SPAMS

v1.2

Generated by Doxygen 1.9.1

Chapter 1

Main Page

A single particle model (SPM) by Group B for the 2023 PX915 group project.

1.1 Introduction

SPAMS models the charging and discharging of a lithium ion battery using a Crank-Nicolson semi-implicit finite difference scheme to obtain the concentration of lithium in a sphere, c(iapp, r), at each time step, and includes the following features:

- Apply a constant, stepwise or custom current
- Options for parallelism
- Extend the model to a full battery
- Uncertainty quantification

1.2 Contributors

Fraser Birks, Laura Cairns, Sebastian Dooley, Arielle Fitkin, Jake Eller, and Yu Lei

HetSys CDT, University of Warwick

Chapter 2

Maths and Theory

A half single particle model by Group B for the 2023 PX915 group project.

2.1 1 Mathematics and Algorithms

A Crank-Nicolson semi-implicit finite difference scheme is used to approximate the partial differential equation (PDE) solution to obtain the concentration of lithium in the sphere, c(i_{app}, r), at each time step. This is accurate to second order both spatially and temporally, i.e. (Δ r^2) & (Δ r^2)

A key benefit of the Crank-Nicolson scheme is that it is unconditionally stable for the spherically symmetric diffusion equation (the PDE of interest), thus not restricting the user's choice of step size spatially or temporally. Although, due to being second order, the accuracy may be impacted by a larger step size.

The implicit nature of the Crank-Nicolson scheme makes it ideal for numerically approximating solutions to both stiff and highly nonlinear diffusion problems, providing the user a strong base from which to update this solution framework.

Additionally, the Crank-Nicolson scheme is consistent meaning that 'the error of the numerical method converges to zero as the grid spacing, or time step, reduces to zero from above, under certain regularity conditions on the solution.'

The consistency and stability of the Crank-Nicolson scheme means it satisfies the Lax equivalence theorem, which states that a numerical method is convergent if and only if it is both consistent and stable.

Specific Equation:

$$\frac{c_i^{j+1}-c_i^j}{\Delta t} = \frac{D}{2r_i^2}\left[\left(r_i^2\frac{c_{i+1}^{j+1}-2c_i^{j+1}+c_{i-1}^{j+1}}{\Delta r^2} + r_i^2\frac{c_{i+1}^{j}-2c_i^{j}+c_{i-1}^{j}}{\Delta r^2}\right) + r_i\left(\frac{c_{i+1}^{j+1}-c_{i-1}^{j+1}}{2\Delta r} + \frac{c_{i+1}^{j}-c_{i-1}^{j}}{2\Delta r}\right)\right]$$

Generally:

$$\frac{u_i^{j+1}-u_i^j}{\Delta t} = \frac{1}{2}\left[F_i^{j+1}(u,r,t,\frac{\partial u}{\partial r},\frac{\partial^2 u}{\partial r^2}) + F_i^{j}(u,r,t,\frac{\partial u}{\partial r},\frac{\partial^2 u}{\partial r^2})\right].$$

This is an average of standard forward and backward Euler methods.

Boundary conditions are treated through the use of ghost nodes, which assign values for the function of interest to regions just beyond the domain of the problem in an attempt to approximate the first derivatives present in the Neumann boundary conditions specified.

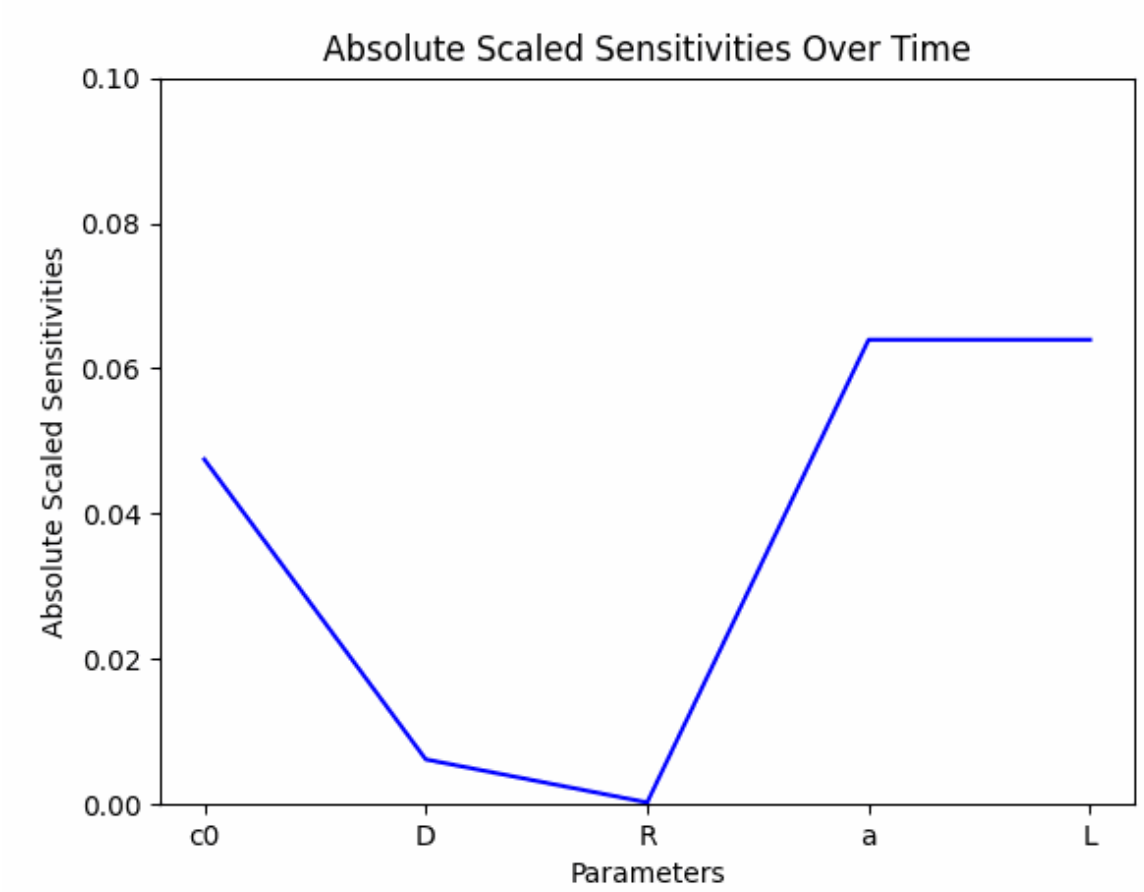
Chapter 3

Uncertainty Quantification

Below are the results of some uncertainty quantification that has been carried out for SPAMS.

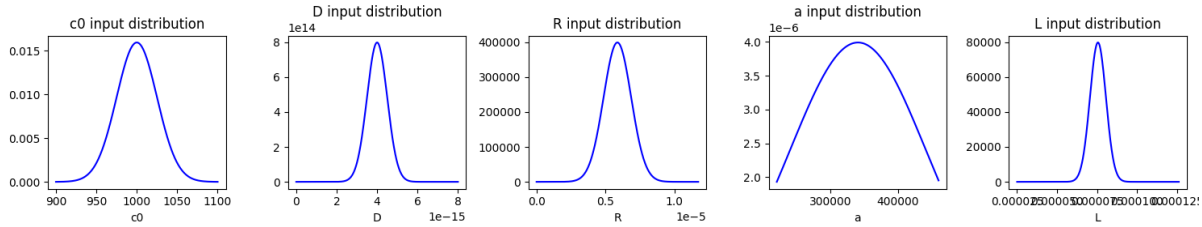
3.1 Sensitivity Analysis

An animation showing first order sensitivity analysis.

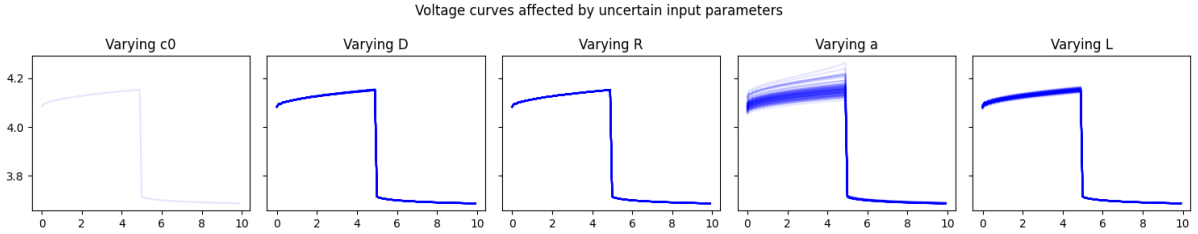


3.2 Uncertainty Propagation

Input Parameter Distributions



Resulting distribution of v(t)



Chapter 4

Modules Index

4.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 checkpointing Module Reference

Module to read and write checkpoint files.

6.1.1 Detailed Description

Module to read and write checkpoint files.

Module contains functions that write checkpoint files and read them in as user input.

6.2 convergence_test Namespace Reference

Test convergence of final voltage and lithium mass loss of unit test with dt and number of nodes.

Functions

- def run_dt_convergence ()
- def run_node_convergence ()

6.2.1 Detailed Description

Test convergence of final voltage and lithium mass loss of unit test with dt and number of nodes.

6.2.2 Function Documentation

6.2.2.1 run_dt_convergence()

```
def convergence_test.run_dt_convergence ( )
```

Definition at line 8 of file convergence_test.py.

6.2.2.2 run_node_convergence()

```
def convergence_test.run_node_convergence ( )
```

Definition at line 36 of file convergence_test.py.

6.3 mixed_parallel_strat_test Namespace Reference

Test the runtime of the unit test for different provided processes in the mixed parallel strategy.

Variables

- list num_cores = [1,2,4,8]
- list num_nodes = [2000]
- times = np.zeros([len(num_cores),len(num_nodes)])
- start = time.time()
- nodenum
- n
- dt
- end = time.time()
- label

6.3.1 Detailed Description

Test the runtime of the unit test for different provided processes in the mixed parallel strategy.

6.3.2 Variable Documentation

6.3.2.1 dt

```
dt
```

Definition at line 17 of file mixed_parallel_strat_test.py.

6.3.2.2 end

```
end = time.time()
```

Definition at line 18 of file mixed_parallel_strat_test.py.

6.3.2.3 label

```
label
```

Definition at line 24 of file mixed_parallel_strat_test.py.

6.3.2.4 n

```
n
```

Definition at line 17 of file mixed_parallel_strat_test.py.

6.3.2.5 nodenum

```
nodenum
```

Definition at line 17 of file mixed_parallel_strat_test.py.

6.3.2.6 num_cores

```
num_cores = [1,2,4,8]
```

Definition at line 9 of file mixed_parallel_strat_test.py.

6.3.2.7 num_nodes

```
list num_nodes = [2000]
```

Definition at line 11 of file mixed_parallel_strat_test.py.

6.3.2.8 start

```
start = time.time()
```

Definition at line 15 of file mixed_parallel_strat_test.py.

6.3.2.9 times

```
times = np.zeros([len(num_cores),len(num_nodes)])
```

Definition at line 12 of file mixed_parallel_strat_test.py.

6.4 MKL_threads_test Namespace Reference

Test the runtime of the unit test for different numbers of threads.

Variables

- list num_threads = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,20]
- list num_nodes = [100,500,1000,2000]
- times = np.zeros([19,4])
- start = time.time()
- nodenum
- n
- dt
- end = time.time()
- label

6.4.1 Detailed Description

Test the runtime of the unit test for different numbers of threads.

6.4.2 Variable Documentation

6.4.2.1 dt

```
dt
```

Definition at line 20 of file MKL_threads_test.py.

