KI PROGRAMMERING PROSJEKT 1 - JAX CONTROLLER

Kontrollerene

Standard PID Controller

Det er implementert en standard PID kontroller, som har to metoder. Control_signal metoden tar inn et error, integralet av error, og derivatet av error og kalkulerer et kontroll signal basert på tre parametere (kp, ki og kd). Den andre metoder heter update_params, hvor kontrolleren tar inn parametere (kp, ki og kd), gradienter av den forrige epoken med hensyn på parametrene og en learning_rate som bestemmer hvor mye parametrene skal oppdatere seg basert på gradientene.

NeuralNet PID Controller

Det er også implementert en NeuralNet PID kontroller, som har fire metoder. Den har en metode som generer et kontroll signal basert på vektene og brasene i nettet (predict metoden). Den har også en metode for å initiere vektene og biasene i nettverket. Til slutt har den også en metode for å oppdatere parameterne basert på sigmoid, relu eller tanh.

Modellene (plants)

Alle modeller har metode for input, output og dynamikk, som beskriver hvordan modellen oppfører seg basert på inputs. Jeg legger ved koden for dynamikken i systemene som beskrivelse på hvordan de fungerer.

Bathtub plant

```
# Simulate the noise in draining
noise = rnd.uniform(-self.noise_level, self.noise_level)
control_signal = max(min(control_signal, 50), 0)
# Calculate the new water level
if control_signal < 0:
    control_signal = 0
self.water_level += control_signal - self.drain_cross_section * ((9.81*self.water_level)**0.5) + noise
if self.water_level < 0:
    self.water_level = 0
```

Cournot plant

```
# Simulate the noise
noise = rnd.uniform(-self.noise_level, self.noise_level)
control_signal = max(min(control_signal, 1), -1)
# Calculate the new quantity
self.q0 = max(min(self.q0 + control_signal, 1), 0)
self.q1 = max(min(self.q1 + noise, 1), 0)
# Calculate the new profit
self.p0 = self.q0 * ((self.p_max - self.q0 - self.q1) - self.c_m*self.p_max)
```

Population plant

```
# Simulate the noise
noise = rnd.uniform(-self.noise_level, self.noise_level)
# Maximum hunters are 100, and minimum are 0
control_signal = max(min(control_signal, 100), 0)
# Calculate the new stock
self.stock = max((self.stock+self.stock*self.growth_factor-control_signal*self.stock*self.hunter_rate + noise), 0)
```

Standard PID Controller - Bathtub Plant

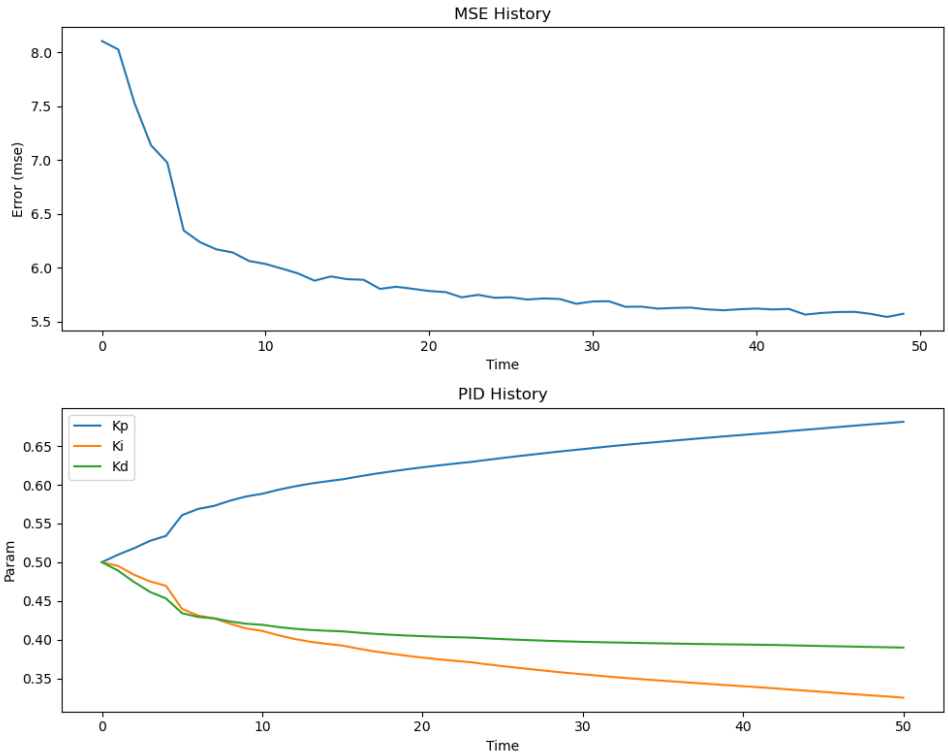
Parametere viser hvordan kontrolleren og modellen ble initiert. Progresjon viser hvordan MSE, og de ulike parameterne (kp, kd og ki) endret seg over tid. Her ser vi at MSE begynner å stabilisere seg ettersom parameterne stiller seg inn, og med flere timesteps og epoker kunne vi forventet enda bedre resultat. Denne modellen var veldig lite sensitiv til startverdier og learning_rate.

