Fine Flux Integration Methods for the SPN Solver in Cocagne

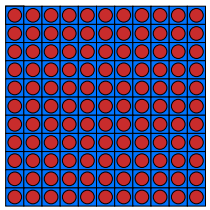
D. Couyras, F. Févotte, L. Plagne

M&C 2013, Sun Valley, ID, USA

EDF

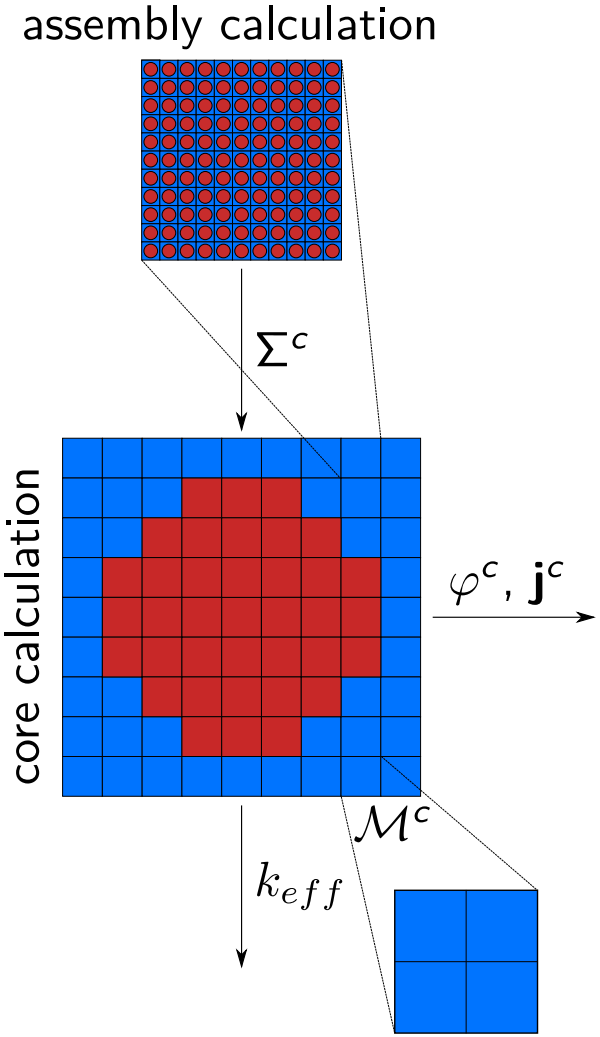
2-step calculation scheme

assembly calculation



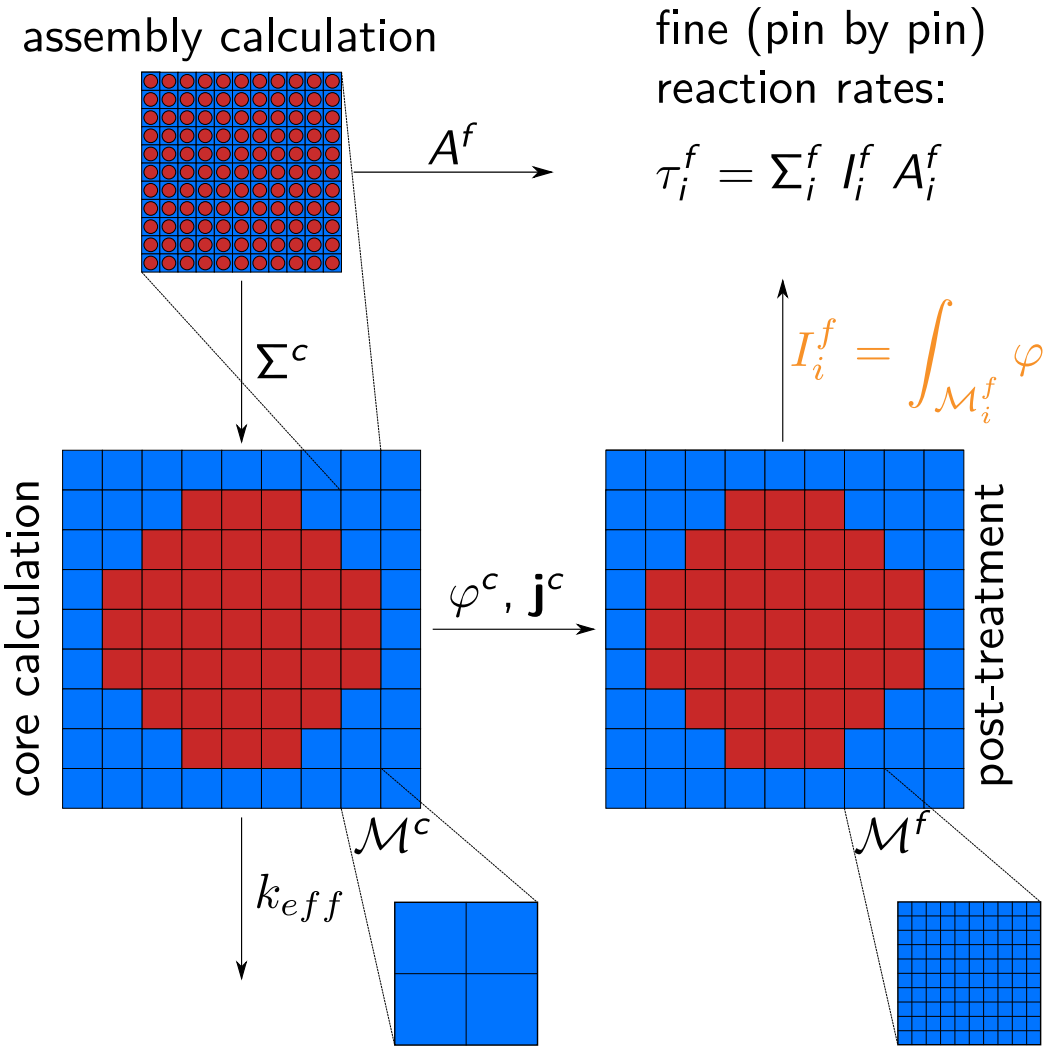
- 2D Assembly calculation
  - **APOLLO2 - REL2005**
  - 281 energy groups
  - very fine spatial discretization
  - Σ: homogenized & collapsed cross sections

2-step calculation scheme



- 2D Assembly calculation
  - very fine spatial discretization
- 3D Core calculation
  - **COCAGNE SPN solver**
  - 2 energy groups
  - coarse spatial mesh $\mathcal{M}^c$ (1×1 to 4×4 meshes/ass.)
  - k_{eff}: multiplication factor
  - φ^c, $\mathbf{j}^c$: coarse flux, current

2-step calculation scheme



- 2D Assembly calculation
  - very fine spatial discretization
