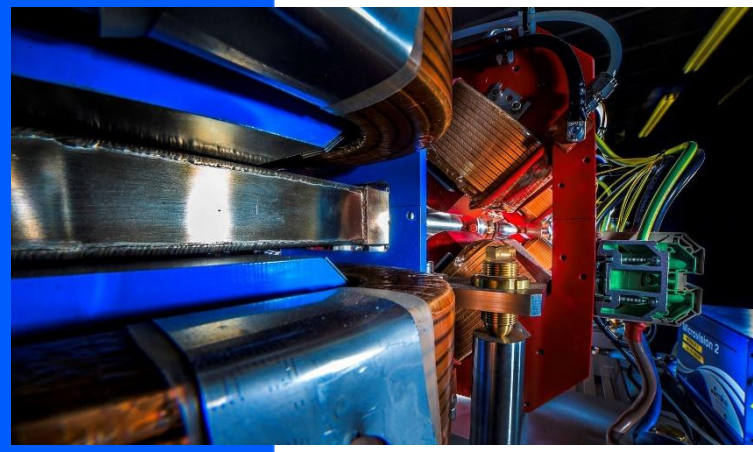
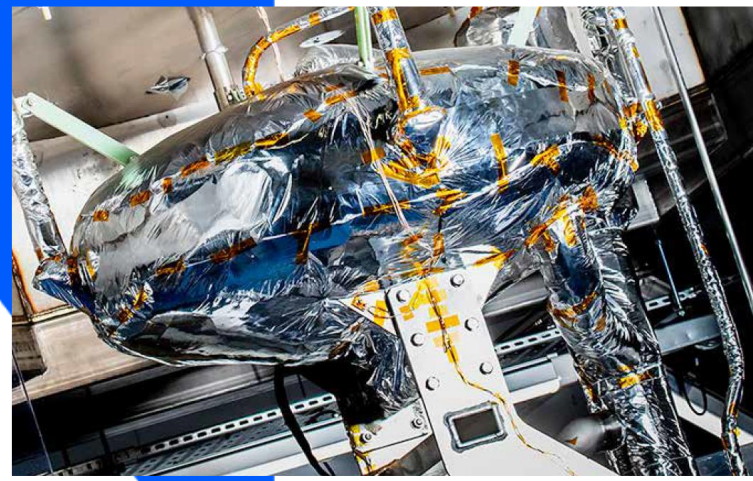
UKRI Science and Technology Facilities Council

ASTeC

Making a brighter future through advanced accelerators

Experience with a Digital Twin Deployment on the CLARA Facility





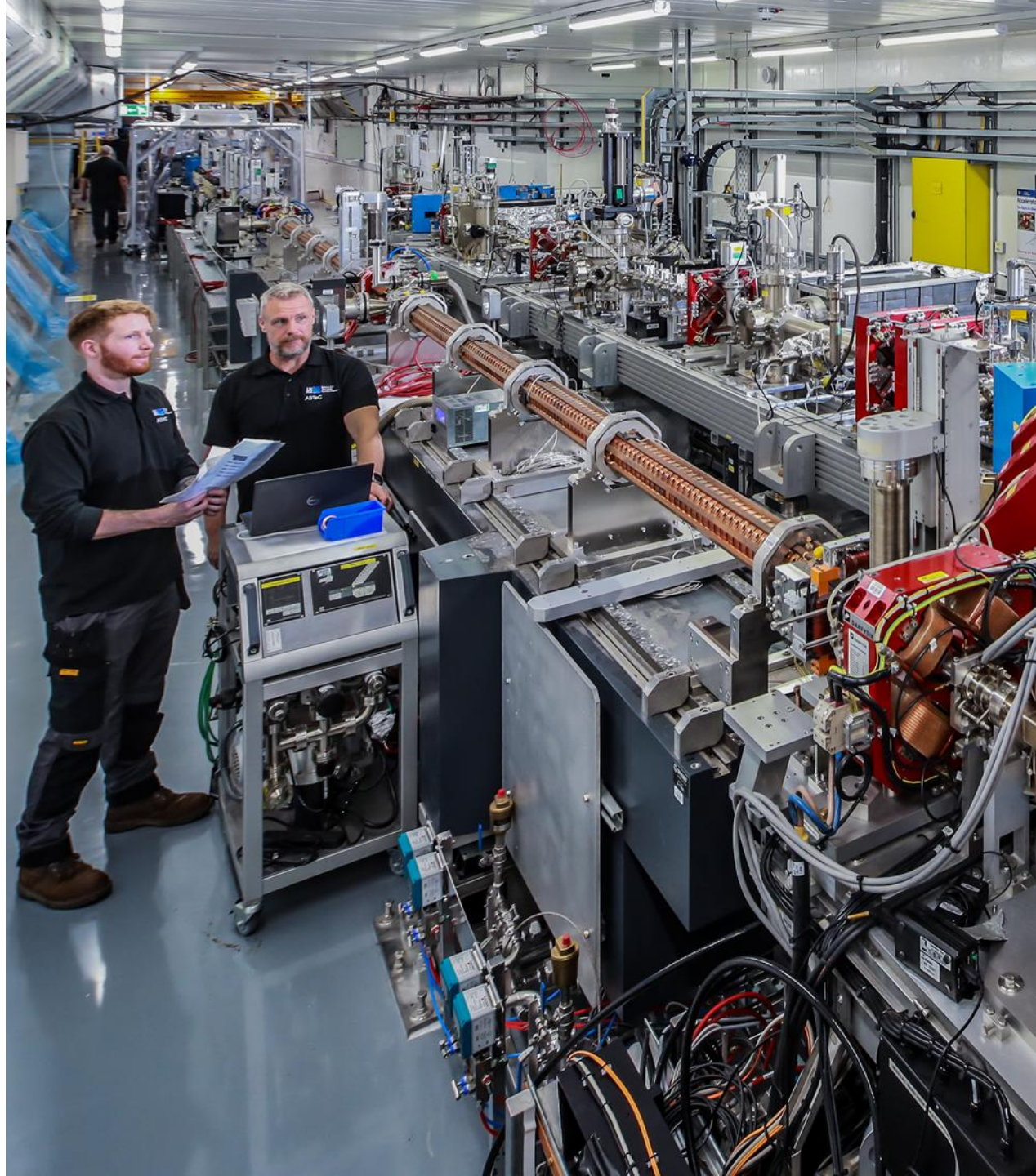
Matthew King

On behalf of: *Alexander Brynes, Nasiq Ziyan, James Jones, Mark Johnson, David Dunning*

Contact: matthew.king@stfc.ac.uk

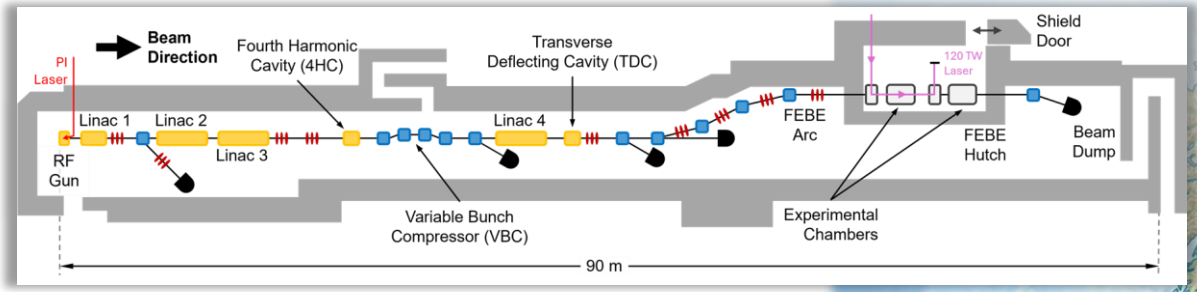
Contents

- Introduction to AI and Digital Twins at CLARA and other UK facilities
- Digital Twin framework
  - Framework overview
  - Lattice description
  - Digital Twin cookbook
  - Deployment example
- Lessons learnt
- The need for an AI test facility