6.4.2.2 end

```
end = time.time()
```

Definition at line 21 of file MKL_threads_test.py.

6.4.2.3 label

```
label
```

Definition at line 27 of file MKL_threads_test.py.

6.4.2.4 n

```
n
```

Definition at line 20 of file MKL_threads_test.py.

6.4.2.5 nodenum

```
nodenum
```

Definition at line 20 of file MKL_threads_test.py.

6.4.2.6 num_nodes

```
list num_nodes = [100,500,1000,2000]
```

Definition at line 11 of file MKL_threads_test.py.

6.4.2.7 num_threads

```
list num_threads = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,20]
```

Definition at line 10 of file MKL_threads_test.py.

6.4.2.8 start

```
start = time.time()
```

Definition at line 19 of file MKL_threads_test.py.

6.4.2.9 times

```
times = np.zeros([19,4])
```

Definition at line 12 of file MKL_threads_test.py.

6.5 nc_output Module Reference

Writes results to a netCDF file.

Functions/Subroutines

- subroutine check (status)
- subroutine output_cstorage (cstorage, n, tsteps, R, time_axis, electrode_charge, filename)

6.5.1 Detailed Description

Writes results to a netCDF file.

6.5.2 Function/Subroutine Documentation

6.5.2.1 check()

```
subroutine nc_output::check (
             integer, intent(in) status )
```

Definition at line 9 of file nc_output.f90.

6.5.2.2 output_cstorage()

```
subroutine nc_output::output_cstorage (
             real(real64), dimension(n, tsteps), intent(in) cstorage,
             integer, intent(in) n,
             integer, intent(in) tsteps,
             real(real64), intent(in) R,
             real(real64), dimension(tsteps), intent(in) time_axis,
             character(len=*), intent(in) electrode_charge,
             character(len=*), intent(in) filename )
```

Definition at line 20 of file nc_output.f90.

6.6 plotter Namespace Reference

Functions for visualisation of solver outputs.

Functions

- def read_output_file (filename, step_num=None)

  Reads in data from user input and solver output using NetCDF.
- def read_input_current (filename, step_num=None)

  Reads in the applied current.
- def animated_conc_plot (intervaltime, dr, tsteps, nodenum, cstore, time_axis, SaveFinalState=False, SparsifyAnimation=False)

  Saves animation of lithium concentration over time.
- def gen_half_cell_voltage (edge_conc_vals, i_app_data, electrode, tsteps, pos_params=None, neg_↩ params=None)

  Generate the voltage against time profile for a half cell and return voltage.
- def voltage_current_plot (electrode, cstore, time_axis, i_app_data, tsteps, pos_params=None, neg_↩ params=None)

  Calculates voltages and plots as a function of time.
- def plot_halfcell_GITT_result (filename, start_times, electrode, pos_params=None, neg_params=None, Animation=False, SaveFinalState=False, SparsifyAnimation=True, animation_interval_time=10)

  Generates the voltage current plot and optionally an animation for a half-cell GITT test.
- def gen_plots (filename, pos_params=None, neg_params=None, animation_interval_time=10, SaveFinal↩ State=True, SparsifyAnimation=False)

  A function which can be called to generate both the voltage-time, current-time plots and the animation.
- def full_battery_GITT_plots (filename_positive, filename_negative, start_times, pos_params=None, neg_↩ params=None, SparsifyAnimation=True, animation_interval_time=10)
- def get_avg_li_conc (c_vals, r_vals)

  Gets the average concentration of lithium at a given timestep using trapezium rule inside the sphere.

6.6.1 Detailed Description

Functions for visualisation of solver outputs.

(Plots are saved locally.)

Contains functions that produce:

- An animation showing the concentration of lithium as a function of time and space. (There is an additional feature for saving the concentration profile at the end of the simulation as a png.)
- Plots of both voltage and applied current over time.

6.6.2 Function Documentation

6.6.2.1 animated_conc_plot()

```
def plotter.animated_conc_plot (
              intervaltime,
              dr,
              tsteps,
              nodenum,
              cstore,
              time_axis,
              SaveFinalState = False,
              SparsifyAnimation = False )
```

Saves animation of lithium concentration over time.

Saves the animation 'concentration_animation.gif', that displays the evolution of lithium concentration over time. If passed the argument 'SaveFinalState = True', it saves an image 'final_state.png' of lithium concentration across the sphere at the final timestep. If passed the argument 'SparsifyAnimation = True', it saves 1 in every 10 timesteps rather than every single one.

Definition at line 89 of file plotter.py.

6.6.2.2 full_battery_GITT_plots()

```
def plotter.full_battery_GITT_plots (
              filename_positive,
              filename_negative,
              start_times,
              pos_params = None,
              neg_params = None,
              SparsifyAnimation = True,
              animation_interval_time = 10 )
```

```
@brief Build an animation of the concentrations, applied current with time and applied voltage with time
in a full battery GITT test.
@details Builds the full dataset from the different GITT test files. Generates the voltage for each timestep,
plots the full voltage-time, current-time and concentration of positive and negative electrode animations.
Also prints lithium mass loss in first step, to give an indication on how good the model is. Acceptable values
are =<10^-6 kg.
@param[in] filename_positive     The input positive filename for the simulation data, excluding any file exten
@param[in] filename_negative     The input negative filename for the simulation data, excluding any file exten
@param[in] start_times  The list of start times for each current step in a GITT test. Array of floats
@param[in] pos_params   [K, a, cmax, L] for the positive electrode
@param[in] neg_params   [K, a, cmax, L] for the negative electrode
@param[in] SparsifyAnimation True to construct animation using 1 of every 10 timesteps
@param[in] animation_interval_time    time between frames in animation(ms)
```

Definition at line 419 of file plotter.py.

6.6.2.3 gen_half_cell_voltage()

```
def plotter.gen_half_cell_voltage (
              edge_conc_vals,
              i_app_data,
              electrode,
              tsteps,
              pos_params = None,
              neg_params = None )
```

Generate the voltage against time profile for a half cell and return voltage.

. Return the voltage for a given applied current and edge concentration values for a given electrode. If the concentration at the edge of the sphere at anytime goes below 0, computing the voltage will not work. This is because negative concentration is unphysical. In this case, a ValueError will be returned.

```
@param[in] edge_conc_vals   Lithium concentration at the sphere edge  for all time steps
@param[in] i_app_data       Vector of applied current density at each timestep
@param[in] electrode        Electrode type of the half cell
@param[in] tsteps           The number of timesteps in the simulation
@param[in] pos_params       [K, a, cmax, L] for the positive electrode, None if the positive is not needed
@param[in] neg_params       [K, a, cmax, L] for the negative electrode, None if the negative is not needed
```

Definition at line 166 of file plotter.py.

6.6.2.4 gen_plots()

```
def plotter.gen_plots (
              filename,
              pos_params = None,
              neg_params = None,
              animation_interval_time = 10,
              SaveFinalState = True,
              SparsifyAnimation = False )
```

A function which can be called to generate both the voltage-time, current-time plots and the animation.

Calls the relevant functions to generate an animation plot and a voltage-time, current-time set of plots.

Parameters

in	*filename*	The input filename for the simulation data, excluding any file extension. String.
in	*pos_params*	[K, a, cmax, L] for the positive electrode, None if the positive is not needed
in	*neg_params*	[K, a, cmax, L] for the negative electrode, None if the negative is not needed
in	*animation_interval_time*	time between frames in animation(ms)
in	*SaveFinalState*	True to save the final state of the animation as a seperate PNG
in	*SparsifyAnimation*	True to construct animation using 1 of every 10 timesteps

Definition at line 402 of file plotter.py.

6.6.2.5 get_avg_li_conc()

```
def plotter.get_avg_li_conc (
              c_vals,
              r_vals )
```

Gets the average concentration of lithium at a given timestep using trapezium rule inside the sphere.

!

Uses the fact that the average concentration of lithium in a sphere at any timestep can be found from $\frac{3}{R^{3}}\int_{0}^{R} c(r)r^{2}dr. Note, that this seems weird. It is weird. The reason this is necessary is because the concentration values in the sphere are defined per active electrode volume, i.e e_act∗L∗A, not per sphere volume. Therefore, to calculate the lithium mass loss in the whole system it is necessary to first find the concentration as if each sphere was uniformly filled (that is what this function does), and then after that multiply this average concentration by the active electrode volume. Take it up with whoever came up with the single particle model, not me. !

Parameters

in	*c_vals*	Array of concentration values covering the whole sphere radius, floats. !
in	*r_vals*	Array of radius values that correspond the concentration values given in c_vals, floats.

Definition at line 577 of file plotter.py.

6.6.2.6 plot_halfcell_GITT_result()

```
def plotter.plot_halfcell_GITT_result (
              filename,
              start_times,
              electrode,
              pos_params = None,
              neg_params = None,
              Animation = False,
              SaveFinalState = False,
              SparsifyAnimation = True,
              animation_interval_time = 10 )
```

Generates the voltage current plot and optionally an animation for a half-cell GITT test.

Concatentates the output files generated by a GITT test and then produces a voltage-time and current-time plot and an animation seperately.

Parameters

in	*filename*	The input filename for the simulation data, excluding any file extension. String.
in	*start_times*	The list of start times for each current step in a GITT test. Array of floats
in	*electrode*	The electrode type, either positive or negative, string.
in	*pos_params*	[K, a, cmax, L] for the positive electrode, None if the positive is not needed
in	*neg_params*	[K, a, cmax, L] for the negative electrode, None if the negative is not needed
in	*Animation*	True to generate an animation of concentration in the sphere, False otherwise
in	*SaveFinalState*	True to save the final state of the animation as a seperate PNG
in	*SparsifyAnimation*	True to construct animation using 1 of every 10 timesteps
in	*interval_time*	time between frames in animation(ms)

Definition at line 355 of file plotter.py.

6.6.2.7 read_input_current()

```
def plotter.read_input_current (
              filename,
              step_num = None )
```

Reads in the applied current.

Parameters

in	*filename*	The name of the input file, this must be a string and have max 50 characters. No file extension is required.
in	*step_num*	Specifies the step number to read for a GITT simulation where there are multiple current steps (defualt = none for serial)

Definition at line 55 of file plotter.py.

6.6.2.8 read_output_file()

```
def plotter.read_output_file (
              filename,
              step_num = None )
```

Reads in data from user input and solver output using NetCDF.

Parameters

in	*filename*	The name of the input file, this must be a string and have max 50 characters. No file extension is required.
in	*step_num*	Specifies the step number to read for a parallel simulation (defualt = none for serial)

Definition at line 14 of file plotter.py.

6.6.2.9 voltage_current_plot()

```
def plotter.voltage_current_plot (
              electrode,
              cstore,
              time_axis,
              i_app_data,
              tsteps,
```

```
pos_params = None,
neg_params = None )
```

Calculates voltages and plots as a function of time.

Calculates the voltage at all time points and saves plots of both Li concentration at the sphere edge and voltage as a function of time. This function has different settings determined by the value of electrode.

Parameters

in	*electrode*	Charge on electrode
in	*cstore*	The full matrix of concentration values (all nodes at all timesteps)
in	*time_axis*	Axis containing all timestep values
in	*i_app_data*	Full set of applied current density data
in	*tsteps*	The number of timesteps
in	*pos_params*	[K, a, cmax, L] for the positive electrode, None if the positive is not needed
in	*neg_params*	[K, a, cmax, L] for the negative electrode, None if the negative is not needed

Definition at line 322 of file plotter.py.

