Enabling Co-Simulation of Tokamak Plant Models and Plasma Control Systems

by

M.L. Walker

General Atomics, USA

Presented at the 11th IAEA Technical Mtg on Control, Data Acquisition and Remote Participation for Fusion Research

8-12 May 2017
Co-simulation = simultaneous execution of independent simulation codes that exchange data.

PCS-Plant co-simulation routinely used at DIII-D for > 20 yrs.
- Supports PCS development & operation
- Used by EAST, KSTAR, NSTX-U > 10 yrs

Similar capability planned to support ITER PCS development & use.
- ITER Plasma Control System Simulation Platform (PCSSP, in Simulink)

We discuss & clarify issues of simulation use to support control development and operation.
Overview

- Representative use cases for co-simulation
  - Four discussed in some detail to motivate discussion that follows

- Simulation Prediction and Reproduction Accuracy

- Requirements for a PCS-Plant co-simulation

- Comparison of proposed PCS simulation alternative methods

- Conclusions and Recommendations
Co-Simulation Use Cases
Representative use cases for co-simulation

- Many co-sim use cases defined to support ITER PCS development/use
- Four discussed here:
  - Pulse schedule development
  - Pulse schedule validation
  - Control problem diagnosis
  - Control system development
- Could be supported using PCS code itself or a PCS model in Simulink.
  - Both imply requirements on real-time PCS development and long-term maintenance
- Discussion focus is ITER, but most issues relevant to existing devices
Use case 1: Pulse schedule development

- Develop pulse schedule = define reference trajectories and specify controller parameters
  - Trial & error process requires use of PCS-Plant co-simulations
  - Long pulse => re-tune references & parameters over time
    - Tune near start then progress forward in time => multiple simulations
  - Later phase control depends on Plant/PCS states at end of prior phase.
    - Must archive & restore Plant/PCS states to enable simulation re-start.

- Requires predictive PCS-Plant simulation.

- Co-simulation routinely used to develop DIII-D pulse schedule, especially for unusual or difficult control.
Use case 2: Pulse schedule validation

• Pulse schedule validation = evaluate pulse schedule for risk to device.

• ITER schedule validation method not officially defined, envisioned as two-step process:
  – Execute PCS-Plant co-simulation with PCS input = pulse schedule under test.
  – Evaluate output signals to verify pulse risk is low.

• Accurate prediction critical to ensure only low-risk pulses pass validation and enter operation.

• DIII-D does not require pulse validation prior to experimental use.
  – But, simulation results often examined to verify control in DIII-D operational constraints.
Use case 3: Control problem diagnosis

- Diagnosing control problem appearing during experiment typically requires reproducing then isolating it.

- **First check – is it a PCS bug?**
  - Feed experiment data into PCS-Sim to isolate specific time step(s) when behavior occurred.
    - To **guarantee behavior again observed, PCS-Sim must provide exact reproduction** of real-time calculations.
    - Then isolate problem source and analyze to determine cause.

- **If process verifies correct PCS execution, then problem is either**
  A. Unexpected change in behavior of Plant system, or
  B. Inadequate selection of control parameters (e.g., gains).
Plant change (A) versus inadequate control parameters (B)

• Distinguish - run PCS-Plant co-sim with experiment pulse schedule.
  • A: If plant simulation behaves differently from experiment, then:
    – Run multiple PCS-Plant co-simulations against Plant model
      • With Plant model changes to test candidate problems.
  • B: If not, problem = control parameters => need improvement:
    – Run PCS-Plant co-simulation to evaluate new parameter settings.

• If long discharge & PCS-Sim can't initiate at $t > 0$, diagnosing can take multiple hours/days.
  – ITER operating procedure likely to suspend operation during that time.

• DIII-D control problems often diagnosed using simulation.
  – Most common = PCS bugs, then inadequate control gains, only a few unexpected Plant changes.
    • For some Plant changes, operation cannot proceed until problem fixed.
Use case 4: Control system development

- Simulation use depends on development model for developing, testing, & implementing in PCS.