AI and Digital Twins at CLARA and other UK facilities

- Long-standing software development towards AI and digital twins for accelerators at STFC Daresbury Laboratory's CLARA accelerator since ~2015
- CLARA is a 250 MeV electron accelerator test facility

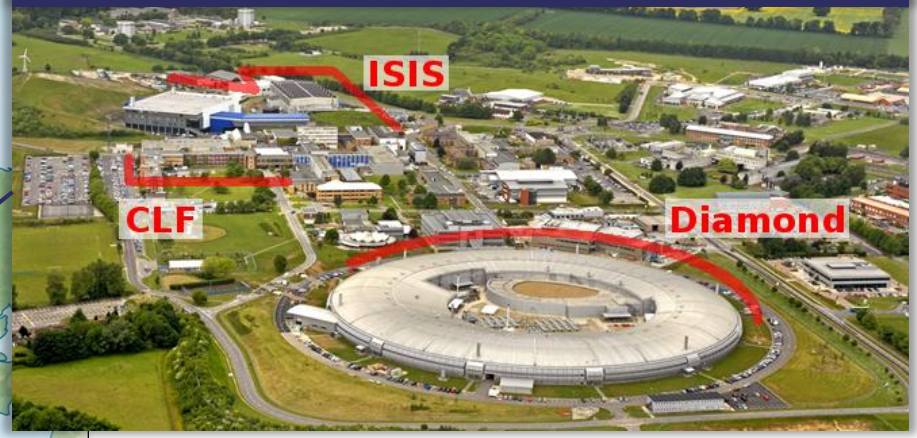


Daresbury Lab/Campus

First 'Intelligent Controls for Particle Accelerators Workshop' Hosted at Daresbury 01 Feb 2018

- Since 2022, there has been close collaboration between **CLARA** and **ISIS** neutron and muon source (both working with **SLAC**) to develop cross-facility AI and DT tools
- This work is extending with support from the Ada Lovelace Centre + Scientific Computing department
- Parallels to MOAT

Rutherford Appleton Lab/Harwell Campus



Science and Technology Facilities Council

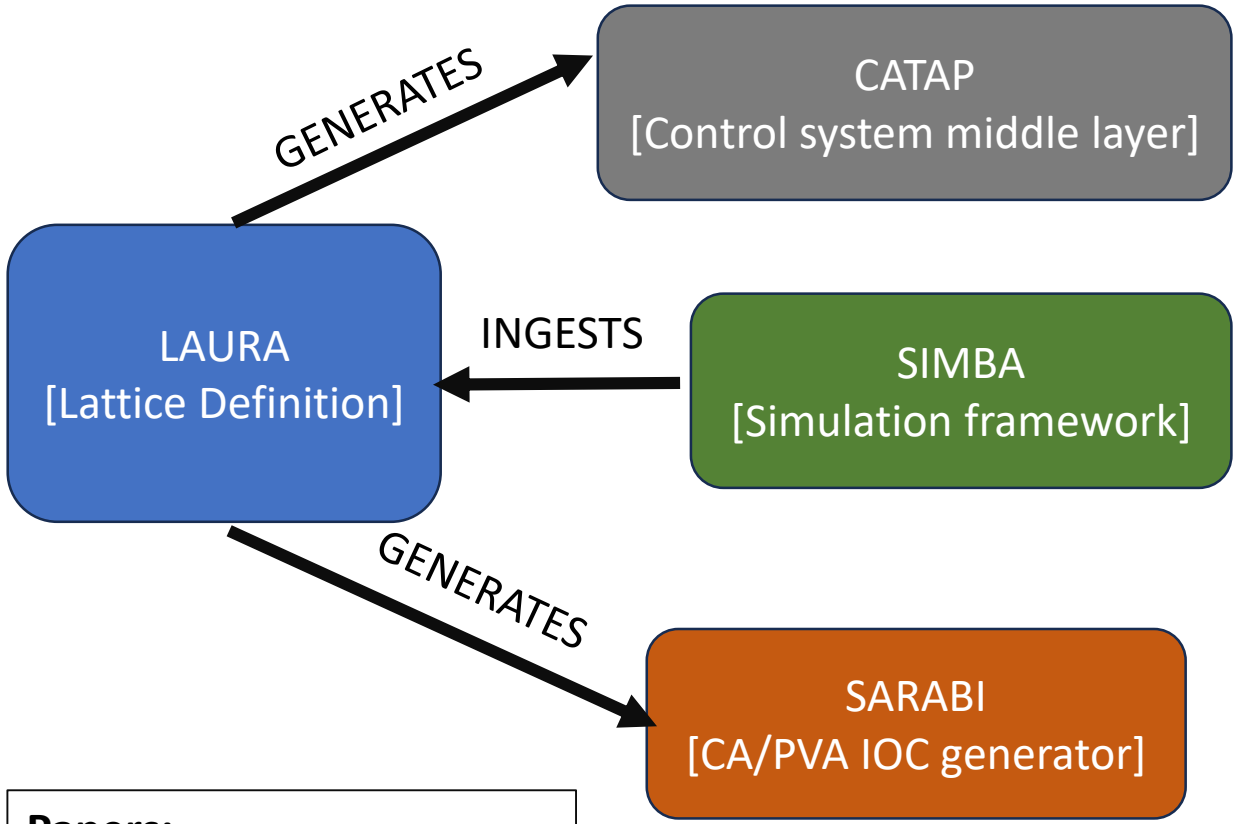
Software Packages @ ASTeC

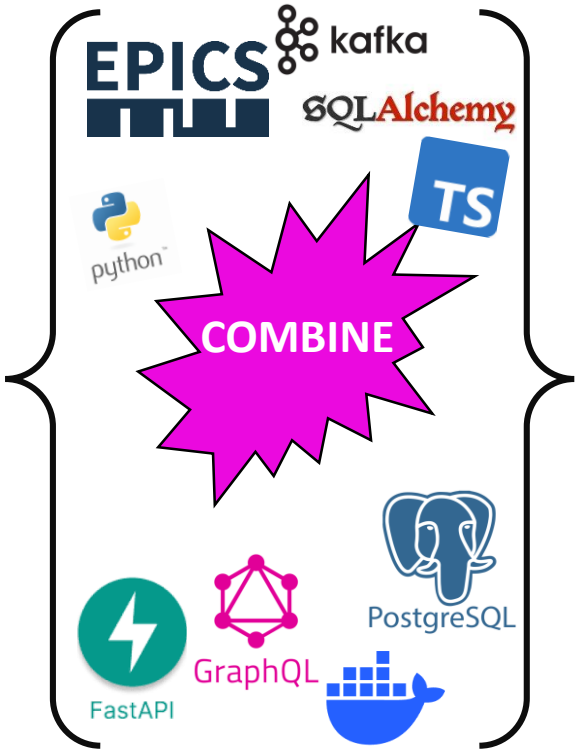
- **LAURA** (*Lattice Architecture for Unified Representation of Accelerators*)
  - YAML-based lattice definition
  - Standardized, extensible schemas
  - arXiv, Github, RTD
- **CATAP** (*Controls Abstraction Towards Accelerator Physics*)
  - Templated Middle Layer for high-level control
  - Generated based on ***LAURA*** definitions
  - arXiv, Github
- **SARABI** (Soft Architecture for Rendering Automated Backend IOCs)
  - Templated CA/PVA IOCs
  - Generated base on ***LAURA*** definitions
  - arXiv, Github
- **SIMBA** *(Simulations for Integrated Modeling of Beams in Accelerators)*
  - Unified simulation framework for particle tracking using ***LAURA*** definitions
  - Supports:
    - ASTRA, GPT, ELEGANT, Ocelot, CSR-Track, Cheetah, Wake-T, Xsuite
    - Beam-based surrogate models
  - arXiv, Github, RTD
- **JANUS** *(Joint Accelerator Network for Unified Simulation)*
  - Digital Twin framework generated from ***LAURA*** definitions
  - ***SIMBA*** handles tracking
  - ***CATAP*** handles control system interactions
  - arXiv, Github

JANUS Overview

A generalizable Digital Twin framework for particle accelerators:

Virtual prototyping | Online/Offline optimisation | Integrated training environment





JANUS

A Digital Twin Framework

DTs generated from LAURA Lattice Definitions

Papers:

CATAP: arXiv:2509.19794

SIMBA/LAURA: arXiv:2604.19103

JANUS: arXiv:2604.19101

Science and Technology Facilities Council

Lattice Description

How do you describe an accelerator lattice?

