Using Python and PySD library to conduct sensitivy analysis in Vensim models

Robinson Salazar Rua and Peter Hovmand

January 1, 2024

1 Install the packages

This step involves setting up the Python environment with necessary libraries like `pandas`, `numpy`, `matplotlib`, `pysd`, and using `pip`. It's crucial for accessing various tools and functions for data manipulation and visualization.

[1]:
```
%%capture
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install pysd
!pip install netCDF4
```

2 Load the libraries

After installing the packages, let's import these libraries into our script. This enables the use of their functionalities, essential for data processing and analysis.

[2]:
```
import os
import zipfile
import itertools
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pysd
```

3 Set the working directory

This is a fundamental step for file management in Python, ensuring that scripts can find and store data correctly.

[3]:
```
os.chdir("C:/Users/rss188/Desktop/Python_ChickenModel")
os.getcwd()
```

[3]:
```
'C:\\Users\\rss188\\Desktop\\Python_ChickenModel'
```

4 Unzip model and data

This segment teaches how to extract data from a zip file into the working directory.

[4]:
```
# Path to the zip file
zip_file_path = 'C:/Users/rss188/Desktop/Python_ChickenModel/SFD chicken model␣
 ↪V2.zip'

# Directory where you want to extract the contents
extract_to_folder = 'C:/Users/rss188/Desktop/Python_ChickenModel'  # Replace␣
 ↪with your desired path

# Create the directory if it doesn't exist
os.makedirs(extract_to_folder, exist_ok=True)

# Unzip the file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_to_folder)

print(f"Extracted contents of {zip_file_path} to {extract_to_folder}")
```

```
Extracted contents of C:/Users/rss188/Desktop/Python_ChickenModel/SFD chicken
model V2.zip to C:/Users/rss188/Desktop/Python_ChickenModel
```

5 Read the original model and print the results

This section is essential for executing System Dynamics models in Python, focusing on loading the model from a file and outputting initial and final simulation results for analysis.

[5]:
```
# Import the pysd library to handle System Dynamics models
import pysd

# Load the Vensim model from the specified file path
model = pysd.read_vensim("C:/Users/rss188/Desktop/Python_ChickenModel/SFD␣
 ↪chicken model V2.mdl")

# Run the simulation on the loaded model
results = model.run()

# Print the first few rows of the simulation results
print(results.head())
```

```
   FINAL TIME  INITIAL TIME  SAVEPER  TIME STEP  Births  Chickens  \
0          50             0        1          1   200.0    1000.0
1          50             0        1          1   200.0    1000.0
2          50             0        1          1   200.0    1000.0
3          50             0        1          1   200.0    1000.0
4          50             0        1          1   200.0    1000.0
```

```
   Cross roading  Death risk  Deaths  Egg production    Eggs  \
0            1.0         0.2   200.0           200.0  1000.0
1            1.0         0.2   200.0           200.0  1000.0
2            1.0         0.2   200.0           200.0  1000.0
3            1.0         0.2   200.0           200.0  1000.0
4            1.0         0.2   200.0           200.0  1000.0

   Fertility effect  Incubation time  Initial chicken population  \
0               0.2                5                        1000
1               0.2                5                        1000
2               0.2                5                        1000
3               0.2                5                        1000
4               0.2                5                        1000

   Initial number of eggs  Max chicken capacity  Normal death risk  \
0                    1000                  1000                0.2
1                    1000                  1000                0.2
2                    1000                  1000                0.2
3                    1000                  1000                0.2
4                    1000                  1000                0.2

   Normal fertility rate  Stress
0                    0.2     1.0
1                    0.2     1.0
2                    0.2     1.0
3                    0.2     1.0
4                    0.2     1.0
```

5.1 Set new conditions and print the results

This segment is crucial for modifying simulation parameters in a System Dynamics model and displaying the updated outcomes, enabling a tailored analysis of different scenarios.

[6]:
```
# Load the System Dynamics model from a specified Vensim file
model = pysd.read_vensim("C:/Users/rss188/Desktop/Chicken model/SFD chicken␣
 ↪model/SFD chicken model V2.mdl")

# Define new parameters to alter the model's behavior
params = {'Normal fertility rate': 0.3, 'Normal death risk': 0.2}

# Specify which variables to include in the simulation results
return_columns = ['Eggs', 'Chickens', 'Egg production', 'Births', 'Deaths']

# Define the range of simulation time steps to be returned
return_timestamps = range(51)

# Run the model simulation with the specified parameters and settings
```

```
results = model.run(params=params, return_columns=return_columns,␣
 ↪return_timestamps=return_timestamps, initial_condition='current')

# Print the first few rows of the simulation results for initial insight
print(results.head())
```

