Example 1

April 5, 2019

1 Discrete Markov process experimental model 1

```
In [1]: import sys

        sys.path.insert(0, "..\\..\\src")
        sys.path.insert(0, "..\\..\\src\\env")
        sys.path.insert(0, "..\\..\\src\\walker")
        sys.path.insert(0, "..\\..\\src\\model")
        sys.path.insert(0, "..\\..\\src\\model\\markov\\")
```

```
In [2]: #sys.path
```

```
In [3]: from model.markov.markov_chain_model import MarkovChainModel
        from model.markov.master_equation_integrator import MasterEquationIntegrator
```

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt
```

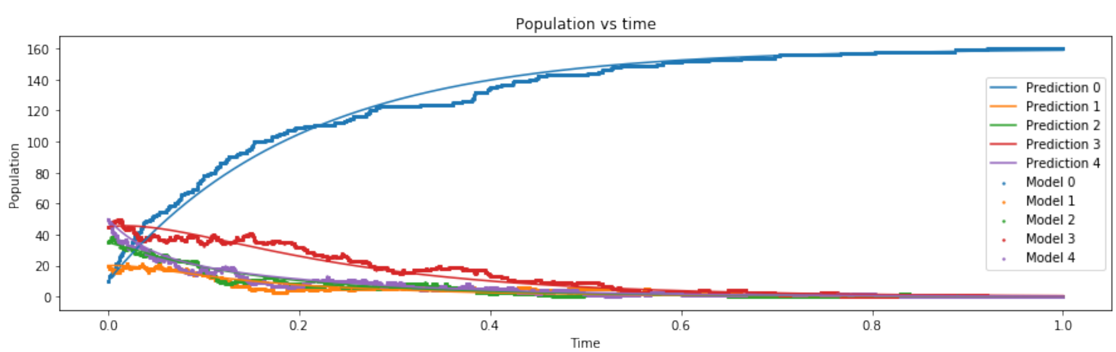
1.1 Small number of walkers

```
In [5]: population = 1 * np.array([10, 20, 35, 45, 50])

        t_1, t_2 = 0, 1
        dt = 1e-4
        time = np.arange(t_1, t_2 + dt, dt)

        model = MarkovChainModel(node_population = population, dt = dt)

        integrator = MasterEquationIntegrator()

        transition_matrix = [[0, 0, 0, 0, 0],\
                             [1, 0, 1, 1, 1],\
                             [1, 1, 0, 1, 0],\
                             [1, 0, 0, 0, 1],\
                             [1, 1, 1, 1, 0]]

        transition_matrix = np.array(transition_matrix, dtype = np.float) * 5
```

```
model.add_transition_probabilities_to_nodes_(transition_matrix)

model.run(time = t_2)

ts, arr = model.get_population_time_series(nodes = [0, 1, 2, 3, 4])

pred_t, pred_y = integrator(transition_matrix, population, (t_1, t_2), t_eval = time)
```

```
'[====================]        Progress: 100%'
```

```
In [6]: fig, ax = plt.subplots()

        fig.set_figwidth(15)

        i = 0

        for ar in arr:
            ax.scatter(ts, ar, s = 2, label = 'Model ' + str(i))
            i += 1

        i = 0

        for y in pred_y:
            ax.plot(pred_t, y, label = 'Prediction ' + str(i))
            i += 1

        ax.set_ylabel('Population')
        ax.set_xlabel('Time')
        ax.set_title('Population vs time')

