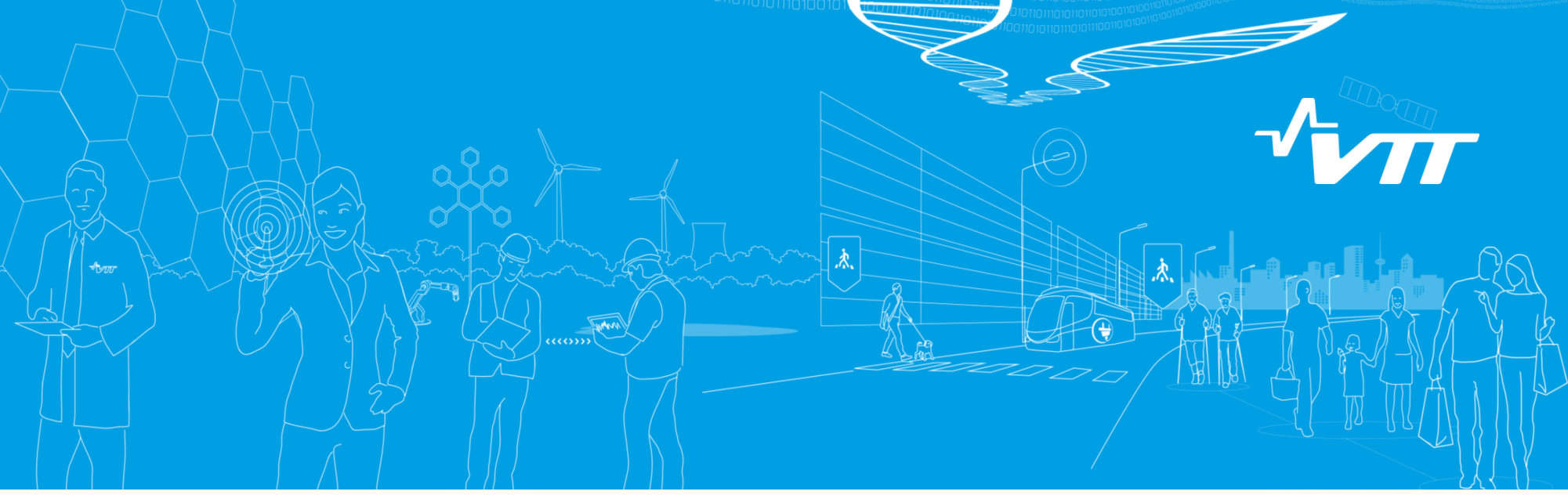


Coupled time dependent simulations with Serpent 2.1.27

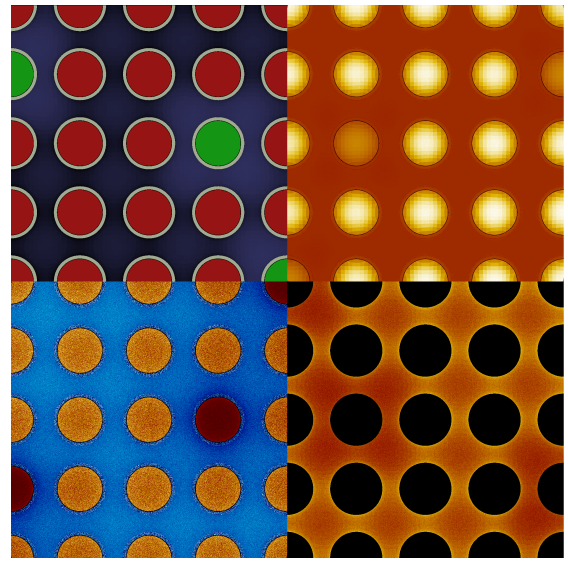
Serpent UGM, Politecnico di Milano
Sept. 26, 2016
V. Valtavirta
VTT Technical Research Center of Finland

Background

Coupled multi-physics simulations have been a part of Serpent 2 development.

Coupled simulations in steady state are starting to be routine.

Example, coupled neutronics CFD-calculations exhibiting fission power – fuel temperature – coolant temperature – coolant density coupling. $\rightarrow$



Dynamic simulation mode for problems without delayed neutron simulation has been available for a couple of years[1].

The simulation mode has been successfully applied to coupled super prompt critical fuel behavior – neutronics[2] and solid-mechanics – neutronics[3] problems.

---

[1] J. Leppänen. "*Development of a Dynamic Simulation Mode in Serpent 2 Monte Carlo Code.*" In proc. Proc. M&C 2013. Sun Valley, ID, May 2013.

[2] V. Valtavirta et al. "*Simulating Fast Transients with Fuel Behavior Feedback Using the Serpent 2 Monte Carlo code.*" In proc. Physor 2014. Kyoto, Japan, 2014.

[3] M. Aufiero et al. "*Serpent-OpenFOAM Coupling in Transient Mode: Simulation of a Godiva Prompt Critical Burst.*" In proc. Proc. M&C + SNA + MC 2015. Nashville, USA, 2015.

Background

Recently a new delayed neutron emission model was implemented in Serpent[4] enabling time dependent simulations for transients with significant effects from delayed neutrons.

The new simulation mode was applicable to coupled transients only by running a separate Serpent simulation for each time-interval.