Parametere:

```
[Bathtub]
waterlevel = 110
cross_section = 3
noise_level = 0.01
target_level = 100
```

```
[Bathtub_PID_Parameters]
start_kp = 0.5
start_ki = 0.5
start_kd = 0.5
direction = -1
standard_learning_rate = 0.001
```

Progresjon:



Standard PID Controller - Cournot Plant

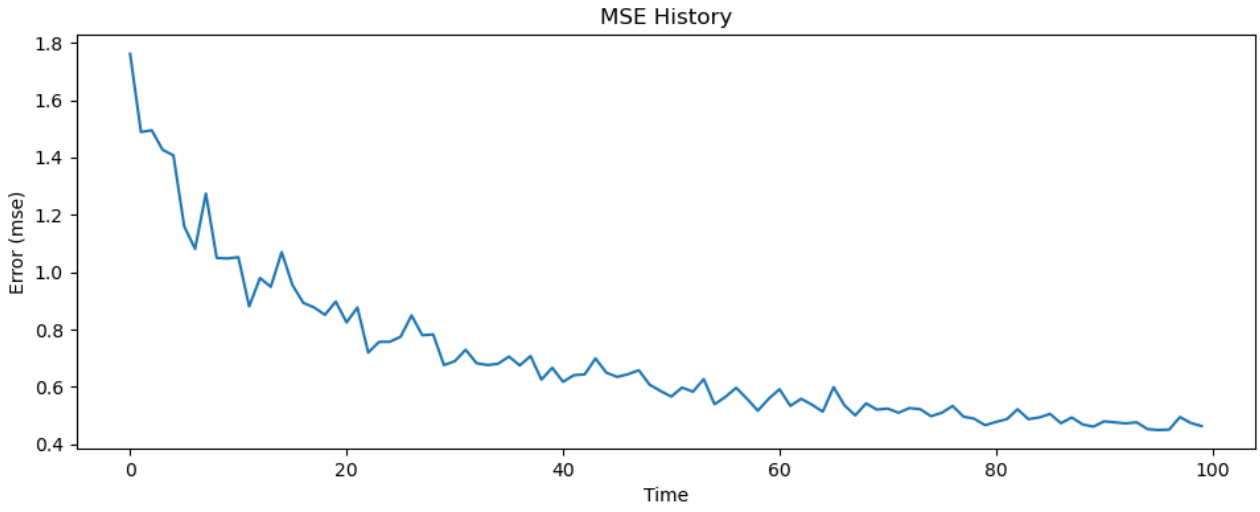
Parametere viser hvordan kontrolleren og modellen ble initiert. Progresjon viser hvordan MSE, og de ulike parameterne (kp, kd og ki) endret seg over tid. Her ser vi at MSE begynner å stabilisere seg ettersom parameterne stiller seg inn, og med flere timesteps og epoker kunne vi forventet enda bedre resultat. Denne modellen er meget sensitiv til startverdier og learning_rate. Jeg måtte starte kontrolleren med ekstremt små verdier for at den ikke skulle «overshoote» kontroll signal og læring.