        ax.legend()
        plt.show()
```




```
In [7]: model.write_population_data(path = "../../data/out/model/markov/example_11.txt")
```

1.2 10 times more walkers

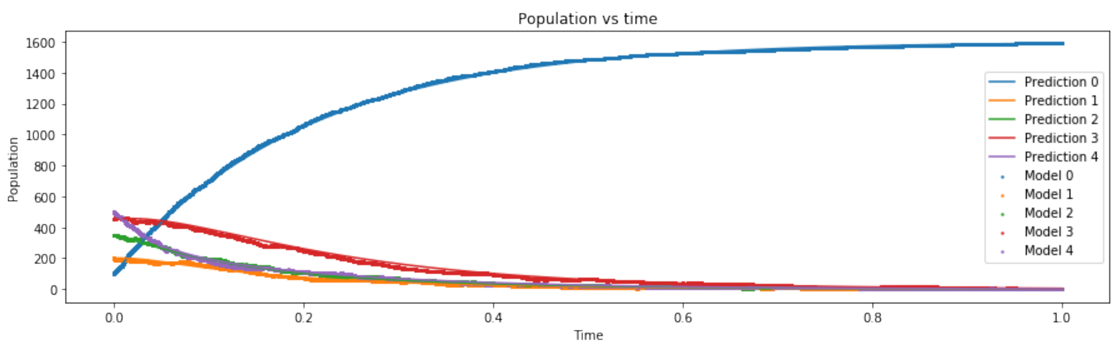
```
In [8]: population = 10 * np.array([10, 20, 35, 45, 50])

        t_1, t_2 = 0, 1
        dt = 1e-5
        time = np.arange(t_1, t_2 + dt, dt)

        model = MarkovChainModel(node_population = population, dt = dt)

        integrator = MasterEquationIntegrator()

        transition_matrix = [[0, 0, 0, 0, 0],\
                             [1, 0, 1, 1, 1],\
                             [1, 1, 0, 1, 0],\
                             [1, 0, 0, 0, 1],\
                             [1, 1, 1, 1, 0]]

        transition_matrix = np.array(transition_matrix, dtype = np.float) * 5

        model.add_transition_probabilities_to_nodes_(transition_matrix)

        model.run(time = t_2)

        ts, arr = model.get_population_time_series(nodes = [0, 1, 2, 3, 4])

        pred_t, pred_y = integrator(transition_matrix, population, (t_1, t_2), t_eval = time)
```

```
'[====================]       Progress: 100%'
```

```
In [9]: model.write_population_data(path = "../../data/out/model/markov/example_12.txt")
```

```
In [10]: fig, ax = plt.subplots()

         fig.set_figwidth(15)

         i = 0

         for ar in arr:
             ax.scatter(ts, ar, s = 2, label = 'Model ' + str(i))
             i += 1

         i = 0

         for y in pred_y:
             ax.plot(pred_t, y, label = 'Prediction ' + str(i))
             i += 1
```

```
ax.set_ylabel('Population')
ax.set_xlabel('Time')
ax.set_title('Population vs time')

ax.legend()
plt.show()
```



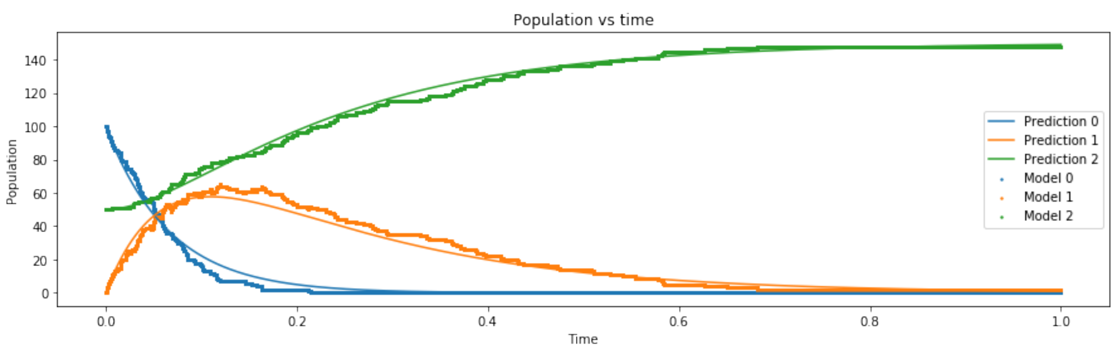
In [11]:
```
fig.savefig("../../data/out/model/markov/example_12.png")
```

1.3 Linear chain model, small number of walkers

In [12]:
```
population = 10 * np.array([10, 0, 5])