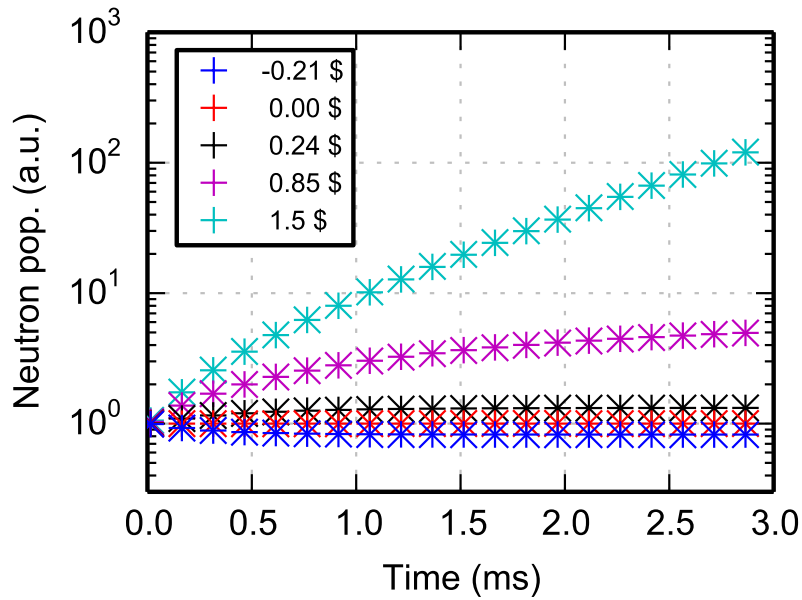


Figure 1: Comparison between theoretical prediction and Serpent results for infinite homogeneous reactor point-kinetics transients. Theoretical (+) and Serpent (x).

[4] V. Valtavirta, M. Hessan, and J. Leppänen. "*Delayed Neutron Emission Model for Time Dependent Simulations with the Serpent 2 Monte Carlo Code – First Results.*" In proc. Proc. Physor 2016. Sun Valley, ID, USA, 2016.

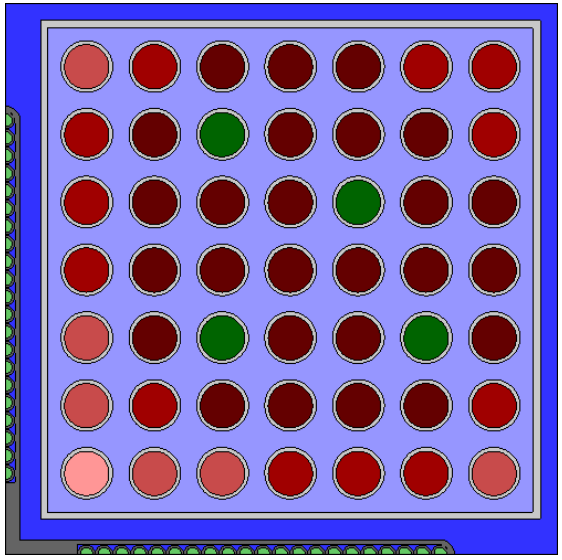
Outline

Some modifications were required to combine the time dependent simulation mode using delayed neutrons with the coupled transient calculation mode used before for super prompt critical transients.

Capability to use velocity and acceleration based universe transformations was added in order for easy simulation of control rod movement.

This presentation will:

1. Shortly describe the coupled transient calculation mode (compared to, e.g., original dynamic mode).
2. Describe the velocity and acceleration based universe transformations.
3. Show an example of a control rod drop-out initiated transient simulated using the internal FINIX fuel behavior module.



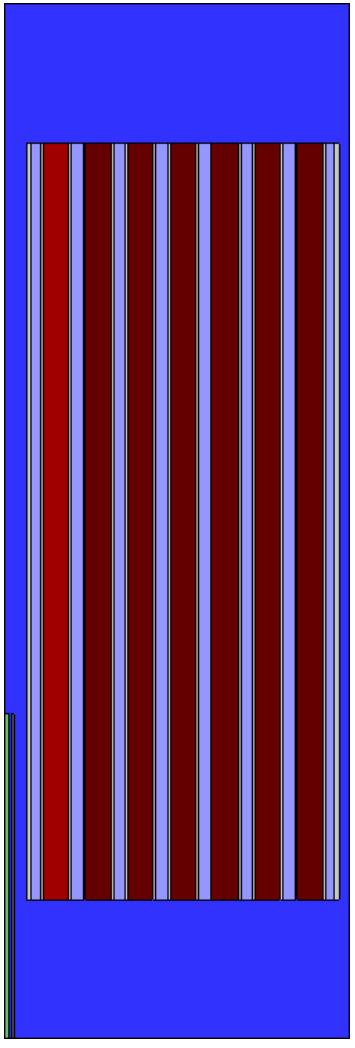
Outline

Some modifications were required to combine the time dependent simulation mode using delayed neutrons with the coupled transient calculation mode used before for super prompt critical transients.

Capability to use velocity and acceleration based universe transformations was added in order for easy simulation of control rod movement.

This presentation will:

1. Shortly describe the coupled transient calculation mode (compared to, e.g., external source mode).
2. Describe the velocity and acceleration based universe transformations.
3. Show an example of a control rod drop-out initiated transient simulated using the internal FINIX fuel behavior module.



Coupled dynamic simulation mode

Two main choices:

- Capability to exchange power / temperature / density data and apply transformations/deformations during the simulation.
- Capability to iterate a certain time-interval to obtain a tightly coupled solution.

Traditional dynamic mode:

0a: Divide source size into N_b batches.
0b: Divide simulation time into N_t time intervals.
1: **for** b = 1,. . . , N_b
2: **for** t = 1,. . . , N_t
3: Do population control for batch b.
4: Simulate batch b through time-interval t.
5: (Next time interval)
6: **end for**
7: Collect results from batch b for all time-intervals.
8: (Next batch)
9: **end for**
10: Collect and print results

Coupled dynamic simulation mode

Two main choices:

- Capability to exchange power / temperature / density data and apply transformations/deformations during the simulation.
- Capability to iterate a certain time-interval to obtain a tightly coupled solution.