- 3D Core calculation
  - coarse spatial mesh $\mathcal{M}^c$ (1×1 to 4×4 meshes/ass.)
  - φ^c, $\mathbf{j}^c$: coarse flux, current
- Post-treatment
  - fine spatial mesh $\mathcal{M}^f$ (pin-by-pin)
  - **I^f: pin-by-pin-averaged flux**
  - A^f: fine shape function
  - τ^f: pin-by-pin reaction rate

Outline

- Context
  - DIABOLO SPN Solver
  - Present fine flux integration method
- New Fine Flux Integration methods
  - Poisson
  - StrawHat
- Real-World results
  - UOX cases
  - UOX/MOX interfaces
- Conclusions – Perspectives

COCAGNE's SPN solver: DIABOLO

Equations and implementation

- Cartesian solver, for the Simplified Transport (SPN) equations

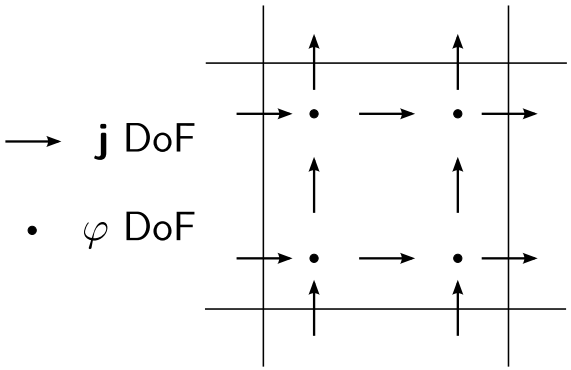
- SP_1 equations:

$$\begin{cases} \dfrac{1}{D}\,\mathbf{j} + \nabla\varphi = 0, \\ \operatorname{div}(\mathbf{j}) + \Sigma\,\varphi = s. \end{cases}$$