It depends who you ask!

Physicist:
Components that use EM fields to direct, steer and focus beams of particles.

Description format:
Simulation code lattice files

Controls Engineer:
Elements that can be used to change or monitor the beam using setpoints, currents and voltages

Description format:
Control system configuration files (i.e. EPICS database)

RF / Magnet / Diagnostics expert:
In-depth knowledge of the physics and design of specific components.

Description format:
Models of components and their fields; direct control of hardware

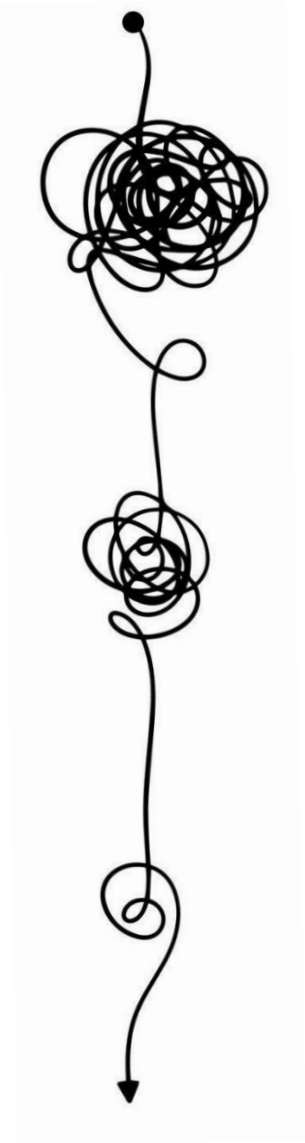
Mechanical Engineer:
Elements physically connected, aligned and placed on girders.

Description format:
CAD models and drawings

Can there be an all-encompassing description of a lattice that will satisfy everyone?
(Perhaps not, but for a digital twin/shadow we at least need to unify the control system and simulation descriptions.)

History of Lattice Descriptions @ CLARA

- **Pre-2017 lattice definitions:**
  - Simulation: Mathematica/MATLAB projects
  - Control System: PVs for control defined in applications (hard-coded)
  - Engineering: Centralized server for diagrams etc.
- **2017-2019 lattice definitions:**
  - Simulation: YAML definitions with package-specific schema
  - Control System: Middle-layer specific format with bespoke parser
  - Engineering: Centralized server for diagrams etc.
- **2019 – 2025 lattice definitions:**
  - Simulation: YAML definitions with package-specific schema
  - Control System: YAML definitions with middle-layer specific schema
  - Engineering: Centralized server for diagrams etc.
- **2025 – Current lattice definitions:**
  - Simulation: ***LAURA*** lattice definitions
  - Control System: ***LAURA*** lattice definitions
  - Engineering: Centralized server for diagrams etc. (linkable in ***LAURA*** lattice)
  - Concurrent with PALS efforts in US



How to Design a Digital Twin

(A recipe of trial and error..)

Digital Twin Cookbook

Recipe:
Digital Twin Loop à la Accelerator

Serves
Continuous optimization and analysis of one accelerator system

Preparation Time
A few months of integration

Cooking Time
Forever (or until someone switches the machine off)

Ingredients

Machine

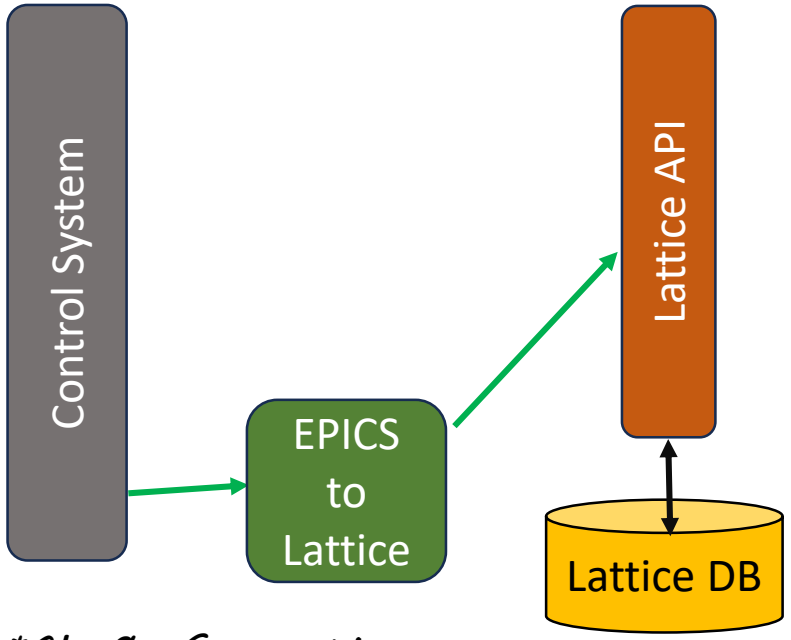
- 1 functioning accelerator
- Network access to the control system
- A list of machine parameters
- Method for publishing results back to operators
- Knowledge of:
  - What can be changed through the control system
  - What can be measured through the control system

Simulation

- 1 functioning simulation model
- A method for interrogating simulation results
- Knowledge of:
  - Machine-Simulation mappings
  - Unit conversions
  - Lookup tables
  - Calibration curves

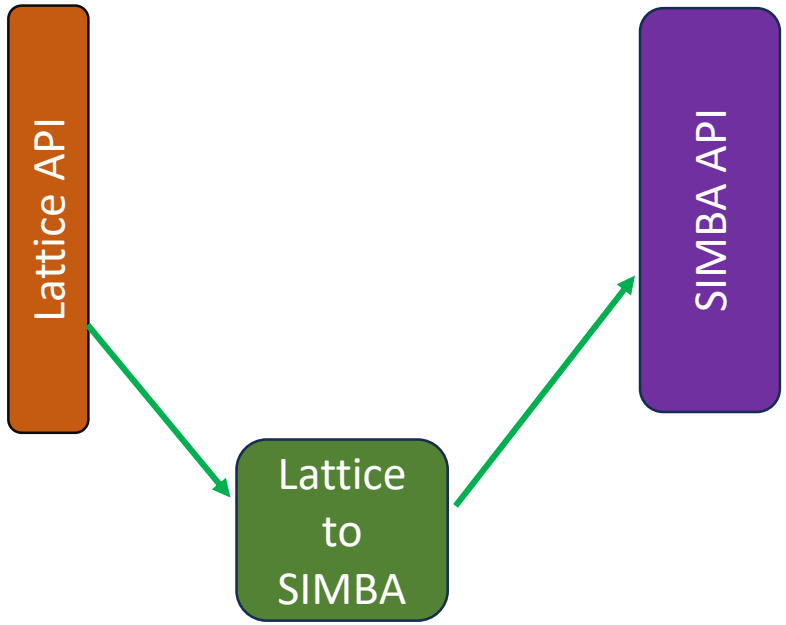
Digital Twin Cookbook

Step 1: Harvest Fresh Machine Settings...
Collect the latest settings from the accelerator*:



*Chef's Suggestion:
A control system middle layer, such as **CATAP** can make it easier to grab machine parameters

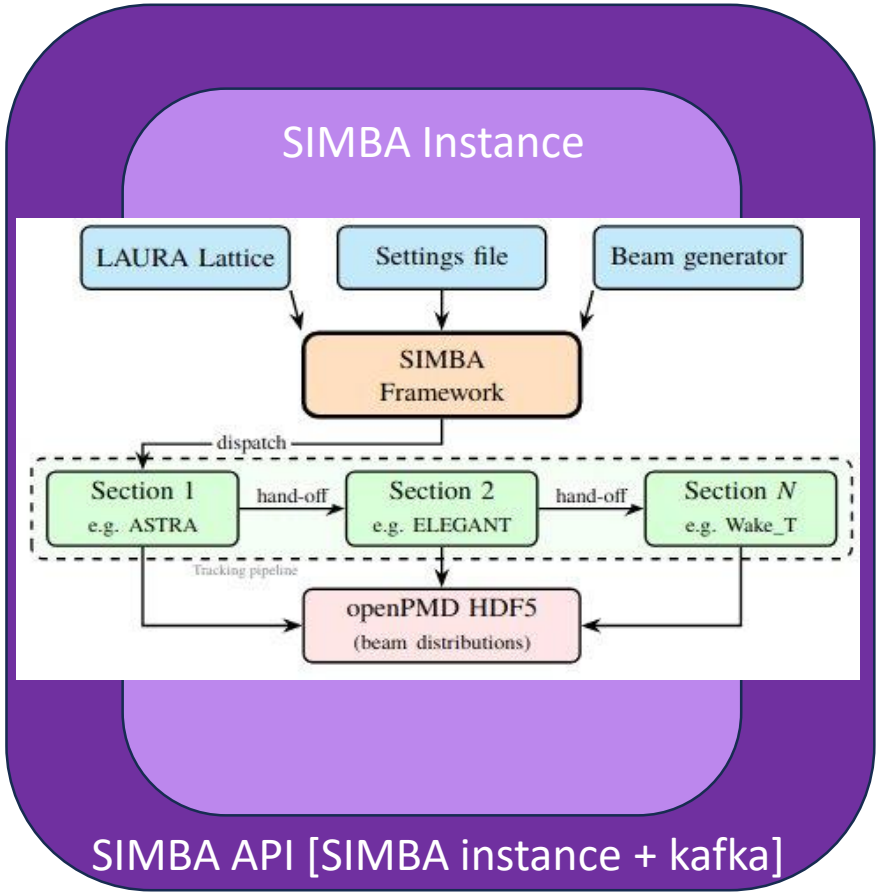
Step 2: Prepare the Simulation Inputs
Translate machine settings into simulation parameters*.



*Chef's Suggestion:
Keeping your conversions and mappings in a **LAURA** lattice can quicken this process!

Digital Twin Cookbook

Step 3: Start the Tracking*



*Chef's Suggestion:
The **SIMBA** software package can ingest a **LAURA** lattice for tracking

Step 4: Wait Until Tender

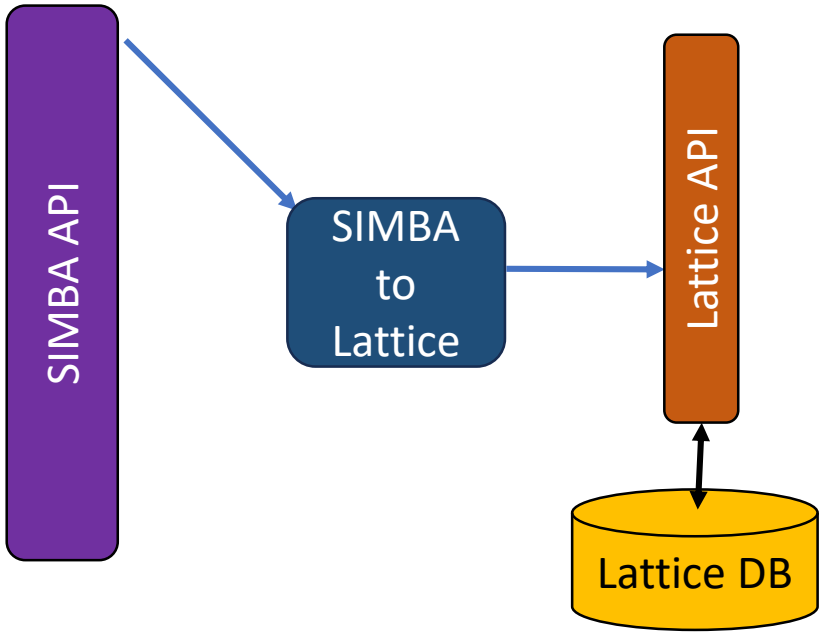
Monitor simulation progress using one of the following techniques:

- **Traditional slow-cooker method**
  - Poll the simulation status periodically
- **Modern smart-kitchen method**
  - Subscribe to a message broker, kafka for example.
  - Receive a notification when the simulation is finished
- **Traditional tracking can be slow!**
  - **SIMBA** is able to use AI-guided culinary automation (surrogate models)

Digital Twin Cookbook

Step 5: Retrieve the Results

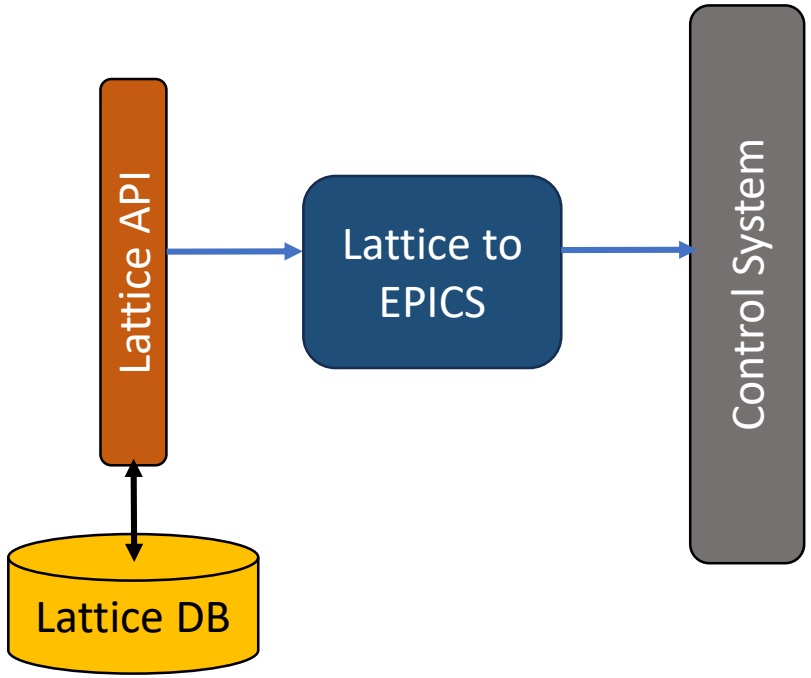
Extract the simulation outputs



- Store settings and results in database
  - Beam distributions can be challenging for large macroparticle numbers

Step 6: Plate and Serve

Publish the simulation results



Step 7: Repeat

Return to Step 1 and continue indefinitely…

Thanks to the database, preparation times reduce with use!

Digital Twin Cookbook

Chef's Notes

The secret to a successful Digital Twin is maintaining strict consistency of parameter mappings and units.

Most failed recipes result from bad conversions between machine and simulation parameters.

Once you can verify your simulation pairs well with the machine, you can begin to investigate advanced "twinning"

- See future recipes...