Coupled dynamic mode (simplified):

0a: Divide source size into N_b batches.
0b: Divide simulation time into N_t time intervals.
1: **for** t = 1,. . . , N_t
2: **for** b = 1,. . . , N_b
3: Do population control for batch b.
4: Simulate batch b through time-interval t.
5: Collect results from batch b for time-interval t.
6: (Next batch)
7: **end for**
8: Collect and print results for time-interval t
9: (batch)
10: **end for**

Coupled dynamic simulation mode

Coupled dynamic mode (less simplified):

0a: Divide source size into N_b batches.
0b: Divide simulation time into N_t time intervals.
1: **for** t = 1,. . . , N_t
2: **while** iterating
3: **for** b = 1,. . . , N_b
4: Get batch b for beginning of time interval t.
5: Do population control for batch b.
6: Simulate batch b through time-interval t.
7: Collect results from batch b for time-interval t.
8: Store batch b for end of time interval t.
9: (Next batch)
10: **end for**
11 Collect, relax and print results for time-interval t.
12: Get updated solution from coupled code.
13: (Next iteration or finish)
14: **end while**
15: (Next time interval)
16: **end for**

Coupled dynamic simulation mode

Main differences to initial dynamic simulation mode:

- All batches are first simulated through first time interval[5].
- Output is limited to current time-interval, archiving of results during simulation is currently left to user.
- Time-intervals can be iterated for a tighter coupling.
- Local temperatures are interpolated between BOI and EOI values.
- Geometry and material densities currently use BOI values.

More information in separate presentation about coupled calculations with Serpent.

[5] Compare: First batch is first simulated through all time intervals

Moving geometry

Serpent supports various transformations applied to the geometry[6]

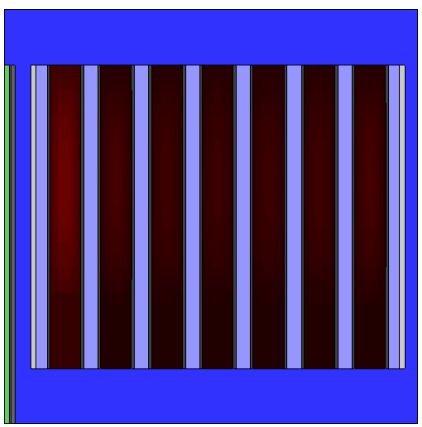
```
trans s ...  (surface transformation)
trans u ...  (universe transformation)
trans f ...  (fill transformation)
```

For example

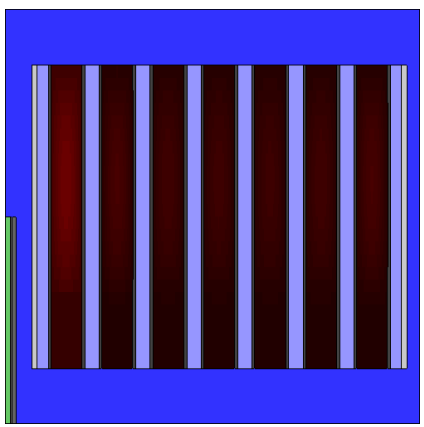
```
trans s s01 0.0 0.0 100
```

translates surface `s01` to the positive z-direction by 100 cm.

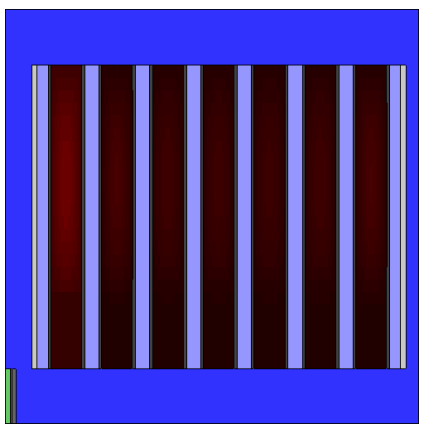
These transformations are useful, among other things, for moving control rod banks:



`trans s s2 0 0 182.88`



`trans s s2 0 0 0`



`trans s s2 0 0 -182.88`

[6] See the relevant part of the input syntax manual in the Serpent Wiki.

Moving geometry

New velocity and acceleration based transformations can be applied with

```
transv <type> <name> <vx0> <vy0> <vz0>
```

and

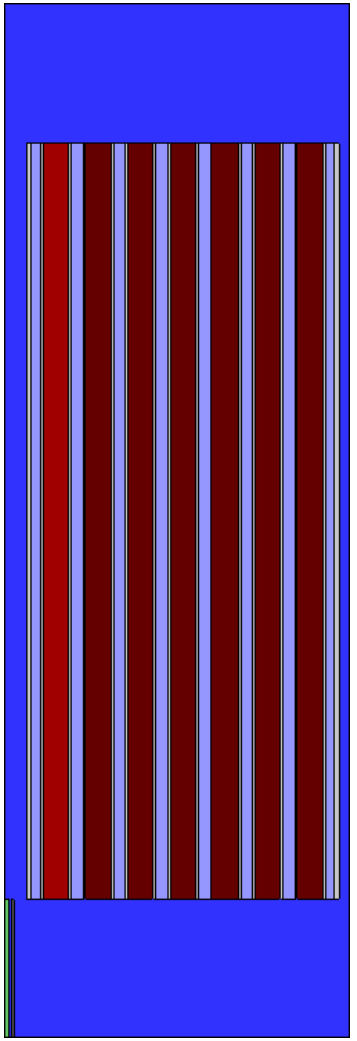
```
transa <type> <name> <ax0> <ay0> <az0>
```

For example

```
trans  s s2 0 0 -182.88
transv s s2 0 0  750.0
transa s s2 0 0 -981.0
```

will

translate surface `s2` downwards by $-182.88\,\mathrm{cm}$
give the surface an initial upwards velocity of $750\ \frac{\mathrm{cm}}{\mathrm{s}}$
and an initial downwards acceleration of $-981\ \frac{\mathrm{cm}}{\mathrm{s}^2}$.



