Functions

Commands from tools.py:

print\_heading (MyConstraints):

Prints the constraint values used for the interventions applied.

MyConstraints = keyword

print\_output (alloc, cost, linenum=""):

Makes an array from the alloc given from the interventions allocation percentages and cost associated with each allocation. The number of lines printed to the output is based on the value of linenum.

alloc = values

cost = values

linenum = “” (Default, prints all lines)

= number (Prints until that line number is reached)

calc\_interventions (alloc, OrigParams, MyConstraints):

Returns the ModifiedParams results taken from the StochCalc.ModelParamList . Calculates the values for the interventions to use in the analysis based on the parameters called for computation. Constraint information is taken from MyConstraints and OrigParams in StochLib.pyModelParams() is taken to generate ModifiedParams. Parameters used are "theta\_1", "delta\_1", "delta\_2" which correspond to xxx.

alloc = values

OrigParams = list of parameters

MyConstraints = keyword

Commands from constraints.py:

constraints\_help():

Returns the parameters used. A help file to describe the meanings of the parameters applied, the allowable range of values for each constraint and acceptable formatting. Takes no arguments.

"theta\_1" = "fraction of infected cases diagnosed and hospitalized", value < 10

"beta\_H" = "contact rate for hospitalized cases", 1 < value < 17

"delta\_2" = "fatality rate of hospitalized patients", 2 < value < 40

setup\_constraints (filename, valid\_interventions):

Returns all the constraints to be used for subsequent analysis. It checks to make sure that valid constraints are selected and used as input for the analysis.

MyConstraints = keyword

valid\_interventions = array listing of all interventions to be applied

Commands from modelfit.py:

parse\_data (filename, country):

Returns an array listing of the day and cases associated with that particular day for s specific country. The function takes the directory listing of the file path of the raw data csv file and extracts the number of cases versus time for a given country. Requirement: country name should match the string in the .csv file.

filename = input file with country string header, must be in .csv format

days = array listing for the specific day containing data on the number of cases

cases = array listing containing the number of cases reported

fit\_params (data\_file, country, N, plot\_fit=False):

Returns an array listing containing the list of parameter for a specific country.

N= array listing of parameters from Legrand paper, values must be float

days = array listing for the specific day containing data on the number of cases

cases = array listing containing the number of cases reported

SIRode (P, t, N, betaI, betaH, betaF, alpha, gammah, gammadh, gammaf, gammai, gammad, gammaih, theta1, delta1, delta2)

Returns an interpolated value based on the specific fit ordinary differential equation (ODE) equation. The ODE equation is then integrated to generate discrete values for the time series data taken from the array listing file containing days and cases. All parameters listed for this equation are consistent with the parameters used in the Legrand paper.

N= array listing of parameters from Legrand paper, values must be float

LLode(x):

Returns OrigParams. The parameters are first initialized using guesses of the values. The guessed values are then fit with the data, integrated, and then the error of the initial guesses are minimized. Optimal parameters are then generated for use in the computation after the process is iterated until the error is below one percent. All parameter definitions are consistent with the Legrand paper.

N= array listing of parameters from Legrand paper, values must be float

OrigParams = list of parameters

Commands from run\_simulations.py:

check\_total(MyConstraints, OrigParams):

Returns an array listing of needed\_resources. The function checks to make sure that the total given is not so large that optimization is pointless.

total = value, must be less than 100,000.

needed\_resources = array listing of parameters for optimization

OrigParams = list of parameters

MyConstraints = keyword

setup\_stoch\_params(N\_samples, trajectories, t\_final, N, I\_init):

Returns an array listing of StochParams. This function initializes the parameters for optimization run from the Stochpy library of parameters generated from the stochastic computation previously done. All paramaters defined here are consistent with the Legrand paper.

N\_samples = number of times to sample the optimization run to query results

trajectories = value

run\_no\_interventions(OrigParams, StochParams, out\_file, n\_threads=1):

Returns cost when there have been no interventions applied to the model. A stochastic analysis is then performed using the input arguments given and the result generated is the cost associated with a specific intervention applied.

alloc = values

cost = value

OrigParams = list of parameters

MyConstraints = keyword

StochParams = list of parameters from the stochastic analysis

n\_threads=1 (Number of processors to use, OpenMP Parallelization)

out\_file = “NONE” (Default), other option generates and output text file

t\_interventions = value, must be at least zero

run\_with\_interventions (alloc, OrigParams, StochParams, MyConstraints, out\_file, n\_threads=1):

Returns cost when interventions have been applied to the model. A stochastic analysis is then performed using the input arguments given and the result generated is the cost associated with a specific intervention applied. An array printout of MyConstraints and resource allocation with cost values are generated for output into out\_file.

alloc = values

cost = value

OrigParams = list of parameters

MyConstraints = keyword

StochParams = list of parameters from the stochastic analysis

n\_threads=1 (Number of processors to use, OpenMP Parallelization)

out\_file = “NONE” (Default), other option generates and output text file

t\_interventions = value, must be at least zero

run\_optimization (OrigParams, StochParams, MyConstraints, disp, out\_file, n\_threads=1):

