Lehrstuhl für Systemverfahrenstechnik
Wissenschaftszentrum Weihenstephan
Technische Universität München

Interdisciplinary project (IDP)

An application of Industrial Internet of Things (IIoT) in crystallization: development of a sensorized controller platform for seeded glycine crystallization using containerization techniques and communication protocols [100% remote possible, English / German]

Motivation

The crystallization is one of the most important process unit in production but the quality control in terms of the particle size and shape distribution (PSSD) is complicated due to the interplay between the nucleation and crystal growth. Due to the lacking in-situ techniques during the crystallization, the properties of the crystallization product evolved with time is hard to observe and the final product remains uncontrollable with proper sensors which guides the cooling profile of the crystallization.

To solve this problem, two essential aspects must be tackled:

Precise temperature control considering both reactor contents and environmental interactions, achieved via a model predictive controller (MPC) embedding glycine crystallization kinetics. Unlike conventional PI controllers, MPC is more noise-resistant and reaches targets faster.

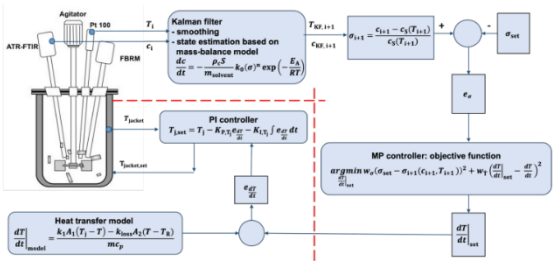

Figure 1 Cascade MPC for constant supersaturation

Accurate state monitoring to correct errors from environmental changes or control actions. Multiple sensors can be used; currently, an inline G-sensor with image analysis tracks single-crystal growth.

State of the art

A key task of this IIoT platform is to decouple the operation of the MPC and G-sensor, enabling decentralized data communication. This is essential for multi-target control, where different objectives need to be addressed sequentially based on production demands. For example, the system should first maintain constant supersaturation (Action 1), then adapt growth parameters based on G-sensor data once crystallization begins (Action 2), and finally switch to controlling the growth rate (Action 3). The challenge is that Action 2 must only start after seeding triggers detectable growth, and stop once stable values are reached to avoid unnecessary fluctuations — at which point Action 3 should take over.

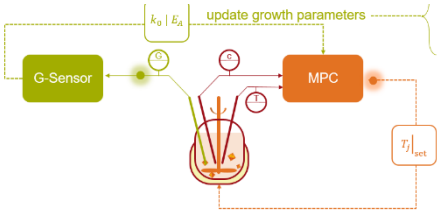

Figure 2 Scheme of adaptive MPC with G-sensor

Currently, the control system uses Matlab, which can run multiple instances but lacks shared workspaces, making data exchange slow due to file-based I/O. Additionally, frequent function changes between Matlab versions complicate maintenance. A critical issue is that Matlab 2023b removed support for the 'Basic256' security policy, which is still required by the OptiMax OPC UA server.

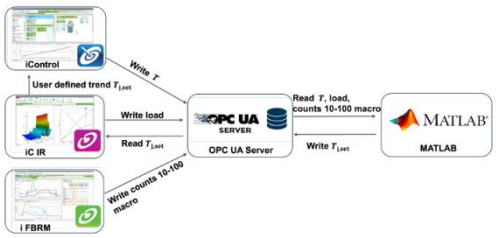

Figure 3 Data flow of the old structure using Matlab

Three possible solutions exist:

- Keep using Matlab versions before 2023b.
- Use two Python scripts and one Matlab instance: one Python client collects data from OptiMax and forwards it to a server for Matlab, while another Python script sends data from Matlab back to OptiMax.
- Fully rebuild the system in Python for better version and dependency control.

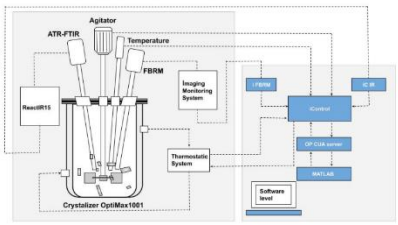

Figure 4 Experimental implementation (w/o G-sensor)

The main bottleneck is the centralized OPC UA communication — if the server fails, all components are impacted.

Working package

- Understanding the *Matlab* codes with theory documents for MPC.
- Transferring the system to python with help of the *Docker* containerization technique.
- Set up data communication: 1) persistent using InfluxDB, 2) instant data exchange with buffer using *RabbitMQ*, 3) data reception using agent *Telegraf* and 4) configuration data using *.txt* or *.config* file.
- Visualization by creating the graphical user interface (GUI) by implementing *Plotly Dash* for each component. There should be some buttons, which can control the start, and end of the processes and the parameters should be able to be updated during the execution of the scripts.
  (Time plan for each time stone can be discussed according to the implementation)

Desired capabilities

- Great knowledge in Python (the key language) and Matlab
- Basic understanding of virtual environment, containerization techniques like Docker, common communication protocols such as MQTT and RabbitMQ, and time series database like InfluxDB
- Basic understanding of system controlling, Kalman filter
- Basic knowledge in image analysis or computer vision

Recommended materials for preparation

- [1] KI-Einsatz in KMU: Einstiegshürden ausräumen [Clearing entry hurdles for AI deployment in SMEs – Artificial intelligence for German SMEs] (Heider, Imanuel [1]; Yu, Huitian [2])
- [2] A master thesis from one of my students: it will be given after contacting me

Supporting courses

Wissenschaftlich-Technisches Rechnen; Modellierung und Simulation disperser Systeme (in English); Wissenschaftliches Rechnen mit Matlab: these courses provide information about Matlab skills in technical aspects and basics to modelling

What you can expect

The framework has been validated by [1]; only the Plotly Dash application remains in draft. Completing it will deepen your understanding of IIoT, data communication, and industrial containerization.

It’s a rare opportunity to apply this to a sensorized MPC system, offering valuable insight into system control, industrial production, and interdisciplinary programming skills.

Technical University of Munich

Chair of Process System Engineering (SVT)

Huitian Yu, M.Sc., M.Sc.

Gregor-Mendel-Straße 4, 85354 Freising

Tel. +49 176 3705 8451

huitian.yu@tum.de