of parallel simulations that will be conducted. Bumblebee will automatically collect sample data from each simulation and include this data in the device statistics.

We will select 2 simulation boxes in order to illustrate this process (without using too many resources). To obtain better statistics, you can increase the number of boxes here, or the simulation time in the parameter set. Note however, that this will also increase the computational cost of the simulation.

The sweep parameters will not be used for this tutorial. Select *Submit Simulation* to submit the job to the server.



The image shows a simulation submission form with a dark grey background. It contains several sections: 'Server' with a dropdown menu set to 'local'; 'Pipeline' with a dropdown menu set to 'Default'; 'Param set' with a yellow highlighted bar labeled 'TAPC-PBE'; a 'Project' section with a dropdown set to 'TAPC' and an 'Updated at' dropdown; a detailed view of the 'TAPC-PBE' parameter set showing its description 'Box with periodic boundary conditions', ID '#37', creation info 'created 3 days ago by Me', and update info 'updated 3 days ago'; a 'Start from disorder instance (simulation box)' section with an information icon and two dropdown menus set to '1' and '2'; a 'Sweep parameter' section with an information icon and four input fields for 'Select or enter parameter', 'from', 'to', and 'number of steps', each with a dropdown arrow; and a 'Preview' button at the bottom.

Server

local

Pipeline

Default

Param set

TAPC-PBE

Project ▾ TAPC Updated at ▾

TAPC-PBE
Box with periodic boundary conditions
#37 · created 3 days ago by Me updated 3 days ago

Start from disorder instance (simulation box) ⓘ

1

to

2

Sweep parameter ⓘ

Select or enter parameter

from

to

number of steps

Preview

Fig. 4.12: Simulation submission form. Note that your server name may differ

Monitoring the Simulation

You can check the progress of the simulation by selecting the *Jobs* tab in the taskbar.



Fig. 4.13: Job overview in the web interface

Selecting the simulation will show jobs for the 2 parallel boxes. Clicking on a running job will bring you to an overview menu showing the current simulation status.

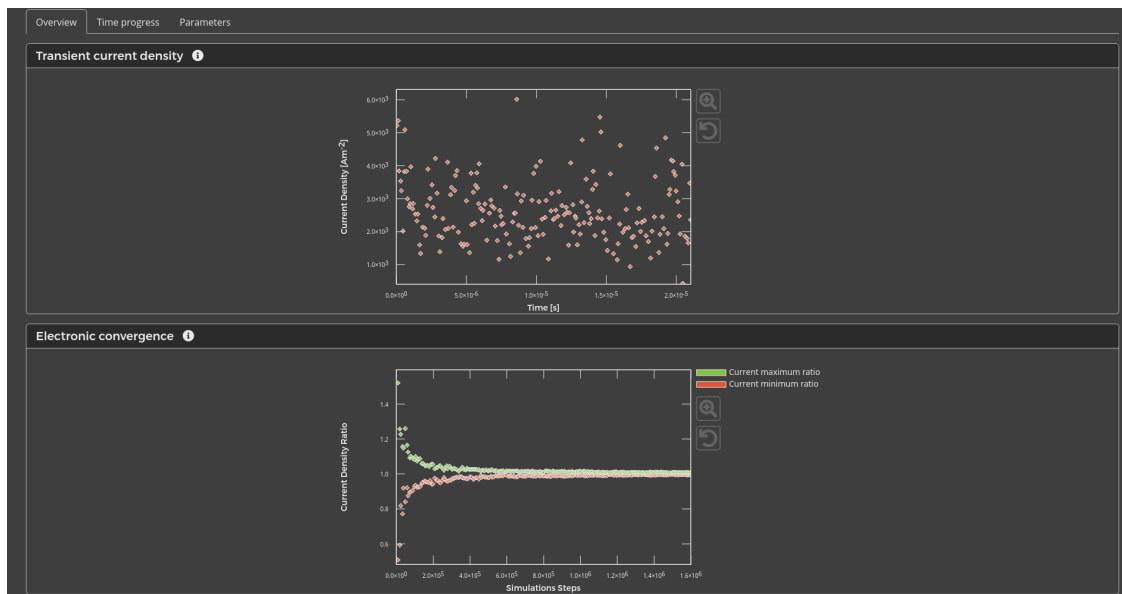


Fig. 4.14: Job monitor in the web interface

The *Overview* tab shows the simulation convergence and transient device parameters. The *Time Progress* tab provides the latest summary of the simulation statistics.

If you wish to terminate a simulation before it has reached the total number of steps, choose the *Select Jobs* option. You can now highlight specific boxes or entire simulations. Selecting the *Stop Selected Jobs* option will send a signal to the job to label the current iteration as the end of the simulation. The job will continue until the next output step has been reached and then finalize the simulation statistics.

The *Kill Selected Jobs* option will immediately terminate jobs, without recovering the output, and should only be used if there are issues with the job itself.

Simulation Output

After the simulation has finished, you can view the results in the *Simulations* tab. Selecting a simulation will bring you to the *Overview* tab. The *Sweeps* tab lists the associated jobs. The job progress can be viewed in the *Jobs* tab.

The simulation output is collected in the *Reports* tab. The web interface handles automatic visualization of various results. An overview of the visualization options is collected in the manual.

The *Single Box* tab shows the results for each job. This allows you to analyze the individual trajectories. The *Multibox* tab provides the aggregate results obtained from collecting the data from multiple boxes. Navigate to the *Files* tab if you want access to the raw simulation output.

Tip: It is possible to add additional boxes to the simulation in order to improve the statistical estimates, even if the simulation has already been completed. Select the simulation and navigate to the *Sweeps* tab. Select the *Add Sweep* option to increase the disorder instance range. This will submit additional jobs to the server. Simulation reports are updated as these jobs complete.

4.1.3 Parameter Screening

Simulation parameters can be screened in order to study device performance under different conditions, or to search for desired material properties.

Create Materials

We start by creating the materials that are used in the stack.

- NPD has a HOMO energy of -5.45 eV and a LUMO energy of -1.4 eV
- mCBP has a HOMO energy of -6 eV and a LUMO energy of -1.5 eV
- Ir(dmp)3 has a HOMO energy of -5 eV and a LUMO energy of -1.7 eV

For this tutorial, we will focus on charge transport only. The *Transport* template can be used when creating new materials. We will use a Gaussian DOS with a standard deviation of 0.1 for both polarons. Excitonic processes will be omitted for now. The exciton energy levels can therefore be put to 0 for all materials.

Create Compositions

Navigate to the *Compositions* tab to access the compositions that make up the device layers. Pure compositions for each of the compounds should be available.

In addition, we are going to create a new host-guest mixture. Select the *New Composition* option. Provide a name for the composition and select the *Basic* template. Selecting the *Save* option will bring you to the composition editor.

You will see a warning in the editor stating that the mole fractions do not sum to 1. This warning will be displayed until the composition is fully defined.

We will create a mixture of 0.9 mCPB and 0.1 Ir(dmp)3. To add a component to the mixture, simply select the material and the desired fraction. The *Create Fraction* button will add the material to the mixture. It is also possible to adjust or remove fractions that were previously included by selecting the editor icons in the fractions list.

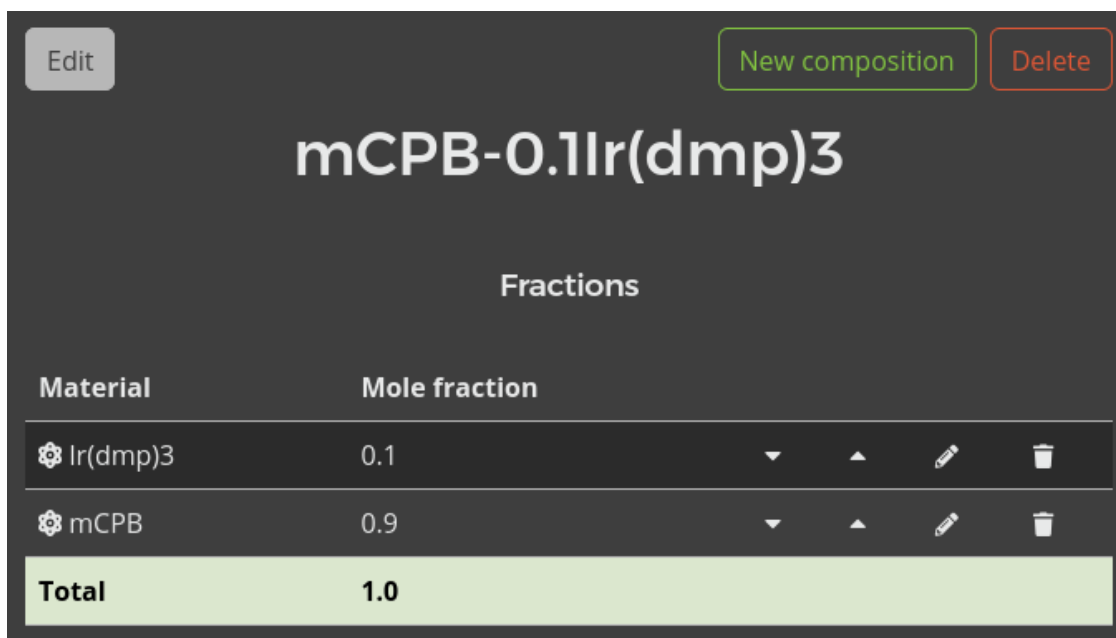


Fig. 4.15: Mixed composition for a host-guest system

Create a Stack

We will create a stack using 3 layers. The outer contact layers are composed of pure NPD. The inner emitter layer contains the host-guest mixture defined earlier.

- Create a 10 nm NPD layer
- Create a 60 nm layer containing the mCPB-Ir(dmp)3 mixture
- Create another 10 nm NPD layer

You will end up with an 80 nm stack.

Create a Parameter Set

For this simulation, we are interested in investigating the device performance at different voltages. We will therefore use the *Voltage Sweep* template when creating our parameter set.

Give a name to the parameter set and add the stack that was created in the previous step. The voltage will be set as part of the parameter screening and does not need to be chosen at this step.

The Fermi levels of the electrodes are automatically matched to the material parameters. These levels can be adjusted to those of the external contacts. We will use an electrode energy level of -5.25 at the anode and -1.2 at the cathode. By using an energy barrier of 0.2 eV compared to the NPD polaron energy levels, we are reducing the rate of polaron exchange processes with the electrodes. This biases the kMC simulation, increasing the number of samples that contain transport processes compared to electrode exchange.

Note: Care should be taken that this artificial barrier does not affect the device statistics. This can be achieved by analyzing the change in device properties when varying the barrier height. Because electrode exchange processes are localized at the edges of the device, a single-layer simulation can be used to perform these screenings, significantly reducing the simulation costs.

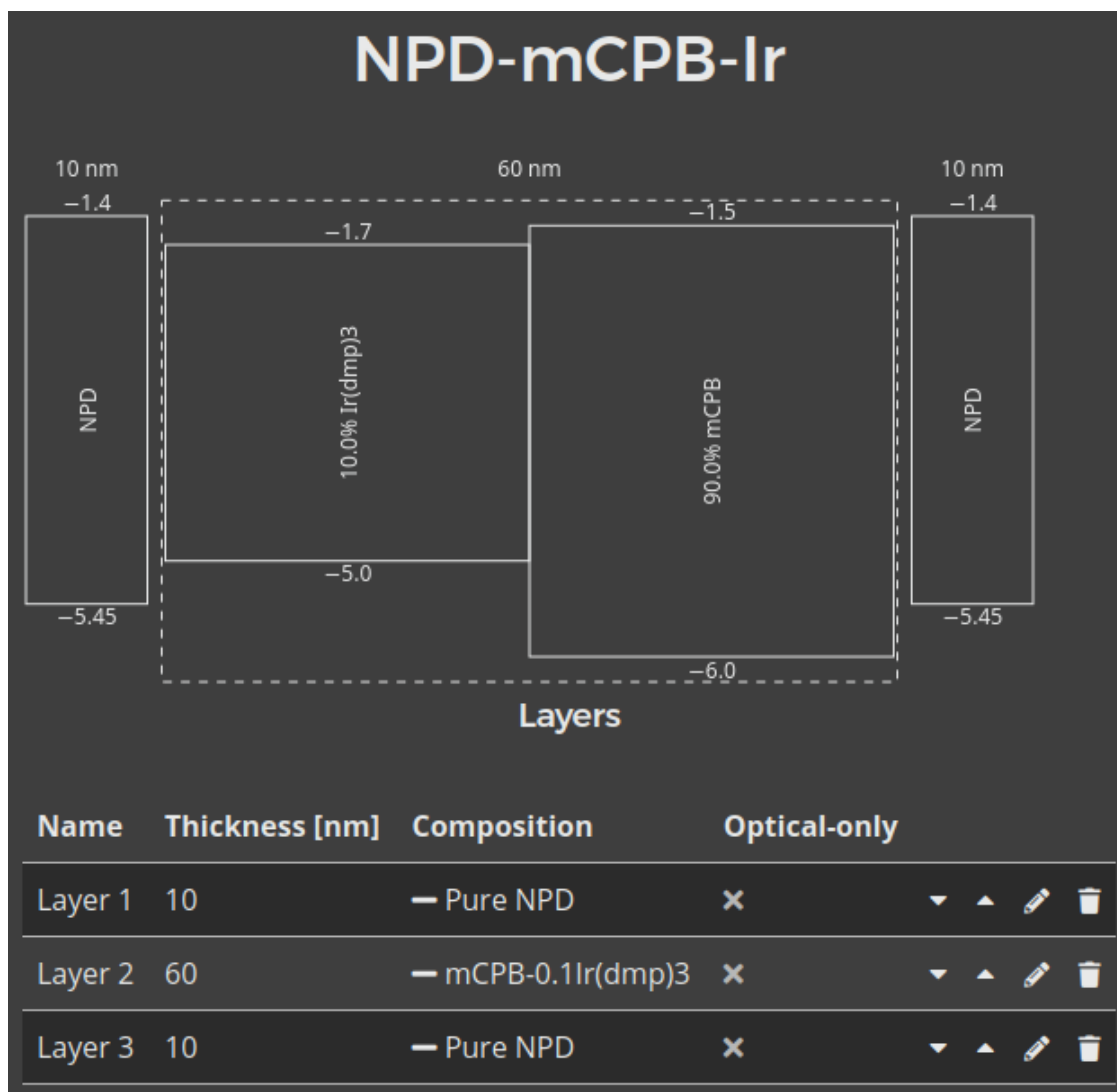


Fig. 4.16: Diagram of the OLED device

Stack

NPD-mCPB-Ir

Project ▾ CPB Updated at ▾

NPD-mCPB-Ir 📄 1

=42 · created 3 days ago by Me updated 3 days ago

Fermi level left electrode (anode) [eV] ⓘ

-5.25 ⬆️⬆️⬆️

Fermi level right electrode (cathode) [eV] ⓘ

-1.6 ⬆️⬆️⬆️

Physical parameters

Voltage [V] ⓘ

0.0 ⬆️⬆️⬆️

Fig. 4.17: Electrode contacts are configured automatically to include an exchange barrier

Note: When operating under high currents, the device sensitivity to the barrier height may become voltage-dependent. Make sure to verify the validity of your data at both extremities of the investigated voltage range.

The *Modules* tab allows you to enable different processes that are allowed to occur during the simulation. You can disable the excitonics module as we will focus on polaron transport for this tutorial.

We will set the number of simulation steps to 1.000.000.000 in the *Termination Criteria* tab. On the *Output* tab, we set the report interval to 100.000 and the output interval to 1.000.000. The parameter set can now be saved.

Starting the Simulation

Navigate to the *Simulations* tab. Specify a name for the simulation and check that the correct server is selected for running the job. Then, select the voltage sweep parameter set created in the previous step. For tutorial purposes, we can set the disorder instances to 1.

Parameter screenings can be specified as part of the simulation setup. The screening allows evaluation of the device behavior for different parameter values. Aside from the sweep parameter, all other conditions will remain unaltered from the default parameter set.

The screening values are obtained through linear interpolation. Minimum and maximum values for the screening parameter are selected and the total number of screening steps is chosen. A uniform stepsize between parameter values is calculated.

For this example, we select the voltage as our screening parameter. The minimum and maximum voltages are set to 1 and 5 V respectively. By choosing 3 voltage steps, this will prompt the screening to perform simulations at 1, 3 and 5 V. A preview of these screening conditions is provided in the web interface.

Note that the number of disorder instances is applied to each screening step. The default of 5 disorder instances would therefore yield 3x5 independent jobs. By using only a single disorder instance, the number of simulation jobs is limited to 3.

We now submit the simulation and wait for the screening steps to complete.

Simulation Output

Current-voltage characteristics can be viewed in the *OLED Report* section of the *Sweep Report* panel.

Tip: After performing an initial screening, it is possible to add additional sample points. After selecting a simulation, the *Sweep* tab contains the *Add Sweep* option to request additional jobs. The output from these jobs is then automatically collected and appended to the current simulation. This option was used in the previous tutorial to improve simulation statistics, but can also be used to extend the sweep range or to add additional points to the parameter screening.

Param set

NPD-mCPB-Ir-Sweep

Project ▾ CPB Updated at ▾

NPD-mCPB-Ir-Sweep
Voltage sweep
#39 · created 3 days ago by Me updated about 9 hours ago

Start from disorder instance (simulation box) ⓘ

1 ▾

to

1 ▾

Sweep parameter ⓘ

voltage

1 ▾

5 ▾

3 ▾

Preview | 1 | 3 | 5 |

Fig. 4.18: Voltage sweep setup in the simulation settings

4.1.4 Exciton Simulation

Opto-electronic processes are defined in the kMC simulation to model OLED emission. In this example, a phosphorescent emitter is considered. Loss processes are included using both Dexter and Förster mechanisms.

Create Materials

To construct the OLED device stack, we will create an electron transport layer, a hole transport layer and a host-guest emitter. This requires the definition of 4 materials.

Phosphorescent Dye

Ir(ppy)_3 is used as the phosphorescent dye. We will select the corresponding template when creating a new material.

On the *Electronic* tab, we specify a HOMO level of -5.27 eV and a LUMO level of -1.86 eV. A Gaussian broadening is enabled by default.

Several material-specific parameters are specified to describe exciton generation and emission. These parameters are set on the *Excitonic* tab of the material editor.

We will set a singlet binding energy of 0.75 eV and a triplet binding energy of 1 eV. By enabling the option to link the singlet and triplet binding energies, the exciton energy levels will be computed automatically based on the exciton binding energy and the HOMO/LUMO levels.

An energy level broadening is defined for the exciton levels, just as we did for the polaron levels, accounting for the variations in molecular parameters due to the inhomogeneous environment of the layer. A Gaussian broadening is used, this time with a width of 0.05 eV.

To describe Dexter-type exciton diffusion, Dexter transfer parameters are specified. We choose a prefactor of 1, with a decay length of 0.3 nm.

Note: The rates of transfer processes is specified in normalized units. I.e. the true prefactor is multiplied by a normalization factor. This time unit is specified in the parameter set.

This decomposition allows us to write the material parameters using convenient factors.

In contrast, the radiative processes are provided in natural units, without normalization. These parameters specify the real frequency in the input, with normalization applied internally by Bumblebee.

By selecting the phosphorescent material template, the singlet fractions will have been set to 0, such that the exciton generation products will exclusively be triplets.

An intersystem crossing rate of 10^{10} s^{-1} will be specified. The reverse intersystem crossing rate is set to 0. This allows any singlets obtained through e.g. exciton transport to be irreversibly converted to triplets.

The radiative decay rate of the triplet excitons is set to $6.1 \cdot 10^5 \text{ s}^{-1}$. The non-radiative decay rate is set to $1.9 \cdot 10^4 \text{ s}^{-1}$. The photoluminescent and electroluminescent quantum yields of the dye are now reported to provide an indication of the molecular emitter efficiency.