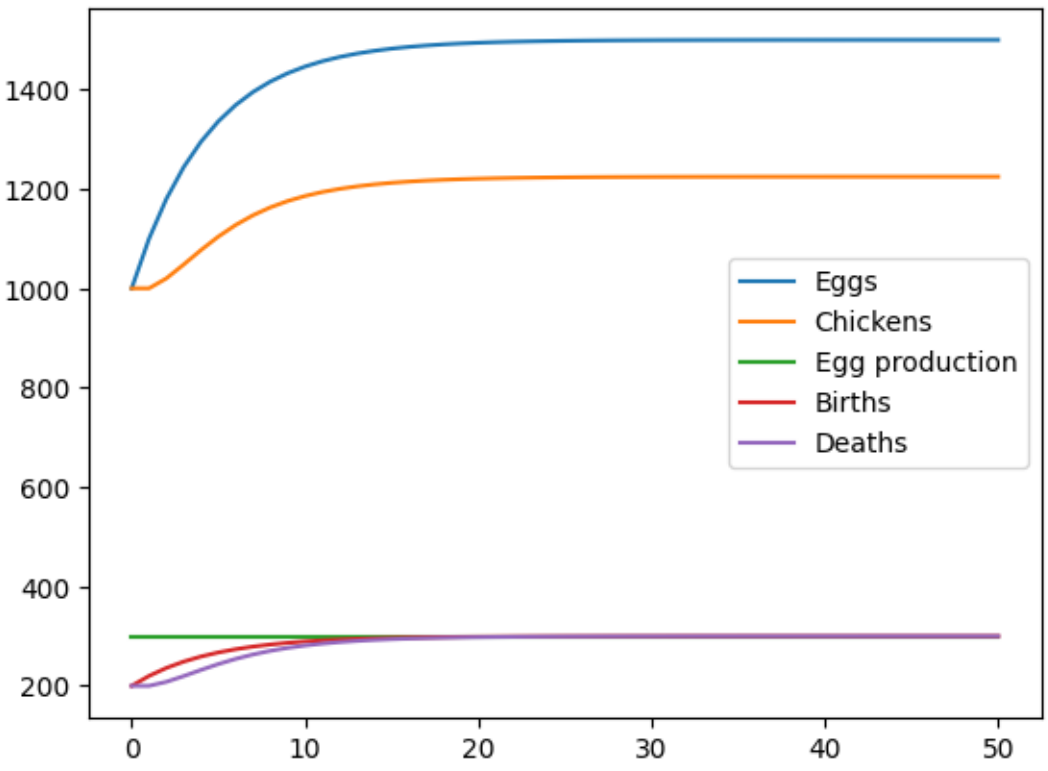
```
     Eggs     Chickens  Egg production  Births      Deaths
0  1000.0  1000.000000           300.0  200.00  200.000000
1  1100.0  1000.000000           300.0  220.00  200.000000
2  1180.0  1020.000000           300.0  236.00  208.080000
3  1244.0  1047.920000           300.0  248.80  219.627265
4  1295.2  1077.092735           300.0  259.04  232.025752
```

5.2 Plot the results from the new conditions

This section is dedicated to visualizing the outcomes of the System Dynamics model under altered conditions, providing a graphical representation of the simulation results for enhanced analysis and understanding.

[7]:
```
# Generate a plot of the simulation results using the new conditions
results.plot()
```

[7]: <Axes: >



6 First sensitivy analysis (Normal fertility rate)

This segment focuses on conducting the first sensitivity analysis of the System Dynamics model, specifically examining how varying the 'Normal fertility rate' parameter influences the overall model behavior and outcomes. This analysis helps in understanding the impact of changes in fertility rate on the system's dynamics.

[8]:
```
# Define a range of values for 'Normal Fertility Rate' to test in the␣
 ↪sensitivity analysis
Fertility_Rate_Values = np.linspace(start=0.2, stop=1, num=5)

# Initialize an empty DataFrame to store results of each simulation run
results = pd.DataFrame()

# Loop through each fertility rate value
for rate in Fertility_Rate_Values:

    # Run the model simulation with the current fertility rate
    single_run_result = model.run(params={'Normal Fertility Rate': rate})

    # Store the 'Chickens' population results in the DataFrame with the rate as␣
 ↪the column name
    results[str(round(rate, 2))] = single_run_result['Chickens']

# Print the first few rows of the results DataFrame for an initial overview
print(results.head())
```