6.7 read_inputs Module Reference

Module to read user inputs.

6.7.1 Detailed Description

Module to read user inputs.

Module contains functions that read user inputs from file and command line.

6.8 sensitivity_analysis Namespace Reference

First order sensitivity analysis.

Functions

- def sensitivity_over_time ()

  Wrapper for the sensitivity analysis code.

Variables

- ani = FuncAnimation(fig, update, frames=len(scaled_sensitivities_matrix), blit=True)
- filename
- writer
- fps

6.8.1 Detailed Description

First order sensitivity analysis.

A python file that is used to produce first order sensitivity analysis Running the file outputs a sensitivity gif.

Credit to Dr. James Kermode for some functions recycled from PX914.

6.8.2 Function Documentation

6.8.2.1 sensitivity_over_time()

```
def sensitivity_analysis.sensitivity_over_time ( )
```

Wrapper for the sensitivity analysis code.

Definition at line 28 of file sensitivity_analysis.py.

6.8.3 Variable Documentation

6.8.3.1 ani

```
ani = FuncAnimation(fig, update, frames=len(scaled_sensitivities_matrix), blit=True)
```

Definition at line 324 of file sensitivity_analysis.py.

6.8.3.2 filename

```
filename
```

Definition at line 325 of file sensitivity_analysis.py.

6.8.3.3 fps

```
fps
```

Definition at line 325 of file sensitivity_analysis.py.

6.8.3.4 writer

```
writer
```

Definition at line 325 of file sensitivity_analysis.py.

6.9 Uncertainty_Propagation Namespace Reference

Performs uncertainty propagation.

Functions

- def in_out_easy_peasy (parameter_np_array)

  Simplification of the solver that allows for input parameters to be supplied and only the QOI to be returned.

Variables

- iapp = np.concatenate((np.array([20.0 for i in range(50)]),np.array([0.0 for i in range(50)])))

  Read in applied current density from csv file iapp_filename = 'WLTP_m10.csv' iapp, iapp_label, tsteps = UI.iapp_↩ read_csv(iapp_filename)
- parameters = parameter_np_array

  END MEAN SET VALUES #########.
- int nprocs = 40

  Call the function to perform the full battery parallel solve.
- output_filename_positive
- output_filename_negative
- cstore_pos

  Obtaining QOI (Voltage) #.
- tsteps
- nodenum
- R_pos
- time_axis
- dr_pos
- electrode_pos
- cstore_neg
- R_neg
- dr_neg
- electrode_neg
- i_app_data_pos = plotter.read_input_current(output_filename_positive)
- i_app_data_neg = plotter.read_input_current(output_filename_negative)
- edge_conc_vals_pos = cstore_pos[:,-1]
- edge_conc_vals_neg = cstore_neg[:,-1]
- list posparams = [K_pos,parameters[3],cmax_pos_sim,parameters[4]]
- list negparams = [K_neg,a_neg,cmax_neg_sim,L_neg]
- pos_voltage = plotter.gen_half_cell_voltage(edge_conc_vals_pos,i_app_data_pos,electrode_pos,tsteps,pos↩ _params=posparams)
- neg_voltage = plotter.gen_half_cell_voltage(edge_conc_vals_neg,i_app_data_neg,electrode_neg,tsteps,neg↩ _params=negparams)
- full_voltage = pos_voltage-neg_voltage
- c0_dist = st.norm(1000.0, 25.0)

Define the input parameter distributions #.

- D_dist = st.norm(4.0e-15, 0.5e-15)
- R_dist = st.norm(5.86e-6, 1.0e-6)
- a_dist = st.norm(3.0∗0.665/5.86e-6, 1.0e5)
- L_dist = st.norm(75.6e-6, 5.0e-6)
- int num_samples = 40
- c0_samples = c0_dist.rvs(num_samples)
- D_samples = D_dist.rvs(num_samples)
- R_samples = R_dist.rvs(num_samples)
- a_samples = a_dist.rvs(num_samples)
- L_samples = L_dist.rvs(num_samples)
- fixed_c0 = c0_dist.mean()
- fixed_D = D_dist.mean()
- fixed_R = R_dist.mean()
- fixed_a = a_dist.mean()
- fixed_L = L_dist.mean()
- int counter = 0
- dt
- time = np.linspace(0.0,dt∗(tsteps-1.),tsteps)

  _, _, tsteps = UI.iapp_read_csv(iapp_filename)
- voltage_curves = np.zeros(tsteps)
- fig

  UQ Plot #.
- axs
- figsize
- sharex
- True
- sharey
- params = np.array([c0_samples[i], fixed_D, fixed_R, fixed_a, fixed_L])
- alpha
- figg

  Input Distributions #.
- axss
- False
- int resol = 1000
- x = np.linspace(900, 1100, resol)

6.9.1 Detailed Description

Performs uncertainty propagation.

Applies normal distributions to parameter inputs and produces a 5 subplot figure that shows the posterior distribution of the voltage curve. Each subplot shows the effect of altering one parameter and keeping the others constant at mean values of input distributions.

6.9.2 Function Documentation

6.9.2.1 in_out_easy_peasy()

```
def Uncertainty_Propagation.in_out_easy_peasy (
              parameter_np_array )
```

Simplification of the solver that allows for input parameters to be supplied and only the QOI to be returned.

Definition at line 32 of file Uncertainty_Propagation.py.

6.9.3 Variable Documentation

6.9.3.1 a_dist

```
a_dist = st.norm(3.0*0.665/5.86e-6, 1.0e5)
```

Definition at line 110 of file Uncertainty_Propagation.py.

6.9.3.2 a_samples

```
a_samples = a_dist.rvs(num_samples)
```

Definition at line 122 of file Uncertainty_Propagation.py.

6.9.3.3 alpha

```
alpha
```

Definition at line 187 of file Uncertainty_Propagation.py.

6.9.3.4 axs

```
axs
```

Definition at line 179 of file Uncertainty_Propagation.py.

6.9.3.5 axss

```
axss
```

Definition at line 233 of file Uncertainty_Propagation.py.

6.9.3.6 c0_dist

```
c0_dist = st.norm(1000.0, 25.0)
```

Define the input parameter distributions #.

Definition at line 107 of file Uncertainty_Propagation.py.

6.9.3.7 c0_samples

```
c0_samples = c0_dist.rvs(num_samples)
```

Definition at line 119 of file Uncertainty_Propagation.py.

6.9.3.8 counter

```
int counter = 0
```

Definition at line 133 of file Uncertainty_Propagation.py.

6.9.3.9 cstore_neg

```
cstore_neg
```

Definition at line 87 of file Uncertainty_Propagation.py.

6.9.3.10 cstore_pos

```
cstore_pos
```

Obtaining QOI (Voltage) #.

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.11 D_dist

```
D_dist = st.norm(4.0e-15, 0.5e-15)
```

Definition at line 108 of file Uncertainty_Propagation.py.

6.9.3.12 D_samples

```
D_samples = D_dist.rvs(num_samples)
```

Definition at line 120 of file Uncertainty_Propagation.py.

6.9.3.13 dr_neg

```
dr_neg
```

Definition at line 87 of file Uncertainty_Propagation.py.

6.9.3.14 dr_pos

```
dr_pos
```

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.15 dt

```
dt
```

Definition at line 167 of file Uncertainty_Propagation.py.

6.9.3.16 edge_conc_vals_neg

```
edge_conc_vals_neg = cstore_neg[:,-1]
```

Definition at line 91 of file Uncertainty_Propagation.py.

6.9.3.17 edge_conc_vals_pos

```
edge_conc_vals_pos = cstore_pos[:,-1]
```

Definition at line 90 of file Uncertainty_Propagation.py.

6.9.3.18 electrode_neg

```
electrode_neg
```

Definition at line 87 of file Uncertainty_Propagation.py.

6.9.3.19 electrode_pos

```
electrode_pos
```

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.20 False

```
False
```

Definition at line 233 of file Uncertainty_Propagation.py.

6.9.3.21 fig

```
fig
```

UQ Plot #.

Definition at line 179 of file Uncertainty_Propagation.py.

6.9.3.22 figg

```
figg
```

Input Distributions #.

Definition at line 233 of file Uncertainty_Propagation.py.

6.9.3.23 figsize

```
figsize
```

Definition at line 179 of file Uncertainty_Propagation.py.

6.9.3.24 fixed_a

```
fixed_a = a_dist.mean()
```

Definition at line 129 of file Uncertainty_Propagation.py.

6.9.3.25 fixed_c0

```
fixed_c0 = c0_dist.mean()
```

Definition at line 126 of file Uncertainty_Propagation.py.

6.9.3.26 fixed_D

```
fixed_D = D_dist.mean()
```

Definition at line 127 of file Uncertainty_Propagation.py.

6.9.3.27 fixed_L

```
fixed_L = L_dist.mean()
```

Definition at line 130 of file Uncertainty_Propagation.py.

6.9.3.28 fixed_R

```
fixed_R = R_dist.mean()
```

Definition at line 128 of file Uncertainty_Propagation.py.

6.9.3.29 full_voltage

```
full_voltage = pos_voltage-neg_voltage
```

Definition at line 99 of file Uncertainty_Propagation.py.

6.9.3.30 i_app_data_neg

```
i_app_data_neg = plotter.read_input_current(output_filename_negative)
```

Definition at line 89 of file Uncertainty_Propagation.py.

6.9.3.31 i_app_data_pos

```
i_app_data_pos = plotter.read_input_current(output_filename_positive)
```

Definition at line 88 of file Uncertainty_Propagation.py.

6.9.3.32 iapp

```
iapp = np.concatenate((np.array([20.0 for i in range(50)]),np.array([0.0 for i in range(50)])))
```

Read in applied current density from csv file iapp_filename = 'WLTP_m10.csv' iapp, iapp_label, tsteps = UI.iapp_↩ read_csv(iapp_filename)

Definition at line 54 of file Uncertainty_Propagation.py.

6.9.3.33 L_dist

```
L_dist = st.norm(75.6e-6, 5.0e-6)
```

Definition at line 111 of file Uncertainty_Propagation.py.

6.9.3.34 L_samples

```
L_samples = L_dist.rvs(num_samples)
```

Definition at line 123 of file Uncertainty_Propagation.py.

6.9.3.35 neg_voltage

```
neg_voltage = plotter.gen_half_cell_voltage(edge_conc_vals_neg,i_app_data_neg,electrode_neg,tsteps,neg↩
_params=negparams)
```

Definition at line 98 of file Uncertainty_Propagation.py.

6.9.3.36 negparams

```
list negparams = [K_neg,a_neg,cmax_neg_sim,L_neg]
```

Definition at line 95 of file Uncertainty_Propagation.py.

6.9.3.37 nodenum

```
nodenum
```

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.38 nprocs

```
nprocs = 40
```

Call the function to perform the full battery parallel solve.

Definition at line 77 of file Uncertainty_Propagation.py.

6.9.3.39 num_samples

```
int num_samples = 40
```

Definition at line 116 of file Uncertainty_Propagation.py.

6.9.3.40 output_filename_negative

```
output_filename_negative
```

Definition at line 79 of file Uncertainty_Propagation.py.

6.9.3.41 output_filename_positive

```
output_filename_positive
```

Definition at line 79 of file Uncertainty_Propagation.py.

6.9.3.42 parameters

```
parameters = parameter_np_array
```

END MEAN SET VALUES #########.

Definition at line 59 of file Uncertainty_Propagation.py.

6.9.3.43 params

```
params = np.array([c0_samples[i], fixed_D, fixed_R, fixed_a, fixed_L])
```

Definition at line 185 of file Uncertainty_Propagation.py.