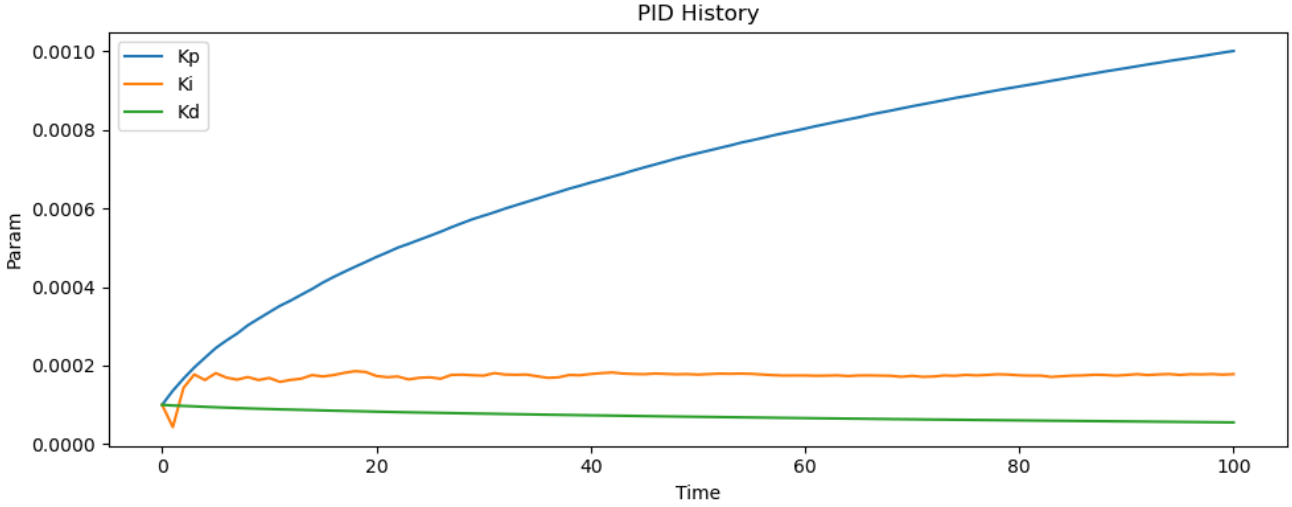
Parametere

```
[Cournot]
q0 = 0.5
q1 = 0.5
noise_level = 0.001
p_max = 50
c_m = 0.1
target_level = 20
```

```
[Cournot_Standard_PID_Parameters]
start_kp = 0.0001
start_ki = 0.0001
start_kd = 0.0001
direction = -1
standard_learning_rate = 0.00000001
```

Progresjon





Standard PID Controller - Population Plant

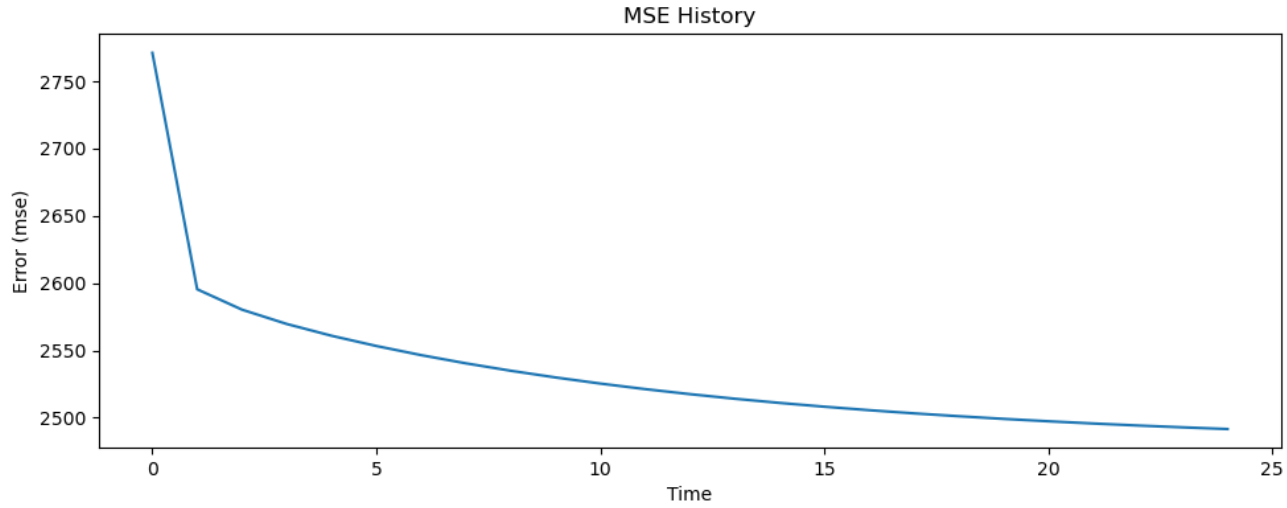
Parametere viser hvordan kontrolleren og modellen ble initiert. Progresjon viser hvordan MSE, og de ulike parameterne (kp, kd og ki) endret seg over tid. Her ser vi at MSE begynner å stabilisere seg ettersom parameterne stiller seg inn, og med flere timesteps og epoker kunne vi forventet enda bedre resultat. Denne modellen er meget sensitiv til startverdier og learning_rate. Som i cournot modellen måtte jeg starte kontrolleren med ekstremt små verdier for at den ikke skulle «overshoote» kontroll signal og læring.