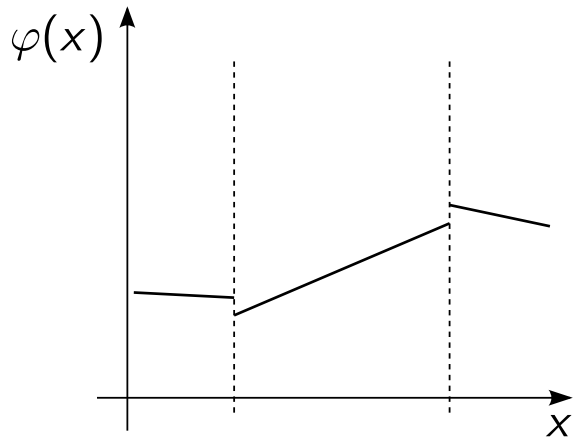
- Discretized system with RT_k FEM:

$$\begin{cases} \mathbb{A}_k\, J - \mathbb{B}_k\, \Phi = 0, \\ {}^t\mathbb{B}_k\, J + \mathbb{T}_k\, \Phi = S_k. \end{cases}$$

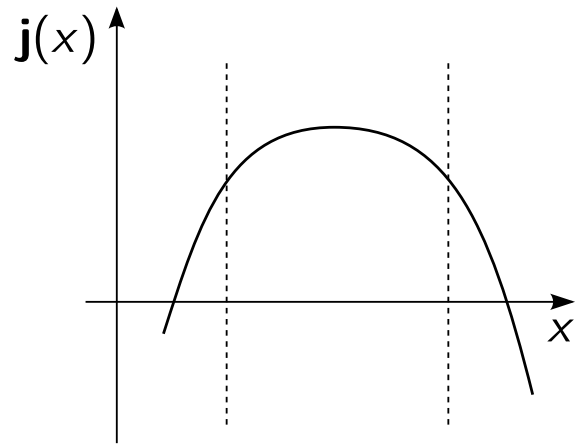
- RT_1 DoFs in a mesh cell:



- RT_1 flux:



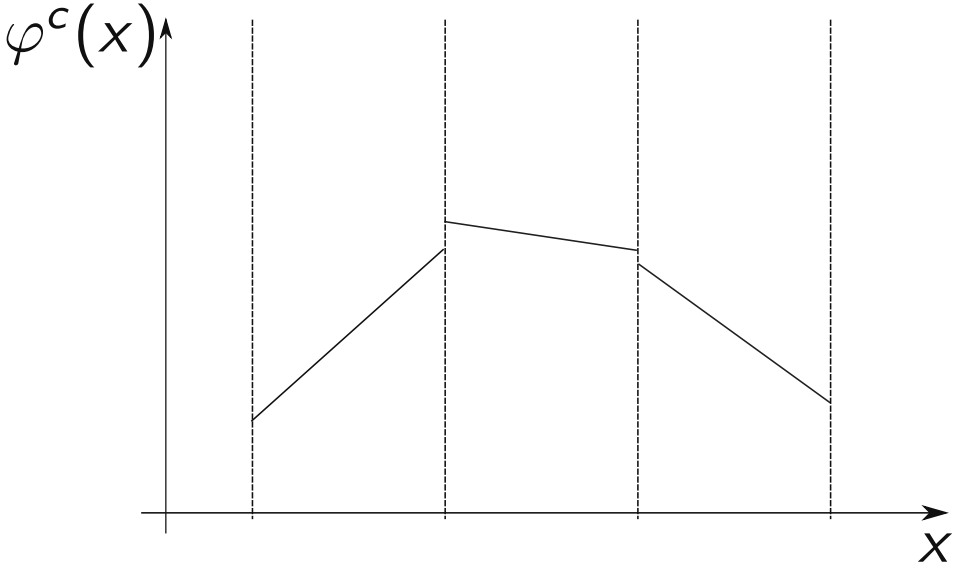
- RT_1 current:



Direct flux integration method

Principle

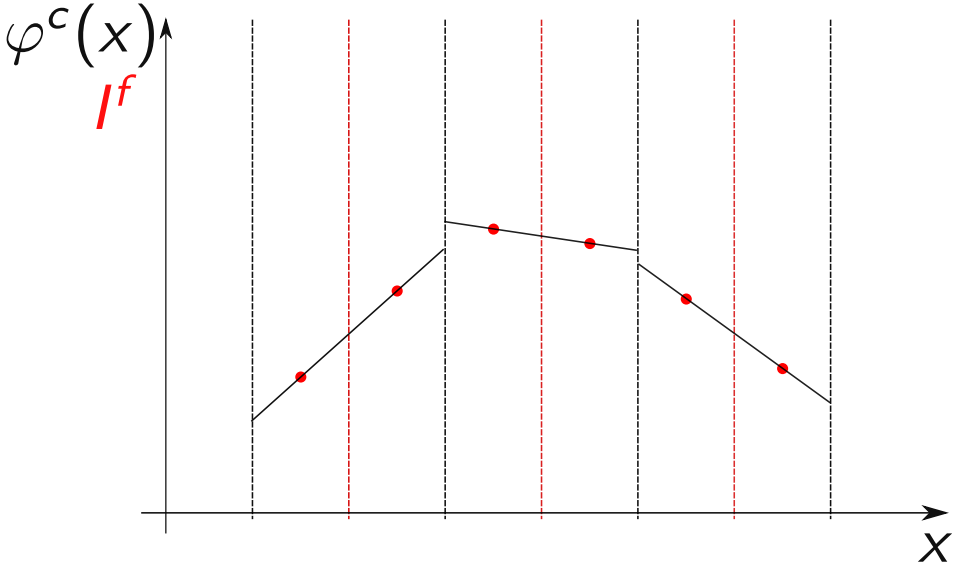
- Direct integration of the coarse flux
  - Example of a 1D RT_1 coarse flux:



Direct flux integration method

Principle

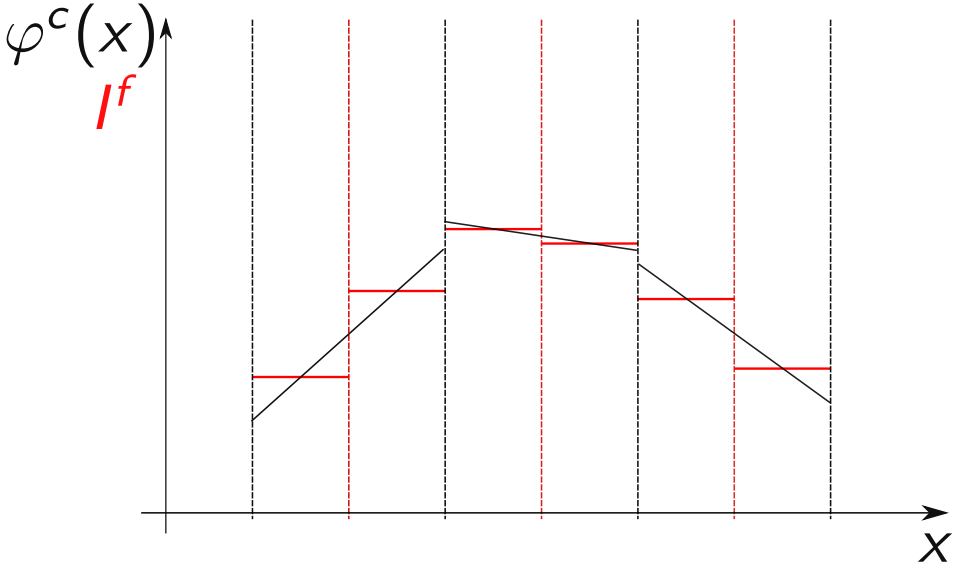
- Direct integration of the coarse flux
  - Example of a 1D RT_1 coarse flux: $I_i^f = \frac{1}{|\mathcal{M}_i^f|} \int_{\mathcal{M}_i^f} \varphi^c(x)\, dx$



Direct flux integration method

Principle

- Direct integration of the coarse flux
  - Example of a 1D RT_1 coarse flux:



$$I_i^f = \frac{1}{|\mathcal{M}_i^f|}\int_{\mathcal{M}_i^f}\varphi^c(x)\,dx$$

$$= \frac{1}{|\mathcal{M}_i^f|}\int \varphi^c(x)\ \underbrace{\mathbb{1}_{\mathcal{M}_i^f}(x)}_{RT_0 \text{ basis function}}\ dx$$