Moving geometry

Implementation

Let's say a transformation of the form

```
trans  <type> <name> <dx0> <dy0> <dz0>
transv <type> <name> <vx0> <vy0> <vz0>
transa <type> <name> <ax0> <ay0> <az0>
```

is applied.

Reminder: The simulation time can be divided into time-intervals with population control at each interval boundary by linking a time binning (set up using the tme card) to the set nps option.

Then at time t at time interval $t \in [t_i, t_{i+1})$ the total transformation is calculated from

$$\Delta x = dx0 + vx0 \times t_i + \frac{1}{2} ax0 \times t_i^2$$

in the x-direction and similarly in the y- and z-directions. Multiple separate translations, velocities and accelerations can be applied if needed.

The movement happens at time-interval boundaries and the BOI geometry is used throughout the interval[7].

[7] Using the exact time t would also be possible, but has not been tested.

Moving geometry

Time limits for transformations

Sometimes it is nice to limit the movement to certain times. This can be done with

`trans<-/v/a> <type> <name> tlim <tBeg> <tEnd> <tType> <x0> <y0> <z0>`

The transformation will be active in the interval $t \in [t_{\mathrm{Beg}}, t_{\mathrm{End}})$ and the velocity and the acceleration transformations will be calculated as

$$\Delta x = vx0 \times (t - t_{\mathrm{Beg}})$$

and

$$\Delta x = \frac{1}{2} ax0 \times (t - t_{\mathrm{Beg}})^2$$

The effect is that of the velocity or acceleration starting to affect the system at t_{Beg}. What happens after t_{End} depends on the `<tType>` parameter.

Moving geometry

Time limits for transformations

Sometimes it is nice to limit the movement to certain times. This can be done with

`trans<-/v/a> <type> <name> tlim <tBeg> <tEnd> <tType> <x0> <y0> <z0>`

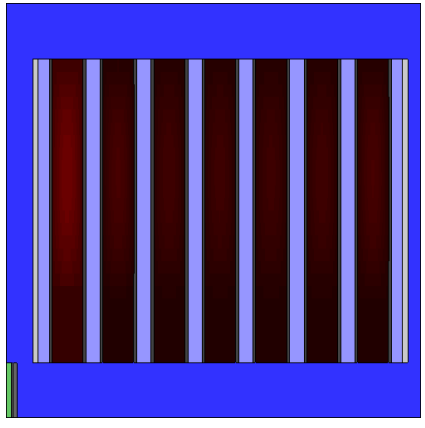
There are currently three possibilities for `<tType>` for controlling what happens after t_{End}:

`<tType>` = 1. Transformation stays as it is at t_{End} (movement just stops).
`<tType>` = 2. The transformation is removed (as if it never happened).
`<tType>` = 3. Like 1, but velocity accumulated during acceleration remains.

The `<tType>` = 2 option is probably not very useful.

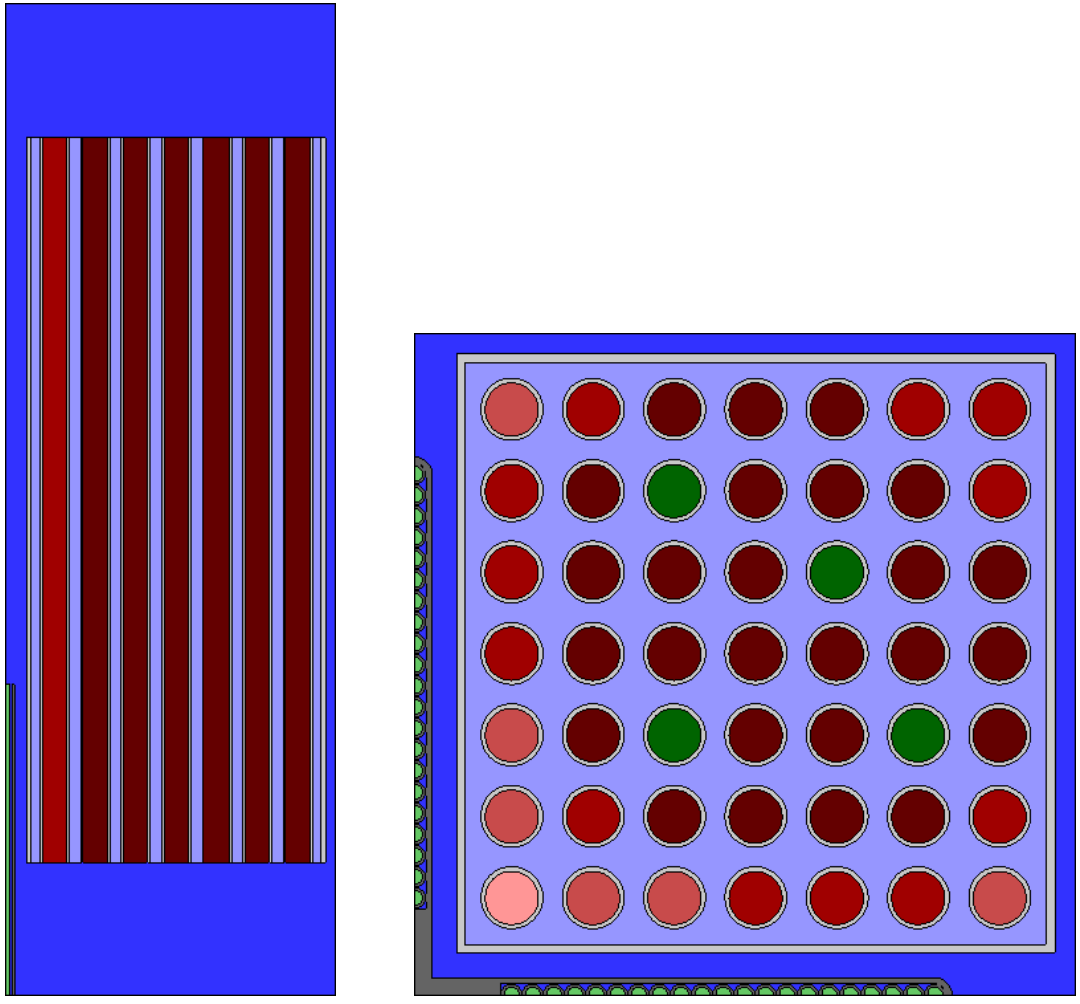
Example:

```
trans  s s2 0 0 -182.88
transa s s2 tlim 0.1 0.5 <tType> 0 0  1000.0
transa s s2 tlim 0.5 0.9 <tType> 0 0 -1000.0
```