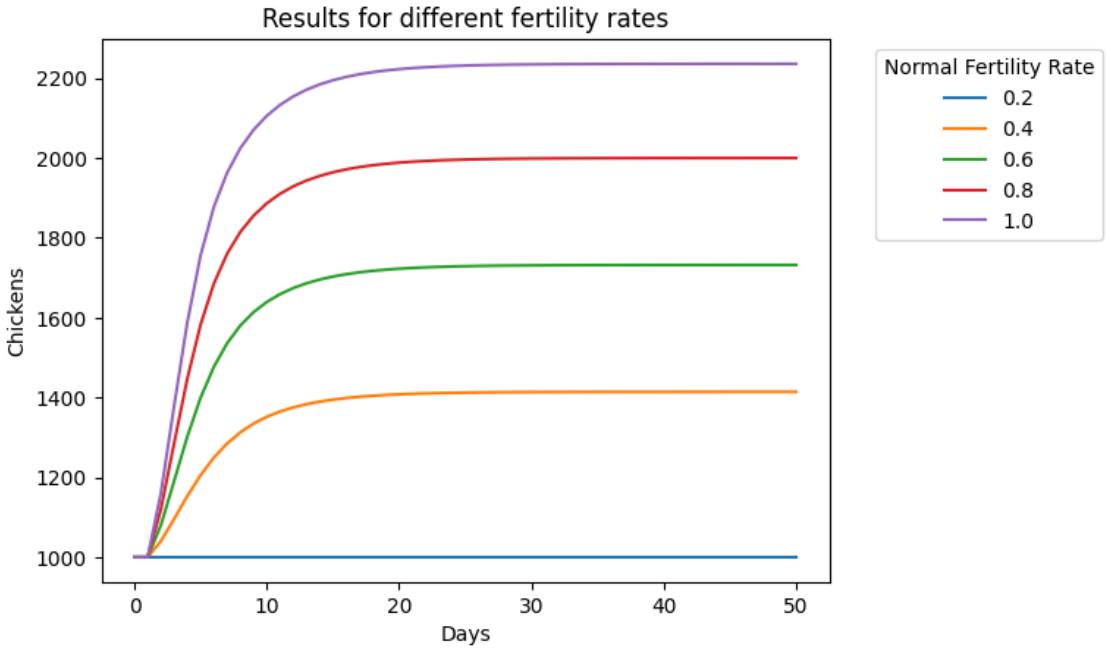
```
     0.2          0.4          0.6          0.8          1.0
0  1000.0  1000.000000  1000.000000  1000.000000  1000.000000
1  1000.0  1000.000000  1000.000000  1000.000000  1000.000000
2  1000.0  1040.000000  1080.000000  1120.000000  1160.000000
3  1000.0  1095.680000  1190.720000  1285.120000  1378.880000
4  1000.0  1153.177068  1302.357176  1447.613317  1589.017989
```

6.1 Plot the results from the first sensitivity analysis

This section is centered on visualizing the data obtained from the first sensitivity analysis, which focused on the 'Normal fertility rate' parameter. It involves creating plots to illustrate how varying this rate impacts the model's outcomes, offering a clear graphical representation for easier interpretation and comparison of results.

[9]:
```
# Generate a plot of the simulation results with different fertility rates
results.plot()

# Set the label for the x-axis as 'Days'
plt.xlabel('Days')

# Set the label for the y-axis as 'Chickens'
```

```
plt.ylabel('Chickens')

# Add a title to the plot
plt.title('Results for different fertility rates')

# Add a legend to the plot, with a title, and position it outside the plot area
plt.legend(title='Normal Fertility Rate', bbox_to_anchor=(1.05, 1), loc='upper␣
 ↪left')

# Display the plot
plt.show()
```



7 Second sensitivy analysis (Normal fertility rate and Normal death risk)

This segment conducts a second, more comprehensive sensitivity analysis on the System Dynamics model, simultaneously examining the effects of variations in two key parameters: 'Normal fertility rate' and 'Normal death risk'. This dual-parameter analysis aims to understand the interplay and combined impact of these factors on the model's behavior and outcomes, providing deeper insights into the system's dynamics.