6.9.3.44 pos_voltage

```
pos_voltage = plotter.gen_half_cell_voltage(edge_conc_vals_pos,i_app_data_pos,electrode_pos,tsteps,pos↩
_params=posparams)
```

Definition at line 97 of file Uncertainty_Propagation.py.

6.9.3.45 posparams

```
list posparams = [K_pos,parameters[3],cmax_pos_sim,parameters[4]]
```

Definition at line 94 of file Uncertainty_Propagation.py.

6.9.3.46 R_dist

```
R_dist = st.norm(5.86e-6, 1.0e-6)
```

Definition at line 109 of file Uncertainty_Propagation.py.

6.9.3.47 R_neg

```
R_neg
```

Definition at line 87 of file Uncertainty_Propagation.py.

6.9.3.48 R_pos

```
R_pos
```

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.49 R_samples

```
R_samples = R_dist.rvs(num_samples)
```

Definition at line 121 of file Uncertainty_Propagation.py.

6.9.3.50 resol

```
int resol = 1000
```

Definition at line 234 of file Uncertainty_Propagation.py.

6.9.3.51 sharex

```
sharex
```

Definition at line 179 of file Uncertainty_Propagation.py.

6.9.3.52 sharey

```
sharey
```

Definition at line 179 of file Uncertainty_Propagation.py.

6.9.3.53 time

```
time = np.linspace(0.0,dt*(tsteps-1.),tsteps)
```

_, _, tsteps = UI.iapp_read_csv(iapp_filename)

Definition at line 169 of file Uncertainty_Propagation.py.

6.9.3.54 time_axis

```
time_axis
```

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.55 True

```
True
```

Definition at line 179 of file Uncertainty_Propagation.py.

6.9.3.56 tsteps

```
tsteps
```

Definition at line 86 of file Uncertainty_Propagation.py.

6.9.3.57 voltage_curves

```
def voltage_curves = np.zeros(tsteps)
```

Definition at line 172 of file Uncertainty_Propagation.py.

6.9.3.58 x

```
x = np.linspace(900, 1100, resol)
```

Definition at line 237 of file Uncertainty_Propagation.py.

6.10 unit_test Namespace Reference

Function that runs a unit test with a provided number of nodes, a provided timestep and a provided number of cores.

Functions

- def run_unit_test (nodenum=500, dt=1.0, num_cores=1)

6.10.1 Detailed Description

Function that runs a unit test with a provided number of nodes, a provided timestep and a provided number of cores.

The unit test is just a full battery simulation with the default (Chen 2020) parameters, where a gradually ramping current is applied for an amount of time followed by a period of rest.

6.10.2 Function Documentation

6.10.2.1 run_unit_test()

```
def unit_test.run_unit_test (
              nodenum = 500,
              dt = 1.0,
              num_cores = 1 )
```

Definition at line 12 of file unit_test.py.

6.11 user_input Namespace Reference

Sets up the user inputs and executes the solver.

Variables

- bool checkpoint = False
- string solver_input_filename = 'user_input'
- tsteps

  SET VALUES #########.
- dt
- n
- c0
- D
- R
- a
- L
- iapp
- iapp_label
- electrode_charge
- float K_pos = 3.42E-6
- float K_neg = 6.48E-7
- float cmax_pos_sim = 63104.00
- float cmax_neg_sim = 33133.00
- list plot_params_pos = [K_pos,a,cmax_pos_sim,L]

  END SET VALUES #########.
- pos_params

6.11.1 Detailed Description

Sets up the user inputs and executes the solver.

This file contains all input parameters for the solver and can be changed by the user. This file and all tuneable parameters are broken down in full detail in the user tutorial (see Tutorial.ipynb).

The following options can be controlled by editing this file:

- Outputting stdout to a file.
- Changing the name of this user input file.
- Set the values for the following input parameters:
  - tsteps
  - dt
  - n
  - c0
  - D
  - R
  - a
  - L
  - iapp
  - iapp_label
- These parameters can either take default values or be entered manually.
- For iapp there are additional options for setting constant or stepwise current and reading in values from a csv file.

The user does not need to edit anything past #END SET INPUT PARAMETER VALUES

The rest of the file calls various functions to:

- Check that the filename and parameters are valid
- Write the parameters to the file
- Call the solver and plotter.

6.11.2 Variable Documentation

6.11.2.1 a

`a`

Definition at line 48 of file user_input.py.

6.11.2.2 c0

```
c0
```

Definition at line 48 of file user_input.py.

6.11.2.3 checkpoint

```
bool checkpoint = False
```

Definition at line 40 of file user_input.py.

6.11.2.4 cmax_neg_sim

```
float cmax_neg_sim = 33133.00
```

Definition at line 56 of file user_input.py.

6.11.2.5 cmax_pos_sim

```
float cmax_pos_sim = 63104.00
```

Definition at line 55 of file user_input.py.

6.11.2.6 D

```
D
```

Definition at line 48 of file user_input.py.

6.11.2.7 dt

```
dt
```

Definition at line 48 of file user_input.py.

6.11.2.8 electrode_charge

```
electrode_charge
```

Definition at line 48 of file user_input.py.

6.11.2.9 iapp

```
iapp
```

Definition at line 48 of file user_input.py.

6.11.2.10 iapp_label

```
iapp_label
```

Definition at line 48 of file user_input.py.

6.11.2.11 K_neg

```
float K_neg = 6.48E-7
```

Definition at line 53 of file user_input.py.

6.11.2.12 K_pos

```
float K_pos = 3.42E-6
```

Definition at line 52 of file user_input.py.

6.11.2.13 L

```
L
```

Definition at line 48 of file user_input.py.

6.11.2.14 n

```
n
```

Definition at line 48 of file user_input.py.

6.11.2.15 plot_params_pos

```
list plot_params_pos = [K_pos,a,cmax_pos_sim,L]
```

END SET VALUES #########.

Definition at line 81 of file user_input.py.

6.11.2.16 pos_params

```
pos_params
```

Definition at line 84 of file user_input.py.

6.11.2.17 R

```
R
```

Definition at line 48 of file user_input.py.

6.11.2.18 solver_input_filename

```
solver_input_filename = 'user_input'
```

Definition at line 41 of file user_input.py.

6.11.2.19 tsteps

```
tsteps
```

SET VALUES #########.

Definition at line 48 of file user_input.py.

6.12 user_input_full_battery_GITT Namespace Reference

Example script for a full battery model.

Variables

- string output_filename_positive = 'current_step_pos'

  Uncomment to print stdout to file sys.stdout = open('test.txt', 'w')
- string output_filename_negative = 'current_step_neg'
- tsteps

  Import default values from https://doi.org/10.1149/1945-7111/ab9050 Use UI.set_defaults_pos() for positive electrode and ..._neg() for negative electrode.
- dt = 1.0
- n = 200
- c0
- D_neg
- R_neg
- a_neg
- L_neg
- iapp_neg
- iapp_label_neg
- electrode_charge_neg
- D_pos
- R_pos
- a_pos
- L_pos
- iapp_pos
- iapp_label_pos
- electrode_charge_pos
- float c0_pos = 30000.0

  Set values ######.
- float c0_neg = 1.0
- float K_pos = 3.42E-6
- float K_neg = 6.48E-7
- float cmax_pos_sim = 63104.00
- float cmax_neg_sim = 33133.00
- int nprocs = 40

  Check parameters are valid ######.
- int nsteps = 10
- list currents = [20.0 for i in range(nsteps)]
- list start_times = [2150.0∗i for i in range(nsteps)]
- list run_times = [150.0 for i in range(nsteps)]
- list wait_times = [2000.0 for i in range(nsteps)]
- list params_pos = [dt, c0_pos, D_pos, R_pos, a_pos, L_pos, electrode_charge_pos]
- list params_neg = [dt, c0_neg, D_neg, R_neg, a_neg, L_neg, electrode_charge_neg]
- list plot_params_pos = [K_pos,a_pos,cmax_pos_sim,L_pos]

  Call the function to perform the full battery parallel solve.
- list plot_params_neg = [K_neg,a_neg,cmax_neg_sim,L_neg]
- pos_params
- neg_params
- SparsifyAnimation
- True
- animation_interval_time

6.12.1 Detailed Description

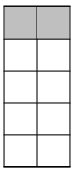
Example script for a full battery model.

This is a running script which exists to demonstrate how one uses the GITT function to run simulations in parallel on a full battery. This simulation completes 10 current cycles on each electrode over 2 cores each - parallelising perfectly over 40 cores.

These simulations consist of applying a step function of current to a battery which turns on to a fixed value for a given time interval, then back to 0, and then on again, etc.

The ascii art below indicates schematically how a current-time step looks during this experiment.

I |_____ _____



|_____|_____|_____t

To run one of these simulations, one needs the following things:

- A vector containing the fixed values of current applied for each block of current
- A vector containing the start times of each current block
- A vector containing the duration of each current block

6.12.2 Variable Documentation

6.12.2.1 a_neg

`a_neg`

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.2 a_pos

`a_pos`

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.3 animation_interval_time

```
animation_interval_time
```

Definition at line 89 of file user_input_full_battery_GITT.py.

6.12.2.4 c0

```
c0
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.5 c0_neg

```
float c0_neg = 1.0
```

Definition at line 48 of file user_input_full_battery_GITT.py.

6.12.2.6 c0_pos

```
float c0_pos = 30000.0
```

Set values ######.

Definition at line 47 of file user_input_full_battery_GITT.py.

6.12.2.7 cmax_neg_sim

```
float cmax_neg_sim = 33133.00
```

Definition at line 56 of file user_input_full_battery_GITT.py.

6.12.2.8 cmax_pos_sim

```
float cmax_pos_sim = 63104.00
```

Definition at line 55 of file user_input_full_battery_GITT.py.

6.12.2.9 currents

```
list currents = [20.0 for i in range(nsteps)]
```

Definition at line 72 of file user_input_full_battery_GITT.py.

6.12.2.10 D_neg

```
D_neg
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.11 D_pos

```
D_pos
```

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.12 dt

```
float dt = 1.0
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.13 electrode_charge_neg

```
electrode_charge_neg
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.14 electrode_charge_pos

```
electrode_charge_pos
```

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.15 iapp_label_neg

```
iapp_label_neg
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.16 iapp_label_pos

```
iapp_label_pos
```

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.17 iapp_neg

```
iapp_neg
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.18 iapp_pos

```
iapp_pos
```

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.19 K_neg

```
float K_neg = 6.48E-7
```

Definition at line 53 of file user_input_full_battery_GITT.py.

6.12.2.20 K_pos

```
float K_pos = 3.42E-6
```

Definition at line 52 of file user_input_full_battery_GITT.py.

6.12.2.21 L_neg

```
L_neg
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.22 L_pos

```
L_pos
```

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.23 n

```
int n = 200
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.24 neg_params

```
neg_params
```

Definition at line 89 of file user_input_full_battery_GITT.py.

6.12.2.25 nprocs

```
int nprocs = 40
```

Check parameters are valid ######.

Manually set up applied current and parallelisation ###### Number of parallel processors being utilised

Definition at line 66 of file user_input_full_battery_GITT.py.

6.12.2.26 nsteps

```
int nsteps = 10
```

Definition at line 69 of file user_input_full_battery_GITT.py.

6.12.2.27 output_filename_negative

```
output_filename_negative = 'current_step_neg'
```

Definition at line 39 of file user_input_full_battery_GITT.py.