Control rod drop transient

Peach Bottom 2 like BWR assembly with control rod inserted 90 cm above the bottom of active fuel. The transient starts from power level of 3 MW, neutronics is solved by Serpent and fuel behavior is solved by FINIX for 28 unique pin positions. Coolant behavior is ignored.



Control rod drop transient

System made critical at a high power level of 3 MW via iteration of coolant boron with multi-physics solution. The fuel temperature distribution corresponds to the power level, but the coolant TH-distribution is constant:

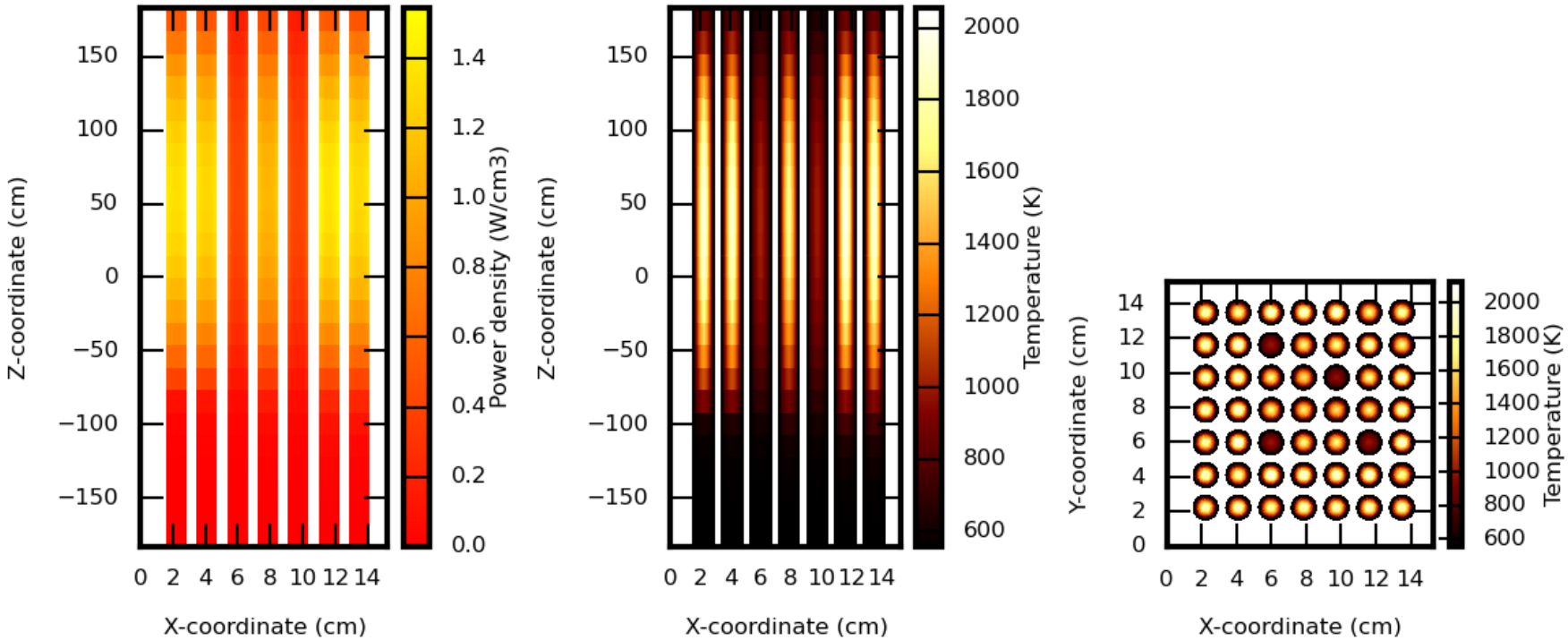


Figure 2: Initial power density and temperature distribution at power level of 3 MW.

The transient is initiated by letting the control rod fall from the system due to gravitational acceleration.

Control rod drop transient

The simulation has to be executed in several parts:

1. Obtain coupled critical solution for the steady state:
-Coupled Serpent-FINIX solution of steady state. (can cheat a bit)
-Change coolant boron to get closer to criticality.

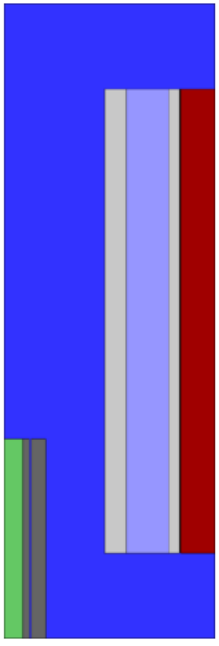
2. Generate initial source for the transient:
-Standalone Serpent using the steady state fuel behavior solution.
-Save live neutron source, and point-wise precursor source and normalization of both with set savesrc.