Note: Förster processes will be configured in the stack editor. Because Förster transfer describes a dipolar process, the rate parameters exhibit strong variations with molecular environment. The stack editor allows definition of custom rates for inter-layer transfer processes, and allows definition of multiple intra-layer Förster processes to account for more complex rate expressions.

Energy levels

Singlet binding energy [eV] ⓘ

0.75

Triplet binding energy [eV] ⓘ

1.0

☒ Link singlet and triplet binding energies ⓘ

Singlet energy [eV] ⓘ

2.66

Triplet energy [eV] ⓘ

2.41

Excitonic DOS type ⓘ

Gaussian

σ_S [eV] ⓘ

0.05

σ_T [eV] ⓘ

0.05

Fig. 4.19: Exciton binding energies and energy levels can be linked automatically



Transfer

Dexter prefactor ⓘ

1.0

Singlet wavefunction decay length [nm] ⓘ

0.3

Triplet wavefunction decay length [nm] ⓘ

0.3

Fig. 4.20: Rate constants for Dexter-type exciton transfer

Host

CBP is used as a host material. Select the appropriate template when creating a new material entry.

We use a HOMO level of -6.08 eV and a LUMO level of -1.75 eV. A Gaussian broadening is enabled by default. For the excitons, we use a singlet binding energy of 1 eV and a triplet binding energy of 1.7 eV. For Dexter-type exciton transfer, a prefactor of 0.95 is used along with a decay length of 0.3.

The singlet-triplet generation ratio will be set to 0.25 (corresponding to a statistical 1:3 distribution of singlet and triplet excitons).

Thermalization losses during exciton transport from the dye through the host are included by setting the non-radiative decay rates to 10^5 s^{-1} for singlets and 10^4 s^{-1} for triplets. The radiative decay rates are set to 0.

Electron Transport Layer

TPBi is used as an electron transport layer. Select the *Transport* layer when creating a new material.

We use a HOMO level of -6.2 eV and a LUMO level of -1.7 eV. For the excitons, we use a singlet binding energy of 0.75 eV and a triplet binding energy of 1 eV. For Dexter-type exciton transfer, a prefactor of 1 is used along with a decay length of 0.3.

To mimic the effect of an exciton diffusion barrier in the stack, a non-radiative decay rate of 10^8 s^{-1} is specified for both excitons.

Photophysics

Singlet fraction (for electronic generation) ⓘ

0.0

Singlet fraction (for annihilation) ⓘ

0.0

Intersystem crossing rate [s^{-1}] ⓘ

1000000000000.0

Reverse intersystem crossing rate [s^{-1}] ⓘ

0.0

Singlet radiative decay rate [s^{-1}] ⓘ

0.0

Singlet non-radiative decay rate [s^{-1}] ⓘ

0.0

Triplet radiative decay rate [s^{-1}] ⓘ

610000.0

Triplet non-radiative decay rate [s^{-1}] ⓘ

19000.0

PLQY ⓘ

97.0 %

ELQY ⓘ

97.0 %

Fig. 4.21: Förster prefactors, intersystem crossing frequencies and singlet-triplet distributions. The PLQY and ELQY are reported automatically based on the provided rates

Hole Transport Layer

TAPC is used as the hole transport layer. We use a HOMO level of -5.5 eV and a LUMO level of -0.96 eV. For the excitons, we use a singlet binding energy of 1 eV and a triplet binding energy of 1.59 eV. For Dexter-type exciton transfer, a prefactor of 1 is used along with a decay length of 0.3. A non-radiative decay rate of 10^8 s^{-1} is specified for both excitons.

Create Compositions

We will create a host-guest mixture containing 0.9 CBP and 0.1 Ir(ppy)₃.

Create a Stack

We start by composing the stack layers. We use a 20 nm TAPC hole transport layer, a 30 nm host-guest layer in the center and a 20 nm TPBi electron transport layer.

The stack editor now allows us to define the Förster radii. Förster rates are defined for various processes, including diffusion, quenching and annihilation. Because the Förster rates depend on the molecular environment, separate reactions have to be specified for each pair of materials. To streamline this process, the web interface provides the option to automatically configure the most common processes for the current stack. The mechanisms that are included are based on the material templates.

For this tutorial, we will use the default Förster interactions. This will include the triplet diffusion, triplet quenching and exciton annihilation reactions.

For illustration purposes, we will use the manual option to add singlet quenching to our list of reactions. Selecting the option *Add Förster Interaction* will open a new menu.

We select Ir(ppy)₃ as the donor material, i.e. the material containing the singlet. We then select all 4 materials as possible acceptors. In the list of processes, we select both singlet-electron and singlet-hole quenching. We will specify a Förster radius of 1 nm. Clicking the *Generate* button will now automatically configure both quenching reactions for each donor-acceptor pair. Each of these processes will have the same Förster radius by default. Selecting the radius of a specific reaction allows you to change the value. Here, we will use a 1.5 nm radius for the quenching reactions inside Ir(ppy)₃. Select the *Save* button to add the reactions to the stack.

Note: For inter-layer Dexter transport, the rate constants are obtained as a geometric average of the layer parameters. Custom parameters can be specified for individual layer pairs to overrule this behavior.

Create a Parameter Set

We will use the single voltage point template to create the parameter set. We select our stack and set a default voltage of 5 V. The maximum number of simulation steps will be set at 1.000.000.000.

Excitonic processes are included in the simulation by enabling the exciton module. The single voltage point template includes this option by default. You can check the module configuration by navigating to the *Modules* tab.

Förster interactions ⓘ						
Add Förster interaction		Add default Förster interactions			Delete all	
Interaction	Donor layer	Donor material	Acceptor layer	Acceptor material	Förster radius [nm]	
Triplet diffusion						
Triplet diffusion	EMI	Ir(ppy)3	EMI	Ir(ppy)3	1.5	🗑️
Triplet-hole quenching						
Triplet-hole quenching	EMI	Ir(ppy)3	ETL	TPBi	3.5	🗑️
Triplet-hole quenching	EMI	Ir(ppy)3	HTL	TAPC	3.5	🗑️
Triplet-hole quenching	EMI	Ir(ppy)3	EMI	CBP	3.5	🗑️
Triplet-hole quenching	EMI	Ir(ppy)3	EMI	Ir(ppy)3	3.5	🗑️
Triplet-electron quenching						

Fig. 4.22: Part of the default Förster interactions

Add interaction

Donor

▼ HTL

TAPC

▼ EMI

Ir(ppy)3

CBP

▼ ETL

TPBi

Acceptor

▼ HTL

TAPC

▼ EMI

Ir(ppy)3

CBP

▼ ETL

TPBi

Interactions

Interactions

Singlet diffusion

Singlet-hole quenching

Singlet-electron quenching

Singlet-singlet annihilation

Singlet-triplet annihilation

Triplet diffusion

Triplet-hole quenching

Triplet-electron quenching

Triplet-singlet annihilation









Triplet-triplet annihilation

Förster radius [nm]

1

Generate

Fig. 4.23: Editor for manual configuration of Förster processes

Generated Interactions					
Interaction	Donor layer	Donor material	Acceptor layer	Acceptor material	Förster radius [nm]
Singlet-hole quenching					
Singlet-hole quenching	EMI	Ir(ppy)3	ETL	TPBi	1 
Singlet-hole quenching	EMI	Ir(ppy)3	EMI	CBP	1 
Singlet-hole quenching	EMI	Ir(ppy)3	EMI	Ir(ppy)3	1.5 
Singlet-hole quenching	EMI	Ir(ppy)3	HTL	TAPC	1 
Singlet-electron quenching					
Singlet-electron quenching	EMI	Ir(ppy)3	ETL	TPBi	1 
Singlet-electron quenching	EMI	Ir(ppy)3	EMI	CBP	1 
Singlet-electron quenching	EMI	Ir(ppy)3	EMI	Ir(ppy)3	1.5 
Singlet-electron quenching	EMI	Ir(ppy)3	HTL	TAPC	1 

Cancel
Save

Fig. 4.24: Manually-generated list of Förster interactions


Custom prefactors 					
Add prefactor					
Species	Donor layer	Donor material	Acceptor layer	Acceptor material	Value

Fig. 4.25: Editor for manual configuration of inter-layer Dexter rate constants

Starting the Simulation

A voltage sweep can be performed to investigate the roll-off in device efficiency at higher voltages. We select a voltage range from 3 to 6 V and select 7 voltage points. A single disorder instance can be selected to decrease the simulation runtime.

If you wish to limit the computational time required for this tutorial, you can perform the single voltage point simulation instead. This will use the 5 V default chosen in the parameter set.

Simulation Output

The distribution of carriers over the stack layers can be viewed in the *Profiles* section of the *Sweep Report* panel.

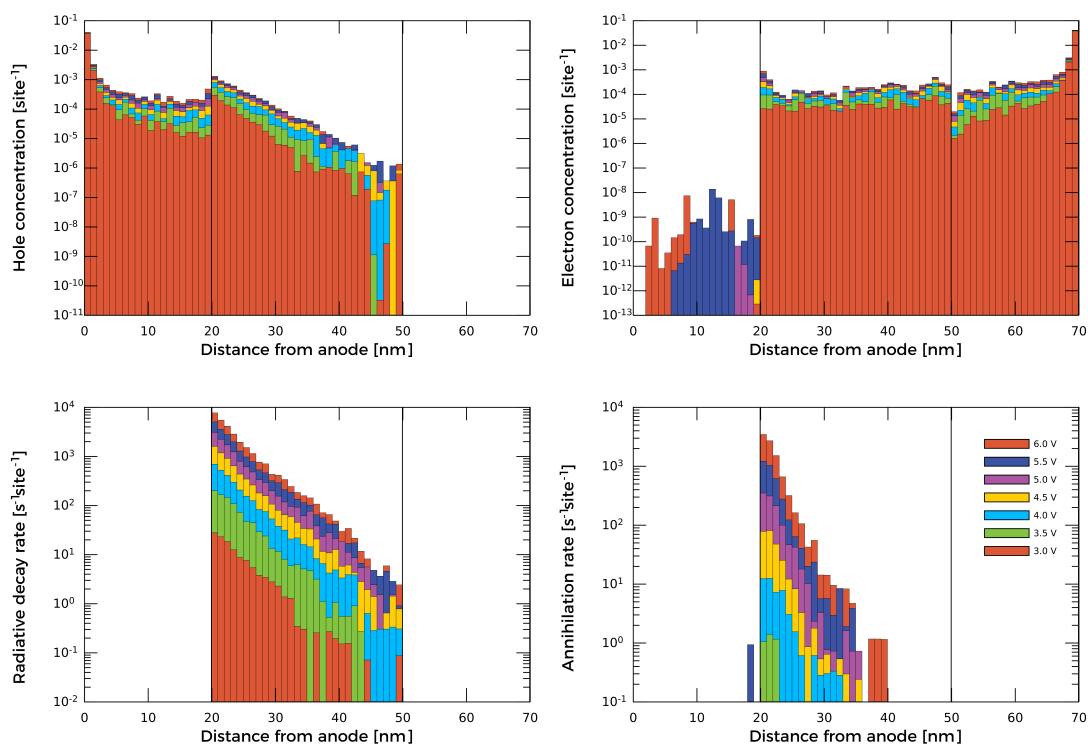


Fig. 4.26: Voltage-dependent carrier densities in the OLED stack

A summary of the excitonic process frequencies is provided in the *OLED Report* section, along with the current-voltage characteristics and device efficiency.

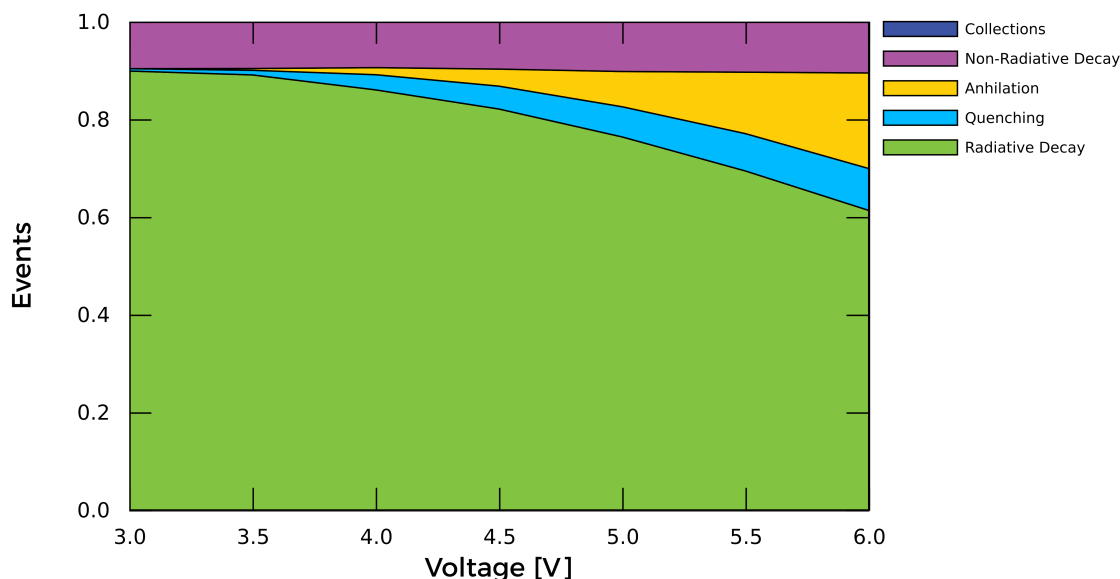


Fig. 4.27: OLED efficiency roll-off following voltage-dependence of device loss processes

4.1.5 Acceleration

Bumblebee provides acceleration methods to enhance sampling effectiveness and reduce computation time.

Module Selection

A rate booster is available to promote anisotropic charge hopping along the electrode current. This increases the rate at which the system equilibrates and enhances the sampling of rare events that are typically of interest to OLED performance.

The relative rates of the parallel and perpendicular transfer processes can be modified. Specifying a priority factor of 1.2 for the carrier transport will increase the parallel current sampling frequency by 20%.

Note: The acceleration module adds an anisotropic bias to the distribution of process frequencies. This bias can alter the kMC trajectories, influencing the device statistics.

Similar to the determination of the electrode transfer barrier, it is necessary to verify that the priority factor does not alter the device behavior.

A single disorder instance can be evaluated in order to determine the correct acceleration settings. Once determined, the parameter set can be updated. Subsequent simulations can be performed at significantly reduced cost.

Because the acceleration module does not alter the actual opto-electronic process parameters, the same acceleration settings can be used at various currents. Care should be taken, however, that the priority factor was determined at a current where opto-electronic processes were energetically accessible.

For example: when annihilation processes are included in the simulation, you should confirm that sampling of these processes occurred in the disorder instance that was used to determine the acceleration settings.

The necessary process event counts can be found in the *Profiles* section of the *Sweep Report*.

Configuring Output

Writing output to the file system can become a bottleneck for the kMC simulation, as write speeds are typically slow compared to the evaluation of processes in Bumblebee.

Faster simulation times can be obtained by choosing longer output intervals. This does come with a risk that fewer checkpoints are generated to enable simulation restarts in case of job interruptions. As the average device statistics typically contain the simulation properties of interest, the reduced amount of intermediate output itself is often considered inconsequential.

Aside from reducing the reporting interval, the content of the simulation output itself can also be reduced. The *Output* settings in the parameter sets allow users to disable undesired statistics from the output by de-selecting the appropriate set of files. This reduced the memory volume of the output, particularly for transient data, event logs or 3D distributions.

Cost Scaling

The cost of a Bumblebee simulation scales with the volume of the device. Modeling of devices with many layers or large surface area will require longer simulation times, simply because the carriers have to traverse longer distances and sampling has to be conducted over a larger number of gridpoints.

For devices with a large surface area, one has the choice between conducting a single large-area simulation or splitting the total surface area between multiple parallel instances. This choice typically has little effect on the overall simulation cost. Parallel distribution tends to be preferred when a larger number of cores is available, as utilization of multiple CPUs allows faster access to simulation results. The required number of CPU-hours nevertheless remains mostly unaltered.

When conducting simulations on very large stacks, the time required to equilibrate the device increases, as does the required number of samples to properly characterize the device. Since most large-stack OLED devices tend to represent multi-stack junctions, it is possible to coarse-grain the simulation by splitting up the stack. Simulations can be conducted on individual stack segments. By matching the current-voltage profiles of the segments, the behavior of the multi-junction system can be reproduced. Note however, that this is only possible when interactions between the junctions are negligible. For cases where coarse-graining is not viable, utilization of the acceleration module is recommended to improve sampling efficiency.

4.2 Advanced Features

These tutorials provide examples of advanced simulations in Bumblebee.

4.2.1 Layer Morphology

Compositions are used to include multiple materials in a single layer. This allows for the generation of host-guest systems or molecular mixtures. Composites can also be used to model the microscopic roughness of layer interfaces.

Bumblebee provides both *Basic* and *Advanced* compositions.

Basic Composition

In the previous tutorials, *Basic* compositions were generated. These compositions assume that the materials are randomly distributed within a layer. These distributions are generated at the start of the simulation. When multiple disorder instances are used, each instance will have its own unique distribution.

The resulting layers have a homogeneous material distribution. The *Advanced* composition allows for the definition of gradients to specify more complex layer morphologies.

Layer Gradients

When creating a new composition, select the *Advanced* composition type.

New composition

Name
Linear

Description

Optional

Project
Tutorial

Group
Project

Advanced options

Composition type
Advanced

Cancel Save as Save

Fig. 4.28: Select the advanced composition option to access the morphology editor

In the advanced composition editor, we start by adding materials to the layer. For this example, we will create a CBP-Ir(ppy)₃ host-guest system using the materials from a previous tutorial.

The first fraction that is added to the layer will be labeled as the background. The background material is used to close the material balance at each gridpoint, assuring that the fractions sum to 1. If no morphology is specified, the background

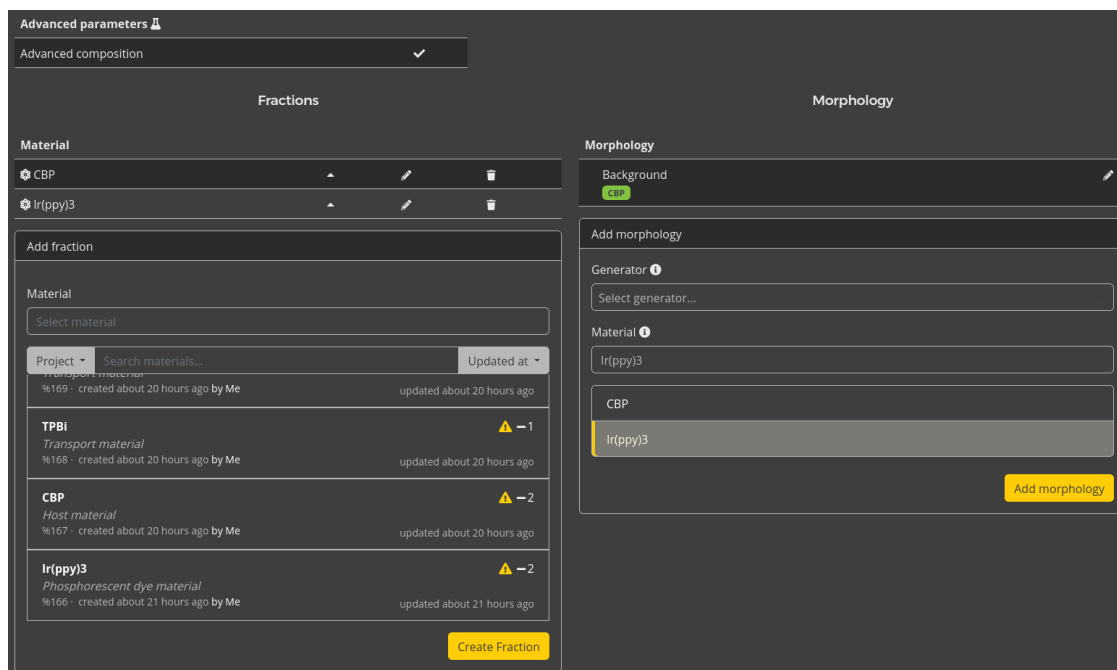


Fig. 4.29: Overview of the morphology editor for advanced materials

material will make up the entire layer.