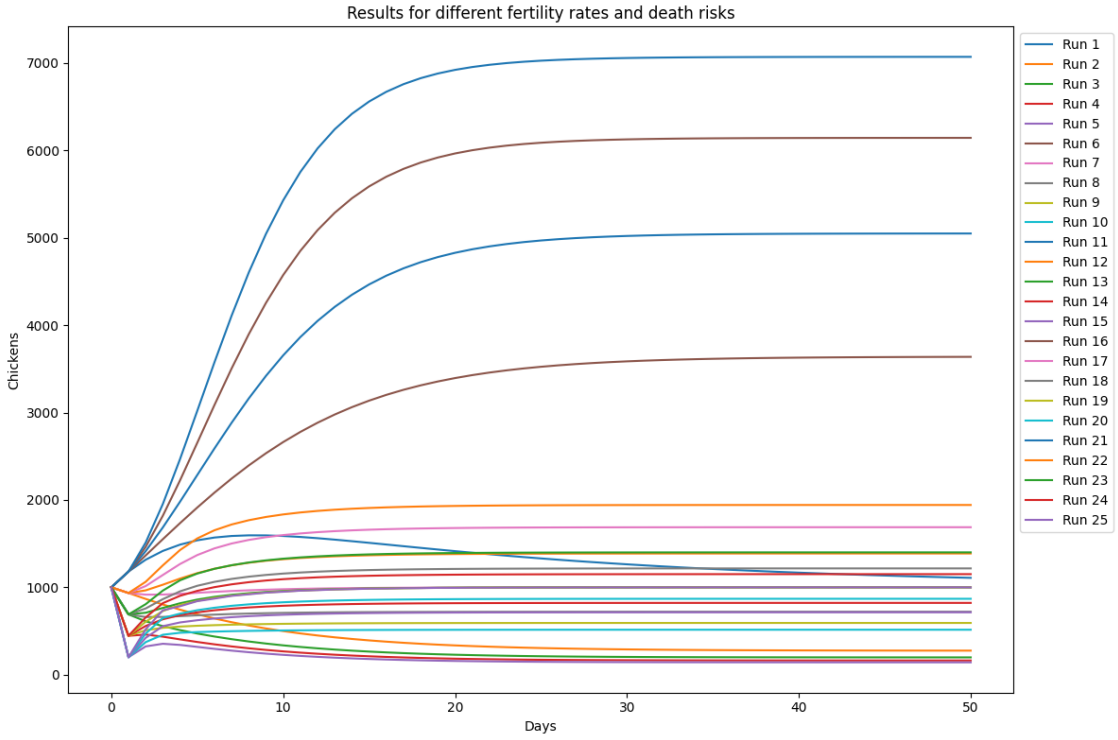
[10]:
```
# Define a range of values for 'Normal Fertility Rate' and 'Normal Death Risk'␣
 ↪for sensitivity analysis
Fertility_Rate_Values = np.linspace(start=0.02, stop=1, num=5)
```

```
Death_Risk_Values = np.linspace(start=0.02, stop=1, num=5)

# Create all possible combinations of Fertility Rate and Death Risk values
parameter_combinations = list(itertools.product(Fertility_Rate_Values,␣
 ↪Death_Risk_Values))

# Initialize a list to store results of each simulation run
results_list = []

# Initialize a variable to track the number of each simulation run
run_number = 1

# Loop through each combination of Fertility Rate and Death Risk
for Fertility_Rate, Death_Risk in parameter_combinations:

    # Run the model with the current combination of parameters
    single_run_result = model.run(params={'Normal Fertility Rate':␣
 ↪Fertility_Rate, 'Normal Death Risk': Death_Risk})

    # Append selected results and parameters to the results list with the␣
 ↪current run number
    results_list.append({
        'Run Number': f'Run {run_number}',
        'Normal Fertility Rate': Fertility_Rate,
        'Normal Death Risk': Death_Risk,
        'Eggs': single_run_result['Eggs'],
        'Chickens': single_run_result['Chickens']
    })

    # Increment the run number for the next iteration
    run_number += 1

    # Delete the single run result to free up memory
    del single_run_result

# Convert the list of results into a DataFrame
results = pd.DataFrame(results_list)

# Define the file path to save the results as a CSV file
csv_file_path = r'C:/Users/rss188/Desktop/Chicken model/SFD chicken model/
 ↪model_results.csv'

# Save the results DataFrame to a CSV file
results.to_csv(csv_file_path, index=False)
```

7.1 Plot the results from the second sensitivity analysis

This section involves creating visual representations of the data obtained from the second sensitivity analysis, which explores the combined effects of varying both 'Normal fertility rate' and 'Normal death risk'. The focus is on graphically illustrating how these dual-parameter changes impact the model's behavior, facilitating a more nuanced understanding of the interactions between these two key factors.