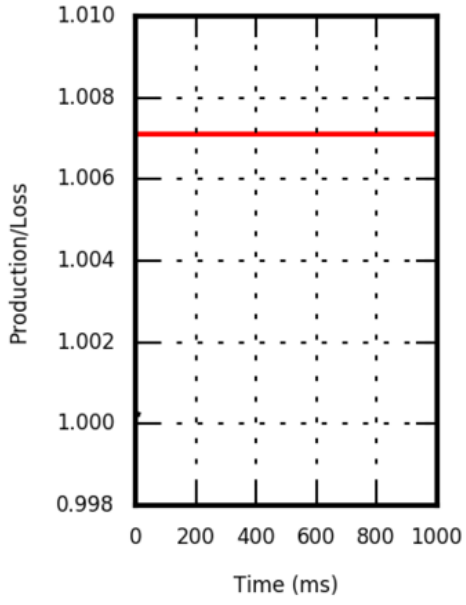
3. Run transient:
-Coupled Serpent-FINIX solution for each time interval.
-Iterate each time interval two times. (stupid)
-FINIX uses previous time interval power as an initial guess for upcoming time interval.

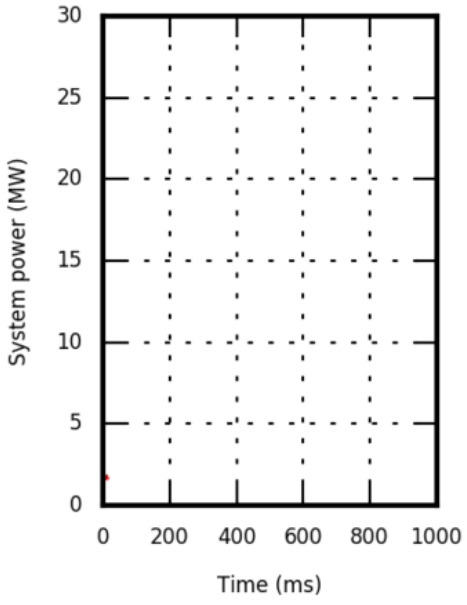
See Wiki:Transient simulations.

Separate talk covers Serpent-FINIX steady state and transient calculations.

Results 1/3







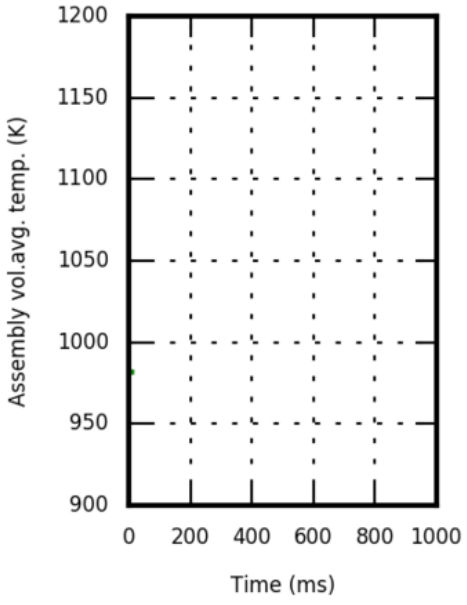
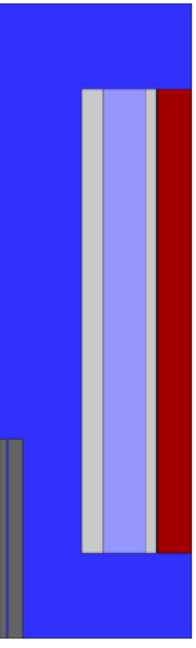
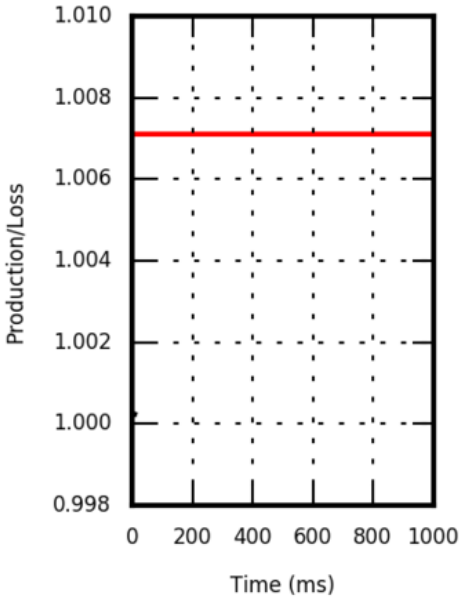
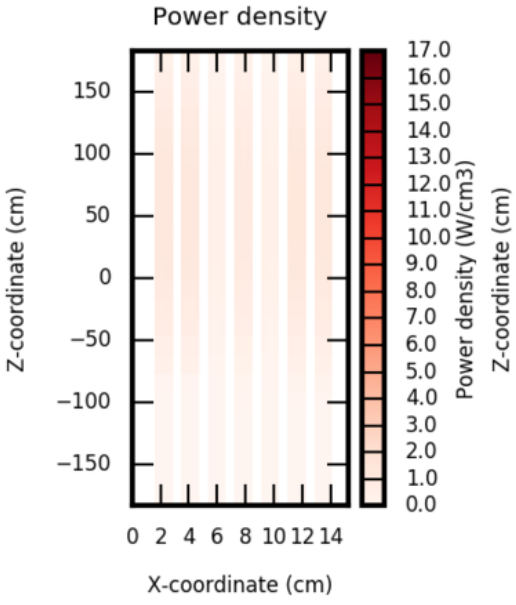


Figure 3: From left: **1.** Geometry plot. **2.** Momentary neutron production / loss. **3.** System power. **4.** Assembly volume averaged fuel temperature.