From the creators of: SIMBA, CATAP, and LAURA

JANUS

A Digital Twin Framework

DTs generated from LAURA Lattice Definitions

JANUS Framework

Accelerator Lattice

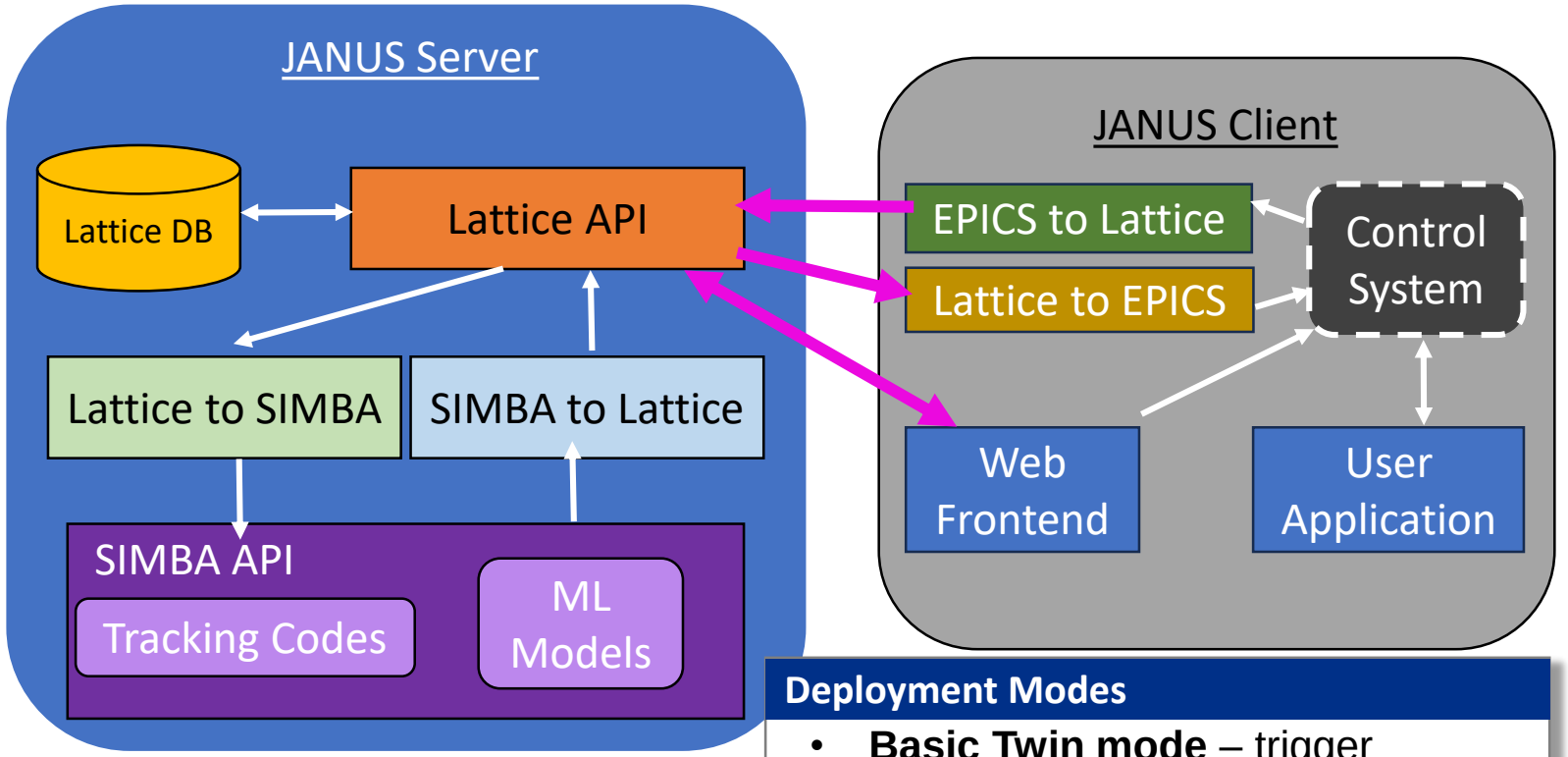
- The **ground source of truth** about the accelerator, containing:
  - Element definitions
  - Control system information
- Based on LAURA format

Virtual Control System

- **Generated procedurally** based on the lattice
- Sends signals to **update simulation model** when changes are made
- Can be swapped for physical controls

Simulation Model

- Can use **particle tracking codes** or **AI models**
- Chain tracking together for multiple machine sections defined in LAURA
- Based on SIMBA package



Lattice Database

- Each run is associated with a UUID and saved to a database
- Settings and Results are stored in SQL
- To be used for surrogate model training
- Accessible using GraphQL interface via Lattice API

Deployment Modes

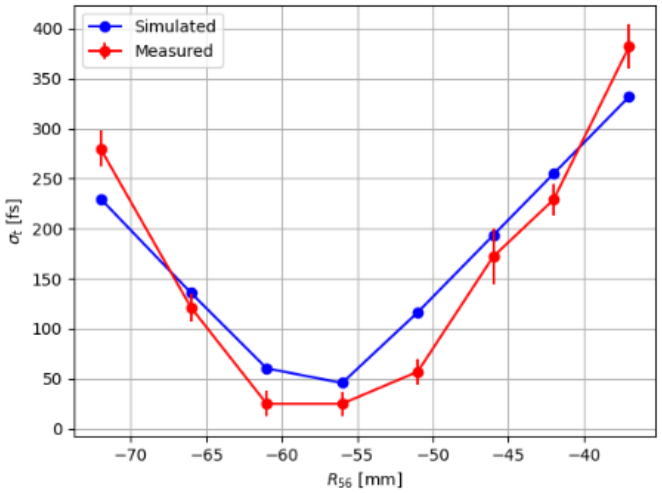
- **Basic Twin mode** – trigger simulations based on physical control system values, results sent back
- **Dev mode** – trigger simulations based on virtual control system values
- **Advanced Twin mode** – Bi-directional, autonomous feedback between physical control system and JANUS

UKRI Science and Technology Facilities Council

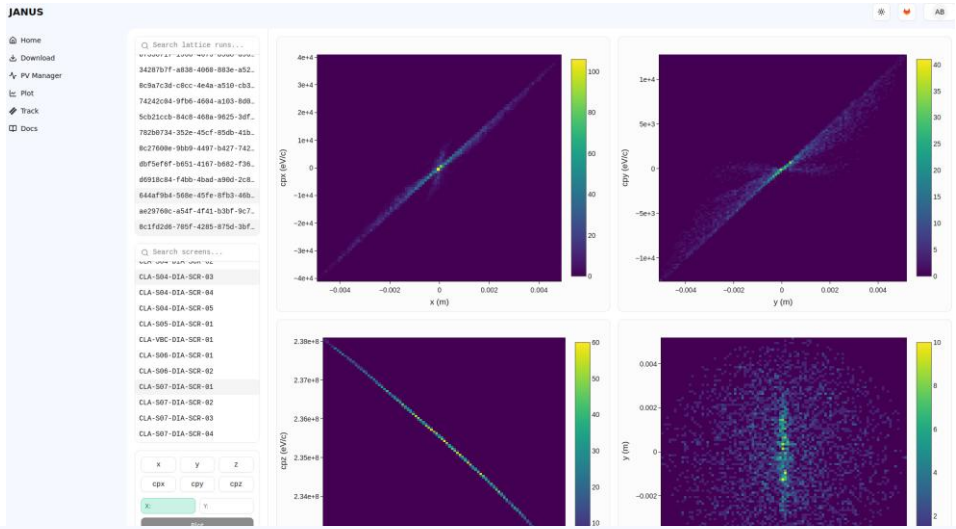
JANUS Framework - Control Room Deployment