$\Rightarrow$ I_i^f can be identified to an RT_0 flux discretization on $\mathcal{M}^f$

Let us try and solve the following fine RT_0 SP_1 system:

$$\begin{cases} \mathbb{A}_0^f J - \mathbb{B}_0^f \Phi = 0, \\ {}^t\mathbb{B}_0^f J + \mathbb{T}_0^f \Phi = S_0^f. \end{cases}$$

Fine Flux Integration

Fine system resolution

- Solve the fine SP_1 RT_0 system:

$$\begin{cases} \mathbb{A}_0^f J - \mathbb{B}_0^f \Phi = 0, \\ {}^t\mathbb{B}_0^f J + \mathbb{T}_0^f \Phi = S_0^f. \end{cases}$$

- **Question:**
  Could we avoid entirely solving this using the coarse solution (Φ^c, J^c)?

  - project/interpolate (Φ^c, J^c) on $\mathcal{M}^f$ to initialize the fine solver
    $\rightarrow$ open the way to multi-level/multigrid methods; see the perspectives.
  - **solve only subset of the problem at the fine level**
    $\rightarrow$ solutions explored in the following.

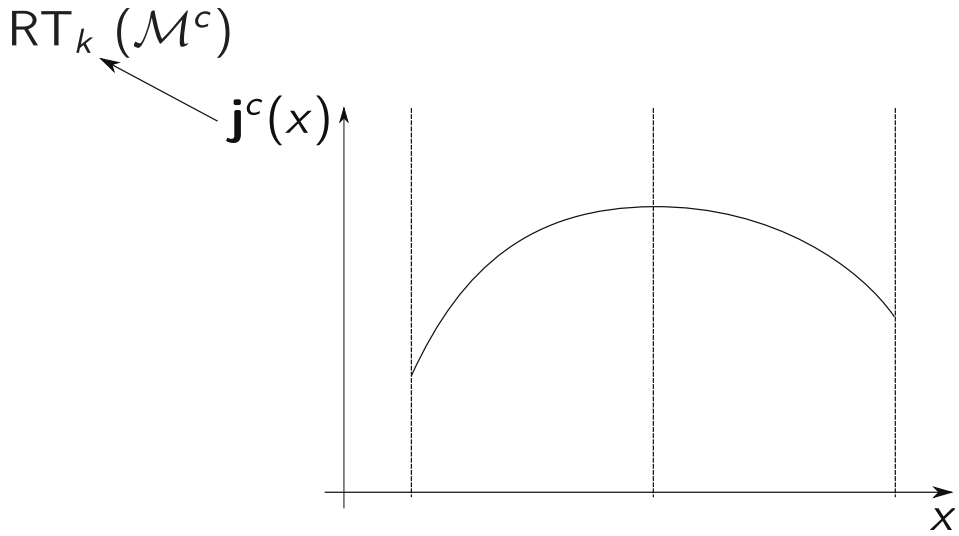
Current-based integration methods

Principle

$$\begin{cases} \mathbb{A}_0^f \, J^f - \mathbb{B}_0^f \, \Phi^f = 0 \\ {}^t\mathbb{B}_0^f \, J^f + \mathbb{T}_0^f \, \Phi^f = \mathcal{S}_0^f \end{cases}$$

- **Idea**:
  - project the current only, and treat it as a known source term;
  - ignore the last set of equations:

$$\mathbb{B}_0^f \, \Phi^f = \mathbb{A}_0^f \, J_{proj}^f$$



- **Advantage**:
  - $\mathbb{B}$ does not contain any physical data.

- **Problem**:
  - this system is not square!