Results 2/3







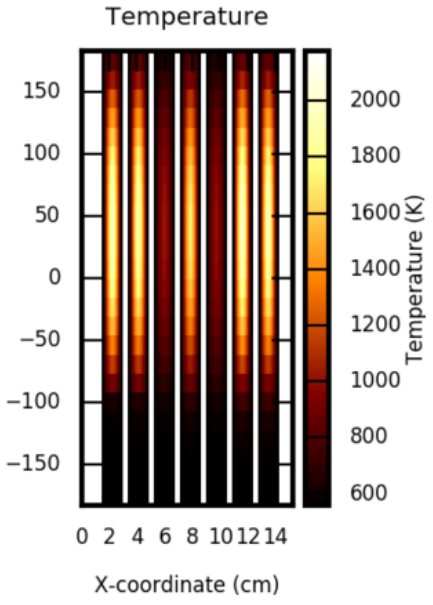
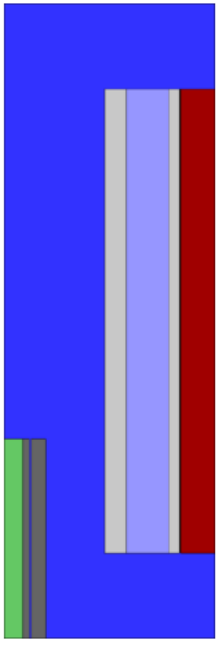
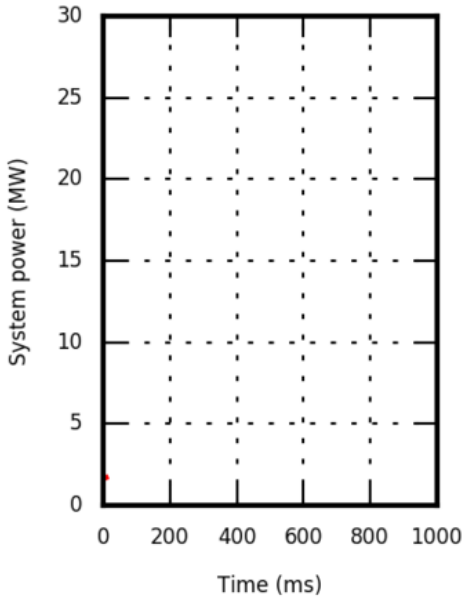
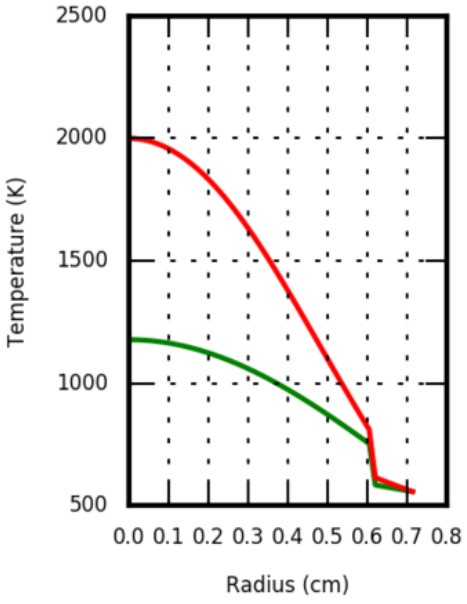


Figure 4: From left: **1.** Geometry plot. **2.** Momentary neutron production / loss. **3.** Power density in a diagonal cut through assembly. **4.** Temperature distribution in a diagonal cut through the assembly.

Results 3/3







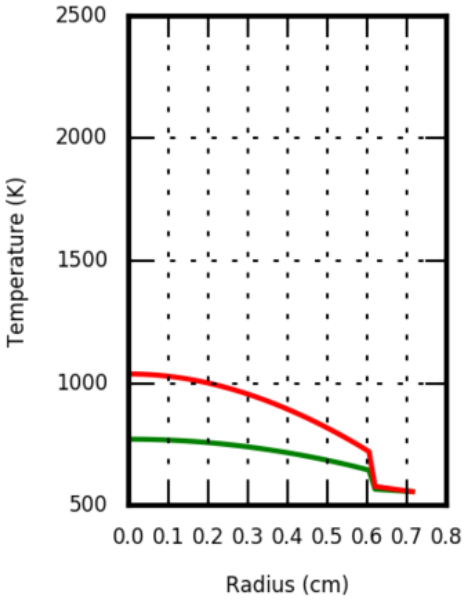


Figure 5: From left: **1.** Geometry plot. **2.** System power. **3.** Radial temperature profiles in axial layers 8/24 and 16/24. Corner rod. **4.** Radial temperature profiles in axial layers 8/24 and 16/24. BA-rod.

Comments on simulation

Conducting the simulation in three parts (criticality iteration, source generation, transient simulation) was quite straightforward.

Transient simulation took most of the time (150 hours on a 32 core cluster node).

The standard relative errors on local power tallies are lost due to the power relaxation but are probably quite high.

Source generation took 4 hours with 32 cores. The generated source contained 112 million live neutrons and 6 million precursors. Compare to 0.6 million initial neutrons simulated for first interval and 120 million primary neutrons sampled throughout the simulation.

Smart timestep iteration could have saved maybe 1/3 of the simulation time. Application of convergence criteria should be implemented to FINIX simulations.

Final verdict: Coupled transient simulations for small systems and short timescales (N seconds) are computationally expensive but plausible.

Limitations

The fission power estimate is currently based on the -8 response function. Time dependent decay heat is not explicitly modeled.