[11]:
```
# Create a figure for plotting with specified dimensions (width 12 inches,␣
 ↪height 8 inches)
plt.figure(figsize=(12, 8))

# Iterate through each row in the 'results' DataFrame
for index, row in results.iterrows():
    # For each row, plot the 'Chickens' data, using the 'Run Number' as the␣
 ↪label for each line
    plt.plot(row['Chickens'], label=row['Run Number'])

# Set the label for the x-axis to 'Days'
plt.xlabel('Days')

# Set the label for the y-axis to 'Chickens'
plt.ylabel('Chickens')

# Set the title of the plot to describe the content
plt.title('Results for different fertility rates and death risks')

# Place a legend on the plot, positioning it in the upper left corner outside␣
 ↪of the main plot area
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))

# Adjust the layout for better appearance and to fit all elements within the␣
 ↪figure bounds
plt.tight_layout()

# Display the plot
plt.show()
```



8 Third sensitivy analysis (Normal fertility rate, Normal death risk, Max chicken capacity)

This segment undertakes a third sensitivity analysis on the System Dynamics model, expanding the scope to include three key parameters: 'Normal fertility rate', 'Normal death risk', and 'Max chicken capacity'. The analysis aims to explore and understand the combined effects and interactions of these parameters on the model's behavior, providing a more comprehensive view of how these factors collectively influence the system's dynamics.

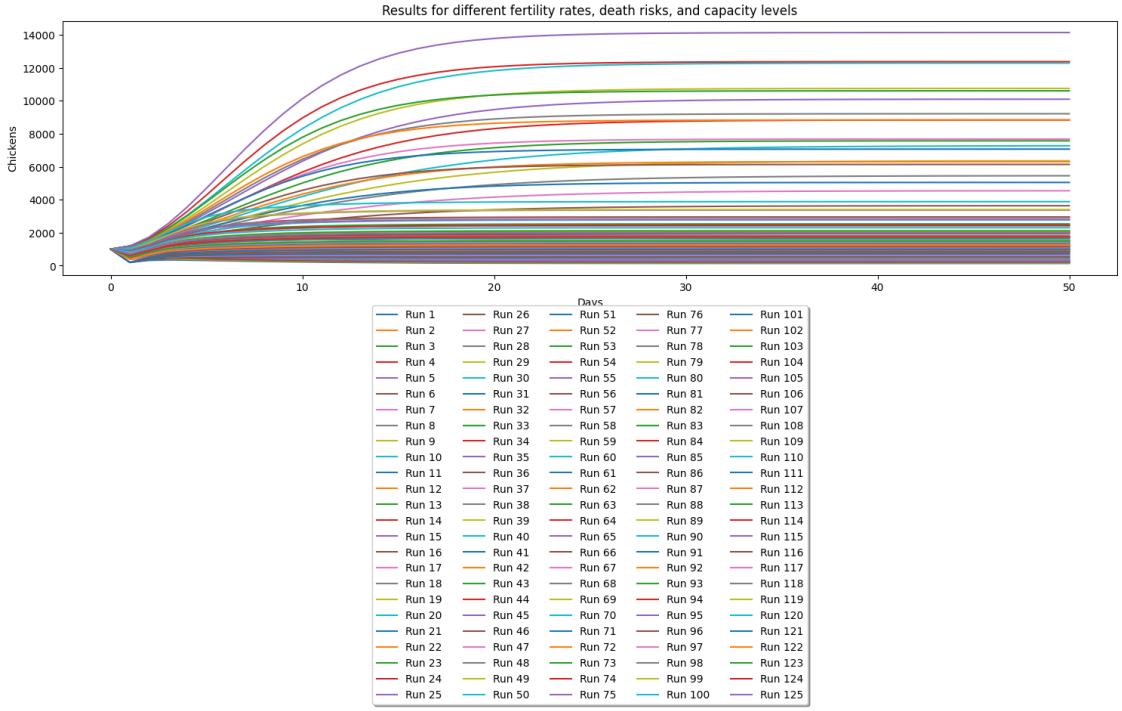
[12]:
```
# Create arrays of values for the 'Normal Fertility Rate', 'Normal Death Risk',␣
↪and 'Max Chicken Capacity' parameters
Fertility_Rate_Values = np.linspace(start=0.02, stop=1, num=5)
Death_Risk_Values = np.linspace(start=0.02, stop=1, num=5)
Chicken_Capacity_Values= np.linspace(start=1000, stop=2000, num=5)

# Generate all possible combinations of the three parameters
parameter_combinations = list(itertools.product(Fertility_Rate_Values,␣
↪Death_Risk_Values, Chicken_Capacity_Values))

# Initialize an empty list to store the results of each simulation run
results_list = []
```

```
# Set a counter to keep track of each simulation run
run_number = 1

# Iterate through each combination of the three parameters
for Fertility_Rate, Death_Risk, Chicken_Capacity in parameter_combinations:

    # Run the model simulation with the current combination of parameters and␣
 ↪for a specified range of time steps
    single_run_result = model.run(params={'Normal Fertility Rate':␣
 ↪Fertility_Rate, 'Normal Death Risk': Death_Risk, 'Max chicken capacity':␣
 ↪Chicken_Capacity}, return_timestamps=range(51))

    # Append a dictionary containing the run number, parameter values, and␣
 ↪relevant results to the results list
    results_list.append({
        'Run Number': f'Run {run_number}',
        'Normal Fertility Rate': Fertility_Rate,
        'Normal Death Risk': Death_Risk,
        'Max chicken capacity': Chicken_Capacity,
        'Eggs': single_run_result['Eggs'],
        'Chickens': single_run_result['Chickens']
    })

    # Increment the run number for the next iteration
    run_number += 1

    # Delete the result of the current simulation run to free up memory
    del single_run_result

# Convert the list of results into a DataFrame
results = pd.DataFrame(results_list)

# Define the file path to save the results as a CSV file
csv_file_path = r'C:/Users/rss188/Desktop/Chicken model/SFD chicken model/
 ↪model_results.csv'

# Save the DataFrame as a CSV file at the specified path
results.to_csv(csv_file_path, index=False)
```