- Control development model is sequence of activities:
  1. Controller development
  2. Controller validation
  3. Controller implementation in PCS
  4. Controller verification in PCS
  5. PCS verification
  6. PCS validation

(But can change order, e.g., if entire PCS code generated from Simulink.)

- All non-trivial DIII-D control built first in development environment, e.g., Matlab/Simulink.
  - Most new controllers tested with co-simulation.
Simulation Prediction and Reproduction Accuracy
Need accurate Simulation Prediction & Reproduction (P/R)

• Primary simulation objective = reproduce past / predict future discharge behavior.

• P/R accuracy matters greatly, but no simple metric exists.
• DIII-D uses vague "good enough" criterion based on experience.
  – DIII-D PCS architecture precludes doing better than that.
• This won't work for ITER - effective simulation needed long before experiments.
  – More precise and demanding criterion required.

• Required P/R accuracy for ITER depends on the use case.
  – Extreme accuracy only needed by PCS-Sim and only some use cases.
  – Pulse schedule validation has demanding requirements - key component of device-protection first line of defense.
  – Pulse schedule & control system development have looser P/R requirements.
Defining & Achieving Simulation P/R Accuracy

• How to define metric for P/R accuracy?

• **Exact P/R of PCS execution - exactly predict/reproduce PCS output**
  - Can be achieved and obviate need to define metric for PCS-Sim
  - BUT, imposes requirements on PCS architecture

• Executing PCS code on real-time hardware cannot provide it.
  - Calculation-time dependencies are created by transferring data between separate PCS processes.
  - Can’t reproduce these when connected to non-realtime simulation without explicit provisions in PCS architecture.
  - This is what limits DIII-D PCS P/R capability.
How PCS calculation-time dependency can occur

- Fast-sampled process A needs input from slow-sampled process B - processes not synchronized

- B sends data to A when computed.
- Two sources of ambiguity:
  - Does A operate on data available at cycle start or data present later in cycle?
    - These data can be different.
  - Time during process B cycle when data sent is variable (magnitude of variability amplified for illustration).
Resolving PCS calculation time dependencies

- How to know which B output actually processed by A?

- Remove B output variability – require transmission only at cycle end.

- Still not clear which B data sample used by A.
- Require PCS processes use only data at cycle start, then clear on which data A should operate.
- How to enforce in simulation?
  - Simulated process calculation time not equal to real calculation.

- Some approaches for handling this – example in paper.
- Main point: To predict/reproduce PCS execution, execution must be reproducible.
  - Each such issue may => another requirement on real-time PCS.
Can't avoid limits on Plant Simulation P/R accuracy

- **Must define** quantifiable P/R accuracy metric(s), e.g., prediction error bound(s).
  - Needed to make decisions

- **Example - pulse validation:**
  - Add **error bounds** to Sim-predicted signals
  - Compare with threshold - accept/reject candidate pulse schedule

- **Avoid conservative operation** => stringent accuracy requirement
  - Feedback is forgiving => decreases Plant–Sim P/R requirement.

- **Reduce or eliminate inaccuracy whenever possible. Candidates:**
  - PCS - exact P/R is possible
  - Actuator and diagnostic models - very accurate P/R is possible.
Requirements for PCS-Plant Co-simulation
Requirements for PCS-Plant co-simulation

- **Co-simulation** => PCS-Sim & Plant-Sim execute separately.
  - Each evolves its own simulated time.
- **Simulated times must be equal when data exchanged.**
  - When diagnostic data sampled or commands transmitted by PCS.
  - Only a minimum requirement.
- Each Sim must ensure its P/R as accurate as possible.
- Both Plant and PCS contain multiple sub-processes, each with own characteristic time scale.
  - Must simulate time-dependent data flow between sub-processes.
  - Plant-Sim: Simulink (in case of PCSSP) handles it.
  - PCS-Sim: Method must correctly simulate interactions.
  - Calculation-time dependency => requirements on real-time PCS.
PCS-Sim Requirements for accurate P/R