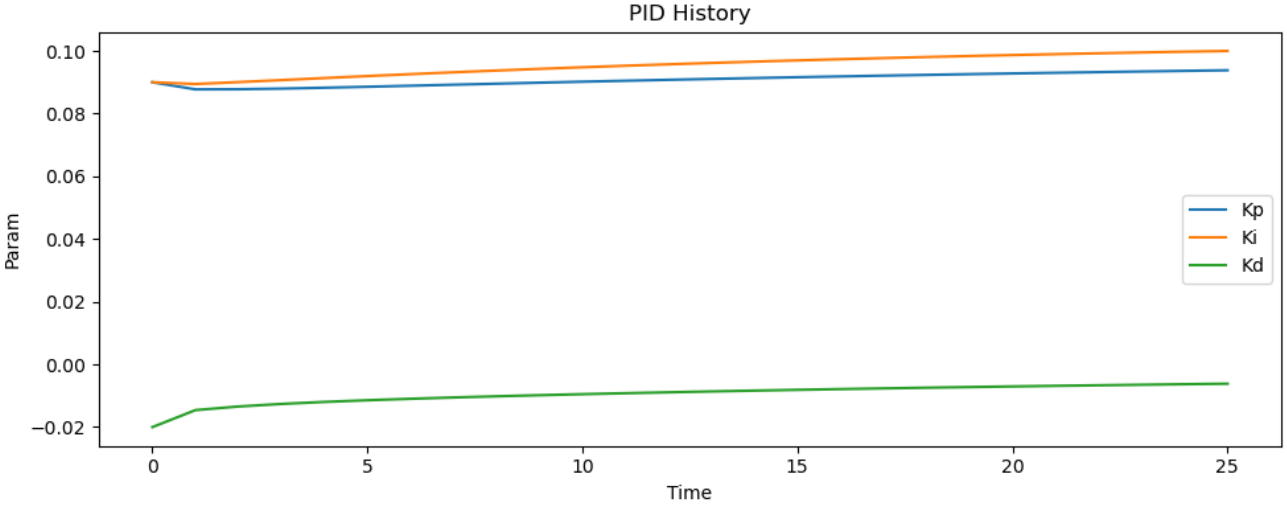
Parametere

```
[Population]
stock = 10000
hunter_rate = 0.001
growth_rate = 0.05
noise_level = 0.001
target_level = 10000
```

```
[Population_Standard_PID_Parameters]
start_kp = 0.09
start_ki = 0.09
start_kd = -0.02
direction = -1
standard_learning_rate = 0.0000001
```