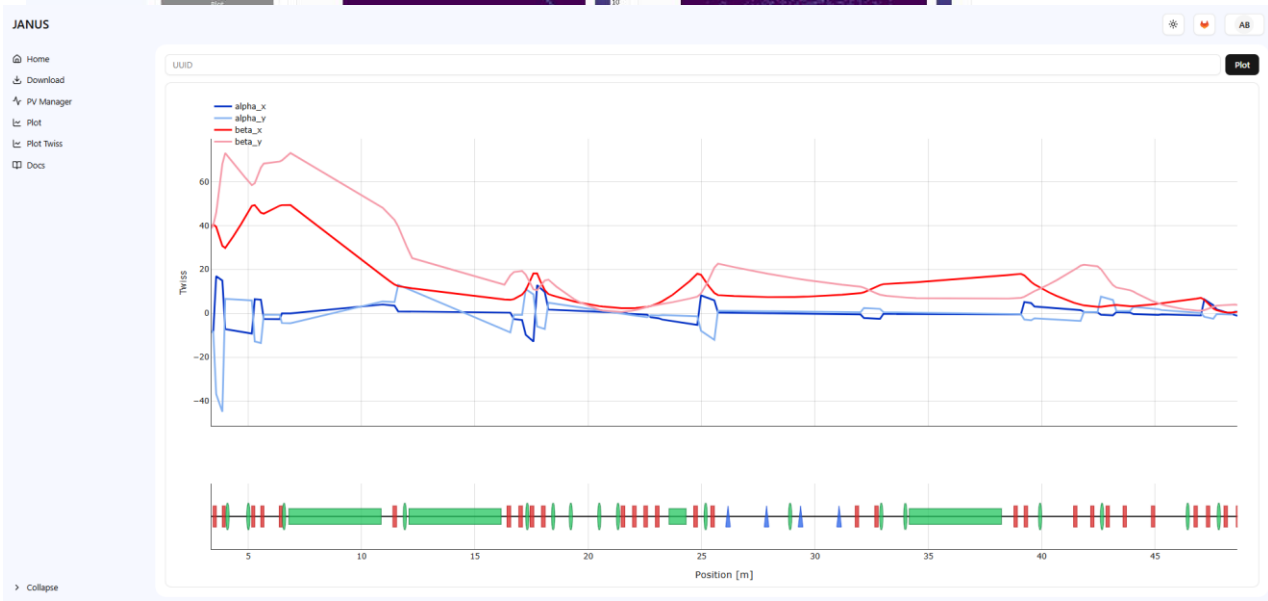
Web-based interface to JANUS:

- PVs can be **monitored** and specific PVs used to **trigger simulations**.
- **Real-machine version** of JANUS can be accessed from **any control room** PC.
- **Same Framework**, accessing Physical IOCs
- **Virtual diagnostics** (6D phase spaces, Twiss) are available for **all entries** in the database, also available via **PVAccess** on physical machine.



Comparison between measured and simulated bunch lengths after final linac (compression scan) -- both set up only using the control system.





Lesson Learnt

During design/deployment of JANUS:

- A consistent, structured, ontology that you can query is *vital*
  - Reliable foundation to build software stack from
  - Software can become more templated
  - Rich-text descriptions can be useful for LLMs
- Multiple access points provide flexibility:
  - Run simulations via the web
  - Access database via the web
  - App prototyping with virtual control system with/without simulation data

Model Matching Lessons:

- There are plenty of simulation parameters that aren't one-to-one with the machine
- Machines are complicated and modelling every interaction is complex
- Focusing on how information is exchanged between machine and simulation is worthwhile
  - There are lots of "gotchas" hidden here
- The measure of trust is not clear when the machine can drift, setups can change, and hardware can break!
  - Context and intention of the data is not always captured

Conclusions

- None of the items in this talk would have been possible without CLARA acting as a test facility
- To develop an advanced digital twin needs continued regular access to beamtime:
  - **Parasitic:** Useful, but it can make data hard to contextualize
  - **Designated:** needs preparation to get quality data
  - Significant work is still required:
    - **Advancing DT:** Model matching, Optimizations, Automated feedback etc.
    - **Extending DT:** Connection to experiments, Robotics, Non-simulation parameters
- *Developing an advanced digital twin solution and an AI test facility in combination can have widespread benefits for all accelerators*

Papers:
CATAP: arXiv:2509.19794
SIMBA/LAURA: arXiv:2604.19103
JANUS: arXiv:2604.19101

Thank you

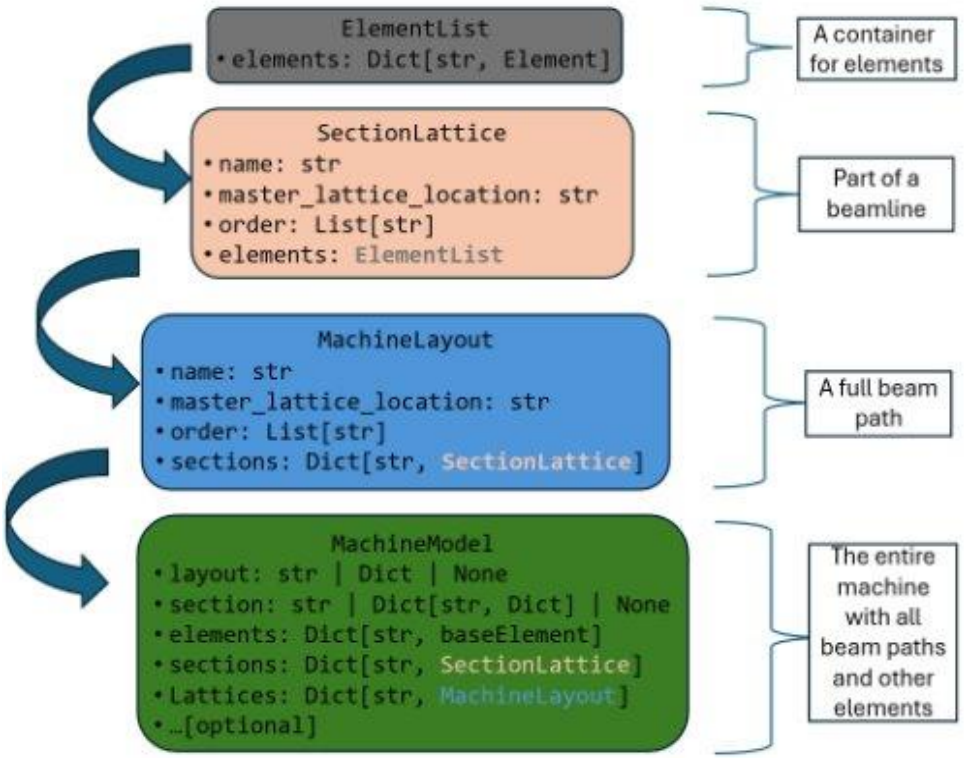
On behalf of: *Alexander Brynes, Nasiq Ziyan, James Jones, Mark Johnson, David Dunning*

Contact: matthew.king@stfc.ac.uk

Appendix

LAURA

- Attempts to provide an "all-encompassing" definition for an accelerator
- Element definitions can contain:
  - physical, simulation, controls, electrical, manufacturer information
- *Elements* are chained together into *sections*
- *Machines* are created from an ordered chain of *sections*
- LAURA API allows this to be done in either YAML or programmatically in python
- Translator Module:
  - Export the lattice to codes supported by ***SIMBA***
  - Import lattice in supported codes: ELEGANT