Morphology generators are provided to add gradients to the layer composition.

Linear

We will use CBP as the background material. You can select the pencil icon to adjust the background material if necessary.

Selecting the linear gradient generator, we can add the Ir(ppy)3 dye to the layer. The material fraction is specified at the layer edges. A linear interpolation is used to determine the fractions at the interior gridpoints.

For this example, we use fractions of 0.4 and 0.8. Selecting *Add Morphology* will save the gradient as part of the current composition. Multiple gradients can also be combined to create more complex morphologies.

Trapezoid

To compare the different generators, we will now create a new composition.

The trapezoid generator can be used to combine multiple linear interpolations. In order to create a trapezoidal gradient, the material fraction can be specified at various locations inside the layer. Linear interpolation is used to determine the fractions at the remaining gridpoints.

We specify the locations as a fraction of the layer width. This allows the morphology to be used with different stacks. At each location, we specify the fraction of Ir(ppy)3. The background material (CBP) is used to close the balance.

Clicking the check mark allows additional points to be added to the trapezoidal gradient. We will use a simple 3-point trapezoid for this example.

If the list of locations does not include the edges of the layer ($x = 0$, $x = 1$), all fractions outside the trapezoid range will be set to 0.

Add morphology

Generator ⓘ

Linear gradient

Linear gradient

Fraction at $x = 0$

0.25

Fraction at $x = 1$

0.75

Material ⓘ

Ir(ppy)3

CBP

Ir(ppy)3

Add morphology

Fig. 4.30: Linear gradient editor

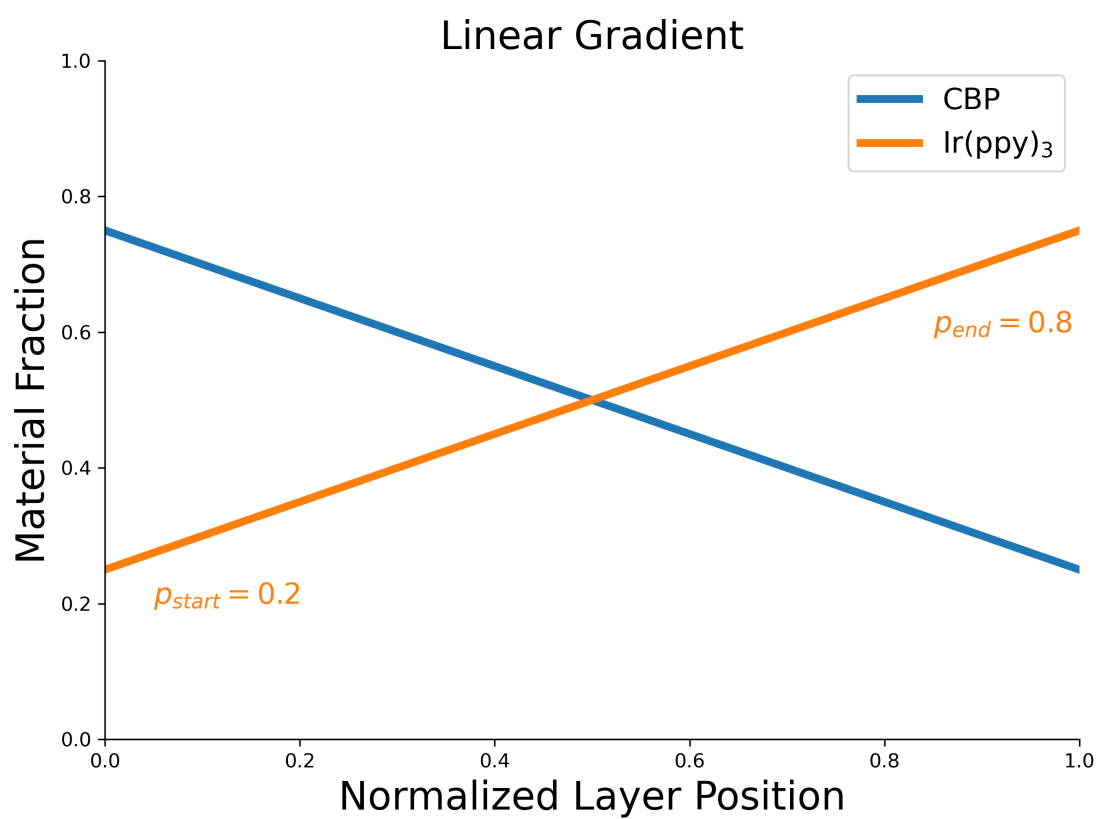


Fig. 4.31: Linear dye gradient in the host-guest system

Add a morphology

Generator ⓘ

Trapezoid gradient

Trapezoid gradient

0.25	↕	0.4	↕	🗑️
0.5	↕	0.8	↕	🗑️
0.75	↕	0.6	↕	🗑️
Position x	↕	Fraction	↕	✅

Material ⓘ

Ir(ppy)3

CBP

Ir(ppy)3

Create Morphology

Fig. 4.32: Trapezoidal gradient editor

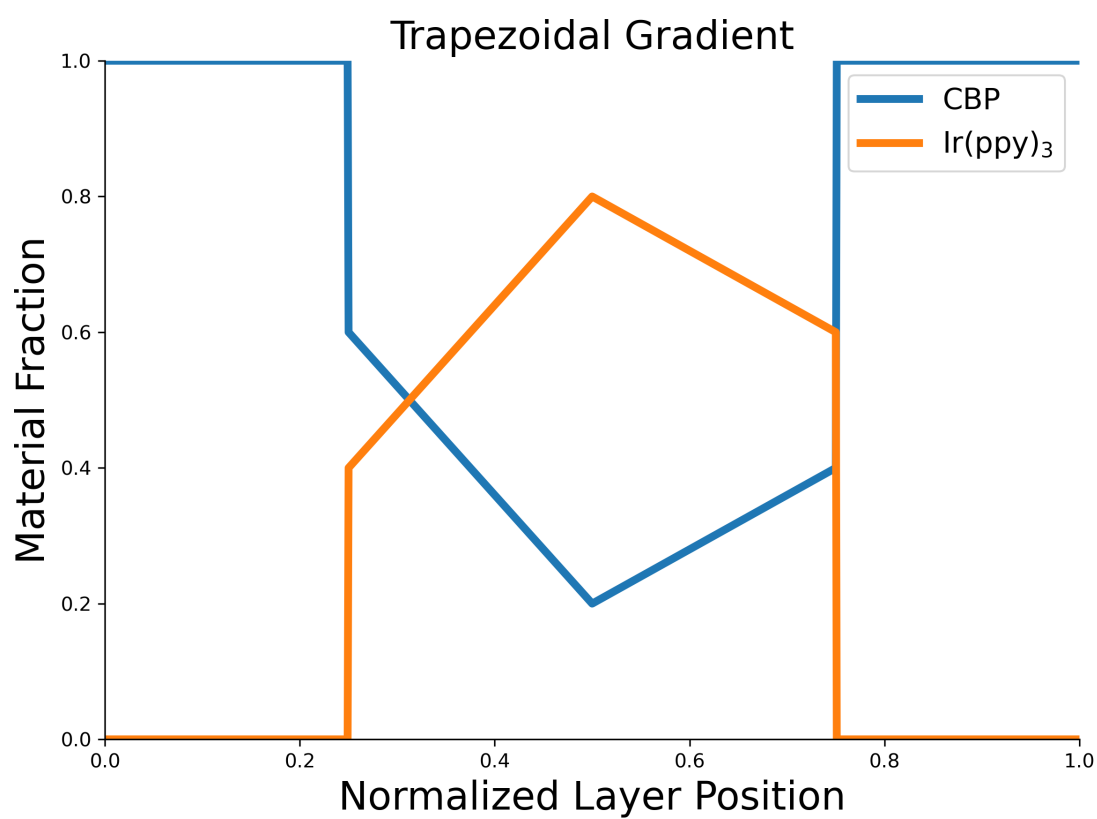


Fig. 4.33: Trapezoidal dye gradient in the layer center

Custom

We will create a new composition to illustrate the use of custom gradients.

The screenshot shows a software interface for creating a custom gradient. The main window is titled "Add morphology". Inside, there's a section for "Generator" with a dropdown menu set to "Custom gradient". Below this, there are two input fields: "x start" with the value 0.2 and "x end" with the value 0.8. Underneath these is a "Formula" field containing the expression $x^{**}x$. Below the formula field is a "Material" dropdown menu set to "Ir(ppy)3". A list of materials is shown below the dropdown, with "Ir(ppy)3" selected. At the bottom right of the window is a yellow button labeled "Add morphology".

Fig. 4.34: Custom gradient editor

The custom gradient option allows user-defined mathematical functions to be used for determining the material fraction.

We start by defining the start and end points of the generator. Locations outside of this range will have a fraction of 0. For locations inside the range, a function is specified to convert the position x into a fraction.

The supported mathematical operations are:

- Basic arithmetic operations: +, -, *, \
- Exponentiation and modulation: **, %
- Absolute values: abs()
- Basic mathematical functions: sqrt(), exp(), log(), log10()
- Basic trigonometric functions: sin(), cos(), tan()
- Hyperbolic functions: sinh(), cosh(), tanh()

- Inverse trigonometric functions: `asin()`, `acos()`, `atan()`
- Inverse hyperbolic functions: `arcsinh()`, `arccosh()`, `arctanh()`

Warning: If the custom function returns a fraction that is less than 0 or larger than 1, the morphology generator will return an error and your simulation will not succeed.

For this example, we will use an exponential function x^x in the range of [0.2, 0.8].

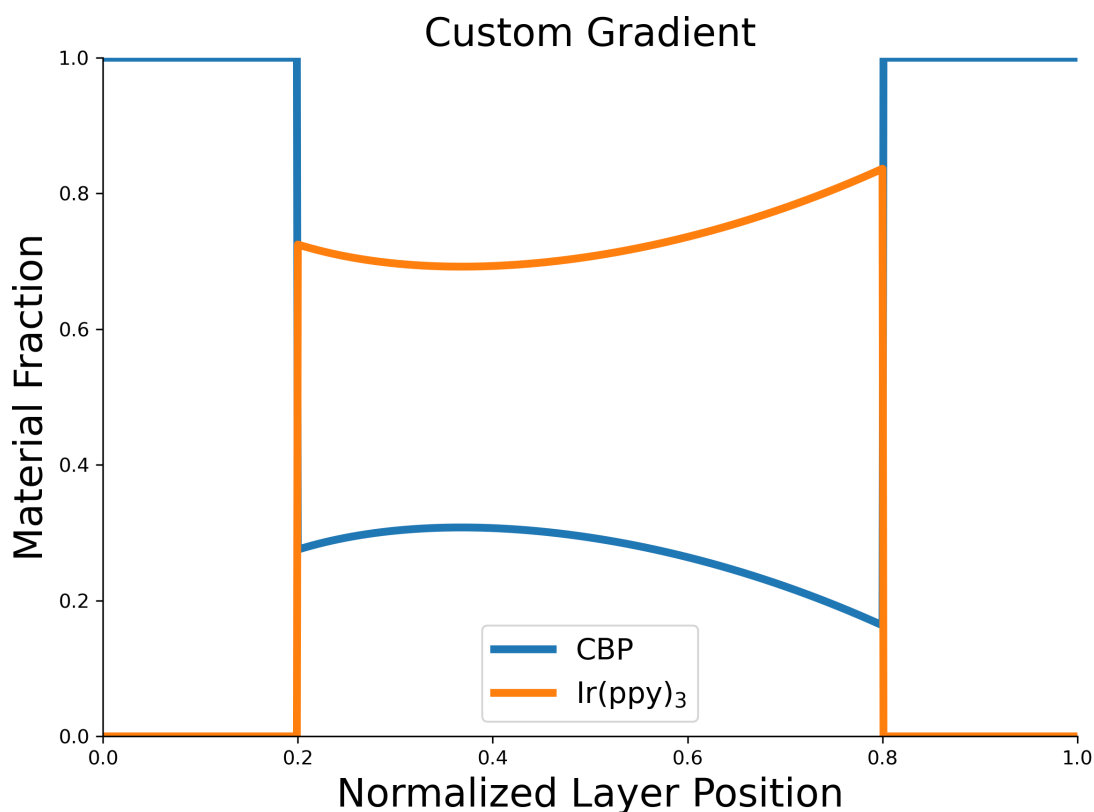


Fig. 4.35: Exponential dye gradient in the layer center

Multiple Gradients

We will create a new composition to illustrate the use of multiple gradients.

The various generators can be combined in order to create more complex morphologies. When doing this, you have to be careful that the sum of the fractions does not exceed 1.

For this example, we will consider a host-guest system containing 2 dyes: Ir(ppy)₃ and Ir(dmp)₃.

CBP is used as the host material and is therefore set as the background material. We will use a linear generator to create a static Ir(dmp)₃ fraction of 0.1. The trapezoid is used to add an Ir(ppy)₃ gradient to the edges. The custom generator is used to add Ir(ppy)₃ to the center of the layer.

Morphology

Background
CBP

Linear gradient
Ir(dmp)3
Concentration at x = 0: 0.1
Concentration at x = 1: 0.1

Trapezoid gradient
Ir(ppy)3
Positions: 0, 0.25, 0.75, 1
Concentrations: 0.35, 0, 0, 0.35

Custom gradient
Ir(ppy)3
Start: 0.25
End: 0.75
Formula: $x**x$

Fig. 4.36: Morphology configuration for multiple gradients in a dual-dye system

Visualizing Gradients

At the end of these steps, you should now have 4 advanced compositions.

You can use these compositions to set up new simulations. The generated morphologies will then be shown in the *Morphology* section of the *Box Report*.

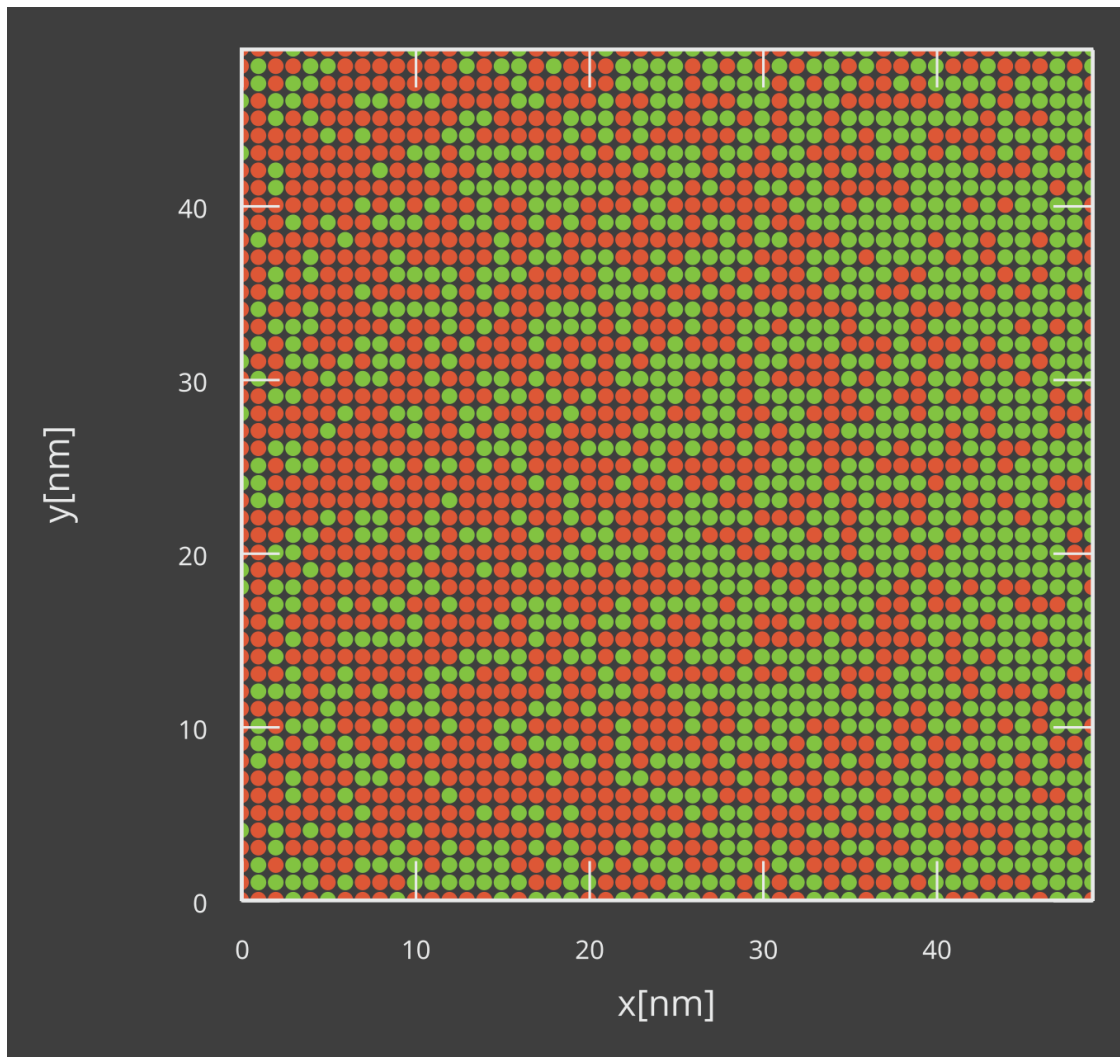


Fig. 4.37: Layer cross-section for the linear dye gradient

Layer Contacts

In order to model the microscopic roughness of layer interfaces, a composite layer can be added to the stack in order to describe the nanoscale spatial mixing of the layer components.

Using the materials defined in an earlier tutorial, we create a basic composition of 0.5 TAPC and 0.5 CBP.