Returns the optimized final\_cost and resource allocation associated with the final\_cost. This function computes the final\_cost values after optimization has been performed based on the parameters given from StochParams. Error handling is performed for values that do not correspond to cases where optimization is not needed.

alloc = values

cost = value

final\_cost = value generated after optimization is performed if needed

OrigParams = list of parameters

MyConstraints = keyword

disp=False (Default)

= True (generates the plot profile in a pop-out window)

StochParams = list of parameters from the stochastic analysis

n\_threads=1 (Number of processors to use, OpenMP Parallelization)

out\_file = “NONE” (Default), other option generates and output text file

t\_interventions = value, must be at least zero

Commands from \_\_init\_\_.py:

get\_data\_path(path):

Returns the path directory. The data\_file\_default and constraints\_file\_default are also generated for their respective directories.

path = directory listing of the files

setup\_model(data\_file=data\_file\_default, constraints\_file=constraints\_file\_default,

plot\_fit=True, N\_samples=200, trajectories=20, t\_final=250., N=200000,

I\_init=3, valid\_interventions='all', country="Sierra Leone"):

Returns params (list of parameters based on the specific country selected).

data\_file

constraints\_file

plot\_fit=True (Default, data fitting applied)

=False (data fitting is ignored)

N\_samples = number of times to sample the optimization run to query results

trajectories = value

N= array listing of parameters from Legrand paper, values must be float

valid\_interventions = array listing of all interventions applicable, Default = ‘all’

country = specified country based on data, Default = “Sierra Leone”

= other options: “Liberia”, “Guinea”

t\_final = value

I\_init = value

OrigParams = list of parameters

MyConstraints = keyword

disp=False (Default)

= True (generates the plot profile in a pop-out window)

StochParams = list of parameters from the stochastic analysis

n\_threads=1 (Number of processors to use, OpenMP Parallelization)

out\_file = “NONE” (Default), other option generates and output text file

t\_interventions = value, must be at least zero

optimize\_with\_setup (params, disp=True, out\_noiv\_file="out\_noiv.csv", \

out\_iv\_file="out.csv", figure\_file="out.png", plot=True, \

n\_threads=1):

OrigParams, StochParams, MyConstraints = params

cost\_noiv = run\_no\_interventions(OrigParams, StochParams, out\_noiv\_file, \

n\_threads=n\_threads)

xmin, final\_cost = run\_optimization(OrigParams, StochParams, MyConstraints, \

disp, out\_iv\_file, n\_threads=n\_threads)

if plot:

plot\_output(out\_noiv\_file, out\_iv\_file, figure\_file)

return xmin, final\_cost

def run\_simulation\_with\_setup(alloc, params, out\_noiv\_file="out\_noiv.csv", \

out\_iv\_file="out.csv", figure\_file="out.png", plot=True, \

n\_threads=1):

OrigParams, StochParams, MyConstraints = params

cost\_noiv = run\_no\_interventions(OrigParams, StochParams, out\_noiv\_file, \

n\_threads=n\_threads)

final\_cost = run\_with\_interventions(alloc, OrigParams, StochParams, \

MyConstraints, out\_iv\_file, n\_threads=n\_threads)

if plot:

plot\_output(out\_noiv\_file, out\_iv\_file, figure\_file)

return final\_cost

######################

def optimize(disp=True, out\_noiv\_file="out\_noiv.csv", \

out\_iv\_file="out.csv", figure\_file="out.png", plot=True, \

n\_threads=1, \*\*kwds):

params = setup\_model(\*\*kwds)

xmin, final\_cost = optimize\_with\_setup(params, disp=disp, out\_noiv\_file=out\_noiv\_file, \

out\_iv\_file=out\_iv\_file, figure\_file=figure\_file, plot=plot, \

n\_threads=n\_threads)

return xmin, final\_cost

def run\_simulation(alloc, disp=True, out\_noiv\_file="out\_noiv.csv", \

out\_iv\_file="out.csv", figure\_file="out.png", plot=True, \

n\_threads=1, \*\*kwds):

params = setup\_model(\*\*kwds)

final\_cost = run\_simulation\_with\_setup(alloc, params, out\_noiv\_file=out\_noiv\_file, \

out\_iv\_file=out\_iv\_file, figure\_file=figure\_file, \

plot=plot, n\_threads=n\_threads)

return final\_cost

Class Objects

Commands from const\_function.py:

CostFunction:

Returns a display of print out of resource allocation and cost in real-time computation if disp=True. A callable object that must be minimized as part of the optimization computation based on the interventions, associated costs, and resource allocation.

OrigParams = list of parameters

StochParams = list of parameters from the stochastic analysis

MyConstraints = keyword

disp=False (Default)

= True (generates the plot profile in a pop-out window)

n\_threads=1 (Number of processors to use, OpenMP Parallelization)

Log\_list = array listing during computation, Default = []

alloc = values

cost = values

out\_file = “NONE” (Default), other option generates and output text file

Commands from constraints.py:

Constraints(filename):

A callable object to hold optimization parameters: total resources, resource costs and effects, and time to start interventions.

filename = input file to parse the parameters

total = value, must be a positive value

cost = value, cannot be negative

t\_interventions = value, must be at least zero

all\_interventions = array listing of 'beta\_H', 'delta\_2', 'theta\_1