Velocity and acceleration based transformations are only used for linear velocity and acceleration (no rotating systems at the moment).

Temperature and density of materials can change in time dependent fashion, but material compositions currently cannot[8]

[8] Delayed neutron precursors *are* tracked separately.

Summary

Serpent 2.1.27 can model coupled transient scenarios starting from a critical system.

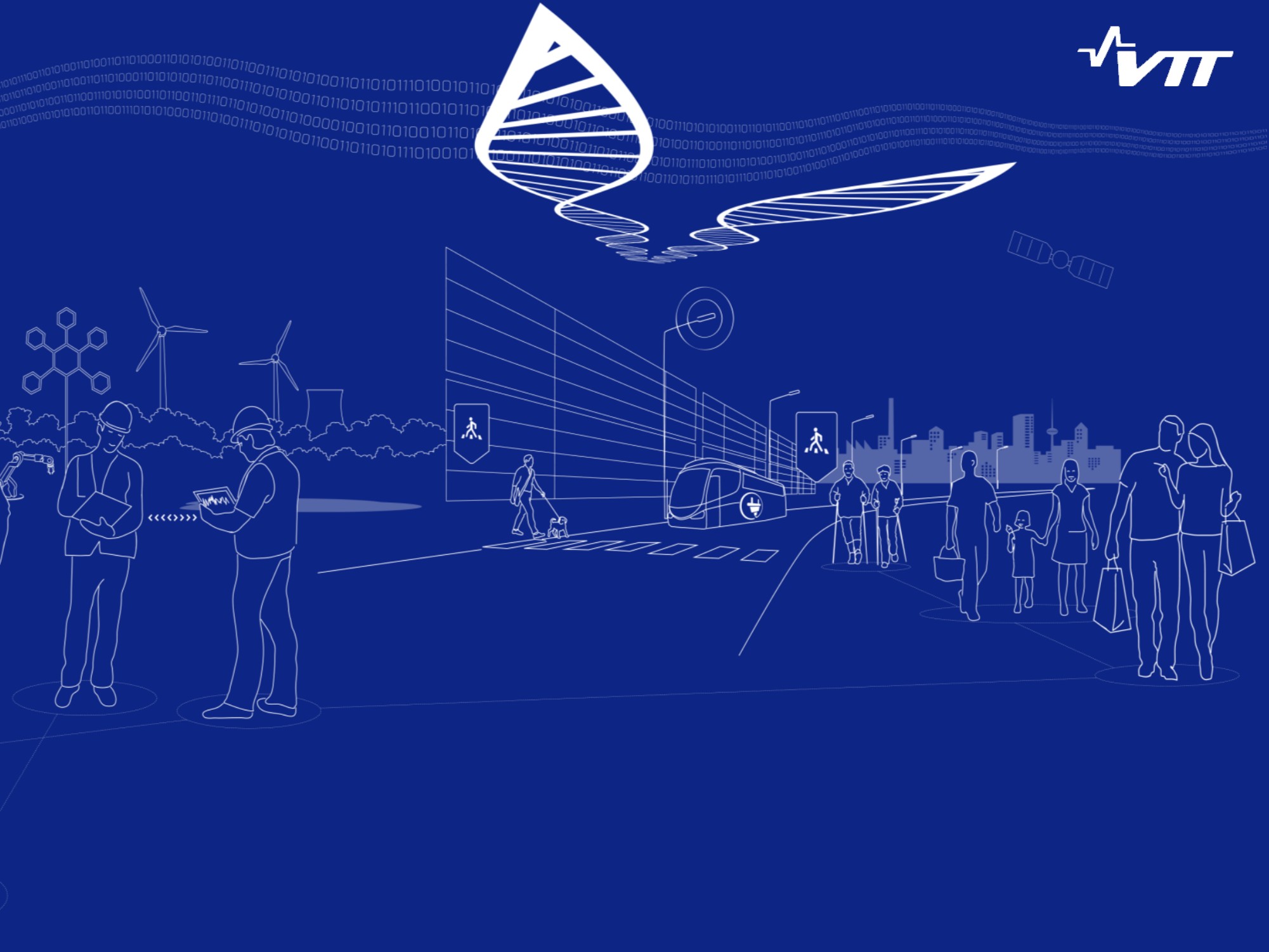
This requires the combination of multiple recent developments:

New time dependent delayed neutron emission model.
Revised coupled transient calculation routine.
Moving geometry transformations.

The new capabilities will be validated against the SPERT-III E-core (LWR fuel) transients and TRIGA pulse data obtained from the Jožef Stefan Institute.

Modeling the validation cases may bring up some additional features required for realistic coupled transient simulations.

You can try it yourself. Instructions for coupled transient modeling using the Minimal Serpent Coupling Script will be in a separate presentation.



M. Aufiero et al. "*Serpent-OpenFOAM Coupling in Transient Mode: Simulation of a Godiva Prompt Critical Burst.*" In proc. Proc. M&C + SNA + MC 2015. Nashville, USA, (2015).

J. Leppänen. "*Development of a Dynamic Simulation Mode in Serpent 2 Monte Carlo Code.*" In proc. Proc. M&C 2013. Sun Valley, ID, (May 2013).

V. Valtavirta, M. Hessan, and J. Leppänen. "*Delayed Neutron Emission Model for Time Dependent Simulations with the Serpent 2 Monte Carlo Code – First Results.*" In proc. Proc. Physor 2016. Sun Valley, ID, USA, (2016).

V. Valtavirta et al. "*Simulating Fast Transients with Fuel Behavior Feedback Using the Serpent 2 Monte Carlo code.*" In proc. Physor 2014. Kyoto, Japan, (2014).