We create a new stack containing 3 layers:

- A 20 nm TAPC layer
- A 1 nm layer containing the TAPC/CBP mixture

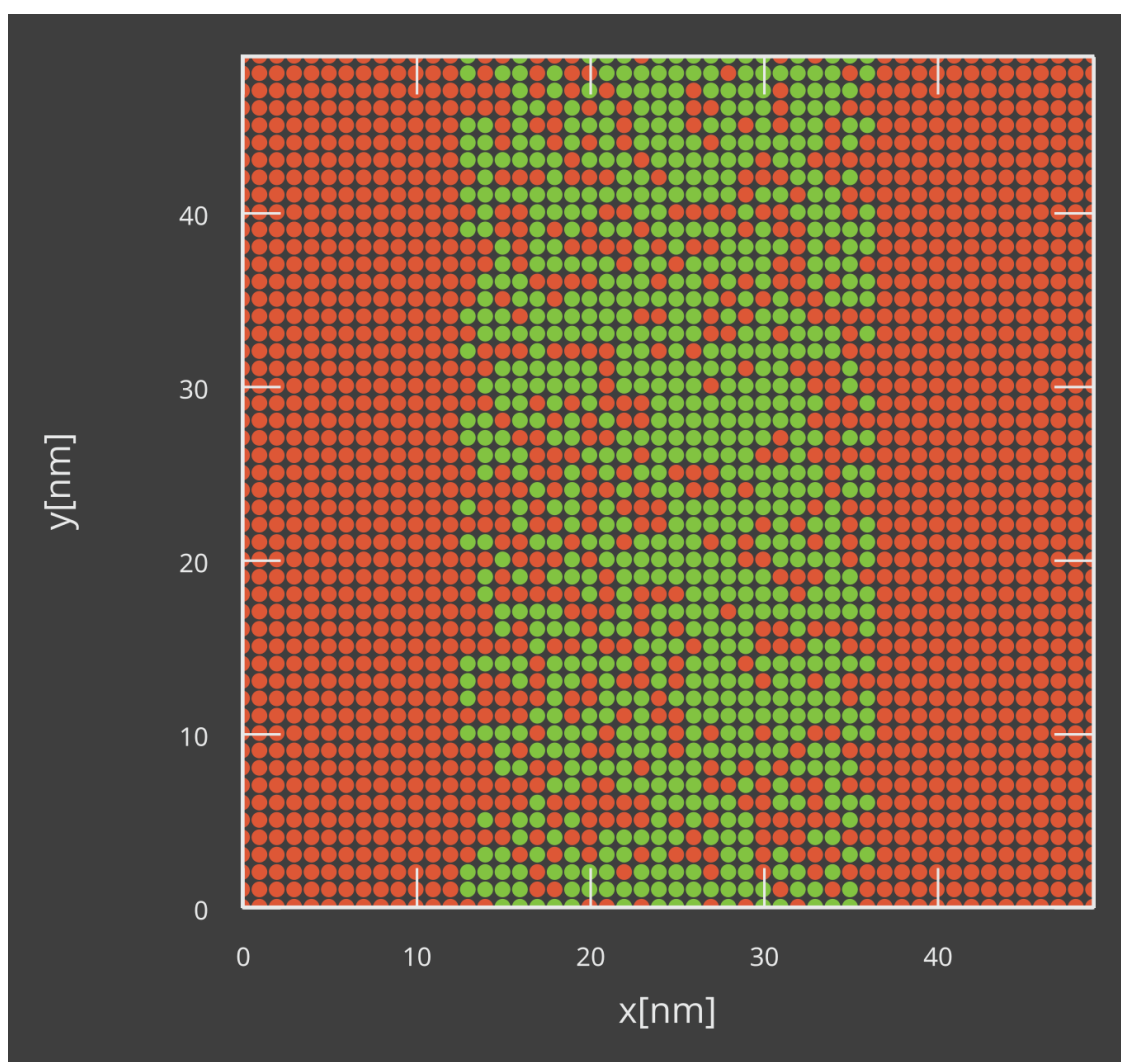


Fig. 4.38: Layer cross-section for the trapezoidal dye gradient

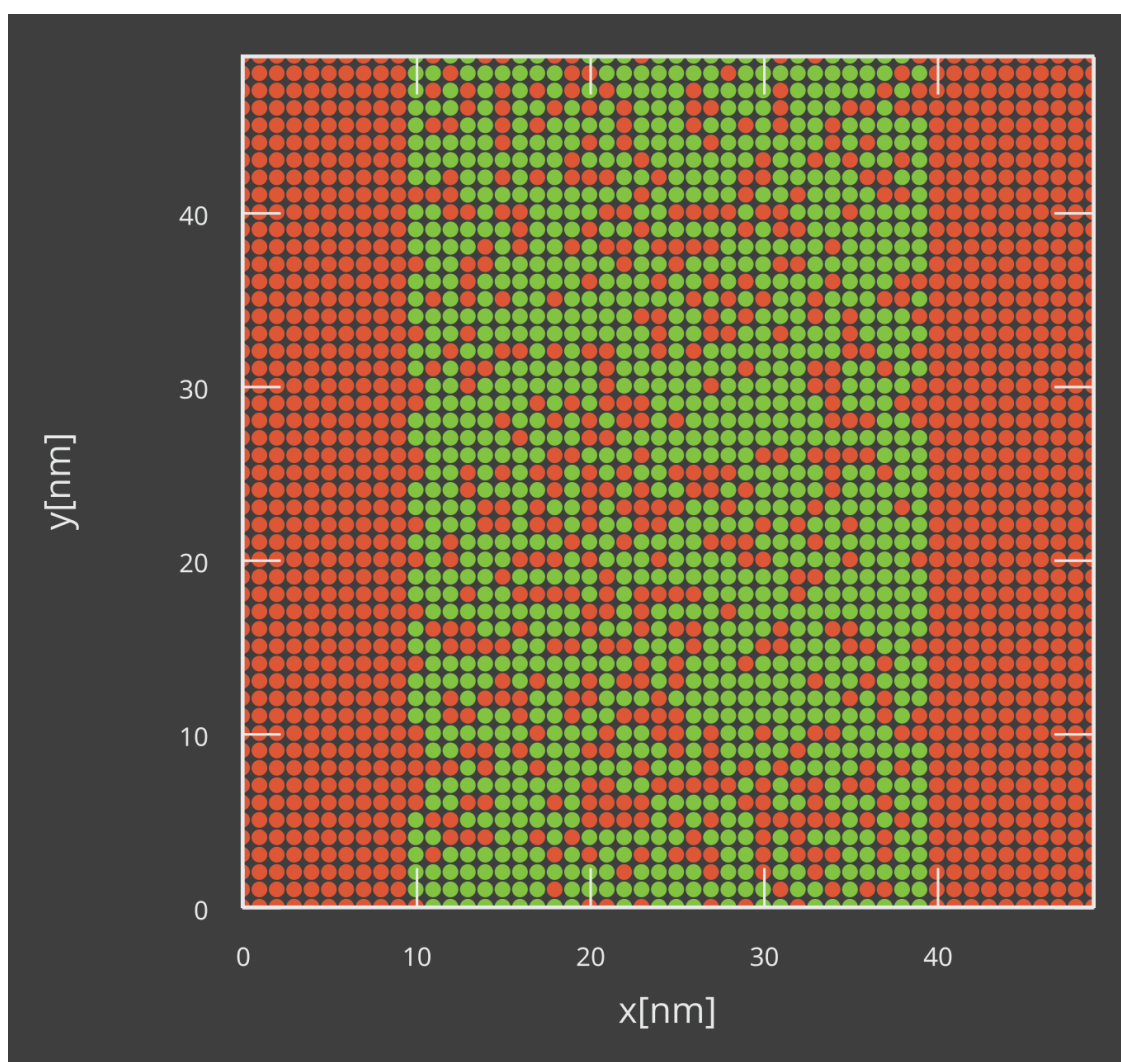


Fig. 4.39: Layer cross-section for the exponential dye gradient

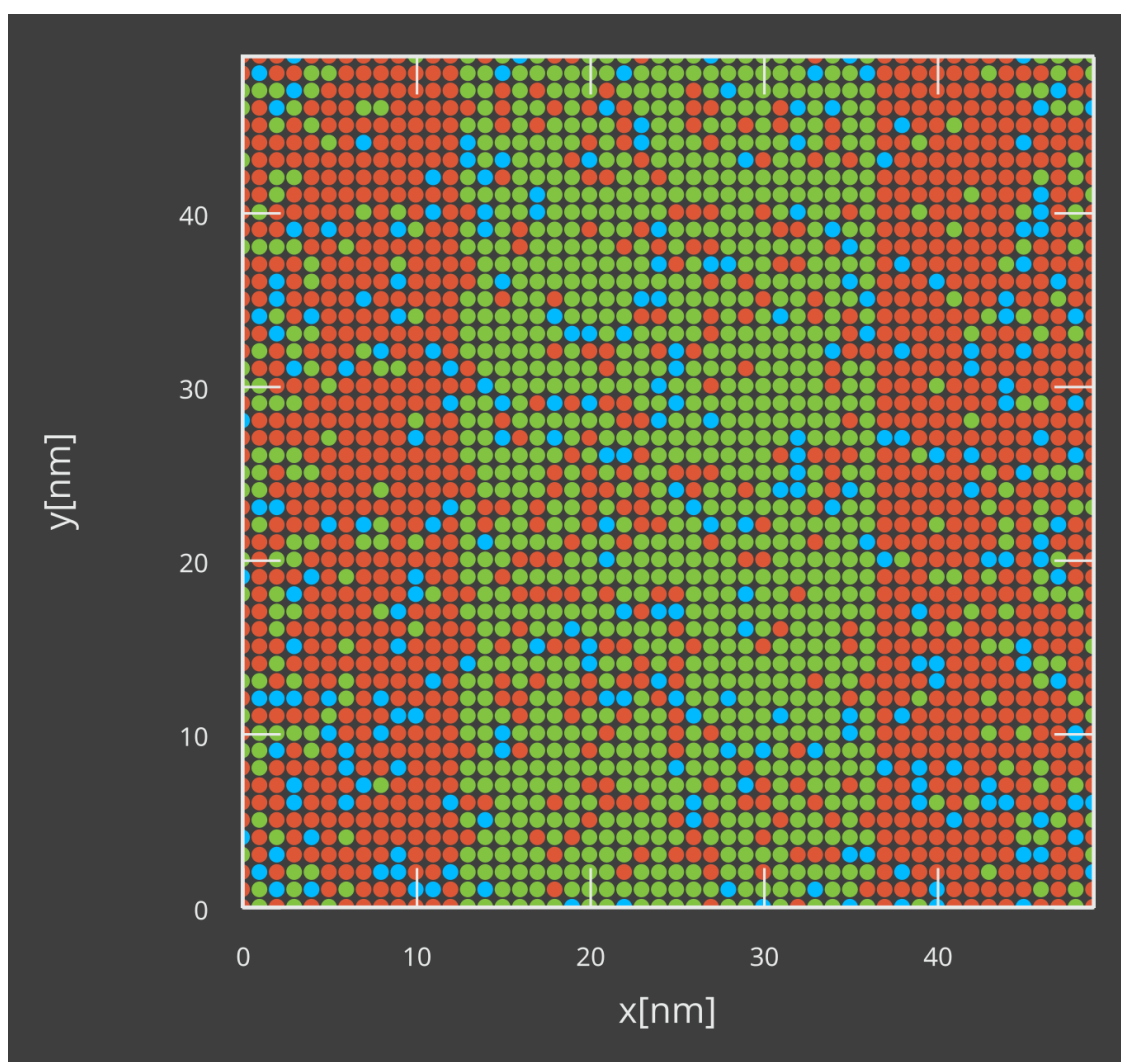


Fig. 4.40: Layer cross-section for the dual-dye system

- A 20 nm layer containing a CBP/Ir(ppy)₃ host-guest system with 10% Ir(ppy)₃

The resulting morphology localizes the disorder in the interfacial layer.

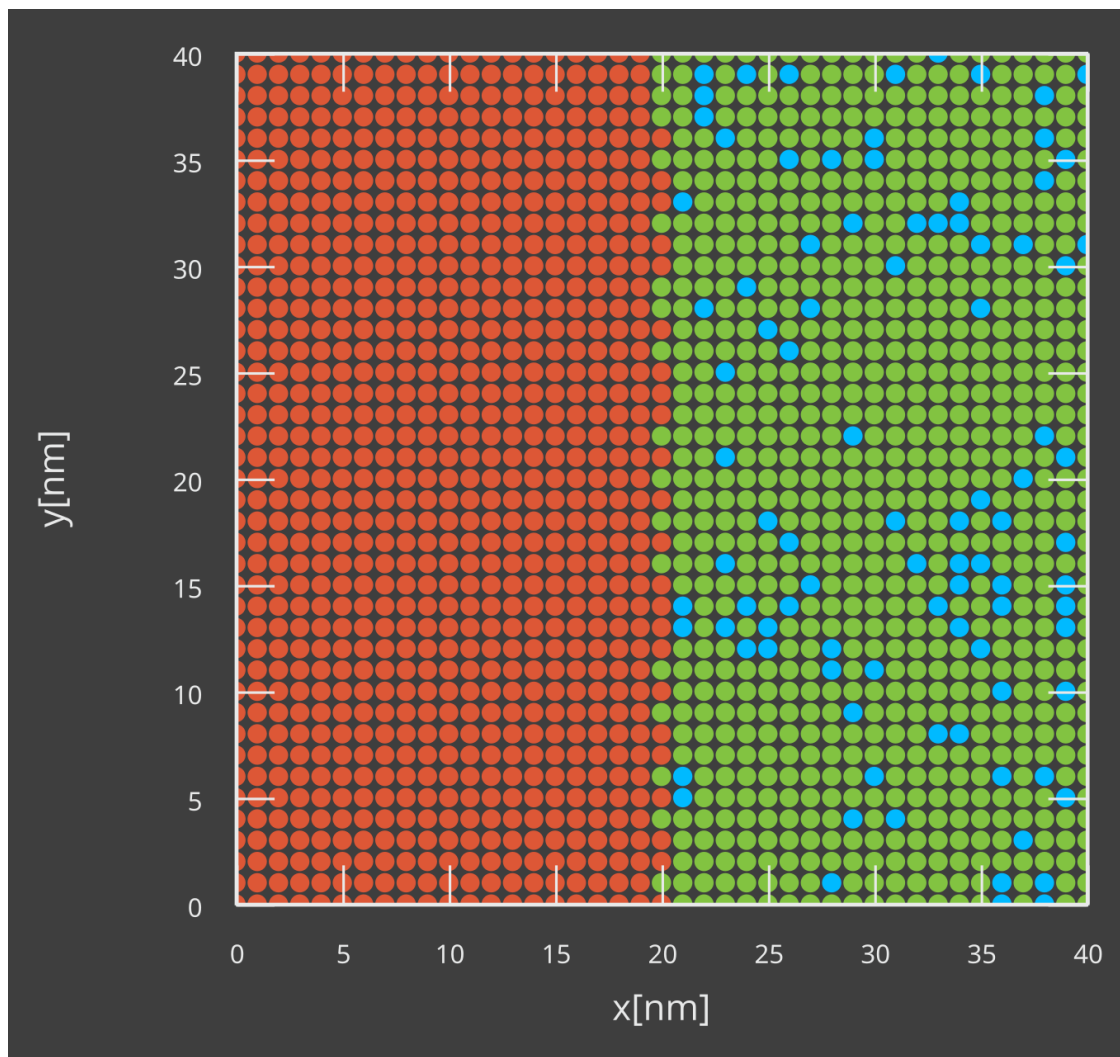


Fig. 4.41: Layer cross-section for a stack with inter-layer interfacial disorder

4.2.2 Polymeric OLED

For polymeric OLED materials, the charge transport along the polymer backbone can differ from the transport between adjacent chains. Polymeric morphologies can be created to account for this behavior during Bumblebee simulations.

Create Materials

Polymer

In this tutorial, we will consider a SY-PPV PLED. We start by generating a new material for the polymer. We will use the *Fluorescent Dye* template.

We set a HOMO level of -5.4 eV and a LUMO level of -2.8 eV. For the polymer, we will disable the energy level broadening by selecting a delta function. For the excitons, we use a singlet binding energy of 1.3 eV and a triplet binding energy of 1.6 eV. A delta function is again used to disable energy level broadening.

An enhanced singlet-triplet generation ratio of 0.4 will be used. The singlet radiative decay rate is set at 10^8 s^{-1} . The non-radiative decay rate is set at $5 \cdot 10^7 \text{ s}^{-1}$.

To describe the preferential carrier hopping along the conjugate backbone, anisotropic hopping rates can be specified in the *Advanced* tab of the materials editor.

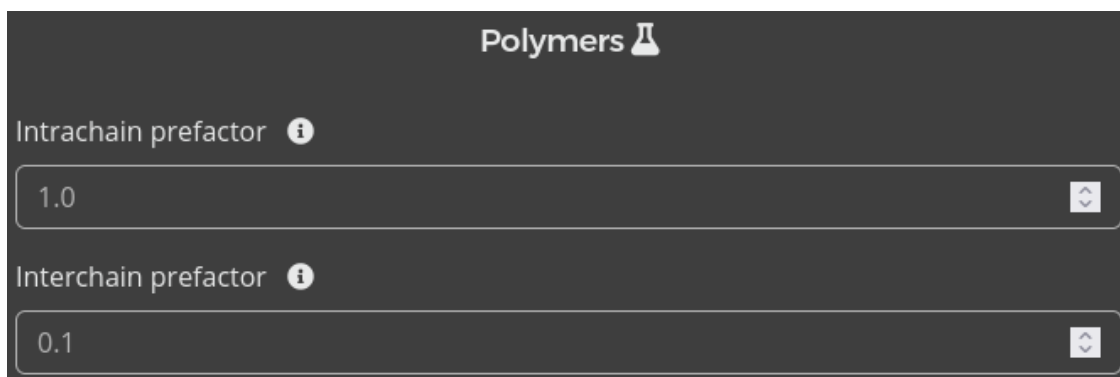


Fig. 4.42: Charge transport anisotropy settings for polymeric materials

We chose here to set an inter-chain prefactor of 0.1 to suppress charge hopping between the chains.

Vacuum Level

Due to the imperfect stacking of polymer chains in the emission layer, voids will be present between the chains. To account for this, we create a vacuum material to represent these voids.

We set a HOMO level of 25 eV and a LUMO level of 50 eV. Energy level broadening is disabled by selecting a delta function. This choice of energy levels creates a large barrier for transfer towards the vacuum, preventing this material from participating in electron transport.

Because the vacuum should not carry any excitons either, the singlet and triplet binding energies can be set at 0. Energy level broadening is disabled by selecting a delta function.

To inhibit transport, the hole mobility, electron mobility and Dexter prefactors are set to 0.

Transport Layer

PEDOT:PSS will be used as a hole transport layer. We select the *Transport* template to create a new material.

We set a HOMO level of -5 eV and a LUMO level of -2.3 eV. A Gaussian energy level broadening is enabled by default. For the excitons, we use a singlet binding energy of 0.7 eV and a triplet binding energy of 1.2 eV.

Create Compositions

In order to include the morphology of the polymer network, we will create an advanced composition.

In the composition editor, we add fractions for both SY-PPV and the vacuum. We set the vacuum as the background material.

We use the polymer generator to create a polymeric morphology. This generator will attempt to fill the layer using polymeric chains obtained through a self-avoiding walk.

A polymer fraction is specified to determine the portion of the grid that will be filled with the polymeric material. The maximum size of the individual chains is set using the chain length parameter. Note that not every polymer will be able to reach the maximum chain length, either due to confinement by neighboring chains or the finite size of the layer.

The behavior of the self-avoiding walker is set using an anisotropic growth vector, which describes the relative probability that chains grow in any given direction. The rigidity parameter restricts the polymer chain from folding back onto itself within a set number of backbone units.

For SY-PPV, we will use a polymer fraction of 0.8, a chain length of 100 and a backbone rigidity of 4. The chain growth probability towards the electrodes will be doubled in order to generate a directed conductor.

Create a Stack

We will create a stack containing 2 layers. The electron transfer is assumed to proceed through a metallic layer.

- Add a 20 nm layer of PEDOT:PSS
- Add a 60 nm layer of the polymer composition

We will enable the default Förster interactions for this stack, removing the processes that involve the vacuum.

Create a Parameter Set

We will use the *Single Voltage Point* template to create a parameter set. The voltage is set to 5 V.

Because the polymer layer contains voids, we have to manually set the electrode energy levels. We use an energy of -4.8 eV for the anode and -3.0 for the cathode.

Starting the Simulation

We start a new simulation using the voltage point parameter set.

We select 5 disorder instances to improve the sampling of the polymer network.

Add morphology

Generator ⓘ

Polymer

Polymers

Fraction ⓘ

0.8

Chain Length ⓘ

25

Rigidity ⓘ

4

Preference for x direction ⓘ

2

Preference for y direction ⓘ

1

Preference for z direction ⓘ

1

Material ⓘ

SY-PPV

Vacuum

SY-PPV

Add morphology

Fig. 4.43: Polymer generation settings

Simulation Output

The morphology of the polymer layer can be viewed in the *Morphology* section of the *Box Report*.