8.1 Plot the results from the third sensitivity analysis

This section is dedicated to graphically presenting the outcomes from the third sensitivity analysis, which explores the interaction between 'Normal fertility rate', 'Normal death risk', and 'Max chicken capacity' in the System Dynamics model. It focuses on visually illustrating the collective impact of these three parameters, allowing for an in-depth analysis of their combined influence on the system's behavior.

[13]:
```
# Create a figure for plotting with specified dimensions (width 15 inches,␣
 ↪height 10 inches)
plt.figure(figsize=(15, 10))

# Iterate through each row in the results DataFrame
for index, row in results.iterrows():
    # For each row, plot the 'Chickens' data, using the 'Run Number' as the␣
 ↪label for each line
    plt.plot(row['Chickens'], label=row['Run Number'])

# Set the label for the x-axis to 'Days'
plt.xlabel('Days')

# Set the label for the y-axis to 'Chickens'
plt.ylabel('Chickens')

# Set the title of the plot to describe the content
plt.title('Results for different fertility rates, death risks, and capacity␣
 ↪levels')

# Place a legend on the plot with a central upper position and additional␣
 ↪styling
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), fancybox=True,␣
 ↪shadow=True, ncol=5)

# Adjust the layout for better appearance and to fit all elements within the␣
 ↪figure bounds
plt.tight_layout()

# Display the plot
plt.show()
```



9 Fourth sensitivy analysis (Normal fertility rate, Normal death risk, Max chicken capacity, Initial number of eggs, Initial number of chickens)

This segment delves into a detailed examination of the System Dynamics model by simultaneously varying five key parameters: 'Normal fertility rate', 'Normal death risk', 'Max chicken capacity', 'Initial number of eggs', and 'Initial number of chickens'. This comprehensive analysis aims to explore and understand the complex interactions and combined effects of these parameters, providing an extensive insight into how they collectively shape the model's behavior and outputs.