Progresjon





NeuralNet PID Controller - Bathtub Plant

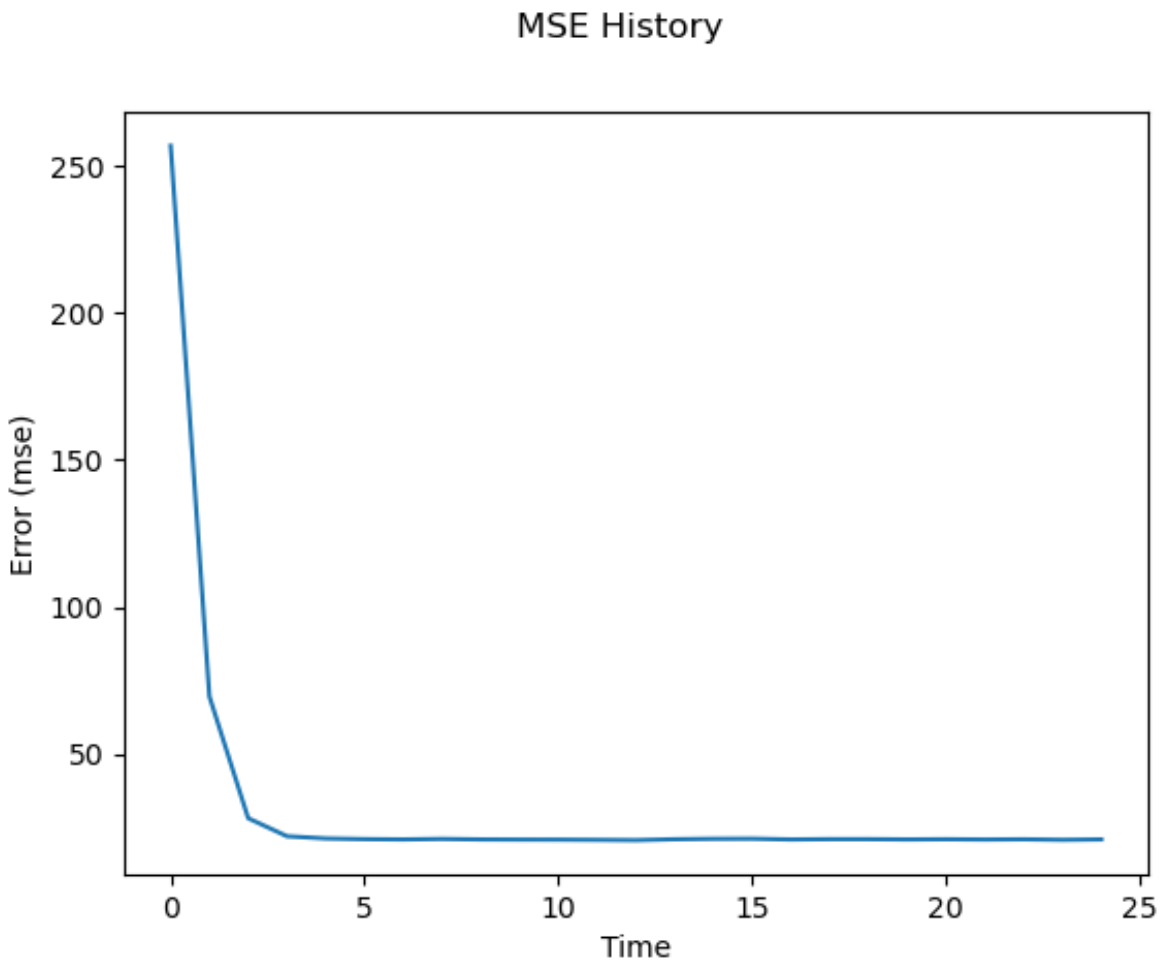
Parametere viser hvordan kontrolleren og modellen ble initiert. Progresjon viser hvordan MSE har endret seg over tid. Her ser vi at MSE begynner å stabilisere seg ettersom parameterne stiller seg inn, og med flere timesteps og epoker kunne vi forventet enda bedre resultat. Denne modellen er ikke betydelig sensitiv til startverdier og learning_rate.

Parametere

```
[Bathtub]
waterlevel = 110
cross_section = 3
noise_level = 0.01
target_level = 100
```

```
[Bathtub_NN_PID_Parameters]
learning_rate = 0.001
factor = 0.1
layer_option = 0
```