6.12.2.28 output_filename_positive

```
output_filename_positive = 'current_step_pos'
```

Uncomment to print stdout to file sys.stdout = open('test.txt', 'w')

Call the plotting function which plots the results of the GITT test with nstep steps.

Set filename to output user input parameters ###### Set string for filename. Do not enter a file extension. Max characters = 50

Definition at line 38 of file user_input_full_battery_GITT.py.

6.12.2.29 params_neg

```
list params_neg = [dt, c0_neg, D_neg, R_neg, a_neg, L_neg, electrode_charge_neg]
```

Definition at line 79 of file user_input_full_battery_GITT.py.

6.12.2.30 params_pos

```
list params_pos = [dt, c0_pos, D_pos, R_pos, a_pos, L_pos, electrode_charge_pos]
```

Definition at line 78 of file user_input_full_battery_GITT.py.

6.12.2.31 plot_params_neg

```
plot_params_neg = [K_neg,a_neg,cmax_neg_sim,L_neg]
```

Definition at line 86 of file user_input_full_battery_GITT.py.

6.12.2.32 plot_params_pos

```
plot_params_pos = [K_pos,a_pos,cmax_pos_sim,L_pos]
```

Call the function to perform the full battery parallel solve.

Definition at line 85 of file user_input_full_battery_GITT.py.

6.12.2.33 pos_params

```
pos_params
```

Definition at line 89 of file user_input_full_battery_GITT.py.

6.12.2.34 R_neg

```
R_neg
```

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.35 R_pos

```
R_pos
```

Definition at line 44 of file user_input_full_battery_GITT.py.

6.12.2.36 run_times

```
list run_times = [150.0 for i in range(nsteps)]
```

Definition at line 74 of file user_input_full_battery_GITT.py.

6.12.2.37 SparsifyAnimation

```
SparsifyAnimation
```

Definition at line 89 of file user_input_full_battery_GITT.py.

6.12.2.38 start_times

```
start_times = [2150.0*i for i in range(nsteps)]
```

Definition at line 73 of file user_input_full_battery_GITT.py.

6.12.2.39 True

```
True
```

Definition at line 89 of file user_input_full_battery_GITT.py.

6.12.2.40 tsteps

```
tsteps
```

Import default values from https://doi.org/10.1149/1945-7111/ab9050 Use UI.set_defaults_pos() for positive electrode and ..._neg() for negative electrode.

Definition at line 43 of file user_input_full_battery_GITT.py.

6.12.2.41 wait_times

```
list wait_times = [2000.0 for i in range(nsteps)]
```

Definition at line 75 of file user_input_full_battery_GITT.py.

6.13 user_input_mod Namespace Reference

Package containing the functions needed to set up the user input parameters and execute the solver.

Functions

- def iapp_read_csv (filename)

  CURRENT DENSITY SET UP ###.
- def iapp_constant_setup (tsteps, iapp)

  Sets up the applied current density 'iapp' as a constant valued array of length tsteps.
- def iapp_step_setup (tsteps, iapp_steps)

  Sets up the applied current density 'iapp' as a stepped array of length tsteps.
- def set_defaults_pos ()

  DEFAULT PARAMETERS ###.
- def set_defaults_neg ()

  Returns default parameters for a negative electrode.
- def verify_params (filename, tsteps, dt, n, c0, D, R, a, L, electrode_charge)

  PARAMETER VERIFICATION ###.
- def verify_iapp (iapp, iapp_label, tsteps)

  Verifies the types and values of the applied current density array and its label.
- def write_to_file (filename, tsteps, dt, n, c0, D, R, a, L, iapp, iapp_label, electrode_charge)

  WRITE TO FILE ###.
- def call_solver (filename, checkpoint, nprocs=1)

  CALL SOLVER ###.
- def get_GITT_initial_concs (currents, run_times, c0, R, a, L, electrode_charge)
- def GITT_half_cell (filename, nprocs, currents, start_times, run_times, wait_times, n, params)

  INITIALISE A FULL GITT TEST IN PARALLEL ####.
- def GITT_full_cell (filename_positive, filename_negative, nprocs, currents, start_times, run_times, wait_↩ times, n, params_pos, params_neg)

  Executes the SPM solver in parallel to run a GITT full cell test over nprocs cores.
- def full_battery_simulation (filename_positive, filename_negative, nprocs=1)

  Executes the SPM solver.
- def optimise_parallelism (n_procs, n_gitt_steps=1, full_battery=False)

  Decides the correct number of threads to use for a given simulation.

6.13.1 Detailed Description

Package containing the functions needed to set up the user input parameters and execute the solver.

Provides options for automatically generating applied current density 'iapp', as a constant or step function, and for reading iapp from a csv file.

Provides the option to set the parameters to default values obtained from **?**.

Contains additional functions for validating the types and values of the input parameters, writing the user input txt file, and executing the solver.

6.13.2 Function Documentation

6.13.2.1 call_solver()

```
def user_input_mod.call_solver (
              filename,
              checkpoint,
              nprocs = 1 )
```

CALL SOLVER ###.

Executes the SPM solver.

```
@details Calls the SPM solver using the subprocess package.
The filename of the user input file is passed to the solver as a command line argument.
Errors from the execution are read in and further execution prevented if necessary.
@param[in] filename: The name of the file containing the desired input, either a checkpoint file or user input
@param[in] checkpoint: Boolean indicating if a checkpoint file is used.

The function works by:
1. Validating whether the input file is a checkpoint file or user input file name.
2. Setting up the solver call line, including the file name.
3. Calling the solver.
```

Definition at line 413 of file user_input_mod.py.

6.13.2.2 full_battery_simulation()

```
def user_input_mod.full_battery_simulation (
              filename_positive,
              filename_negative,
              nprocs = 1 )
```

Executes the SPM solver.

```
@details Calls the SPM solver using the subprocess package for both the anode and cathode simultaneously
The filenames of the user input files for both the anode and cathode are passed to
instances of the solver as command line arguments. Note that if 2 processors are supplied,
both the anode and cathode will run simultaneously.
@param[in] filename_positive: The name of the positive output file, this must be a string and have max 50 char
@param[in] filename_negative: The name of the negative output file, this must be a string and have max 50 char

The function works by:
1. Seting up the solver call line, including the file name.
2. Calling the solver.
```

Definition at line 651 of file user_input_mod.py.

6.13.2.3 get_GITT_initial_concs()

```
def user_input_mod.get_GITT_initial_concs (
              currents,
              run_times,
              c0,
              R,
              a,
              L,
              electrode_charge )


@brief Calculates the initial concentrations for each step in a multi-step parallelised GITT test.

@details Computes initial flat concentrations for the initial step of a parallelised multi-step GITT test.

The formula used for this calculation is:
C(T=t) = C_0 + \frac{i_app * t}{F*e_act * L}, where C_0 is C(T=0).

As current is defined as positive for charging, the sign of the current vector must be flipped when considerin
the positive electrode (cathode) which will discharge during charging.

@param[in] currents  A vector containing current values to apply at each current step, must be floats and have
start_times, run_times, wait_times.
@param[in] run_times  A vector containing the run time of each current step, must be floats and have the same
start_times, currents, wait_times.
@param[in] c0  Initial concentration, must be a float greater than or equal to 0.
@param[in] R  Radius of the sphere, must be a float greater than 0.
@param[in] a  Particle surface area per unit volume, must be a float greater than or equal to 0.
@param[in] L  Electrode thickness, must be a float greater than or equal to 0.
@param[in] electrode_charge  Electrode charge, mus be a single character 'p' or 'n'.
```

Definition at line 475 of file user_input_mod.py.

6.13.2.4 GITT_full_cell()

```
def user_input_mod.GITT_full_cell (
              filename_positive,
              filename_negative,
              nprocs,
              currents,
              start_times,
              run_times,
              wait_times,
              n,
              params_pos,
              params_neg )
```

Executes the SPM solver in parallel to run a GITT full cell test over nprocs cores.

```
@details Executes the SPM solver in parallel to run a test experiment on a full cell using a galvanostatic int
see \cite Weppner1977

Due to the equilibration time between each current step applied, it is possible to pre-compute initial constan
particles in the model by considering the amount of lithium removed in each step.

@param[in] filename_positive  Name of user input file for the cathode, no file extension. Note that a version
each current step applied in the GITT test.
@param[in] filename_negative  Name of user input file for the anode, no file extension. Note that a version of
```

```
each current step applied in the GITT test.
@param[in] nprocs  Number of processors to parallelise current steps over.
Cannot parallelise over more processors than the number of steps applied in the GITT test.
@param[in] currents  A vector containing the values of current to apply at each step, must be floats and have
start_times, run_times, wait_times.
@param[in] start_times  A vector containing the start times of each current step, must be floats and have the
currents, run_times, wait_times.
@param[in] run_times  A vector containing the run time of each current step, must be floats and have the same
start_times, currents, wait_times.
@param[in] wait_times  A vector containing the wait time of each current step, must be floats and have the sam
currents, start_times, run-times.
@param[in], n: The number of nodes to use in the simulation. Must be an integer greater than 100.
@param[in] params: A vector containing the parameters for the simulation: [dt, c0, D, R, a, L]

The function works by:
1. Setting up the solver call line, including the file name.
2. Calling the solver.
```

Definition at line 576 of file user_input_mod.py.

6.13.2.5 GITT_half_cell()

```
def user_input_mod.GITT_half_cell (
              filename,
              nprocs,
              currents,
              start_times,
              run_times,
              wait_times,
              n,
              params )
```

INITIALISE A FULL GITT TEST IN PARALLEL ####.

Executes the SPM solver in parallel to run a GITT half cell test over nprocs cores.

```
@details Executes the SPM solver in parallel to run a test experiment on a half cell using a galvanostatic int
see \cite Weppner1977

Due to the equilibration time between each current step applied, it is possible to pre-compute initial constan
particles in the model by considering the amount of lithium removed in each step.

Errors from execution are read in and further execution is prevented if necessary.

@param[in] filename  Name of user input file, no file extension. Note that a version of this file is created f
@param[in] nprocs  Number of processors to parallelise current steps over.
Cannot parallelise over more processors than the number of steps applied in the GITT test.
@param[in] currents  A vector containing the values of current to apply at each step, must be floats and have
start_times, run_times, wait_times.
@param[in] start_times  A vector containing the start times of each current step, must be floats and have the
currents, run_times, wait_times.
@param[in] run_times  A vector containing the run time of each current step, must be floats and have the same
start_times, currents, wait_times.
@param[in] wait_times  A vector containing the wait time of each current step, must be floats and have the sam
currents, start_times, run-times.
@param[in], n: The number of nodes to use in the simulation. Must be an integer greater than 100.
@param[in] params: A vector containing the parameters for the simulation: [dt, c0, D, R, a, L, electrode_charg
```

Definition at line 513 of file user_input_mod.py.

6.13.2.6 iapp_constant_setup()

```
def user_input_mod.iapp_constant_setup (
              tsteps,
              iapp )
```

Sets up the applied current density 'iapp' as a constant valued array of length tsteps.

```
@param[in] tsteps Number of timesteps, needs to be the length of the array iapp.
@param[in] iapp Constant value of iapp.
@result iapp_arr: An array containing applied current density.
@result iapp_label: String containing a description of iapp.

Function does the following:
1. Verifies that the current density, iapp, is a float.
2. Sets up an array.
```

Definition at line 52 of file user_input_mod.py.

6.13.2.7 iapp_read_csv()

```
def user_input_mod.iapp_read_csv (
              filename )
```

CURRENT DENSITY SET UP ###.