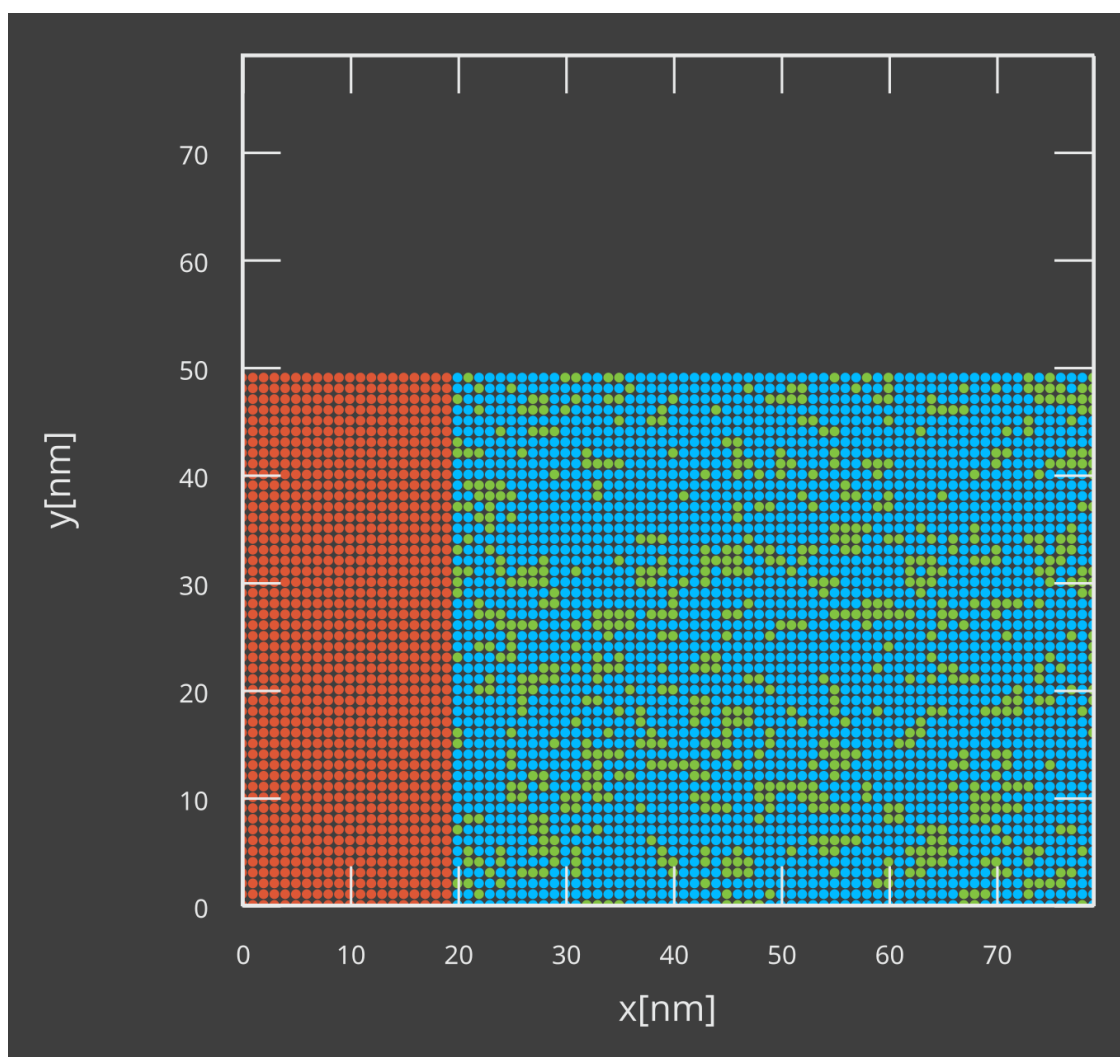


Fig. 4.44: Layer cross-section for the PEDOT:PSS/SY-PPV stack

4.2.3 Device Lifetime

Lifetime simulations can be performed to understand the role of degradation processes on the OLED performance.

Create Materials

Degradation events involve a change in molecular properties of the layer materials. For this reason, we will create both healthy and degraded variants of the dye molecule.

Phosphorescent Dye

Ir(ppy)₃ is used as the phosphorescent dye. We will select the corresponding template when creating a new material.

We specify a HOMO level of -5.27 eV and a LUMO level of -1.86 eV. A Gaussian broadening is enabled by default. For the excitons, we use a singlet binding energy of 0.75 eV and a triplet binding energy of 1 eV. The Dexter-type exciton diffusion prefactor is set to 1, with an associated decay length of 0.3 nm.

An intersystem crossing rate of 10^{10} s^{-1} is specified. The reverse intersystem crossing rate is set to 0. The radiative decay rate of the triplet excitons is set to $6.1 \cdot 10^5 \text{ s}^{-1}$. The non-radiative decay rate is set to $1.9 \cdot 10^4 \text{ s}^{-1}$.

Degraded Dye

The deactivated Ir(ppy)₃ dye molecule loses access to its radiative decay pathway. We create a new material to store these properties.

We specify a HOMO level of -5.3 eV and a LUMO level of -2.25 eV. The Gaussian broadening is maintained. For the excitons, we use a singlet binding energy of 0.85 eV and a triplet binding energy of 1.25 eV. The Dexter-type exciton diffusion prefactor is set to 1, with an associated decay length of 0.3 nm.

An intersystem crossing rate of 10^{10} s^{-1} is specified. The reverse intersystem crossing rate is set to 0. The radiative decay rate of the triplet excitons is set to 0 following deactivation. The non-radiative decay rate is kept at $1.9 \cdot 10^4 \text{ s}^{-1}$.

Host

CBP is used as the host material. Select the appropriate template when creating a new material entry.

We use a HOMO level of -6.08 eV and a LUMO level of -1.75 eV. A Gaussian broadening is enabled by default. For the excitons, we use a singlet binding energy of 1 eV and a triplet binding energy of 1.7 eV. For Dexter-type exciton transfer, a prefactor of 0.95 is used along with a decay length of 0.3.

The singlet-triplet generation ratio will be set to 0.25. Thermalization losses during exciton transport from the dye through the host are included by setting the non-radiative decay rates to 10^5 s^{-1} for singlets and 10^4 s^{-1} for triplets. The radiative decay rates are set to 0.

Electron Transport Layer

TPBi is used as an electron transport layer. Select the *Transport* template when creating a new material.

We use a HOMO level of -6.2 eV and a LUMO level of -1.7 eV. For the excitons, we use a singlet binding energy of 0.75 eV and a triplet binding energy of 1 eV. For Dexter-type exciton transfer, a prefactor of 1 is used along with a decay length of 0.3. A non-radiative decay rate of 10^8 s^{-1} is specified for both excitons.

Hole Transport Layer

TAPC is used as the hole transport layer. We use a HOMO level of -5.5 eV and a LUMO level of -0.96 eV. For the excitons, we use a singlet binding energy of 1 eV and a triplet binding energy of 1.59 eV. For Dexter-type exciton transfer, a prefactor of 1 is used along with a decay length of 0.3. A non-radiative decay rate of 10^8 s^{-1} is specified for both excitons.

Electron Blocking Layer

fac-Ir(pmb)3 is used as an electron blocking layer. Select the *Advanced* template when creating a new material.

We use a HOMO level of -5.2 eV and a LUMO level of -1 eV. A Gaussian broadening of 0.1 eV is enabled for both polarons. For the excitons, we use a singlet binding energy of 0.8 eV and a triplet binding energy of 1.4 eV. The Gaussian broadening is set to 0.05 for both excitons. For Dexter-type exciton transfer, a prefactor of 0.9 is used along with a decay length of 0.3.

For the triplet excitons, we set a radiative decay rate of $3.4 \cdot 10^5 \text{ s}^{-1}$ along with a non-radiative decay rate of $5.7 \cdot 10^5 \text{ s}^{-1}$. The singlet decay rates are kept at 0.

Create Compositions

We will create a host-guest system for the emission layer.

- We use a fraction of 0.9 for the CBP host material
- We use a fraction of 0.1 for the Ir(ppy)3 dye
- We add the degraded Ir(ppy)3 material to the composition with a fraction of 0. This makes the material accessible for degradation simulations

Create a Stack

We will create a new stack and add the different layers.

- Add a 20 nm layer of TAPC
- Add a 5 nm layer of fac-Ir(pmb)3
- Add a 40 nm layer of the CBP/Ir(ppy)3 composite
- Add a 20 nm layer of TPBi

Enable the default Förster interactions to include the relevant excitonic processes.

In the *Degradation Processes* section of the stack editor, we can specify excitonic events that can trigger degradation of the materials. Selecting the *Add Degradation Process* option will open the degradation editor.

We select *Degradation upon exciton generation* as the cause of the degradation event. We select Ir(ppy)3 as the starting material and degraded Ir(ppy)3 as the product material. We assign a probability of 1.0. All annihilation events occurring

Degradation processes ⓘ					
Add degradation process					
Cause	Layer	Material	Resulting material	Probability	
Degradation upon polaron quenching	EMI	Ir(ppy)3	Ir(ppy)3-degraded	0.8	🗑️
Degradation upon annihilation	EMI	Ir(ppy)3	Ir(ppy)3-degraded	1.0	🗑️

Fig. 4.45: Degradation process overview in the stack editor

in the Ir(ppy)3 phase will then cause the dye to be converted into the degraded material that was specified earlier. Select the *Save* option to add this mechanism to the stack.

Multiple independent processes can be added. For the current stack, we will add a second degradation event for *Degradation upon polaron quenching*. We will use a probability of 0.8, such that only 80% of polaron quenching reactions will trigger degradation.

Create a Parameter Set

Selecting the *Lifetime Simulation* template for the parameter set will configure the simulation to include degradation events. The *Degradation* option will have been enabled in the *Modules* tab.

We will set the voltage at 6 V. The remaining parameter settings are kept at their default values.

Running the Simulation

We will configure a simulation using a single disorder instance.

Once the simulation has completed, device degradation statistics are found in the *OLED Degradation* section of the *Multibox Report*.

4.2.4 Photoluminescence

Absorption processes can be included to replicate photoresponse measurements. Device models can be constructed to simulate photovoltaics and photodetectors.

Create Materials

In this tutorial, we will model the photoluminescent behavior of an organic photovoltaic (OPV).

Add degradation

Cause

Degradation upon polaron quenching

Layer

EMI

Material

Ir(ppy)3

Resulting material

Ir(ppy)3-degraded

Probability

0.8

Cancel

Save

Fig. 4.46: Degradation editor settings

Donor

P3HT is used as the donor material. Excitation processes are assumed to be singlet-dominant. We therefore choose to use the *Fluorescent Dye* template.

We set a HOMO level of -5.2 eV and a LUMO level of -3.3 eV. For the excitons, we use a singlet binding energy of 1.2 eV and a triplet binding energy of 1.4 eV. Default Gaussian broadening is used for both polaron and exciton energy levels. The Dexter prefactor for excitonic transfer is set to 1.9.

The fluorescent material template will have automatically configured a radiative singlet decay process. We will set this rate to 10^7 s^{-1} . We will simulate a singlet-mediated device by setting both singlet fractions to 1. ISC rates remain at zero.

Acceptor

PCBM is used as the acceptor material. We will use the *Fluorescent Dye* template to describe the singlet-dominant system.

We set a HOMO level of -6.1 eV and a LUMO level of -3.9 eV. For the excitons, we use a singlet binding energy of 1.6 eV and a triplet binding energy of 1.9 eV. Default Gaussian broadening is used for both polaron and exciton energy levels.

The Dexter prefactor for excitonic transfer is set to 3.1. The singlet radiative decay rate is set to 10^7 s^{-1} . Both singlet fractions are set to 1.

Electron Transport Layer

BCP will be used as the electron transport layer. We select the *Transport* template to create a new material.

We set a HOMO level of -6.3 eV and a LUMO level of -2.9 eV. A Gaussian energy level broadening is enabled by default. For the excitons, we use a singlet binding energy of 1.5 eV and a triplet binding energy of 2.6 eV.

Hole Transport Layer

PEDOT:PSS will be used as the hole transport layer.

We set a HOMO level of -5 eV and a LUMO level of -2.3 eV. A Gaussian energy level broadening is enabled by default. For the excitons, we use a singlet binding energy of 0.7 eV and a triplet binding energy of 1.2 eV.

Create a Stack

We will utilize pure compositions to construct the OPV stack. We start with a 5 nm BCP electron transport layer. We then add 10 nm layers for both the P3HT donor and PCBM acceptor. A 5 nm PEDOT:PSS hole transport layer is added to complete the device.

Enable the default Förster processes to include inter-layer singlet diffusion.

Photo-absorption processes can be added in the stack editor. Select the *Add Absorption* option to define a new absorption process. We select the P3HT material in the donor layer. Absorption is described as a fixed excitation probability per incident photon. We will assume a factor of 0.8, accounting for optical loss processes. Use the *Save* button to commit this process to the stack. Then repeat these steps to add the same reaction to the PCBM material in the acceptor layer.



Fig. 4.47: Absorption configurations in the stack editor



Fig. 4.48: Absorption process editor

Create a Parameter Set

Bumblebee offers two parameter templates for photoluminescent processes.

- The *Photoluminescence* template configures a periodic box for measuring the bulk properties of the layer materials. In addition to configuring photoluminescent simulation, higher resolutions are obtained compared to the default *Periodic Box*
- The *PV/Photodetector* template is used for regular device simulations

We will use the PV template to model the photovoltaic device.

Device Settings

We will set the electrode levels to -4.6 for a silver anode and -4.7 for an ITO cathode contact. The external device voltage will be set to 0.5 V.

Having chosen a PV template, the photoluminescence module should have been enabled automatically in the *Modules* tab.

Fluence

Photo-absorption settings are configured in the *Photoluminescence* tab.

Fig. 4.49: Photoluminescence settings in the parameter set editor

The incident irradiation is set by defining a device fluence. We use a value of 300 photons/s/nm³, in line with ambient solar lighting conditions.

We define absorption processes to yield a singlet exciton inside the OPV device.

When simulating bulk material properties, a minimum exciton density can also be added in this tab. For the current device simulation, we will keep this value at 0.

Starting the Simulation

For this tutorial, we will set up a new simulation using a single disorder instance.

A fluence sweep can be performed to investigate how the device current changes as a function of the irradiation. We choose to vary the fluence from 300 to 1500 photons/s/nm³ in 5 steps.

If you wish to limit the computational time required for this tutorial, you can perform a single-point calculation instead. This will use the 300 photons/s/nm³ default fluence defined in the parameter set.

Simulation Output

The power efficiency of the OPV can be viewed in the *OLED Luminance* section of the *Sweep Report*.

Absorption characteristics can be viewed in the *Transient Photoluminescence* section of the *Multibox Report* for various irradiation densities.

4.2.5 Transient Switching

Kinetic Monte Carlo simulations can be used to investigate the response of organic electronic devices to external perturbations.

Here, we consider the effect of switching the device illumination on the current produced by an organic photovoltaic (OPV).

Create Compositions

The OPV layer materials were created as part of the photoluminescence tutorial. Here, we will consider the photoresponse of a bulk heterojunction morphology for the OPV.

We create an advanced composition using the PCBM and P3HT components. Select PCBM as the background material. A P3HT linear gradient is used ranging from a 0.8 fraction at the anode to 0.2 at the cathode.

Create a Stack

The OPV stack is created by selecting a 5 nm BCP electron transport layer, a 20 nm P3HT/PCBM donor/acceptor heterojunction layer and a 5 nm PEDOT:PSS hole transport layer.

Enable the default Förster processes to include inter-layer singlet diffusion. We add absorption processes to both P3HT and PCBM materials in the heterojunction layer. We use an absorption probability of 0.8 for both materials to account for optical loss processes.

Create a Parameter Set

We will use the PV template parameter set for the photovoltaic device. This will automatically enable the photoluminescence module.

Device Settings

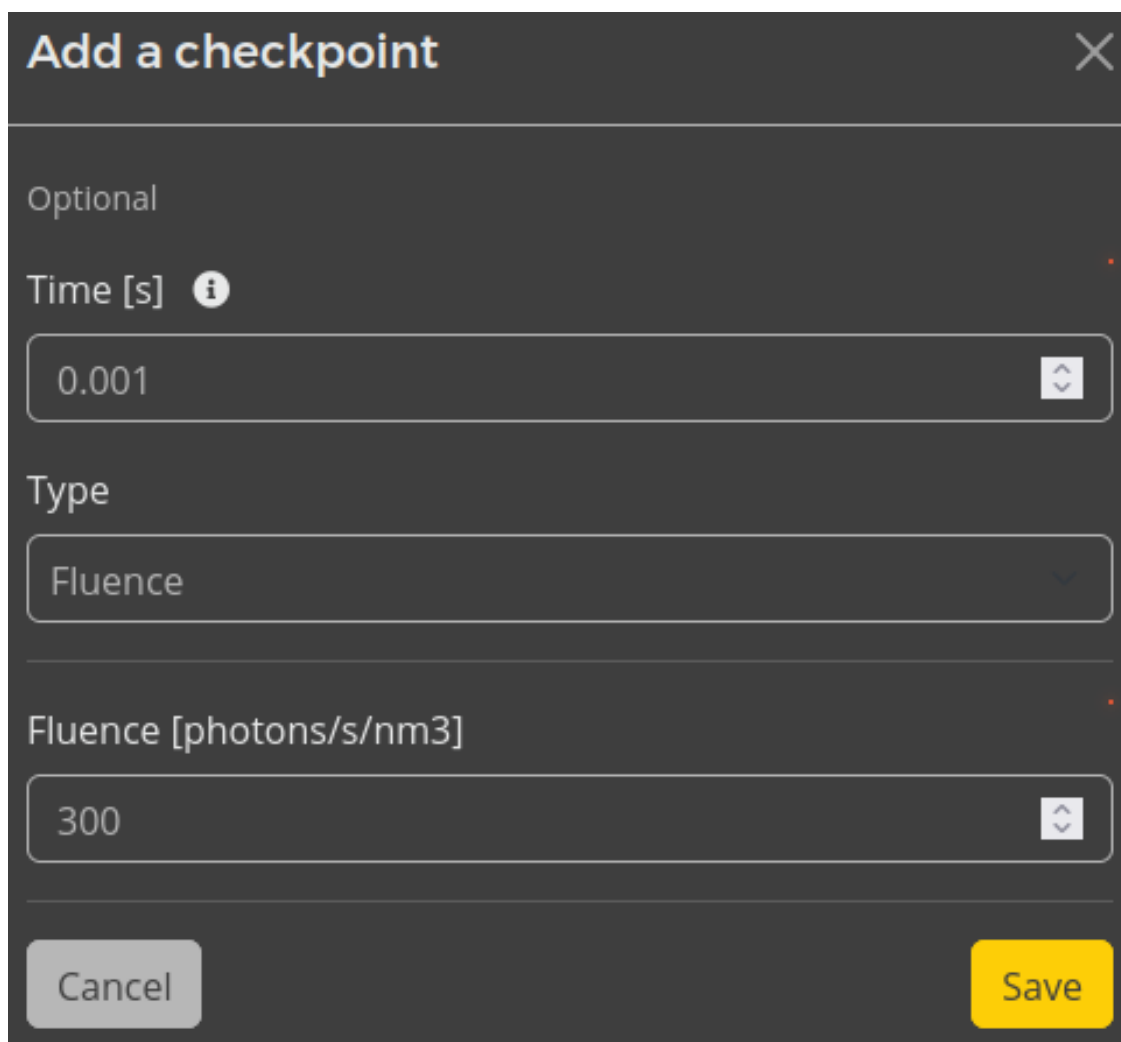
We set the electrode levels to -4.6 for a silver anode and -4.7 for an ITO cathode contact. The external device voltage will be set to 0.5 V.