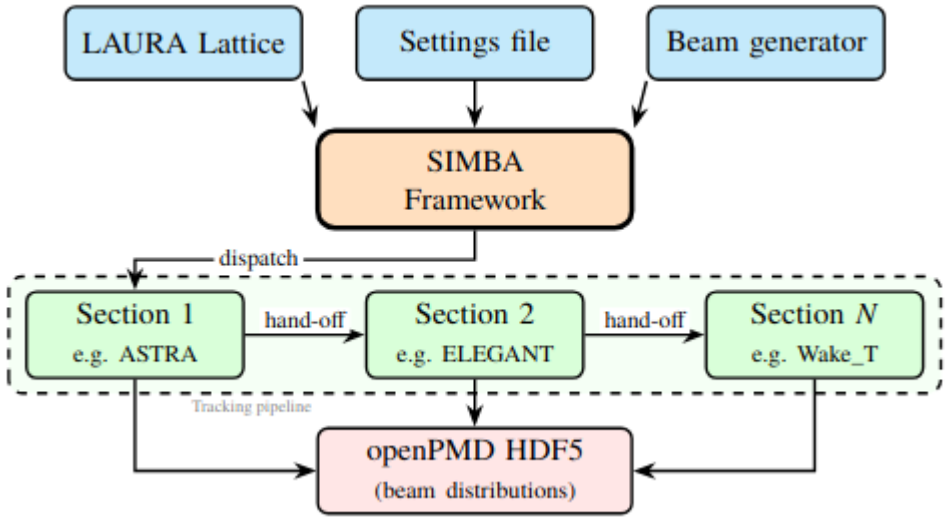
SIMBA

SIMBA is a python package to enable simulations of (linear) particle accelerators.
The lattice is derived from the LAURA structure.

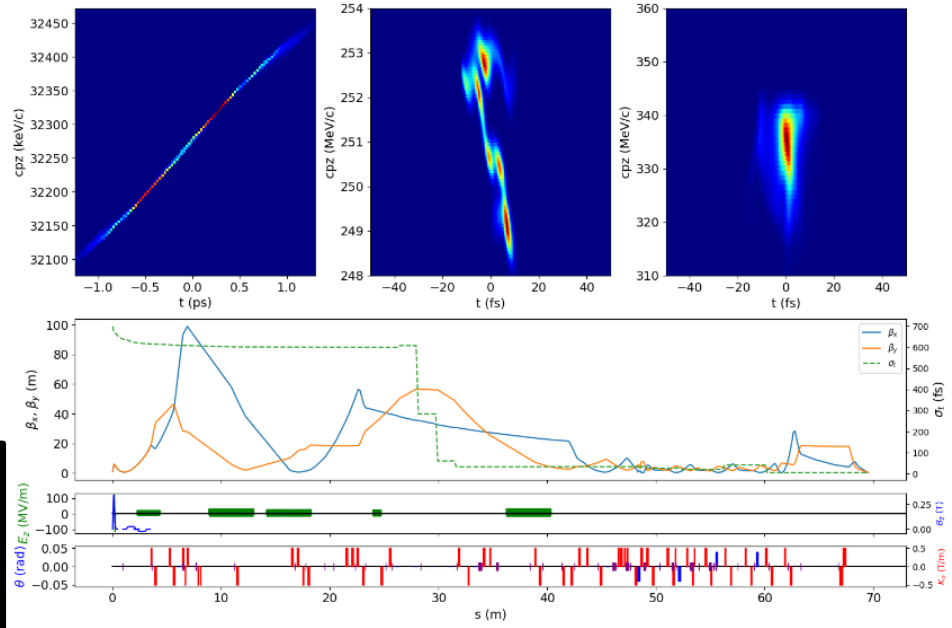
Abstract away the details of underlying tracking codes:

- Consistent beam tracking across multiple beam physics modalities (i.e. RF gun, linac, FEL, plasma...).
- Does not require expertise in multiple codes!
- Can also call ML models of lattice sections.

SIMBA repo



Data flow in SIMBA



SIMBA

- Abstract input files away
- Interact only with python methods
- Code-agnostic input/output
- Convert input/output data to standard formats
- *Learn new "abstract" code*

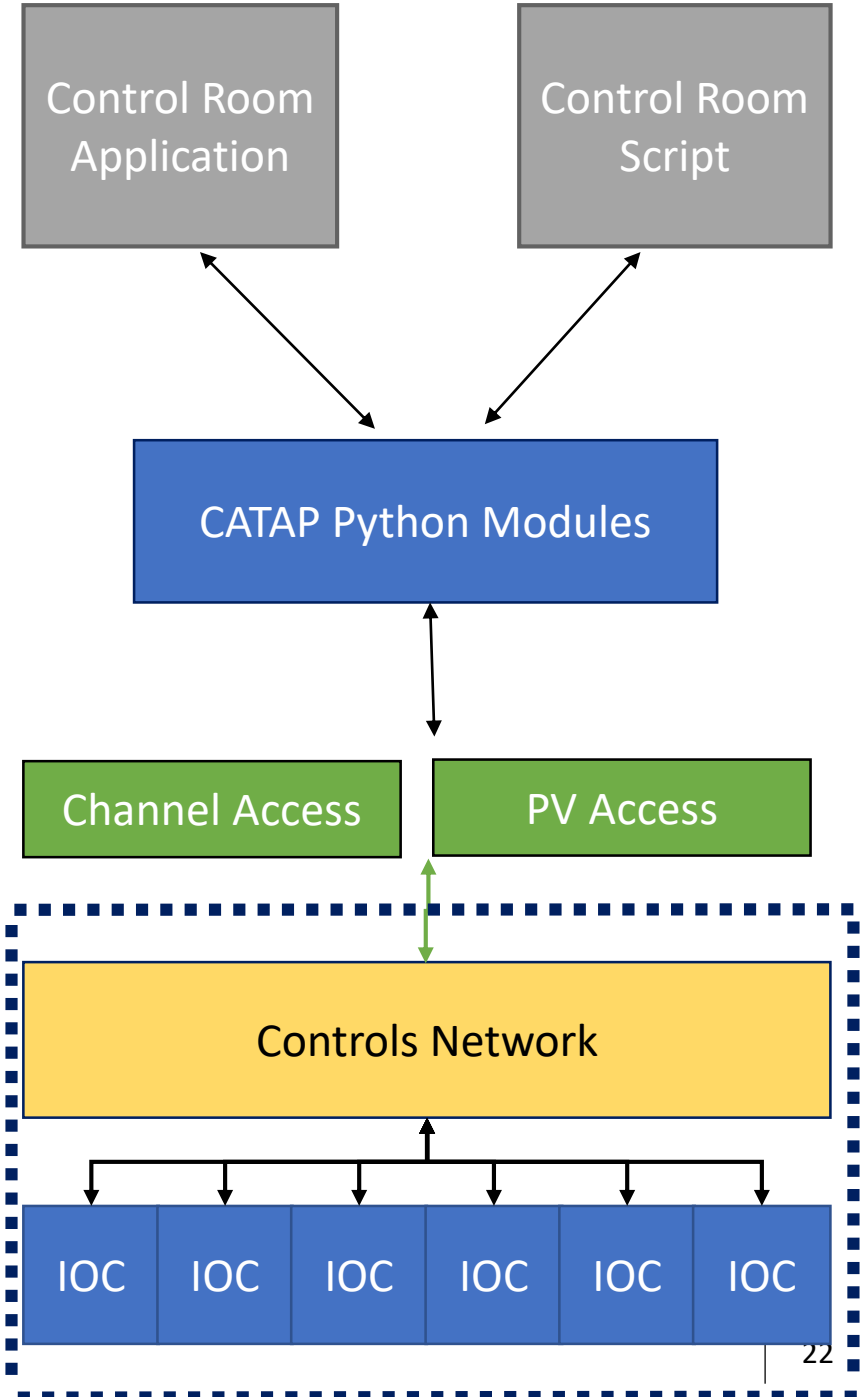
Start-to-end CLARA simulation

CATAP

A middle layer is an abstraction layer between the control system and the control room that simplifies the ways in which operators interact with the machine programmatically.