Reads in the applied current density 'iapp' from a csv file provided by user.

```
@param[in] filename Name of the csv file containing iapp as Time, Current density.
@result iapp_arr: Array containing applied current density.
@result iapp_label: String containing a description of iapp.
@result tsteps: Number of timesteps, found from the length of iapp array.

This function does the following:
1. Reads in the csv file.
2. Parses the file into array of lines.
3. Parses each line at comma.
4. Define tsteps and the iapp_label.
```

Definition at line 16 of file user_input_mod.py.

6.13.2.8 iapp_step_setup()

```
def user_input_mod.iapp_step_setup (
              tsteps,
              iapp_steps )
```

Sets up the applied current density 'iapp' as a stepped array of length tsteps.

```
@param[in] tsteps Number of timesteps, needs to be the length of the array iapp.
@param[in] iapp_steps 2D Array containing heights of the steps and the timesteps at which step occurs, startin
@result iapp_arr: An array containing the applied current density.
@result iapp_label: A string describing the type for iapp.

Function does the following:
1. Sets up the label and initialises the array.
2. Verifies that timesteps are integers and current density values are floats.
3. Verifies that the first timestep is 0.
4. Sets up an array.
```

Definition at line 76 of file user_input_mod.py.

6.13.2.9 optimise_parallelism()

```
def user_input_mod.optimise_parallelism (
              n_procs,
              n_gitt_steps = 1,
              full_battery = False )
```

Decides the correct number of threads to use for a given simulation.

```
@details Given a user supplied number of processors, as well as further details about the simulation
(the number of current steps if it's a GITT test, if the simulation is a full cell or not), this function
decides on the optimum number of threads to use. It never picks a number of threads greater than 4,
as any larger than this and the communication overhead between threads during the matrix solve begins to domin
and the computation time increases. This function therefore allows a balanced optimum parallelisation strategy
@param[in] n_procs: The number of processors that the user is happy for the program to exploit. The program wi
of cores less than or equal to this number. Integer.
@param[in] n_gitt_steps: The number of GITT steps if the experiment being performed is a GITT test. 1 by defau
the experiment not being a GITT test) integer.
@param[in] full_battery: Bool, True means that both sides of a battery are being simulated, whilst False means

The function works by:
1. Deciding what number of threads is best to use given the type of simulation and input parameters, up to a m
   Note that this is based off the principle that parallelisation over independent seperate instances of the s
   than multithreading.
2. Setting the environment variable.
```

Definition at line 686 of file user_input_mod.py.

6.13.2.10 set_defaults_neg()

```
def user_input_mod.set_defaults_neg ( )
```

Returns default parameters for a negative electrode.

Gives the default parameter values, obtained from **?**. THe simulation is set up to run for 100 timesteps of size dt = 0.1s, with n = 1000 spatial nodes. Applied current density is set up as a constant current density of value 0.73mA m^2.

Definition at line 174 of file user_input_mod.py.

6.13.2.11 set_defaults_pos()

```
def user_input_mod.set_defaults_pos ( )
```

DEFAULT PARAMETERS ###.

Returns default parameters for a positive electrode

Gives the default parameter values, obtained from **?** The simulation is set up to run for 100 timesteps of size dt = 0.1s, with n = 1000 spatial nodes. Applied current density is set up as a constant current density of value 0.73mA m^2.

Definition at line 139 of file user_input_mod.py.

6.13.2.12 verify_iapp()

```
def user_input_mod.verify_iapp (
              iapp,
              iapp_label,
              tsteps )
```

Verifies the types and values of the applied current density array and its label.

```
@details Verifies that the parameters output iapp and iapp_label have the correct types, valid values, and cor
If any are found to be invalid, an error is printed and the execution will stop.
@param[in] iapp: Applied current density, 1D array of floats of length tsteps.
@param[in] iapp_label: Labels the type of applied current density, must be a string.
@param[in] tsteps: Number of timesteps, in order to check iapp has the correct length.
```

Definition at line 323 of file user_input_mod.py.

6.13.2.13 verify_params()

```
def user_input_mod.verify_params (
              filename,
              tsteps,
              dt,
              n,
              c0,
              D,
              R,
              a,
              L,
              electrode_charge )
```

PARAMETER VERIFICATION ###.

Verifies the types and values of the input parameters.

```
@details Verifies that output filename, tsteps, dt, c0, D, R, a, and L have the correct type and valid values.
If any are found to be invalid, an error is printed and the execution will stop.

@param[in] filename  The name of the output file, this must be a string and have max 50 characters. No file ex
@param[in] tsteps  Number of timesteps, must be an integer greater than 0.
@param[in] dt  Timestep size, must be a float greater than 0.
@param[in] n  Number of spatial nodes, must be an integer between 100 and 4000.
@param[in] c0  Initial concentration, must be a float greater than or equal to 0.
@param[in] D  Diffusion constant, must be a float.
@param[in] R  Radius of the sphere, must be a float greater than 0.
@param[in] a  Particle surface area per unit volume, must be a float greater than or equal to 0.
@param[in] L  Electrode thickness, must be a float greater than or equal to 0.
@param[in] electrode_charge  Labels charge of the electrode, must b a string, 'p' for positive or 'n' for nega
```

Definition at line 213 of file user_input_mod.py.

6.13.2.14 write_to_file()

```
def user_input_mod.write_to_file (
              filename,
              tsteps,
              dt,
              n,
              c0,
              D,
              R,
              a,
              L,
              iapp,
              iapp_label,
              electrode_charge )
```

WRITE TO FILE ###.

Writes user inputs to a txt file.

```
@details Writes the user inputs to a txt file named 'filename'.
Parameters are output in the format 'parameter = value'.
tsteps, dt, c0, D, R, a, L, electrode_charge are output to the top of the file, followed by an asterix line (*
iapp_label is output below the asterix line, with iapp array the following it, written one element per line.

The function works by:
1. Setting the file name.
2. Making a list of strings to write to the file.
3. Writing to the file.
```

Definition at line 359 of file user_input_mod.py.

6.14 user_input_parallel_simulation Namespace Reference

a running script which exists to demonstrate how one uses the GITT function to run simulations in parallel.

Variables

- string output_filename = 'current_step'
- string electrode = 'positive'
- tsteps
- dt = 1.0
- n
- c0 = 1.0
- D
- R
- a
- L
- iapp
- iapp_label
- electrode_charge
- float K_pos = 3.42E-6
- float K_neg = 6.48E-7
- float cmax_pos_sim = 63104.00

- float cmax_neg_sim = 33133.00
- int nprocs = 5
- int nsteps = 5
- list currents = [20.0 for i in range(nsteps)]
- list start_times = [2150.0∗i for i in range(nsteps)]
- list run_times = [150.0 for i in range(nsteps)]
- list wait_times = [2000.0 for i in range(nsteps)]
- list params = [dt, c0, D, R, a, L,electrode_charge]
- list plot_params_neg = [K_neg,a,cmax_neg_sim,L]
- neg_params
- Animation
- True
- SparsifyAnimation

6.14.1 Detailed Description

a running script which exists to demonstrate how one uses the GITT function to run simulations in parallel.

6.14.2 Variable Documentation

6.14.2.1 a

```
a
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.2 Animation

```
Animation
```

Definition at line 103 of file user_input_parallel_simulation.py.

6.14.2.3 c0

```
float c0 = 1.0
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.4 cmax_neg_sim

```
float cmax_neg_sim = 33133.00
```

Definition at line 74 of file user_input_parallel_simulation.py.

6.14.2.5 cmax_pos_sim

```
float cmax_pos_sim = 63104.00
```

Definition at line 73 of file user_input_parallel_simulation.py.

6.14.2.6 currents

```
list currents = [20.0 for i in range(nsteps)]
```

Definition at line 87 of file user_input_parallel_simulation.py.

6.14.2.7 D

```
D
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.8 dt

```
float dt = 1.0
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.9 electrode

```
electrode = 'positive'
```

Definition at line 37 of file user_input_parallel_simulation.py.

6.14.2.10 electrode_charge

```
electrode_charge
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.11 iapp

```
iapp
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.12 iapp_label

```
iapp_label
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.13 K_neg

```
float K_neg = 6.48E-7
```

Definition at line 71 of file user_input_parallel_simulation.py.

6.14.2.14 K_pos

```
float K_pos = 3.42E-6
```

Definition at line 70 of file user_input_parallel_simulation.py.

6.14.2.15 L

```
L
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.16 n

```
n
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.17 neg_params

```
neg_params
```

Definition at line 103 of file user_input_parallel_simulation.py.

6.14.2.18 nprocs

```
int nprocs = 5
```

Definition at line 82 of file user_input_parallel_simulation.py.

6.14.2.19 nsteps

```
int nsteps = 5
```

Definition at line 84 of file user_input_parallel_simulation.py.

6.14.2.20 output_filename

```
output_filename = 'current_step'
```

Definition at line 36 of file user_input_parallel_simulation.py.

6.14.2.21 params

```
list params = [dt, c0, D, R, a, L,electrode_charge]
```

Definition at line 93 of file user_input_parallel_simulation.py.

6.14.2.22 plot_params_neg

```
plot_params_neg = [K_neg,a,cmax_neg_sim,L]
```

Definition at line 100 of file user_input_parallel_simulation.py.

6.14.2.23 R

```
R
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.24 run_times

```
list run_times = [150.0 for i in range(nsteps)]
```

Definition at line 89 of file user_input_parallel_simulation.py.

6.14.2.25 SparsifyAnimation

```
SparsifyAnimation
```

Definition at line 103 of file user_input_parallel_simulation.py.

6.14.2.26 start_times

```
start_times = [2150.0*i for i in range(nsteps)]
```

Definition at line 88 of file user_input_parallel_simulation.py.

6.14.2.27 True

```
True
```

Definition at line 103 of file user_input_parallel_simulation.py.

6.14.2.28 tsteps

```
tsteps
```

Definition at line 61 of file user_input_parallel_simulation.py.

6.14.2.29 wait_times

```
list wait_times = [2000.0 for i in range(nsteps)]
```

Definition at line 90 of file user_input_parallel_simulation.py.

Chapter 7

File Documentation

7.1 /home/chem/msuvnj/PX915_GroupB_22-23/checkpointing.f90 File Reference

Modules

- module checkpointing

  Module to read and write checkpoint files.

Variables

- integer, intent(in) tstep
- integer, intent(in) tsteps
- real(real64), intent(in) dt
- integer, intent(in) n
- real(real64), dimension(n), intent(in) c
- real(real64), intent(in) d
- real(real64), intent(in) r
- real(real64), intent(in) a_small
- real(real64), intent(in) l
- real(real64), dimension(tsteps), intent(in) iapp
- character(len= $*$), intent(in) electrode_charge
- real(real64), dimension(n, tsteps), intent(in) cstorage
- integer, intent(in), optional freq_in
- integer freq
- integer freq_remainder
- character(len= $*$), intent(inout) filename
- character(len=60) dir
- character(len=60) checkpoint_name
- character(len=120) filepath
- integer ncid
- integer dimid_n
- integer dimid_tsteps
- integer, dimension(1) dimids_parameter
- integer, dimension(2) dimids_cstorage
- integer status

- integer varid_tstep
- integer varid_tsteps
- integer varid_dt
- integer varid_n
- integer varid_c
- integer varid_d
- integer varid_r
- integer varid_a
- integer varid_l
- integer varid_iapp
- integer varid_ec
- integer varid_cstorage
- integer filename_length
- integer file_ext
- integer file_test
- character(len=10) tstep_str
- integer, intent(out) tstep_init
- integer varid
- character(len=5) file_extension
- integer parse_idx
- real(real64) c0

7.1.1 Variable Documentation

7.1.1.1 a_small

```
real(real64), intent(out) a_small
```

Definition at line 42 of file checkpointing.f90.