In the *Photoluminescence* tab, we set the starting fluence to 0. Absorption processes are configured to yield singlet excitons.

Transient Switches

Perturbations to the device operation can be added in the *Transient Parameters* tab.

Device perturbations must first be activated by enabling checkpoints. Individual perturbations can then be added to the simulation by selecting the *Add checkpoint* option.



Add a checkpoint [X]

Optional

Time [s] ⓘ

0.001

Type

Fluence

Fluence [photons/s/nm3]

300

Cancel Save

Fig. 4.50: Checkpoint editor for configuring transient responses

Checkpoints can be provided for various parameters, such as the voltage, current or fluence. Checkpoints occur at a designated time after the simulation has commenced. Multiple checkpoints of different types can also be combined to replicate complex interactions.

After 0.001 seconds, we will set the fluence to 300 photons/s/nm³. Light switching is investigated by alternating the fluence every 0.001 seconds.

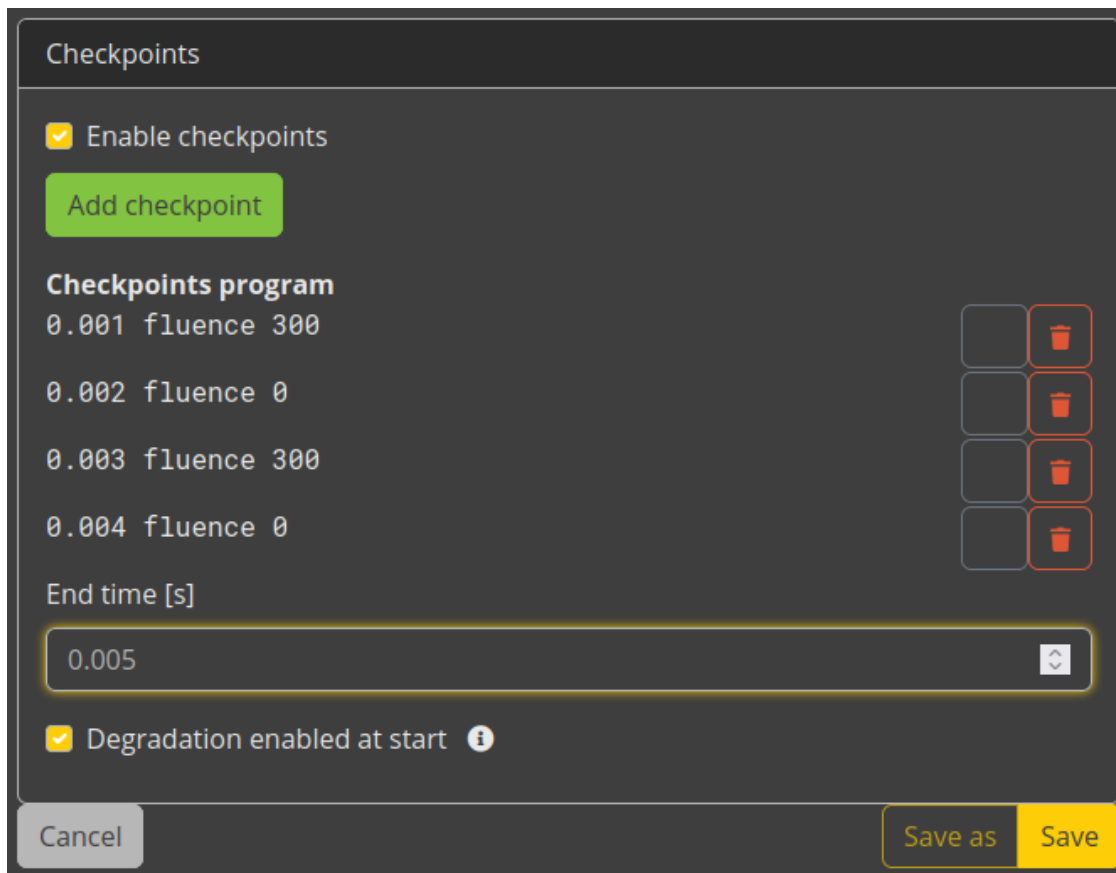


Fig. 4.51: Transient configuration for pulsed illumination

An end time of 0.005 seconds is specified in the *Transient Parameters* tab, after which the simulation concludes. This end time overrides the maximum simulation time defined in the termination criteria.

Running the Simulation

A single disorder instance will be used for this tutorial.

After the simulation has concluded, the transient response of the OPV current to the pulsed illumination can be viewed in the *Transient Photoluminescence* and *Transient OLED Response* sections of the *Multibox Report*.

4.2.6 OFET Simulation

Organic field-effect transistors (OFET) utilize organic electronics as charge transport materials in molecular transistors.

OFET modeling in Bumblebee is performed by adding a source-drain field perpendicular to the primary gate electrodes. This allows the cross-current behavior to be modeled.

The OFET device uses an organic conductor to channel carriers between the transistor source and drain. Dielectric layers are used to shield the gates.

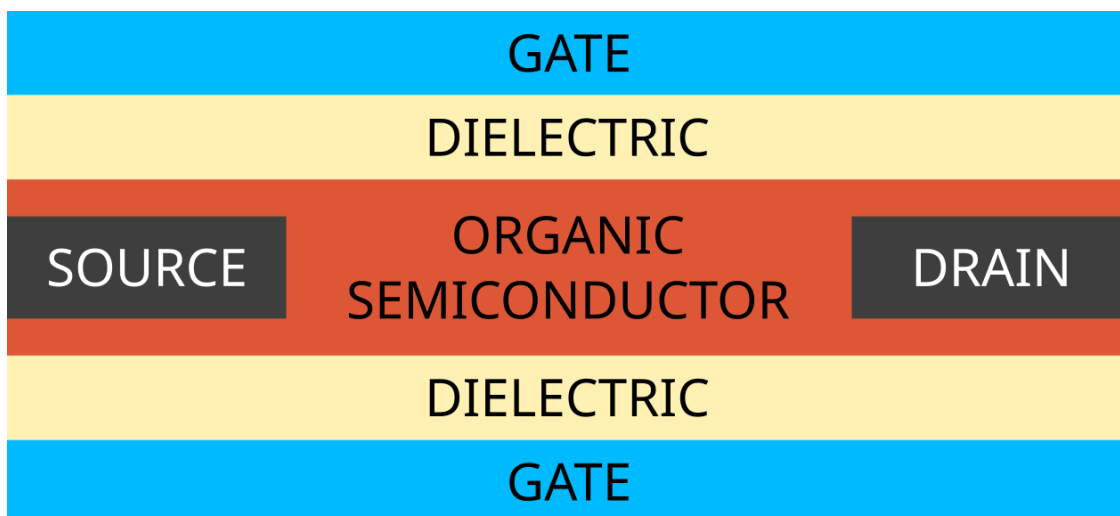


Fig. 4.52: Default probe-field OFET device geometry used by Bumblebee

Bumblebee uses 2 gate electrodes by default. In order to model single-gate OFETs, an insulator can be used to cut off access to one of the gates.

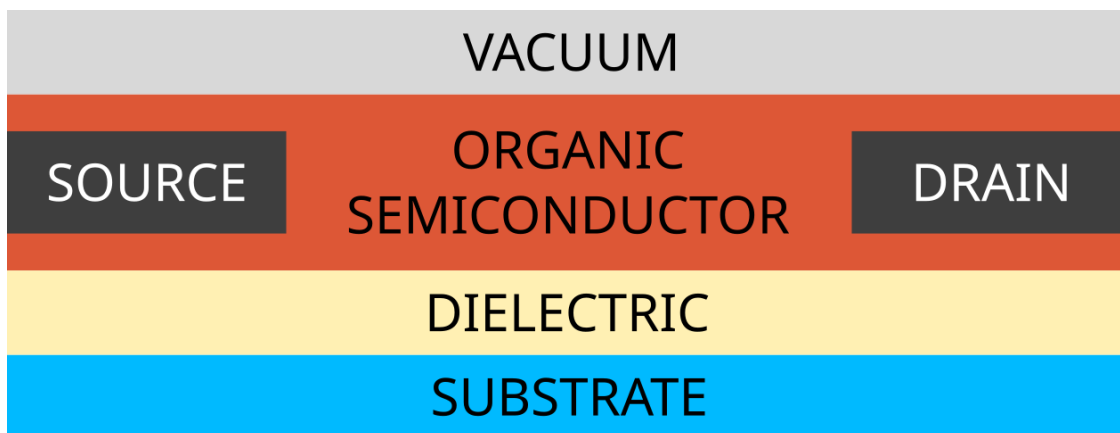


Fig. 4.53: Single-gate OFET device configuration

Create Materials

In this tutorial, we will discuss the modeling of an OFET memory device, using a charge trapping layer for polaron confinement.

We will not focus on the role of excitons in this tutorial. For this reason, the OFET components will be treated as transport layers. (If desired, excitonic processes can be added to the OFET simulation to account for internal loss mechanisms.)

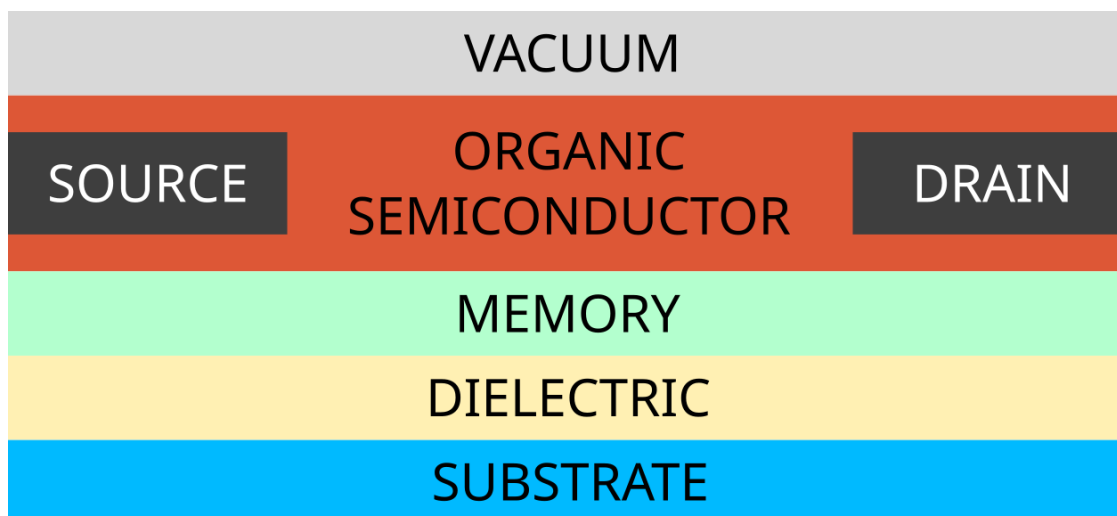


Fig. 4.54: OFET memory device configuration (MOFET)

Conductive Layer

Pentacene is used in the conductive channel. We will use the *Transport* template when creating a new material.

We set a HOMO level of -6.1 eV and a LUMO level of -3.9 eV. For the excitons, we use a singlet binding energy of 1.2 eV and a triplet binding energy of 1.4 eV. Default Gaussian broadening is used for both polaron and exciton energy levels.

The electron mobility prefactor is set to 0.5 to study the effect of a slower diffusion rate of the electron species compared to the holes.

Dielectric Layer

SiO₂ is used as a dielectric. The *Transport* template is used when creating this material.

We set a HOMO level of -9 eV and a LUMO level of -1 eV. For the excitons, we use a singlet binding energy of 2 eV and a triplet binding energy of 2.1 eV. Default Gaussian broadening is used for both polaron and exciton energy levels.

Both electron and hole mobility prefactors are set to 0.1 to account for polaron blocking behavior of the dielectric.

In the *Advanced* tab, we can specify a source-drain injection prefactor. For the dielectric, this value is adjusted to 0 to block off the transistor contacts, which are meant to connect only to the conductive layer.

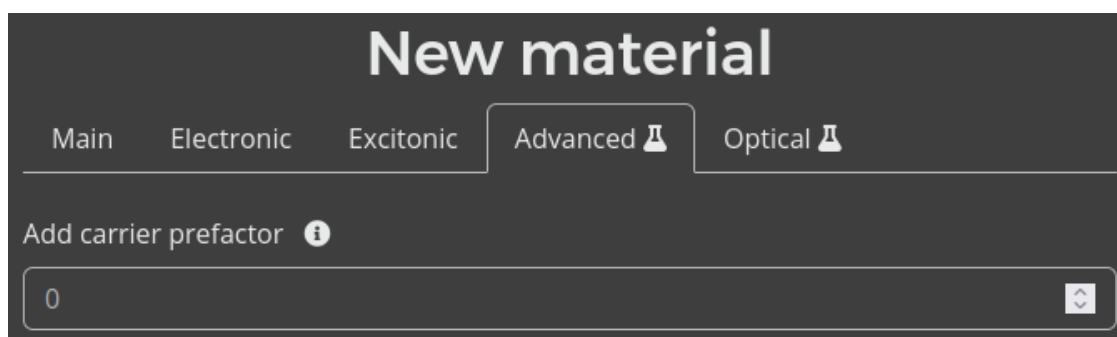


Fig. 4.55: Source-drain injection prefactor setting in the Advanced material configuration

Memory Layer

PVN is used as the charge storing material in the memory layer. We use the *Transport* template to create this material.

We set a HOMO level of -6.2 eV and a LUMO level of -1.2 eV. For the excitons, we use a singlet binding energy of 1.1 eV and a triplet binding energy of 1.5 eV. Default Gaussian broadening is used for both polaron and exciton energy levels.

The source-drain injection prefactor for the memory layer is set to 0, as only the conductive layer interacts with the transistor contacts.

Create a Stack

We will use the anode as the active gate. Therefore, we start by placing a 15 nm SiO₂ layer. This is followed by a 10 nm PVN layer and a 35 nm Pentacene layer.

Create a Parameter Set

We will use the *OFET* template when creating the parameter set.

The device voltage is set to 10 V.

The anode contact is taken as the transistor gate. We assume a Au contact and set the Fermi level to -5.1 eV.

The cathode will represent the interface with the air. We set the Fermi level to 0 eV to avoid interactions with the device.

The *OFET* template should have automatically enabled the *Transistor* module, which adds the source-drain contacts to the device. Polaron injection at the gates is also disabled. This prevents polaron hopping into the vacuum.

Note: If you want to model the OFET in a cross-current setup, with active gates, electrode injection can be re-enabled by adjusting the injection and collection prefactors in the *Advance Parameters* tab.

Note: The exciton module is not enabled by default in the *OFET* template. If you want to model excitonic processes inside the OFET, remember to manually enable this module.

The *Transistor* tab allows configuration of the transistor voltage. The drain-source field is initially set to 3 V. Technical parameters are provided for controlling the gate field.

The single-gate transistor has zero electric field at the vacuum interface. This condition is maintained as charge carriers populate the device, resulting in a change in the electric field inside the stack.

An update is performed at a given field check interval to correct the gate field for the changing polaron distribution. By default, this is implemented as an external field correction. The field is adjusted incrementally using the field control step size. Alternatively, polaron exchange with the gate can be enabled to adjust the carrier distribution.

To avoid oscillations in the electric field, a field control margin is defined. Field updates are only performed when the gate field error exceeds this value.

The adjustment to the gate field is performed to maintain the vacuum level. The position of the vacuum is given by the field check position. For the MOFET stack, this value is set to 60 nm.

Drain-source probing field [V/box length] ⓘ

3

Technicalities

Field check interval ⓘ

10000

Field check position ⓘ

60

Field control margin ⓘ

0.01

☒ Adjust gate field instead of amount of carriers ⓘ

Field control step size [V] ⓘ

0.0001

Cancel Save as Save

Fig. 4.56: Transistor configuration in the parameter set

Starting the Simulation

Sweeps of the gate voltage can be used to obtain OFET transfer curves. For this tutorial, we perform a voltage sweep from -10 to 10 V using 11 steps.

Parameter sweeps can also be performed for the drain-source field strength to investigate the effect of cross-field interactions on the device current.

To simulate memory programming/eraser cycles, transient switches can be used to adjust the device polarity during the simulation. Consult the transient response tutorial for more details.

Simulation Output

Polaron mobility, channel conductivity and transfer curves are available in the *OFET Report* section of the *Sweep Report*. The response of the memory layer can be analyzed in the *Transient OLED Response* section of the *Multibox Report*.

4.2.7 Device Equilibration

When an OLED circuit is connected, the voltage over the device gradually increases until steady-state conditions are reached. This voltage ramp can be included as part of the OLED simulation.

Circuit Closure

We will investigate circuit closure for the phosphorescent OLED constructed in the exciton tutorial.

We create a new voltage-point parameter set. We select the phosphorescent stack and choose a default voltage of 5 V.

In the *Termination Criteria* tab, we set the target convergence to 0.05. This will automatically terminate the simulation once the system is within 5% of the steady-state current.

The *Annealing and Equilibration* tab allows configuration of the voltage ramp. We set the starting voltage to 0 V.

The ramp implementation increments the voltage over a fixed number of intervals, resulting in a stepped gradient. Here, we set the number of steps to 10.

The total duration of the voltage ramp is specified in terms of the number of Monte Carlo steps. We will use the first 2 output intervals (as defined in the output settings).

Separate report and output intervals can be specified during the voltage ramp, allowing you to use different parameters compared to the rest of the simulation. In order to obtain output at the end of every ramp stage, we set both the report and output intervals equal to 1/10th of the ramp duration.

After the simulation has concluded, transient current profiles can be viewed in the *Transient OLED Response* section of the *Multibox Report*.

Circuit Disconnect

When the OLED circuit is disconnected, the external voltage drops near-instantly. This behavior can be investigated using the transient response feature.

During this process, we first want the device to be operating at steady-state conditions, i.e. having a stable internal field. A pre-equilibration stage may be specified in the simulation settings to allow the system to de-correlate from the initial state, approaching the device equilibrium. Simulation statistics will then only be generated for the samples obtained after pre-equilibration.

Annealing phase (voltage steps)

Number of time steps in annealing phase ⓘ

2000000

Starting voltage for annealing phase [V] ⓘ

0.0

Number of ramping steps towards final voltage ⓘ

10

Report interval during annealing phase ⓘ

200000

Output interval during annealing phase ⓘ

1000000

Fig. 4.57: Voltage ramp configuration in the parameter set

We create a new voltage-point parameter set. The starting voltage is kept at 5 V. The target convergence is again set at 5%.

Equilibration phase

Number of time steps in equilibration phase ⓘ

5000000

Report interval during equilibration phase ⓘ

100000

Output interval during equilibration phase ⓘ

1000000

Fig. 4.58: Pre-equilibration stage in the parameter set

The pre-equilibration is configured in the *Annealing and Equilibration* tab. We set a number of equilibration steps equal to 5 times the output interval. As with the voltage ramp, custom report and output intervals may be specified during the pre-equilibration stage. As we are interested in the dynamic behavior here, we will use the regular simulation intervals.

We will move to the *Transient Parameters* tab to configure the voltage switch. We create a new checkpoint at 0.001 seconds, at which point the voltage will be set to 0 V.

We can now run the simulation and analyze the transient current profiles.

4.2.8 Advanced Initialization

The disordered nature of typical OLED materials gives rise to a diversity in molecular environments. This diversity is accounted for by including e.g. energy level broadening and is an important feature in describing charge transport inside the device.

The description of the molecular disorder can have a strong impact on the simulated performance. Various settings are provided to customize the disorder character.