Progresjon



NeuralNet PID Controller - Cournot Plant

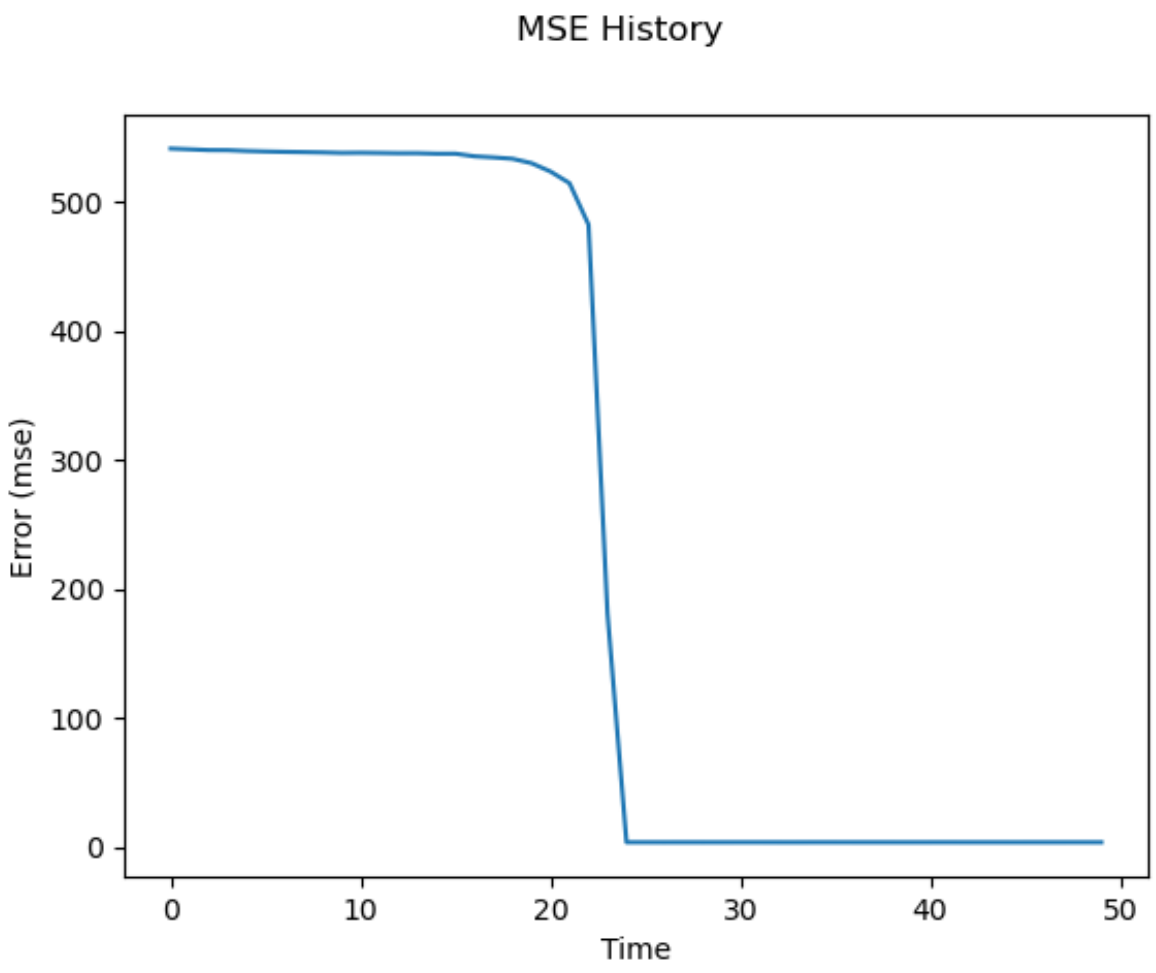
Parametere viser hvordan kontrolleren og modellen ble initiert. Progresjon viser hvordan MSE har endret seg over tid. Her ser vi at MSE begynner å stabilisere seg ettersom parameterne stiller seg inn, og med flere timesteps og epoker kunne vi forventet enda bedre resultat. Denne modellen er noe sensitiv til startverdier og learning_rate. Som man kan se, satt jeg factor (størrelsen på vektene og biasene ved generering) til å være lavere enn ved bathtub modellen. Det kan også se ut som om kontrolleren fant en bratt helning mot et bunnpunkt etter omtrent 23 epoker.

Parametere

```
[Cournot]
q0 = 0.5
q1 = 0.5
noise_level = 0.001
p_max = 50
c_m = 0.1
target_level = 20
```

```
[Cournot_NN_PID_Parameters]
learning_rate = 0.01
factor = 0.01
layer_option = 3
```

Progresjon



NeuralNet PID Controller - Population Plant

Parametere viser hvordan kontrolleren og modellen ble initiert. Progresjon viser hvordan MSE har endret seg over tid. Her ser vi at MSE begynner å stabilisere seg ettersom parameterne stiller seg inn, og med flere timesteps og epoker kunne vi forventet enda bedre resultat. Denne modellen er meget sensitiv til startverdier og learning_rate. Som man kan se, satt jeg factor (størrelsen på vektene og biasene ved generering) til å være lavere enn ved bathtub modellen.

Parametere

```
[Population]
stock = 10000
hunter_rate = 0.001
growth_rate = 0.05
noise_level = 0.001
target_level = 10000
```

```
[Population_NN_PID_Parameters]
learning_rate = 0.01
factor = 0.01
layer_option = 3
```

Progresjon