7.1.1.2 c

```
real(real64), dimension(:), intent(out), allocatable c
```

Definition at line 33 of file checkpointing.f90.

7.1.1.3 c0

```
real(real64) c0
```

Definition at line 298 of file checkpointing.f90.

7.1.1.4 checkpoint_name

```
character(len=60) checkpoint_name
```

Definition at line 66 of file checkpointing.f90.

7.1.1.5 cstorage

```
real(real64), dimension(:, :), intent(out), allocatable cstorage
```

Definition at line 54 of file checkpointing.f90.

7.1.1.6 d

```
real(real64), intent(out) d
```

Definition at line 36 of file checkpointing.f90.

7.1.1.7 dimid_n

```
integer dimid_n
```

Definition at line 68 of file checkpointing.f90.

7.1.1.8 dimid_tsteps

```
integer dimid_tsteps
```

Definition at line 68 of file checkpointing.f90.

7.1.1.9 dimids_cstorage

```
integer, dimension(2) dimids_cstorage
```

Definition at line 68 of file checkpointing.f90.

7.1.1.10 dimids_parameter

```
integer, dimension(1) dimids_parameter
```

Definition at line 68 of file checkpointing.f90.

7.1.1.11 dir

```
character(len=60) dir
```

Definition at line 66 of file checkpointing.f90.

7.1.1.12 dt

```
real(real64), intent(out) dt
```

Definition at line 27 of file checkpointing.f90.

7.1.1.13 electrode_charge

```
character(len=1), intent(out) electrode_charge
```

Definition at line 51 of file checkpointing.f90.

7.1.1.14 file_ext

```
integer file_ext
```

Definition at line 71 of file checkpointing.f90.

7.1.1.15 file_extension

```
character(len=5) file_extension
```

Definition at line 275 of file checkpointing.f90.

7.1.1.16 file_test

```
integer file_test
```

Definition at line 71 of file checkpointing.f90.

7.1.1.17 filename

```
character(len=*), intent(in) filename
```

Definition at line 65 of file checkpointing.f90.

7.1.1.18 filename_length

```
integer filename_length
```

Definition at line 71 of file checkpointing.f90.

7.1.1.19 filepath

```
character(len=120) filepath
```

Definition at line 67 of file checkpointing.f90.

7.1.1.20 freq

```
integer freq
```

Definition at line 60 of file checkpointing.f90.

7.1.1.21 freq_in

```
integer, intent(in), optional freq_in
```

Definition at line 59 of file checkpointing.f90.

7.1.1.22 freq_remainder

```
integer freq_remainder
```

Definition at line 61 of file checkpointing.f90.

7.1.1.23 iapp

```
real(real64), dimension(:), intent(out), allocatable iapp
```

Definition at line 48 of file checkpointing.f90.

7.1.1.24 l

```
real(real64), intent(out) l
```

Definition at line 45 of file checkpointing.f90.

7.1.1.25 n

```
integer, intent(out) n
```

Definition at line 30 of file checkpointing.f90.

7.1.1.26 ncid

```
integer ncid
```

Definition at line 68 of file checkpointing.f90.

7.1.1.27 parse_idx

```
integer parse_idx
```

Definition at line 278 of file checkpointing.f90.

7.1.1.28 r

```
real(real64), intent(out) r
```

Definition at line 39 of file checkpointing.f90.

7.1.1.29 status

```
integer status
```

Definition at line 68 of file checkpointing.f90.

7.1.1.30 tstep

```
integer, intent(in) tstep
```

Definition at line 21 of file checkpointing.f90.

7.1.1.31 tstep_init

```
integer, intent(out) tstep_init
```

Definition at line 166 of file checkpointing.f90.

7.1.1.32 tstep_str

```
character(len=10) tstep_str
```

Definition at line 72 of file checkpointing.f90.

7.1.1.33 tsteps

```
integer, intent(out) tsteps
```

Definition at line 24 of file checkpointing.f90.

7.1.1.34 varid

```
integer varid
```

Definition at line 202 of file checkpointing.f90.

7.1.1.35 varid_a

```
integer varid_a
```

Definition at line 70 of file checkpointing.f90.

7.1.1.36 varid_c

```
integer varid_c
```

Definition at line 69 of file checkpointing.f90.

7.1.1.37 varid_cstorage

```
integer varid_cstorage
```

Definition at line 70 of file checkpointing.f90.

7.1.1.38 varid_d

```
integer varid_d
```

Definition at line 70 of file checkpointing.f90.

7.1.1.39 varid_dt

```
integer varid_dt
```

Definition at line 69 of file checkpointing.f90.

7.1.1.40 varid_ec

```
integer varid_ec
```

Definition at line 70 of file checkpointing.f90.

7.1.1.41 varid_iapp

```
integer varid_iapp
```

Definition at line 70 of file checkpointing.f90.

7.1.1.42 varid_l

```
integer varid_l
```

Definition at line 70 of file checkpointing.f90.

7.1.1.43 varid_n

```
integer varid_n
```

Definition at line 69 of file checkpointing.f90.

7.1.1.44 varid_r

```
integer varid_r
```

Definition at line 70 of file checkpointing.f90.

7.1.1.45 varid_tstep

```
integer varid_tstep
```

Definition at line 69 of file checkpointing.f90.

7.1.1.46 varid_tsteps

```
integer varid_tsteps
```

Definition at line 69 of file checkpointing.f90.

7.2 /home/chem/msuvnj/PX915_GroupB_22-23/convergence_test.py File Reference

Namespaces

- convergence_test

  Test convergence of final voltage and lithium mass loss of unit test with dt and number of nodes.

Functions

- def run_dt_convergence ()
- def run_node_convergence ()

7.3 math.md File Reference

7.4 uq.md File Reference

7.5 /home/chem/msuvnj/PX915_GroupB_22-23/finite_diff_solver.f90 File Reference

Functions/Subroutines

- program main

  Crank-Nicolson finite differences solver.

7.5.1 Function/Subroutine Documentation

7.5.1.1 main()

```
program main
```

Crank-Nicolson finite differences solver.

Definition at line 22 of file finite_diff_solver.f90.

7.6 /home/chem/msuvnj/PX915_GroupB_22-23/mixed_parallel_strat_test.py File Reference

Namespaces

- mixed_parallel_strat_test

  Test the runtime of the unit test for different provided processes in the mixed parallel strategy.

Variables

- list num_cores = [1,2,4,8]
- list num_nodes = [2000]
- times = np.zeros([len(num_cores),len(num_nodes)])
- start = time.time()
- nodenum
- n
- dt
- end = time.time()
- label

7.7 /home/chem/msuvnj/PX915_GroupB_22-23/MKL_threads_test.py File Reference

Namespaces

- MKL_threads_test

  Test the runtime of the unit test for different numbers of threads.

Variables

- list num_threads = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,20]
- list num_nodes = [100,500,1000,2000]
- times = np.zeros([19,4])
- start = time.time()
- nodenum
- n
- dt
- end = time.time()
- label

7.8 /home/chem/msuvnj/PX915_GroupB_22-23/nc_output.f90 File Reference

Modules

- module nc_output

  Writes results to a netCDF file.

Functions/Subroutines

- subroutine check (status)
- subroutine output_cstorage (cstorage, n, tsteps, R, time_axis, electrode_charge, filename)

7.9 /home/chem/msuvnj/PX915_GroupB_22-23/plotter.py File Reference

Namespaces

- plotter

  Functions for visualisation of solver outputs.

Functions

- def read_output_file (filename, step_num=None)

  Reads in data from user input and solver output using NetCDF.
- def read_input_current (filename, step_num=None)

  Reads in the applied current.
- def animated_conc_plot (intervaltime, dr, tsteps, nodenum, cstore, time_axis, SaveFinalState=False, SparsifyAnimation=False)

  Saves animation of lithium concentration over time.
- def gen_half_cell_voltage (edge_conc_vals, i_app_data, electrode, tsteps, pos_params=None, neg_↩ params=None)

  Generate the voltage against time profile for a half cell and return voltage.
- def voltage_current_plot (electrode, cstore, time_axis, i_app_data, tsteps, pos_params=None, neg_↩ params=None)

  Calculates voltages and plots as a function of time.
- def plot_halfcell_GITT_result (filename, start_times, electrode, pos_params=None, neg_params=None, Animation=False, SaveFinalState=False, SparsifyAnimation=True, animation_interval_time=10)

  Generates the voltage current plot and optionally an animation for a half-cell GITT test.
- def gen_plots (filename, pos_params=None, neg_params=None, animation_interval_time=10, SaveFinal↩ State=True, SparsifyAnimation=False)

  A function which can be called to generate both the voltage-time, current-time plots and the animation.
- def full_battery_GITT_plots (filename_positive, filename_negative, start_times, pos_params=None, neg_↩ params=None, SparsifyAnimation=True, animation_interval_time=10)
- def get_avg_li_conc (c_vals, r_vals)

  Gets the average concentration of lithium at a given timestep using trapezium rule inside the sphere.

7.10 /home/chem/msuvnj/PX915_GroupB_22-23/read_inputs.f90 File Reference

Modules

- module read_inputs

  Module to read user inputs.

Variables

- integer num_args
- integer i
- integer parse_idx
- character(len=164) arg
- character(len=60) name
- character(len=104) val
- character(len=104) filename
- integer, intent(out) tsteps
- real(real64), intent(out) dt
- integer, intent(out) n
- real(real64), intent(out) c0
- real(real64), intent(out) d
- real(real64), intent(out) r
- real(real64), intent(out) a_small
- real(real64), intent(out) l
- real(real64), dimension(:), intent(out), allocatable iapp
- character(len=1), intent(out) electrode_charge
- integer, parameter file_id = 1
- integer, parameter n_params_expected = 9
- integer n_params = 1
- integer ios
- logical asterix = .FALSE.
- logical invalid_param = .FALSE.
- character(len=n_params_expected) param_read
- character(len=30) read_temp
- character(len=1), parameter parser ='='
- integer, intent(in) line

7.10.1 Variable Documentation

7.10.1.1 a_small

```
real(real64), intent(out) a_small
```

Definition at line 112 of file read_inputs.f90.

7.10.1.2 arg

```
character(len=164) arg
```

Definition at line 31 of file read_inputs.f90.

7.10.1.3 asterix

```
logical asterix = .FALSE.
```

Definition at line 146 of file read_inputs.f90.

7.10.1.4 c0

```
real(real64), intent(out) c0
```

Definition at line 103 of file read_inputs.f90.

7.10.1.5 d

```
real(real64), intent(out) d
```

Definition at line 106 of file read_inputs.f90.

7.10.1.6 dt

```
real(real64), intent(out) dt
```

Definition at line 97 of file read_inputs.f90.

7.10.1.7 electrode_charge

```
character(len=1), intent(out) electrode_charge
```

Definition at line 122 of file read_inputs.f90.

7.10.1.8 file_id

```
integer, parameter file_id = 1
```

Definition at line 130 of file read_inputs.f90.

7.10.1.9 filename

```
character(len=*), intent(in) filename
```

Definition at line 40 of file read_inputs.f90.

7.10.1.10 i

```
integer i
```

Definition at line 25 of file read_inputs.f90.

7.10.1.11 iapp

```
real(real64), dimension(:), intent(out), allocatable iapp
```

Definition at line 119 of file read_inputs.f90.

7.10.1.12 invalid_param

```
logical invalid_param = .FALSE.
```

Definition at line 149 of file read_inputs.f90.