Charge Carrier Initialization

By default, devices are initialized without charge carriers. Polaron injection instead occurs at the electrode interface. For bulk materials, a minimum charge carrier density is specified to populate the device without direct electrode contacts.

Additional charges can be distributed throughout the device. This can be utilized to accelerate device equilibration, to simulate specific off-equilibrium conditions or to investigate dopants and trap states. Initial carrier concentrations are configured in the stack editor by specifying the fraction of sites that is occupied by a given type of polar or exciton.

Note: The cost of the kinetic Monte Carlo simulations scales directly with the number of carriers in the device, as process rates have to be evaluated for each individual particle. High carrier densities can result in long simulation times.

Add starting condition ✕

Layer

EMI

Material

fac-Ir(ppy)3-CBP

Type

Triplet

Probability (e.g. 0.04 means ~4% of the molecules occupied)

0.02

Cancel Save

Fig. 4.59: Dopant editor

Starting conditions ⓘ

Add starting condition

Layer	Material	Type	Probability	
EMI	fac-Ir(ppy)3-CBP	Triplet	0.02	

Fig. 4.60: Dopant configuration in the stack editor

Consistency in HOMO, LUMO and Exciton Levels

Energy level broadening can be enabled for the OLED materials.

By default, the energy level shifts for the HOMO and LUMO are treated as being uncorrelated. The HOMO shift and the LUMO shift are then computed independently from one another.

A correlated HOMO-LUMO shift can be requested in the parameter set. This preserves the band gap of the material.

Anti-correlation is also supported, such that the shift in the HOMO level is opposite that of the LUMO.

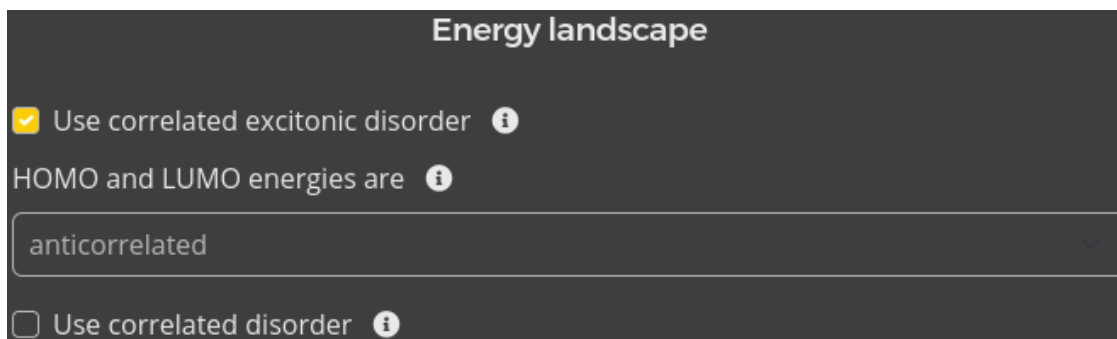


Fig. 4.61: Configuration of energy level shifts in the parameter set

Exciton energies can be correlated to the HOMO-LUMO shifts. The change in exciton binding energy is then scaled to be proportional to the change in band gap.

Dipole Orientation

The polaron and exciton energy level broadening can be modified by specifying a dipole field.


A random arrangement of dipoles can be generated by selecting the *Correlated Disorder* option in the *Energy Landscape* settings in the parameter set. The dipolar coupling at each gridpoint will then be evaluated to determine the spatial energy level distribution.

Alternatively, biased dipole fields can be specified as part of the dipole settings of the materials. Quantile functions are then used to sample three-dimensional vector orientations. Dipole strengths are provided for each material to allow scaling of the interactions based on the composition morphology.


The non-uniform dipole fields must be enabled in the *Dipole-induced Disorder* section of 'Energy Landscape' tab in the parameter set. This will override the random dipole field if both are selected.

By default, interactions across the periodic boundaries are accounted for when calculating the dipolar coupling.

It is possible to override the material-specific quantile functions to eliminate field discontinuities at the material interfaces. Differences in the dipole strength of the materials is still accounted for in this consistent field, such that the material-correlation is preserved.

Dipoles 

Static dipole moment [D] ⓘ

1.2 

Quantile function $f(a,b,c)$ x-axis ⓘ

$2*a-1$

Optional

Quantile function $f(a,b,c)$ y-axis ⓘ

$2*\sqrt{a-a*a}*\sin(2*PI*b)$


Optional

Quantile function $f(a,b,c)$ z-axis ⓘ

$2*\sqrt{a-a*a}*\cos(2*PI*b)$

Optional


Fig. 4.62: Material dipole settings

Dipole-induced disorder 

☒ Add molecular dipole contribution to energy landscape

☒ Include image dipoles (electrodes)

Number of dipole images to consider

100 

☒ Use material-specific dipole distribution

Cancel Save as Save

Fig. 4.63: Dipole field generator in the parameter set

4.3 API

These tutorials document the use of the Python API.

4.3.1 API Access

Bumblebee provides a Python API to allow for automated submission of Bumblebee simulations.

This tutorial documents:

- Installation of the Python API
- How to connect the Python API to the web interface
- How to access, modify or create simulation input
- Job submission and access to simulation output

API Installation

The Bumblebee Python package can be obtained from the Downloads tab in the web interface. Clicking on your username will open a menu, from which you can access Downloads.

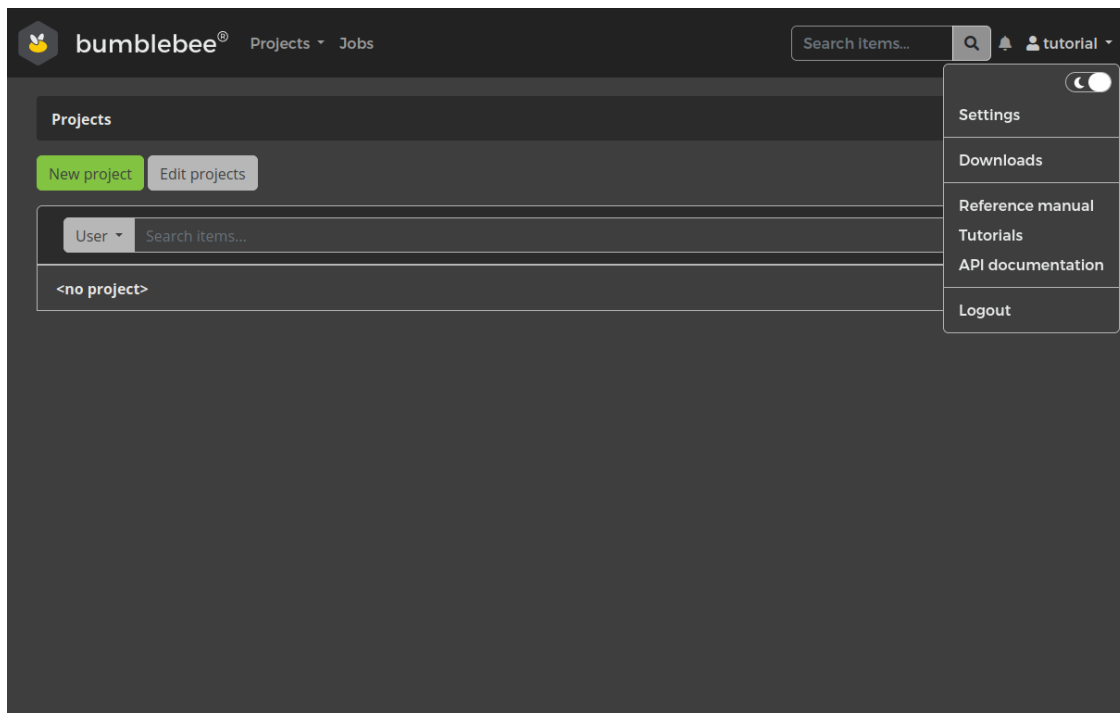


Fig. 4.64: Access Downloads in the web interface

Installation packages are provided as either a *wheel* or *sdist* file.

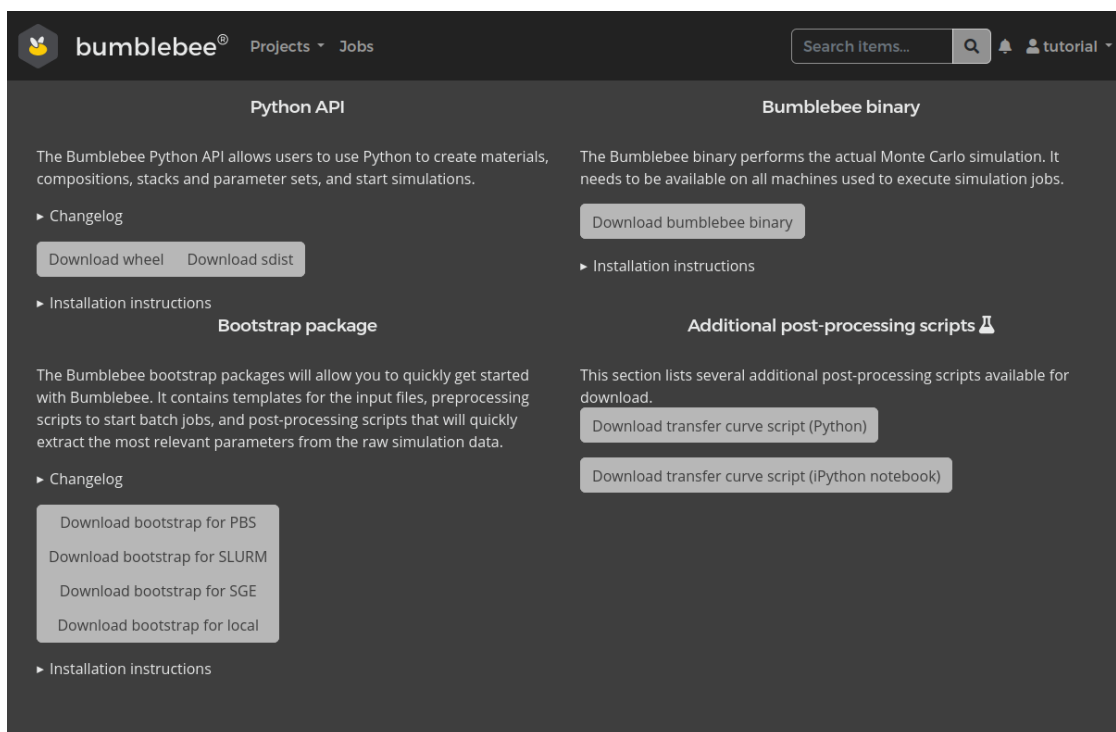


Fig. 4.65: Downloads page containing the Bumblebee Python package

Wheel installation

The Bumblebee Python package can be installed using pip.

To install or upgrade the Bumblebee Python package, download the *wheel* file to your computer and make sure that pip is installed. The API can then be configured from the *wheel* file:

```
python -m pip install bumblebee-api.whl
```

This will install or upgrade the API package. After the installation, Bumblebee can be used in any Python script by including `import bumblebee`.

Sdist installation

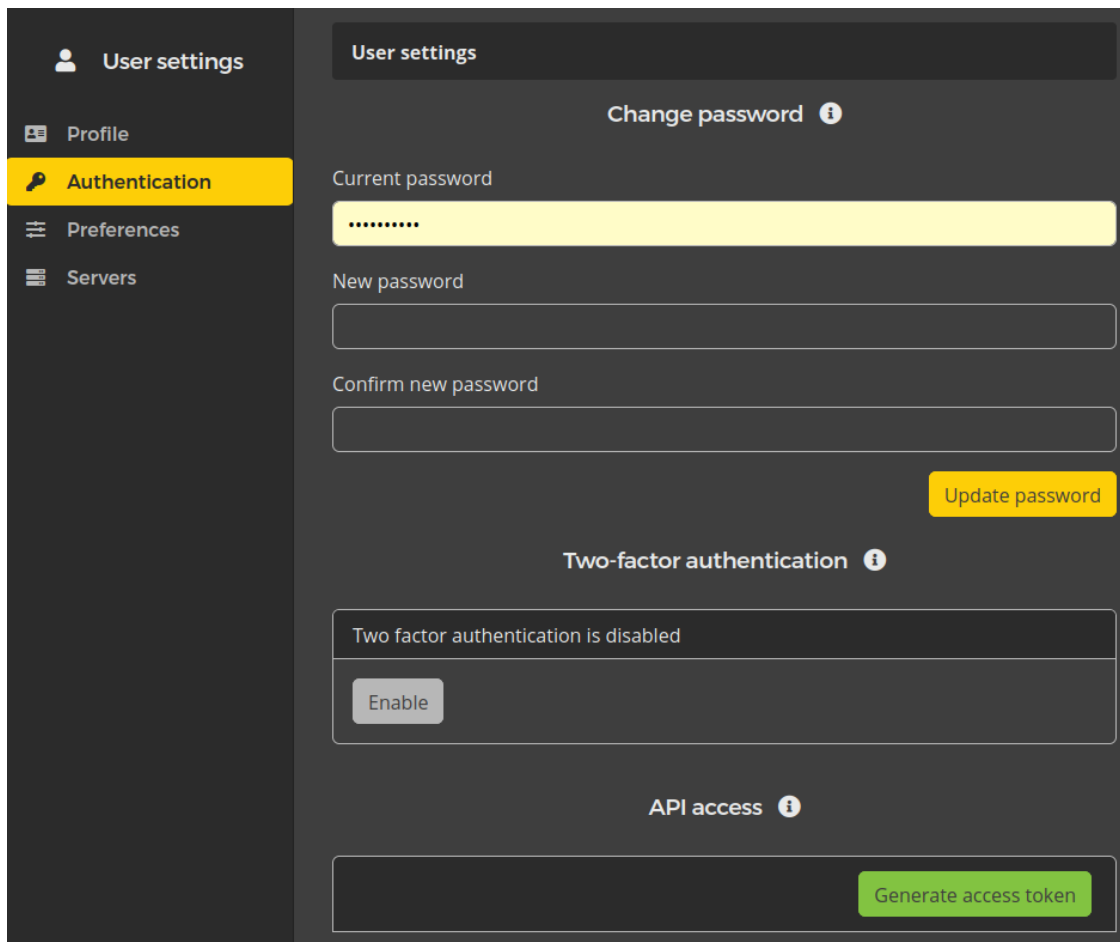
To install or upgrade the Bumblebee Python package, download the *sdist* file to your computer and make sure that pip is installed. The API can then be configured from the *sdist* package:

```
python -m pip install bumblebee-api.tar.gz
```

This will install or upgrade the API package. After the installation, Bumblebee can be used in any Python script by including `import bumblebee`.

Configuring the API Connection

A valid API token is required to connect the Python API to the web interface. Login to your Bumblebee account and go to *Settings/Authentication*. In the *API access* section, the *Generate access token* option will generate and download a new credentials file.



The screenshot shows the 'User settings' page with a sidebar on the left containing 'Profile', 'Authentication' (highlighted), 'Preferences', and 'Servers'. The main content area is titled 'User settings' and contains three sections: 'Change password' with fields for 'Current password', 'New password', and 'Confirm new password', followed by an 'Update password' button; 'Two-factor authentication' with a message 'Two factor authentication is disabled' and an 'Enable' button; and 'API access' with a 'Generate access token' button.

Fig. 4.66: Manage API tokens in the user authentication panel

Warning: The API token provides access to the data stored in your web interface account. If you believe token access has been compromised, you can delete specific tokens from the settings menu.

Establishing the API Connection

In order to use the Python API from a Python script, the Bumblebee Package must be loaded. A connection to the web interface is then made by creating a `Client` object.

```
import bumblebee
bb_client = bumblebee.Client()
```

By default, Python will look for the credentials file in the current working directory. An alternative location can be specified using:

```
bb_client = bumblebee.Client(credentials_path="/home/user/Downloads/bumblebee-api.pem")
```

Note: The API Client requires a network connection to the web interface in order to synchronize with the database.

Creating New Resources

The Bumblebee Client allows for the creation of new resources such as materials, compositions, stacks or parameter sets. This can be achieved by using the `new` method.

```
material = bb_client.new('material')
```

These resources are stored locally inside your Python environment and are not automatically added to the database in the web interface. This allows you to make modifications to the local resources before committing them to persistent storage.

New resources use the same default values as those created by the web interface. The `print` function can be used to see the current resource parameters.

```
print(material)
```

Individual resource parameters can be accessed using their common names. These common names are available as part of the tooltips in the web interface. Simply hover over the info icon to view the API name corresponding to the parameter.

Modifications to the resource parameters can be performed using simple Python assignments.

```
material.name = "Ir (ppy3) "
material.homo = -5.27
material.lumo = -1.86
```

After you have set all the necessary resource parameters, you can upload the resource to the database using the `create` method. This will make the resource available from within the web interface.

```
material.create()
```

Warning: Resources that are not uploaded to the database will be permanently lost when the Python session is closed.

If you wish to make a modification to the resource after it has been added to the database, the `update` method must be used instead.

```
material.sigma_homo = 0.1
material.sigma_lumo = 0.1
material.update()
```

Note: Calling the `create` or `update` methods on a simulation resource will automatically submit the calculation to the selected compute server, similar to the behavior of the web interface itself.

Note: Material parameters that were obtained using the [AMS OLED Workflow](#) can be loaded directly into Python. The Bumblebee API allows you to construct materials using these parameters in order to perform device-level simulations. The construction of compositions using multiple materials will be treated in the [next tutorial](#).

Modifying Existing Resources

To make modifications to existing database entries, the `get` method is used to initialize a local resource.

```
parameter_set_id = 94
parameter_set = bb_client.get("param_set", parameter_set_id)
```

The `get` method retrieves database entries based on their ID. This ID is an integer number assigned to each resource, which can be viewed from the web interface.

A list of resources in the current database can be obtained using the `list` method.

```
compositions = bb_client.list("composition")
for composition in compositions:
    print(comp.id, comp.name)
```

The `list` method also allows filtering of entries based on:

- The project ID
- User access
- Resource names

```
# Obtain a list of compositions in project 2 containing CBP in the name
compositions = bb_client.list("composition", project_id=2, scope="project", search=
    ↪ "CBP")

# Obtain a list of stacks that are shared with the current user
stacks = bb_client.list("stack", scope="user")
```

Similar to the newly-created resources, modifications to these existing database entries are only applied locally. Use the `update` method to upload changes back into the database.

Deleting Resources

Warning: Deleted resources cannot be recovered. Be careful when using delete operations. Resource deletion is recommended to be performed in the web interface in order to use recovery features.

A resource can be deleted from the database by using the `delete` method.

```
material = bb_client.get("material", id=12)
material.delete()
```

Simulation objects are always moved to the trash folder first, in order to prevent accidental deletion of simulation output. Calling the `delete` method on a simulation that is already in the trash folder will permanently remove the resource. The web interface will then also request deletion of the simulation output data on the storage server.

```
simulation = bb_client.get("simulation", id=8)
simulation.delete() # Move to trash
simulation.delete() # Delete permanently
```

Accessing Simulation Output

Simulation output can be accessed through the Python API using the `results` method.

A convenient way to access simulation results is provided by the `results` object that is attached to each simulation resource. The `available` method can be used to list the accessible resources and their corresponding API names.

```
# List available results
simulation.results.available()
```

Result methods provided access to numerical data, which can be used for user-defined analysis and visualization routines. Various result methods allow extraction of data from specific parameter sweeps or trajectories. Always use the `available` method first to confirm that these results are included in the output before calling any access methods.

```
# Obtain the JV curve from the results
jv_result = sim.results.jv()

# Obtain the electron density profile at 5.0 V during the 3rd trajectory
ne_result = sim.results.electron_profile(sweepoint=5, box=3)
```

4.3.2 Job Submission

The Bumblebee Python API can be used to set up new simulations without going through the web interface.