Since the LAURA structure defines the control system variables for an element, we can:

1. Create virtual accelerator elements from the same source of truth as the simulation model.
2. Programmatically construct an entire virtual accelerator.
3. **Programmatically construct accelerator-specific middle layer with a uniform architecture**.




```
from lcls_tools.common.devices.reader import *

areas = ['EMIT2']

for i in areas:
    magnets = create_magnet(area=i)
    magnets.turn_off()
```

Controlling LCLS magnets using [CATAP middle layer](), arXiv.2509:19794

Model Matching with JANUS

Task

- Try to find a set of *tractable* problems to:
  - Determine how close the model is to machine settings;
  - Find out where the discrepancies are;
  - Figure out whether (a) the model and/or (b) the machine need to be adjusted.

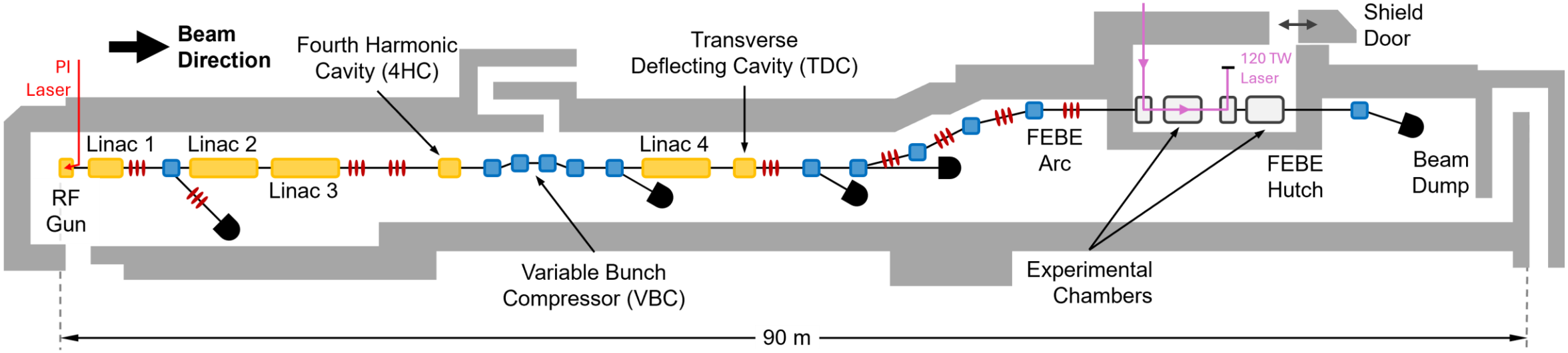
Initial Plans

- Match photoinjector model to machine
- Using single screen, perform an N-dimensional parameter scan, varying as many things as possible (within reasonable limits).
- Monitor beam distribution on screen.
- Simulate these setups in JANUS
- CATAP can load settings onto virtual control system

Next Steps

- Extend these scans to post-linac and spectrometer
- Incorporate Twiss measurements.
- Find discrepancies between the model and the data.
- Use this large dataset to learn how to do more advanced filtering / inference

CLARA FEBE Layout

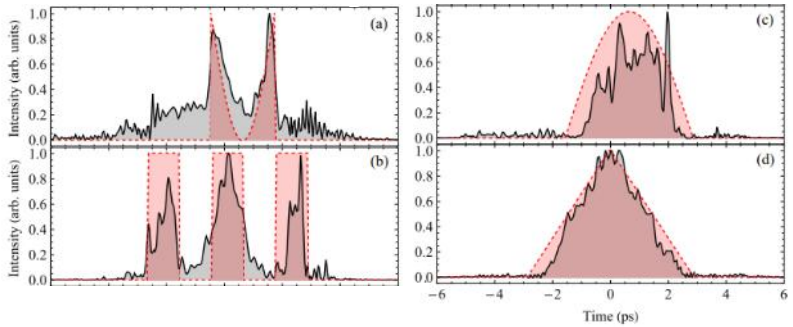


First experiments in the **Full Energy Beam Exploitation** (FEBE) user area took place in early 2026

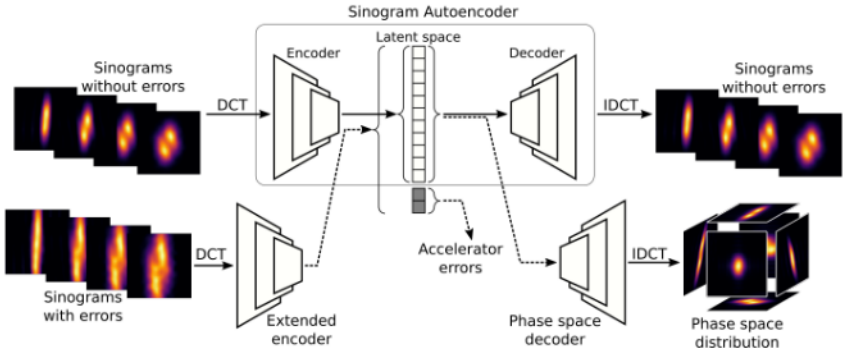
Example Technical System	CLARA Front End (35 MeV)	CLARA (250 MeV)	CLARA FEBE (250 MeV)
RF Structures	2	7	7
Magnets	22	77	126
Diagnostic Systems	13	65	94
Vacuum Gauges and Pumps	57	109	155

Established AI/ML and digital twin expertise within ASTeC

AI-based beam optimisation and



ML-based laser/electron bunch temporal shaping:
https://journals.aps.org/prab/abstract/10.1103/PhysRevAccelBeams.27.041602



(a) Latent space constructed using a sinogram autoencoder.

Electron bunch phase space tomography:
https://iopscience.iop.org/article/10.1088/1748-0221/19/07/P07013

Long-standing community building since

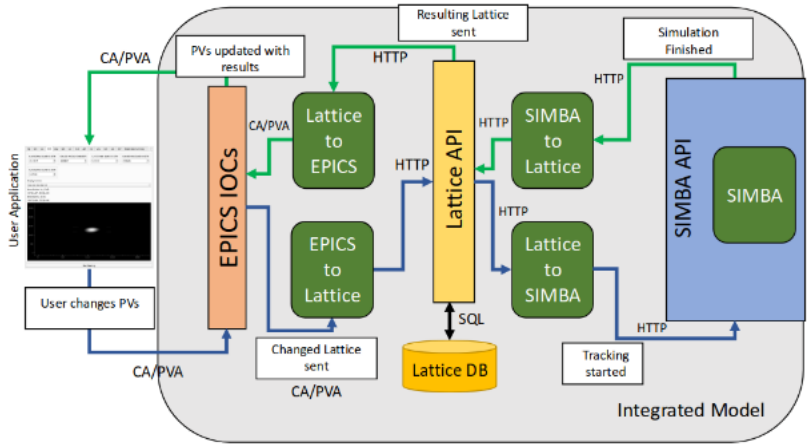
First 'Intelligent Controls for Particle Accelerators Workshop' Hosted at Daresbury 01 Feb 2018

This month, Daresbury was host to the first ever 2-day ICPA workshop

The first Intelligent Controls for Particle Accelerators workshop was held at Daresbury Laboratory, Tuesday-Wednesday 30th-31st January 2018. The goal of the workshop was to ferment cooperation and discuss future developments in this growing field of AI application to Particle Accelerators. A welcome address was given by Lab Director Professor Susan Smith, who made a convincing case for more advanced control methods as Particle Accelerators continue to develop.

STFC[2018]

Auto-Generating Digital Twins for Particle



Closing the Loop: Deploying Auto-Generating Digital Twins for Particle Accelerators
https://arxiv.org/pdf/2604.19101