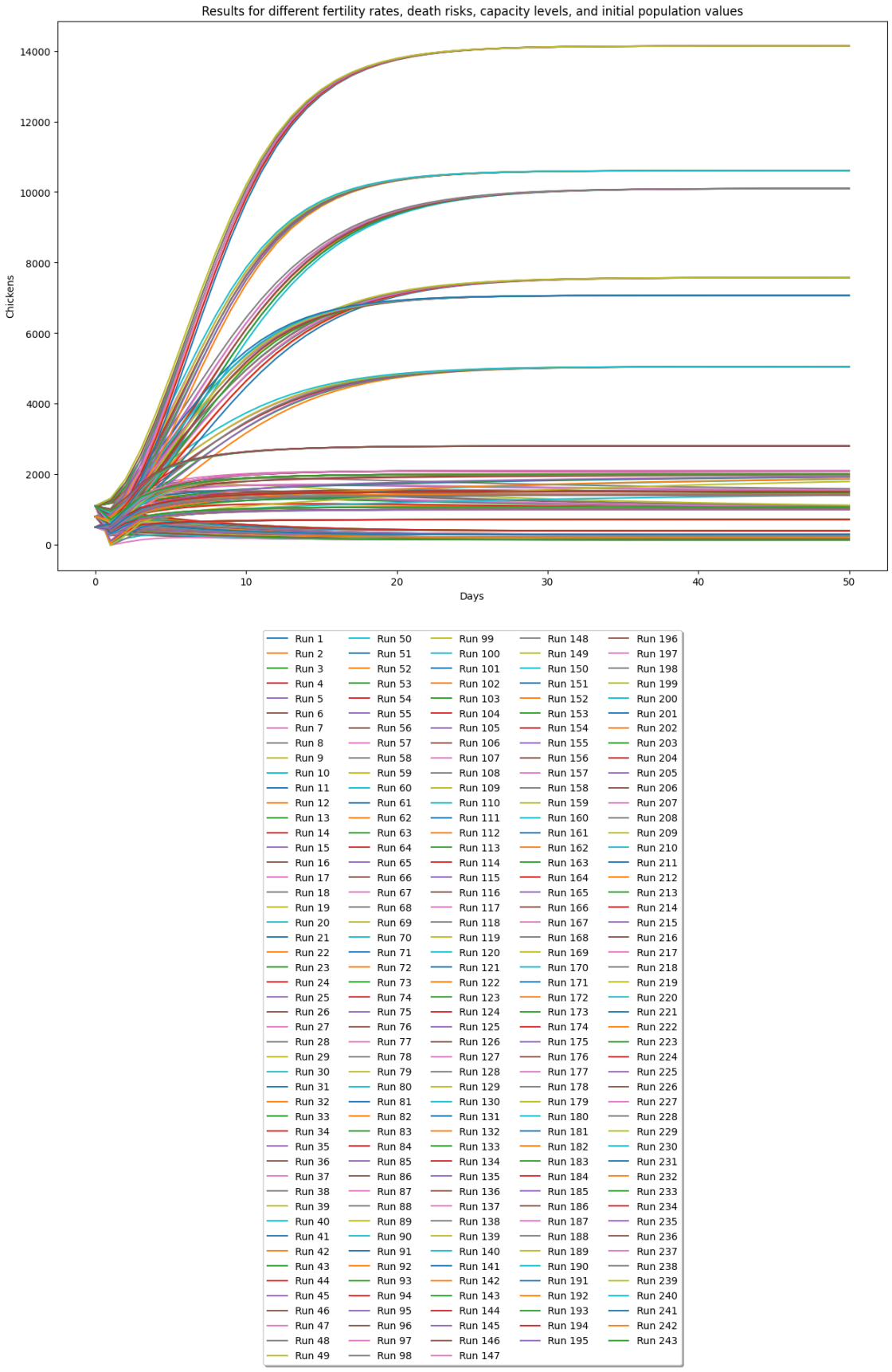
[14]:
```
# Define arrays of values for sensitivity analysis on five parameters
Fertility_Rate_Values = np.linspace(start=0.02, stop=1, num=3)
Death_Risk_Values = np.linspace(start=0.02, stop=1, num=3)
Chicken_Capacity_Values= np.linspace(start=1000, stop=2000, num=3)
Initial_Eggs_Values= np.linspace(start=500, stop=1100, num=3)
Initial_Chickens_Values= np.linspace(start=500, stop=1100, num=3)

# Generate all possible combinations of these five parameters
parameter_combinations = list(itertools.product(Fertility_Rate_Values,␣
 ↪Death_Risk_Values, Chicken_Capacity_Values, Initial_Eggs_Values,␣
 ↪Initial_Chickens_Values))

# Initialize an empty list to store the results of each simulation run
```

```
results_list = []

# Set a counter to keep track of each simulation run
run_number = 1

# Iterate through each combination of the five parameters
for Fertility_Rate, Death_Risk, Chicken_Capacity, Initial_Eggs,␣
 ↪Initial_Chickens in parameter_combinations:

    # Run the model with the current combination of parameters for a specified␣
 ↪range of time steps
    single_run_result = model.run(params={'Normal Fertility Rate':␣
 ↪Fertility_Rate, 'Normal Death Risk': Death_Risk, 'Max chicken capacity':␣
 ↪Chicken_Capacity, 'Initial number of eggs':Initial_Eggs, 'Initial chicken␣
 ↪population':Initial_Chickens}, return_timestamps=range(51))

    # Append a dictionary containing the run number, parameter values, and␣
 ↪relevant results to the results list
    results_list.append({
        'Run Number': f'Run {run_number}',
        'Normal Fertility Rate': Fertility_Rate,
        'Normal Death Risk': Death_Risk,
        'Max chicken capacity': Chicken_Capacity,
        'Initial number of eggs': Initial_Eggs,
        'Initial chicken population': Initial_Chickens,
        'Eggs': single_run_result['Eggs'],
        'Chickens': single_run_result['Chickens']
    })

    # Increment the run number for the next iteration
    run_number += 1

    # Delete the result of the current simulation run to free up memory
    del single_run_result

# Convert the list of results into a DataFrame
results = pd.DataFrame(results_list)

# Define the file path to save the results as a CSV file
csv_file_path = r'C:/Users/rss188/Desktop/Chicken model/SFD chicken model/
 ↪model_results.csv'

# Save the DataFrame as a CSV file at the specified path
results.to_csv(csv_file_path, index=False)
```

9.1 Plot the results from the fourth sensitivity analysis

This section is focused on visually presenting the outcomes of the fourth sensitivity analysis, which examines the interplay between 'Normal fertility rate', 'Normal death risk', 'Max chicken capacity', 'Initial number of eggs', and 'Initial number of chickens' in the System Dynamics model. The aim is to graphically illustrate the complex interactions and combined effects of these five parameters, thereby offering a comprehensive view of their collective impact on the system's behavior through detailed plots.