In this tutorial, we will illustrate how to setup the Ir(ppy3)-based phosphorescent OLED simulations from a [previous GUI tutorial](#).

Create Materials

Materials created using the Python API use the default values defined by the web interface. It is not possible to select a template, therefore requiring all relevant parameters to be set manually.

```
# Create phosphorescent dye
dye = bb_client.new("material")
dye.name = "Irppy3"
dye.homo = -5.27
dye.lumo = -1.86

# Compute the exciton levels from the binding energies
dye.singlet_binding_energy = 0.75
dye.triplet_binding_energy = 1.0
dye.singlet = dye.homo - dye.lumo + dye.singlet_binding_energy
dye.triplet = dye.homo - dye.lumo + dye.triplet_binding_energy

# Configure phosphorescent emission
dye.st_ratio = 0
dye.tta_ratio = 0
dye.isc_rate = 1e10
dye.triplet_rad_decay = 6.1e5
dye.triplet_nonrad_decay = 1.9e4

# Create host matrix
host = bb_client.new("material")
host.name = "CBP"
host.homo = -6.08
host.lumo = -1.75
host.singlet_binding_energy = 1.0
host.triplet_binding_energy = 1.7
host.singlet = host.homo - host.lumo + host.singlet_binding_energy
host.triplet = host.homo - host.lumo + host.triplet_binding_energy
host.dexter_prefactor_singlet = 0.95
host.dexter_prefactor_triplet = 0.95
host.singlet_nonrad_decay = 1e5
host.triplet_nonrad_decay = 1e4

# Create electron transport layer
etl = bb_client.new("material")
etl.name = "TPBi"
etl.homo = -6.2
etl.lumo = -1.7
etl.singlet_binding_energy = 0.75
etl.triplet_binding_energy = 1.0
etl.singlet = etl.homo - etl.lumo + etl.singlet_binding_energy
etl.triplet = etl.homo - etl.lumo + etl.triplet_binding_energy
etl.singlet_nonrad_decay = 1e8
etl.triplet_nonrad_decay = 1e8

# Create hole transport layer
htl = bb_client.new("material")
htl.name = "TAPC"
htl.homo = -5.5
htl.lumo = -0.96
htl.singlet_binding_energy = 1.0
htl.triplet_binding_energy = 1.59
htl.singlet = htl.homo - htl.lumo + htl.singlet_binding_energy
```

(continues on next page)

(continued from previous page)

```
htl.triplet = htl.homo - htl.lumo + htl.triplet_binding_energy
htl.singlet_nonrad_decay = 1e8
htl.triplet_nonrad_decay = 1e8

# Upload the materials to the database
dye.create()
host.create()
etl.create()
htl.create()
```

Note: If the names of the materials have already been used by the web interface, an upload conflict will occur. Adjust the names of the materials to be descriptively unique in order to successfully upload the new materials.

Create Compositions

The `add_fraction` method is used when creating layer compositions. Note that the Python API did not automatically create pure compositions for the materials.

```
# Host-guest mixture
hostguest = bb_client.new("composition")
hostguest.name = "CBP-5Irppy3"
hostguest.add_fraction(0.95, host.id)
hostguest.add_fraction(0.05, dye.id)
hostguest.create()

# Pure HTL
pure_htl = bb_client.new("composition")
pure_htl.name = "Pure TAPC"
pure_htl.add_fraction(1, htl.id)
pure_htl.create()

# Pure ETL
pure_etl = bb_client.new("composition")
pure_etl.name = "Pure TPBi"
pure_etl.add_fraction(1, etl.id)
pure_etl.create()
```

Note: Make sure that the volume fractions add to 1 in order to successfully create a composition.

Note: Make sure to create the materials first, otherwise you will not be able to link to a valid ID when adding to the composition.

Create a Stack

Layers can be added to the stack in a similar way using the `add_layer` method.

```
# Add layers to the stack
stack = bb_client.new("stack")
stack.name = "TAPC-CBP-5Irppy3-TPBi"
stack.add_layer("HTL", 20, pure_htl.id)
stack.add_layer("EMI", 30, hostguest.id)
stack.add_layer("ETL", 20, pure_etl.id)
```

Förster interactions must be specified for each donor-acceptor pair. For this tutorial, we will limit ourselves to interactions between the host and guest. Förster processes are each assigned a unique ID, in accordance with their order in the web interface:

0. Singlet diffusion
1. Singlet-hole quenching
2. Singlet-electron quenching
3. Singlet-singlet annihilation
4. Singlet-triplet annihilation
5. Triplet diffusion
6. Triplet-hole quenching
7. Triplet-electron quenching
8. Triplet-singlet annihilation
9. Triplet-triplet annihilation

```
# Configure Förster interactions inside host-guest system
donor_layer = 1
acceptor_layer = 1
radius = 1.5
# Only consider triplet processes
for interaction in [5,6,7,9]:
    for donor_material in [host.id, dye.id]:
        for acceptor_material in [host.id, dye.id]:
            stack.add_foerster(interaction,
                               donor_layer,
                               donor_mat_index,
                               acceptor_layer,
                               acceptor_mat_index,
                               radius)

# Commit completed stack
stack.create()
```

Additional processes can be specified using the `add_degradation`, `add_absorption`, `add_dopant` and `add_cross_layer_dexter_prefactor` methods.

Create a Parameter Set

We select the newly-created stack when configuring the parameter set. Note that the electrode energy levels are not configured automatically by the Python API.

```
# Link the stack to a new parameter set
parameter_set = bb_client.new("param_set")
parameter_set.name = "SingleVoltage-CBP-5Irppy"
parameter_set.stack_id = stack.id

# Add injection barrier to electrode contacts
parameter_set.fermi_level_left = htl.homo + 0.2
parameter_set.fermi_level_right = etl.lumo - 0.2

# Set the voltage point
parameter_set.voltage = 5
parameter_set.create()
```

Starting the Simulation

Creating a simulation resource will automatically submit the job. The default server will be used to run the simulation, unless specified otherwise.

The simulation type is set using the `add_sweep` method.

```
# Create a single-voltage simulation using the new parameter set
simulation = bb_client.new("simulation")
simulation.name = "1V-CBP-5Irppy"
simulation.add_sweep(parameter_set.id, first_disorder=1, final_disorder=5)

# Submit the simulation
simulation.create()
```

Parameter sweeps can also be specified, analogous to the web interface:

```
# Create a voltage sweep
sweep = bb_client.new("simulation")
sweep.name = "Sweep-CBP-5Irppy"
sweep.add_sweep(parameter_set.id, first_disorder=1, final_disorder=2,
                sweep_variable='voltage', sweep_from=1, sweep_to=5, sweep_steps=5)
```

Simulation Output

The progress of the simulation can be monitored from within the web interface, or using the native monitoring tools of your compute server. Once the simulation has completed, the available output can be listed using the `available` method.

```
simulation.results.available()
```


4.3.3 Customized Parameter Screening

The Python API allows for custom parameter screenings to be conducted beyond the simple one-dimensional sweeps performed by the web interface.

For this tutorial, we will illustrate the analysis of host-guest compositions using the API submissions.

Create a Composition Template

We will be using the basic materials created during the [previous tutorial](#).

We start by creating a default composition using 1% of dye.

```
mixture = bb_client.new("composition")
mixture.name = "CBP-Irppy3"
mixture.add_fraction(0.99, host.id)
mixture.add_fraction(0.01, dye.id)
mixture.create()
```

Create a Stack

We use this composition to create a stack.

```
stack = bb_client.new("stack")
stack.name = "TAPC-CBP-Irppy3-TPBi"
stack.add_layer("HTL", 20, pure_htl.id)
stack.add_layer("EMI", 30, mixture.id)
stack.add_layer("ETL", 20, pure_etl.id)
stack.create()
```

Create a Parameter Set

We configure the parameter set for a 5V simulation.

```
parameter_set = bb_client.new("param_set")
parameter_set.name = "SingleVoltage-CBP-Irppy"
parameter_set.stack_id = stack.id
parameter_set.fermi_level_left = htl.homo + 0.2
parameter_set.fermi_level_right = etl.lumo - 0.2
parameter_set.voltage = 5
parameter_set.create()
```

Submitting Multiple Simulations

We will perform a screening of multiple compositions. By calling the `update` method, we are able to automatically link each new composition to the stack and parameter sets defined earlier.

```
# Define a list of dye fractions
fractions = [0.01, 0.03, 0.05, 0.1, 0.15, 0.25]

# Create and submit a composition for each dye fraction
for dye_fraction in fractions:
```

(continues on next page)

(continued from previous page)

```
# Modify the template composition using the new values
composition.name = "CBP-Irppy3-{}".format(100 * dye_fraction)
composition.fractions[0].fraction = 1.0 - dye_fraction
composition.fractions[1].fraction = dye_fraction
composition.update()

# Submit a new simulation
simulation = bb_client.new("simulation")
simulation.name = "Screening-" + composition.name
simulation.add_sweep(parameter_set.id, first_disorder=1, final_disorder=5)
simulation.create()
```

Simulation Output

The output for each host-guest composition is stored in a unique simulation. The results for each composition can be accessed by calling the `available` method.

```
for simulation in simulations:
    print(simulation.name)
    simulation.results.available()
```

Specific results can now be collated in order to analyze the output for custom parameter screenings.

Note: The `results` methods included in the API only support output extraction for the limited set of pre-defined screening parameters found in the web interface. By storing your custom screening sets in distinct simulation objects, the default post-processing remains accessible to aid in automated data extraction from the Bumblebee output.

Multi-dimensional Parameter Screenings

The Python API can be used to construct screenings of multiple parameters during a single run. Possible implementation strategies include:

- Nesting iterators for multiple variables (e.g. using multiple `for`-loops), or
- Creating a list of specific parameter combinations that you wish to test

This approach can be used to investigate variations in e.g. multi-component compositions, device dimensions, different layer morphologies, degradation scenarios, etc.

4.3.4 Advanced Morphology

This tutorial illustrates the process of specifying complex morphologies using the Python API.

Create Materials

For this tutorial, we will explore dual-dye gradients. We start by defining the necessary materials.

```
dye1 = bb_client.new("material")
dye1.name = "Irppy3"
dye1.homo = -5.27
dye1.lumo = -1.86
dye1.singlet_binding_energy = 0.75
dye1.triplet_binding_energy = 1.0
dye1.singlet = dye1.homo - dye1.lumo + dye1.singlet_binding_energy
dye1.triplet = dye1.homo - dye1.lumo + dye1.triplet_binding_energy
dye1.st_ratio = 0
dye1.tta_ratio = 0
dye1.isc_rate = 1e10
dye1.triplet_rad_decay = 6.1e5
dye1.triplet_nonrad_decay = 1.9e4

dye2 = bb_client.new("material")
dye2.name = "Irdmp3"
dye2.homo = -5.0
dye2.lumo = -1.7
dye2.singlet_binding_energy = 0.7
dye2.triplet_binding_energy = 1.0
dye2.singlet = dye2.homo - dye2.lumo + dye2.singlet_binding_energy
dye2.triplet = dye2.homo - dye2.lumo + dye2.triplet_binding_energy
dye2.st_ratio = 0
dye2.tta_ratio = 0
dye2.isc_rate = 1e10
dye2.triplet_rad_decay = 5.9e5
dye2.triplet_nonrad_decay = 2.2e4

host = bb_client.new("material")
host.name = "CBP"
host.homo = -6.08
host.lumo = -1.75
host.singlet_binding_energy = 1.0
host.triplet_binding_energy = 1.7
host.singlet = host.homo - host.lumo + host.singlet_binding_energy
host.triplet = host.homo - host.lumo + host.triplet_binding_energy
host.dexter_prefactor_singlet = 0.95
host.dexter_prefactor_triplet = 0.95
host.singlet_nonrad_decay = 1e5
host.triplet_nonrad_decay = 1e4
```

Create a Composition

The `add_morphology` method is used to specify advanced layer compositions.

```
# Create a new composition
composition = bb_client.new("composition")

# Always define a background material
composition.add_morphology("background", host.id)

# We use a linear profile for dye1
```

(continues on next page)

(continued from previous page)

```
composition.add_morphology("linear_gradient", dye1.id, p_start=0.1, p_end=0.2)

# We use a trapezoidal profile to add dye2 at the layer boundaries
composition.add_morphology("trapezoid_gradient", dye2.id,
                           trapezoid=[[0, 0.35], [0.25, 0.0], [0.75, 0.0], [1.0, 0.
→35]])

# Submit the composition
composition.create()
```

Create a Stack

We use this composition to create a stack. The procedure is the same as for basic composition types.

```
stack = bb_client.new("stack")
stack.name = "Dual-Dye"
stack.add_layer("EMI", 60, composition.id)
stack.create()
```

Create a Parameter Set

We configure the parameter set for a 5V simulation.

```
parameter_set = bb_client.new("param_set")
parameter_set.name = "SingleVoltage-Dual-Dye"
parameter_set.stack_id = stack.id
parameter_set.fermi_level_left = host.homo + 0.2
parameter_set.fermi_level_right = host.lumo - 0.2
parameter_set.voltage = 5
parameter_set.create()
```

Starting the Simulation

We configure and submit a single-voltage simulation with 5 separate trajectories.

```
simulation = bb_client.new("simulation")
simulation.name = "5V-Dual-Dye"
simulation.add_sweep(parameter_set.id, first_disorder=1, final_disorder=5)
simulation.create()
```

Simulation Output

Once the simulation has completed, the available output can be listed using the `available` method. When then morphologies have successfully been generated, cross-sectional profiles may be visualized using e.g. the `morphology_slice_box` method. Morphologies for the different trajectories can then be compared to investigate correlations between the OLED nanostructure and the device efficiency.

```
# Store the morphology for each trajectory at the first (and only) voltage
material_cross_sections = []
for trajectory in range(1, 5+1):
```

(continues on next page)

(continued from previous page)

```
cross_section = simulation.results.morphology_slice_box(sweepoint=5, ↵  
↵box=trajectory)  
material_cross_sections.append(cross_section)
```


REQUIRED CITATIONS

When you publish results in the scientific literature that were obtained with Bumblebee, you are required to include references to the program package with the appropriate release number.

5.1 General References

For simulations performed using Bumblebee: Bumblebee 2024.1, SCM, Amsterdam, The Netherlands, <http://www.scm.com>. Optionally, you may add the following list of authors and contributors: A. Yakovlev, T. Trnka, B. Klumpers

5.2 Key Publications

Consult the following publications for application examples of Bumblebee for OLED simulation:

- M. Mesta, M. Carvelli, R.J. de Vries, H. van Eersel, J.J.M. van der Holst, M. Schober, M. Furno, B. Lüssem, K. Leo, P. Loeb, R. Coehoorn and P.A. Bobbert, *Molecular-scale simulation of electroluminescence in a multi-layer white organic light-emitting diode*, *Nature Materials* **12**, 652 (2013) (<https://doi.org/10.1038/nmat3622>)
- C. Hauenstein, S. Gottardi, E. Torun, R. Coehoorn and H. van Eersel, *Identification of OLED degradation scenarios by kinetic Monte Carlo simulations of lifetime experiments*, *Frontiers in Chemistry* **9**, 823210 (2021) (<https://doi.org/10.3389/fchem.2021.823210>)
- C. Hauenstein, X. de Vries, C.H.L. Weijtens, P. Imbrasas, P.-A. Will, S. Lenk, K. Ortstein, S. Reineke, P.A. Bobbert, R. Coehoorn and H. van Eersel, *Suppressing exciton deconfinement and dissociation for efficient thermally activated delayed fluorescence OLEDs*, *Journal of Applied Physics* **130**, 155501 (2021) (<https://doi.org/10.1063/5.0062926>)
- H. van Eersel, P.A. Bobbert, R.A.J. Janssen and R. Coehoorn, *Effect of Förster-mediated triplet-polaron quenching and triplet-triplet annihilation on the efficiency roll-off of organic light-emitting diodes*, *Journal of Applied Physics* **119**, 163102 (2016) (<https://doi.org/10.1063/1.4947457>)

Listed below are some of the more-common questions encountered by Bumblebee users.

Note: Issues related to installation of the Bumblebee software are discussed in the [Installation](#) (page 25) guide.

6.1 How do I determine the required number of simulation steps?

The required number of samples is determined by the desired accuracy of the results.

kMC estimates the properties of the OLED device through a stochastic sampling process. Increasing the number of samples results in greater accuracy.

This accuracy differs for each simulation output. Rare simulation events will be sampled less frequently, resulting in poorer statistics.

One strategy to determine the required number of simulation steps is to perform adaptive refinement. Start by running the simulation for a set number of steps, then check the output accuracy. If the results are unsatisfactory, you can add additional trajectories to the simulation in order to increase the number of samples.

6.2 How do I determine the maximum wall time for my workload manager?

The total simulation time varies based on the process complexity, device size and the simulation length. In general, kMC simulations can have a duration from minutes to days.

In order to work well with typical schedulers, it is recommended to use parallel trajectories to distribute the workload for more expensive jobs.

Note that even if a simulation exceeds the allotted wall time, the web interface can still interpret the data obtained thus-far from the intermediate simulation output. As such, there are 2 scheduling strategies:

- You can use a generous wall time along with a short simulation time to split up the sampling process between a large number of instances. When evaluating the simulation output, it is possible to commission additional trajectories to increase the accuracy of the statistical estimates, if deemed necessary
- You can specify a long simulation time (along with a reasonable output interval). The duration of the simulation will then be determined by the wall time. Most simulations will not reach their end, but instead be terminated by the scheduler. Termination will cause a small amount of output to be lost at the end of a simulation. This strategy is only recommended on systems with slow convergence behavior

6.3 Can I recover the results from a killed job?

Bumblebee updates the output files as the simulation progresses. The update frequency can be changed in the parameter set.

The web interface and analysis modules are able to interpret this intermediate data, even when a job was killed by the server.

It is possible to continue a terminated run from the last recorded state. Simply re-run the submission script on the server to resume the simulation.

6.4 How do I perform two-dimensional parameter screenings?

The parameter screening option in the web interface is designed for single-variable screenings.

Even though multiple screenings can be added to a single simulation, the visualization routines are not able to identify this second parameter, resulting in overlapping data in the graphs.

When performing multi-variate device optimization, it is recommended to use the Python API to set custom sample points. The simulation output can then be collected for processing using e.g. the [PLAMS](#) visualization tool set.

6.5 How do I include charge-generation layers in the stack?

Charge-generation layers (CGL) are available starting from the 2025 release of Bumblebee. Consult the [CGL tutorial](#) for more details on this material template.

In older versions of Bumblebee, it is possible to approximate the CGL as an idealized transport layer:

- Use the *Transport* template for the CGL material
- In the stack editor, polaron generation can be enabled by including photoabsorption processes inside the CGL layer. The polaron density can be controlled by the fluence
- For bilayer CGL, localization of the charge generation at the interface can be approximated by using a thin film of 1-2 nm for the absorption region
- Use the *Photovoltaic* template for the parameter set to automatically enable the photoluminescence module during the simulation

6.6 How do I model a tandem stack?

Bumblebee allows any number of layers to be included in the OLED stack. Tandem devices can therefore be treated straightforwardly by including multiple emissive layers.

Assuming proper operation of the polaron injection layers, interactions between light-emission units (LEU) can be minimal. In this regime, it is also possible to run voltage sweep simulations for isolated LEU. Current-voltage profiles can then be aligned to approximate the tandem stack performance. This strategy is primarily recommended for the screening of LEU materials.

When modeling the full tandem stack:

- Explicit inclusion of the interface between LEU can be achieved by using a CGL layer. (See the application notes in the previous segment.)

- The interface can also be modeled implicitly. Instead of adding a layer to the stack, the inter-layer transport parameters can be modified to account for the additional resistance. See the [Advanced Bumblebee tutorials](#) for more details.