7.10.1.13 ios

```
integer ios
```

Definition at line 143 of file read_inputs.f90.

7.10.1.14 l

```
real(real64), intent(out) l
```

Definition at line 115 of file read_inputs.f90.

7.10.1.15 line

```
integer, intent(in) line
```

Definition at line 359 of file read_inputs.f90.

7.10.1.16 n

```
integer, intent(out) n
```

Definition at line 100 of file read_inputs.f90.

7.10.1.17 n_params

```
integer n_params = 1
```

Definition at line 137 of file read_inputs.f90.

7.10.1.18 n_params_expected

```
integer, parameter n_params_expected = 9
```

Definition at line 134 of file read_inputs.f90.

7.10.1.19 name

```
character(len=25), intent(in) name
```

Definition at line 34 of file read_inputs.f90.

7.10.1.20 num_args

```
integer num_args
```

Definition at line 22 of file read_inputs.f90.

7.10.1.21 param_read

```
character (len=n_params_expected) param_read
```

Definition at line 152 of file read_inputs.f90.

7.10.1.22 parse_idx

```
integer parse_idx
```

Definition at line 28 of file read_inputs.f90.

7.10.1.23 parser

```
character(len=1), parameter parser ='='
```

Definition at line 159 of file read_inputs.f90.

7.10.1.24 r

```
real(real64), intent(out) r
```

Definition at line 109 of file read_inputs.f90.

7.10.1.25 read_temp

```
character(len=30) read_temp
```

Definition at line 156 of file read_inputs.f90.

7.10.1.26 tsteps

```
integer, intent(out) tsteps
```

Definition at line 94 of file read_inputs.f90.

7.10.1.27 val

```
character(len=25) val
```

Definition at line 37 of file read_inputs.f90.

7.11 /home/chem/msuvnj/PX915_GroupB_22-23/sensitivity_analysis.py File Reference

Namespaces

- sensitivity_analysis

  First order sensitivity analysis.

Functions

- def sensitivity_over_time ()

  Wrapper for the sensitivity analysis code.

Variables

- ani = FuncAnimation(fig, update, frames=len(scaled_sensitivities_matrix), blit=True)
- filename
- writer
- fps

7.12 /home/chem/msuvnj/PX915_GroupB_22-23/Uncertainty_↩ Propagation.py File Reference

Namespaces

- Uncertainty_Propagation

  Performs uncertainty propagation.

Functions

- def in_out_easy_peasy (parameter_np_array)

  Simplification of the solver that allows for input parameters to be supplied and only the QOI to be returned.

Variables

- iapp = np.concatenate((np.array([20.0 for i in range(50)]),np.array([0.0 for i in range(50)])))

  Read in applied current density from csv file iapp_filename = 'WLTP_m10.csv' iapp, iapp_label, tsteps = UI.iapp_↩ read_csv(iapp_filename)
- parameters = parameter_np_array

  END MEAN SET VALUES #########.
- int nprocs = 40

  Call the function to perform the full battery parallel solve.
- output_filename_positive
- output_filename_negative
- cstore_pos

  Obtaining QOI (Voltage) #.
- tsteps
- nodenum
- R_pos
- time_axis
- dr_pos
- electrode_pos
- cstore_neg
- R_neg
- dr_neg
- electrode_neg
- i_app_data_pos = plotter.read_input_current(output_filename_positive)
- i_app_data_neg = plotter.read_input_current(output_filename_negative)
- edge_conc_vals_pos = cstore_pos[:,-1]
- edge_conc_vals_neg = cstore_neg[:,-1]
- list posparams = [K_pos,parameters[3],cmax_pos_sim,parameters[4]]
- list negparams = [K_neg,a_neg,cmax_neg_sim,L_neg]
- pos_voltage = plotter.gen_half_cell_voltage(edge_conc_vals_pos,i_app_data_pos,electrode_pos,tsteps,pos↩ _params=posparams)
- neg_voltage = plotter.gen_half_cell_voltage(edge_conc_vals_neg,i_app_data_neg,electrode_neg,tsteps,neg↩ _params=negparams)
- full_voltage = pos_voltage-neg_voltage
- c0_dist = st.norm(1000.0, 25.0)

  Define the input parameter distributions #.
- D_dist = st.norm(4.0e-15, 0.5e-15)
- R_dist = st.norm(5.86e-6, 1.0e-6)
- a_dist = st.norm(3.0∗0.665/5.86e-6, 1.0e5)
- L_dist = st.norm(75.6e-6, 5.0e-6)
- int num_samples = 40
- c0_samples = c0_dist.rvs(num_samples)
- D_samples = D_dist.rvs(num_samples)
- R_samples = R_dist.rvs(num_samples)
- a_samples = a_dist.rvs(num_samples)
- L_samples = L_dist.rvs(num_samples)
- fixed_c0 = c0_dist.mean()
- fixed_D = D_dist.mean()
- fixed_R = R_dist.mean()
- fixed_a = a_dist.mean()
- fixed_L = L_dist.mean()
- int counter = 0
- dt
- time = np.linspace(0.0,dt∗(tsteps-1.),tsteps)

_, _, tsteps = UI.iapp_read_csv(iapp_filename)

- voltage_curves = np.zeros(tsteps)
- fig

  UQ Plot #.
- axs
- figsize
- sharex
- True
- sharey
- params = np.array([c0_samples[i], fixed_D, fixed_R, fixed_a, fixed_L])
- alpha
- figg

  Input Distributions #.
- axss
- False
- int resol = 1000
- x = np.linspace(900, 1100, resol)

7.13 /home/chem/msuvnj/PX915_GroupB_22-23/unit_test.py File Reference

Namespaces

- unit_test

  Function that runs a unit test with a provided number of nodes, a provided timestep and a provided number of cores.

Functions

- def run_unit_test (nodenum=500, dt=1.0, num_cores=1)

7.14 /home/chem/msuvnj/PX915_GroupB_22-23/user_input.py File Reference

Namespaces

- user_input

  Sets up the user inputs and executes the solver.

Variables

- bool checkpoint = False
- string solver_input_filename = 'user_input'
- tsteps

  SET VALUES #########.
- dt
- n
- c0
- D
- R
- a
- L
- iapp
- iapp_label
- electrode_charge
- float K_pos = 3.42E-6
- float K_neg = 6.48E-7
- float cmax_pos_sim = 63104.00
- float cmax_neg_sim = 33133.00
- list plot_params_pos = [K_pos,a,cmax_pos_sim,L]

  END SET VALUES #########.
- pos_params

7.15 /home/chem/msuvnj/PX915_GroupB_22-23/user_input_full_battery_GITT.py File Reference

Namespaces

- user_input_full_battery_GITT

  Example script for a full battery model.

Variables

- string output_filename_positive = 'current_step_pos'

  Uncomment to print stdout to file sys.stdout = open('test.txt', 'w')
- string output_filename_negative = 'current_step_neg'
- tsteps

  Import default values from https://doi.org/10.1149/1945-7111/ab9050 Use UI.set_defaults_pos() for positive electrode and ..._neg() for negative electrode.
- dt = 1.0
- n = 200
- c0
- D_neg
- R_neg
- a_neg
- L_neg
- iapp_neg
- iapp_label_neg

- electrode_charge_neg
- D_pos
- R_pos
- a_pos
- L_pos
- iapp_pos
- iapp_label_pos
- electrode_charge_pos
- float c0_pos = 30000.0

  Set values ######.
- float c0_neg = 1.0
- float K_pos = 3.42E-6
- float K_neg = 6.48E-7
- float cmax_pos_sim = 63104.00
- float cmax_neg_sim = 33133.00
- int nprocs = 40

  Check parameters are valid ######.
- int nsteps = 10
- list currents = [20.0 for i in range(nsteps)]
- list start_times = [2150.0∗i for i in range(nsteps)]
- list run_times = [150.0 for i in range(nsteps)]
- list wait_times = [2000.0 for i in range(nsteps)]
- list params_pos = [dt, c0_pos, D_pos, R_pos, a_pos, L_pos, electrode_charge_pos]
- list params_neg = [dt, c0_neg, D_neg, R_neg, a_neg, L_neg, electrode_charge_neg]
- list plot_params_pos = [K_pos,a_pos,cmax_pos_sim,L_pos]

  Call the function to perform the full battery parallel solve.
- list plot_params_neg = [K_neg,a_neg,cmax_neg_sim,L_neg]
- pos_params
- neg_params
- SparsifyAnimation
- True
- animation_interval_time

7.16 /home/chem/msuvnj/PX915_GroupB_22-23/user_input_mod.py File Reference

Namespaces

- user_input_mod

  Package containing the functions needed to set up the user input parameters and execute the solver.

Functions

- def iapp_read_csv (filename)

  CURRENT DENSITY SET UP ###.
- def iapp_constant_setup (tsteps, iapp)

  Sets up the applied current density 'iapp' as a constant valued array of length tsteps.
- def iapp_step_setup (tsteps, iapp_steps)

  Sets up the applied current density 'iapp' as a stepped array of length tsteps.
- def set_defaults_pos ()

DEFAULT PARAMETERS ###.

- def set_defaults_neg ()

  Returns default parameters for a negative electrode.
- def verify_params (filename, tsteps, dt, n, c0, D, R, a, L, electrode_charge)

  PARAMETER VERIFICATION ###.
- def verify_iapp (iapp, iapp_label, tsteps)

  Verifies the types and values of the applied current density array and its label.
- def write_to_file (filename, tsteps, dt, n, c0, D, R, a, L, iapp, iapp_label, electrode_charge)

  WRITE TO FILE ###.
- def call_solver (filename, checkpoint, nprocs=1)

  CALL SOLVER ###.
- def get_GITT_initial_concs (currents, run_times, c0, R, a, L, electrode_charge)
- def GITT_half_cell (filename, nprocs, currents, start_times, run_times, wait_times, n, params)

  INITIALISE A FULL GITT TEST IN PARALLEL ####.
- def GITT_full_cell (filename_positive, filename_negative, nprocs, currents, start_times, run_times, wait_↩ times, n, params_pos, params_neg)

  Executes the SPM solver in parallel to run a GITT full cell test over nprocs cores.
- def full_battery_simulation (filename_positive, filename_negative, nprocs=1)

  Executes the SPM solver.
- def optimise_parallelism (n_procs, n_gitt_steps=1, full_battery=False)

  Decides the correct number of threads to use for a given simulation.

7.17 /home/chem/msuvnj/PX915_GroupB_22-23/user_input_parallel_↩ simulation.py File Reference

Namespaces

- user_input_parallel_simulation

  a running script which exists to demonstrate how one uses the GITT function to run simulations in parallel.

Variables

- string output_filename = 'current_step'
- string electrode = 'positive'
- tsteps
- dt = 1.0
- n
- c0 = 1.0
- D
- R
- a
- L
- iapp
- iapp_label
- electrode_charge
- float K_pos = 3.42E-6
- float K_neg = 6.48E-7
- float cmax_pos_sim = 63104.00

- float cmax_neg_sim = 33133.00
- int nprocs = 5
- int nsteps = 5
- list currents = [20.0 for i in range(nsteps)]
- list start_times = [2150.0∗i for i in range(nsteps)]
- list run_times = [150.0 for i in range(nsteps)]
- list wait_times = [2000.0 for i in range(nsteps)]
- list params = [dt, c0, D, R, a, L,electrode_charge]
- list plot_params_neg = [K_neg,a,cmax_neg_sim,L]
- neg_params
- Animation
- True
- SparsifyAnimation