[15]:
```
# Create a figure for plotting with specified dimensions (width 15 inches,␣
 ↪height 10 inches)
plt.figure(figsize=(15, 10))

# Iterate through each row in the results DataFrame
for index, row in results.iterrows():
    # Plot the 'Chickens' data for each simulation run, using the 'Run Number'␣
 ↪as the label for each line
    plt.plot(row['Chickens'], label=row['Run Number'])

# Set the label for the x-axis to 'Days'
plt.xlabel('Days')

# Set the label for the y-axis to 'Chickens'
plt.ylabel('Chickens')

# Set the title of the plot to describe the content
plt.title('Results for different fertility rates, death risks, capacity levels,␣
 ↪and initial population values')

# Place a legend on the plot with a central upper position, adding styling␣
 ↪features
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), fancybox=True,␣
 ↪shadow=True, ncol=5)

# Display the plot
plt.show()
```


Results for different fertility rates, death risks, capacity levels, and initial population values
Chickens
Days
0
2000
4000
6000
8000
10000
12000
14000
0
10
20
30
40
50
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Run 7
Run 8
Run 9
Run 10
Run 11
Run 12
Run 13
Run 14
Run 15
Run 16
Run 17
Run 18
Run 19
Run 20
Run 21
Run 22
Run 23
Run 24
Run 25
Run 26
Run 27
Run 28
Run 29
Run 30
Run 31
Run 32
Run 33
Run 34
Run 35
Run 36
Run 37
Run 38
Run 39
Run 40
Run 41
Run 42
Run 43
Run 44
Run 45
Run 46
Run 47
Run 48
Run 49
Run 50
Run 51
Run 52
Run 53
Run 54
Run 55
Run 56
Run 57
Run 58
Run 59
Run 60
Run 61
Run 62
Run 63
Run 64
Run 65
Run 66
Run 67
Run 68
Run 69
Run 70
Run 71
Run 72
Run 73
Run 74
Run 75
Run 76
Run 77
Run 78
Run 79
Run 80
Run 81
Run 82
Run 83
Run 84
Run 85
Run 86
Run 87
Run 88
Run 89
Run 90
Run 91
Run 92
Run 93
Run 94
Run 95
Run 96
Run 97
Run 98
Run 99
Run 100
Run 101
Run 102
Run 103
Run 104
Run 105
Run 106
Run 107
Run 108
Run 109
Run 110
Run 111
Run 112
Run 113
Run 114
Run 115
Run 116
Run 117
Run 118
Run 119
Run 120
Run 121
Run 122
Run 123
Run 124
Run 125
Run 126
Run 127
Run 128
Run 129
Run 130
Run 131
Run 132
Run 133
Run 134
Run 135
Run 136
Run 137
Run 138
Run 139
Run 140
Run 141
Run 142
Run 143
Run 144
Run 145
Run 146
Run 147
Run 148
Run 149
Run 150
Run 151
Run 152
Run 153
Run 154
Run 155
Run 156
Run 157
Run 158
Run 159
Run 160
Run 161
Run 162
Run 163
Run 164
Run 165
Run 166
Run 167
Run 168
Run 169
Run 170
Run 171
Run 172
Run 173
Run 174
Run 175
Run 176
Run 177
Run 178
Run 179
Run 180
Run 181
Run 182
Run 183
Run 184
Run 185
Run 186
Run 187
Run 188
Run 189
Run 190
Run 191
Run 192
Run 193
Run 194
Run 195
Run 196
Run 197
Run 198
Run 199
Run 200
Run 201
Run 202
Run 203
Run 204
Run 205
Run 206
Run 207
Run 208
Run 209
Run 210
Run 211
Run 212
Run 213
Run 214
Run 215
Run 216
Run 217
Run 218
Run 219
Run 220
Run 221
Run 222
Run 223
Run 224
Run 225
Run 226
Run 227
Run 228
Run 229
Run 230
Run 231
Run 232
Run 233
Run 234
Run 235
Run 236
Run 237
Run 238
Run 239
Run 240
Run 241
Run 242
Run 243