- To ensure accurate P/R, when starting simulation at t=0:
  - Exchange data with external Plant-Sim
  - Synchronize simulated times of PCS-Sim & Plant-Sim
  - Emulate time-dependent data flows between PCS and Plant
  - Emulate time-dependent data flows between PCS processes

- To ensure accurate P/R, when starting simulation at t>0:
  - Start execution of Plant-Sim & PCS-Sim at user-specified time.
  - Load and set PCS state when starting PCS-Sim.
Some PCS Requirements for accurate P/R

- Archive all samples of input signals used by PCS.
- Archive internal variables representing PCS state at selected times.
- Provide predictable inter-process data flow.

- Other PCS requirements depend on method used for PCS-Sim.

- Other requirements for Sims & PCS for other use cases
  - No significant additional impact on P/R accuracy
Comparison of proposed PCS-Sim alternative methods
Advantages & disadvantages of ITER PCS-Sim options

• **Execute PCS code itself in "simulation mode"**
  – **Primary advantage:**
    • Extensive experience with approach (DIII-D, EAST, KSTAR, NSTX-U).
  – **Disadvantage:**
    • *Additional requirements on PCS (& underlying Real-Time Framework (RTF))*

• **Construct high-fidelity PCS model in PCSSP**
  – **Advantage:**
    • Develop/maintain PCSSP controller = develop/maintain PCS code (generated from PCSSP)
    – Note - RTF not generated from PCSSP
  – **Disadvantages:**
    • No experience that combines scale (entire PCS) and required P/R accuracy.
    • *Real-time PCS requirements not significantly less than PCS code method.*
Substantial uncertainty remains for ITER simulation

• Details to ensure accurate simulation P/R not all known.

• Significant uncertainty remains regarding:
  – Required P/R accuracy for essential ITER use cases
  – Best PCS-Sim method for providing P/R accuracy
  – Resulting PCS requirements necessary to support simulation
Conclusions
Conclusions and Recommendations

- Use case analysis => need for clear definition of desired ITER simulation capability & resulting requirements.
  - Advantages & disadvantages discussed for two PCS-Sim methods.
    - Both => requirements on real-time PCS to support off-line simulation.
  - Recommendation for ITER: Initiate a formal process to:
    - Define essential use cases.
    - Determine required P/R accuracy for essential use cases.
    - Propose accuracy metric(s) applicable to all use cases.
    - Further elaborate pros/cons of candidate methods.
    - Define requirements on real-time PCS and RTF.
Backups
Why is predictable inter-process data flow necessary?

• Calculation-time dependency of real-time not the same in Sim.
  – Different clock times for PCS process calculation.
  – Artificial method needed to ensure correct time dependency.

• Plant-Sim execution & I/O exchange take time, but PCS designed for real-time
  ⇒ Many PCS processes done by next PCS-Plant data exchange time.
  ⇒ Slow controllers outputs available before they would be in real time.
  ⇒ Mechanism needed in PCS-Sim to delay outputs to time when they would be used in real time.
    ⇒ Need to know when they would be used in real-time.

• Similar issue for inter-process data transfers in the real-time PCS.
  – Same mechanism needed.
One possible method to support exact P/R in PCS-Sim

- **Impose additional requirements on ITER PCS real-time execution:**
  - Operate each PCS process with fixed time steps.
  - Each process uses only input available at cycle start.
  - Transmit commands to actuators after fixed delay from cycle start.
  - Synchronize processes to begin & end at integer multiples of $\Delta t_{\text{min}}$.
  - Send inter-process data less than $\Delta t_{\text{min}}$ before end of sender's cycle.
  - Guarantee arrival at receiver before end of sender's cycle.