



Write a non-linear model-predictive controller in Python that optimizes the startup of a steam generator in a power plant.



Designing a non-linear model-predictive controller (NMPC) for a steam generator in a power plant requires expertise in control theory, power plant operations, and deep knowledge of the specific system dynamics. However, I can provide you with a high-level example that you can adapt and refine to suit your specific needs.

Please note that this example is a simplified version, and you should consult with control engineers and power plant experts to implement a real-world NMPC for a steam generator.

1. First, we'll define the steam generator model. In this example, we use a simple first-order non-linear model:

```
python Copy code

import numpy as np

def steam_generator_model(x, u, params):
    # x: state variables (e.g., temperature, pressure)
    # u: control inputs (e.g., fuel flow rate, coolant flow rate)
    # params: model parameters (e.g., heat capacity, time constants)

    # Implement the non-linear model equations here, for example:
    dxdt = -params['alpha'] * x[0] * np.exp(-params['beta'] * x[1]) + params['gamma'] * u[0]

    return dxdt
```

2. Next, we'll implement the cost function that the NMPC will optimize:

```
python Copy code

def cost_function(x, u, setpoints, params):
    # x: state variables
    # u: control inputs
    # setpoints: desired state values (e.g., target temperature and pressure)
    # params: cost function parameters (e.g., weights for different objectives)

    # Define the cost, e.g., weighted sum of squared errors
    cost = params['weight_x'] * (x[0] - setpoints[0])**2 + params['weight_u'] * (u[0] - setpoints[1])**2

    return cost
```

3. Now we'll implement the NMPC optimization using a suitable optimization library, like SciPy:

```
python Copy code

from scipy.optimize import minimize

def nmpc_optimization(x, u_init, setpoints, params, horizon):
    # x: current state variables
    # u_init: initial control inputs for optimization
    # setpoints: desired state values
    # params: model and cost function parameters
    # horizon: prediction horizon for NMPC

    def objective(u):
        cost = 0
        x_pred = x.copy()

        # Loop through the prediction horizon
        for i in range(horizon):
            # Apply the control input and update the state
            x_pred += steam_generator_model(x_pred, u[i], params)

            # Accumulate the cost
            cost += cost_function(x_pred, u[i], setpoints, params)

        return cost

    # Optimize the control inputs
    result = minimize(objective, u_init, method='SLSQP')

    return result.x
```

4. Finally, create a loop that simulates the steam generator startup:

```
python Copy code

import matplotlib.pyplot as plt

# Define parameters, initial conditions, and setpoints
params = {'alpha': 0.1, 'beta': 0.2, 'gamma': 0.3,
          'weight_x': 1, 'weight_u': 0.01}
x_init = np.array([0.0])
u_init = np.array([1.0])
setpoints = np.array([1.0])
horizon = 20

# Simulation settings
simulation_time = 100
sampling_time = 1

# Initialize variables
x = x_init.copy()
x_history = [x_init]

# Simulation loop
for t in range(simulation_time):
    u_opt = nmpc_optimization
```



continue



(x, u_opt, setpoints, params, horizon)

```
makefile Copy code

# Apply the first optimized control input to the system
x += steam_generator_model(x, u_opt[0], params) * sampling_time

# Store the state for visualization
x_history.append(x.copy())
```

Visualize the results

```
x_history = np.array(x_history)
time = np.arange(simulation_time + 1) * sampling_time

plt.figure()
plt.plot(time, x_history, label='Temperature')
plt.plot(time, setpoints * np.ones_like(time), '--', label='Setpoint')
plt.xlabel('Time [s]')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```

```
vbnet Copy code

This example demonstrates a basic NMPC optimization for a steam generator s
```