t_1, t_2 = 0, 1
dt = 1e-5
time = np.arange(t_1, t_2 + dt, dt)

model = MarkovChainModel(node_population = population, dt = dt)

integrator = MasterEquationIntegrator()

transition_matrix = [[0, 3, 0],\
                     [0, 0, 1],\
                     [0, 0, 0]]

transition_matrix = np.array(transition_matrix, dtype = np.float) * 5

model.add_transition_probabilities_to_nodes_(transition_matrix)


model.run(time = t_2)

ts, arr = model.get_population_time_series(nodes = [0, 1, 2])

pred_t, pred_y = integrator(transition_matrix, population, (t_1, t_2), t_eval = time)
```

```
'[===================.]        Progress: 99%'
```

In [13]:
```python
fig, ax = plt.subplots()

fig.set_figwidth(15)

i = 0

for ar in arr:
    ax.scatter(ts, ar, s = 2, label = 'Model ' + str(i))
    i += 1

i = 0

for y in pred_y:
    ax.plot(pred_t, y, label = 'Prediction ' + str(i))
    i += 1

ax.set_ylabel('Population')
ax.set_xlabel('Time')
ax.set_title('Population vs time')

ax.legend()
plt.show()
```



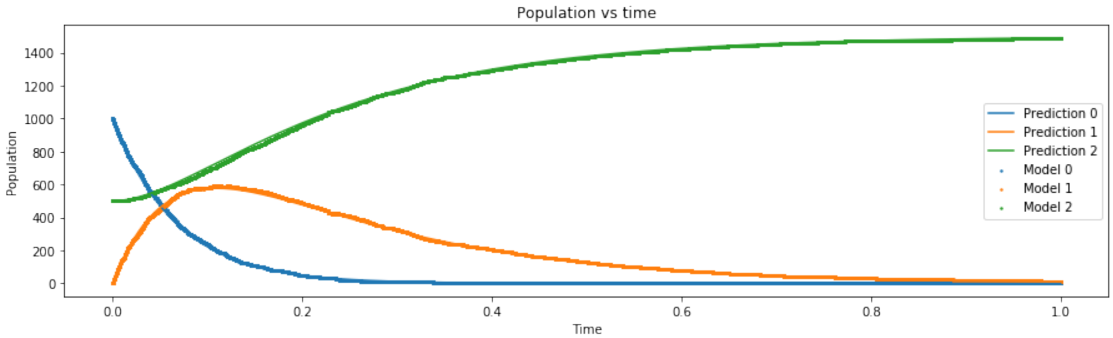
1.4 Linear chain model, 10 times more walkers

In [14]:
```python
population = 100 * np.array([10, 0, 5])

t_1, t_2 = 0, 1
dt = 1e-5
time = np.arange(t_1, t_2 + dt, dt)

model = MarkovChainModel(node_population = population, dt = dt)
```

```
integrator = MasterEquationIntegrator()

transition_matrix = [[0, 3, 0],\
                     [0, 0, 1],\
                     [0, 0, 0]]

transition_matrix = np.array(transition_matrix, dtype = np.float) * 5

model.add_transition_probabilities_to_nodes_(transition_matrix)


model.run(time = t_2)

ts, arr = model.get_population_time_series(nodes = [0, 1, 2])

pred_t, pred_y = integrator(transition_matrix, population, (t_1, t_2), t_eval = time)
```

```
'[====================]        Progress: 100%'
```

```
In [15]: fig, ax = plt.subplots()

         fig.set_figwidth(15)

         i = 0

         for ar in arr:
             ax.scatter(ts, ar, s = 2, label = 'Model ' + str(i))
             i += 1

         i = 0

         for y in pred_y:
             ax.plot(pred_t, y, label = 'Prediction ' + str(i))
             i += 1

         ax.set_ylabel('Population')
         ax.set_xlabel('Time')
         ax.set_title('Population vs time')

         ax.legend()
         plt.show()
```


Population vs time
Population
Time
Prediction 0
Prediction 1
Prediction 2
Model 0
Model 1
Model 2
1400
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0