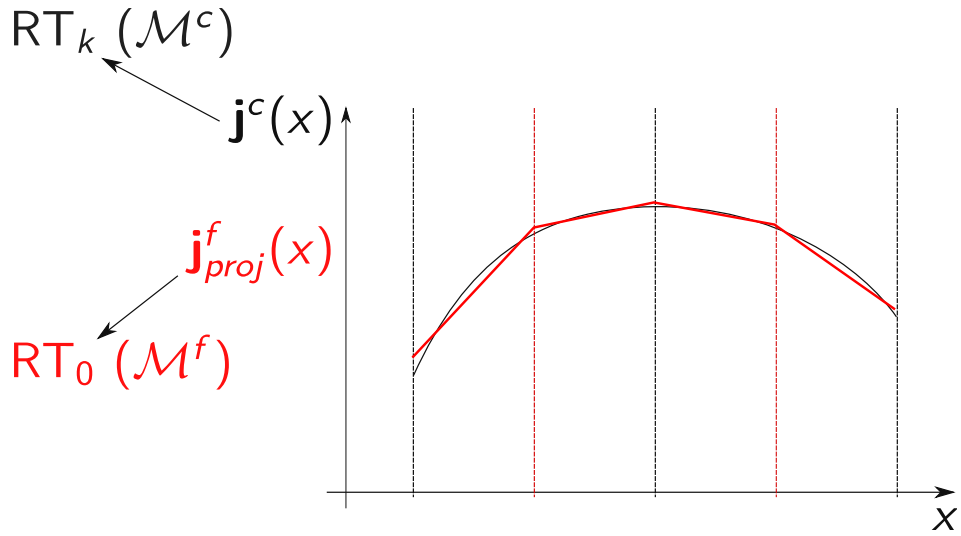
Current-based integration methods

Principle

$$\begin{cases} \mathbb{A}_0^f \, J^f - \mathbb{B}_0^f \, \Phi^f = 0 \\ {}^t\mathbb{B}_0^f \, J^f + \mathbb{T}_0^f \, \Phi^f = \mathcal{S}_0^f \end{cases}$$

- **Idea**:
  - project the current only, and treat it as a known source term;
  - ignore the last set of equations:

$$\mathbb{B}_0^f \, \Phi^f = \mathbb{A}_0^f \, J_{proj}^f$$



- **Advantage**:
  - $\mathbb{B}$ does not contain any physical data.

- **Problem**:
  - this system is not square!

Current-based integration methods

Poisson method

- **Idea:** left-multiply the equation by ${}^t\mathbb{B}$:

$$\mathbb{B}\,\Phi \quad = \quad \mathbb{A}\,J_{proj}$$

Current-based integration methods

Poisson method

- **Idea:** left-multiply the equation by ${}^t\mathbb{B}$:

$$ {}^t\mathbb{B}\,\mathbb{B}\,\Phi_{poisson} = {}^t\mathbb{B}\,\mathbb{A}\,J_{proj} $$

- **Advantages:**
  - ${}^t\mathbb{B}\,\mathbb{B}$ is the classical finite-differences discretization of the Laplace operator;
  - there exist very efficient methods to solve it.

For example, in 1D:

$$ \mathbb{B} = \begin{pmatrix} 1 & & \\ -1 & \ddots & \\ & \ddots & 1 \\ & & -1 \end{pmatrix} \qquad {}^t\mathbb{B}\,\mathbb{B} = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} $$

Current-based integration methods

"StrawHat" method

- **Idea:** ignore the last equation in the system:

$$\mathbb{B}\,\Phi \qquad = \mathbb{A}\,J_{proj}$$

$$\mathbb{B} = \begin{pmatrix} 1 & & & \\ -1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{pmatrix}$$

Current-based integration methods

"StrawHat" method

- **Idea:** ignore the last equation in the system:

$$\tilde{\mathbb{B}}\,\Phi_{strawhat} = \widetilde{\mathbb{A}\,J_{proj}}$$

- **Advantage:**
  - $\tilde{\mathbb{B}}$ is lower triangular.

$$\mathbb{B} = \begin{pmatrix} 1 & & & \\ -1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{pmatrix} \qquad \tilde{\mathbb{B}} = \begin{pmatrix} 1 & & & \\ -1 & \ddots & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}$$

Real-World Results

Real-World results: 3D PWR core computation

Experimental setup

- **Spatial discretization**:

	(x, y)	z
Fine mesh $\mathcal{M}^f$	17×17 cells/ass.	40 cells
Computation mesh $\mathcal{M}^c$	$(1 \times 1) \rightarrow (8 \times 8)$	40 cells
Discretization order	$\mathrm{RT}_0 \rightarrow \mathrm{RT}_2$	RT_2

- **Physical data**: 2-group cross-sections, homogenized at the assembly level.

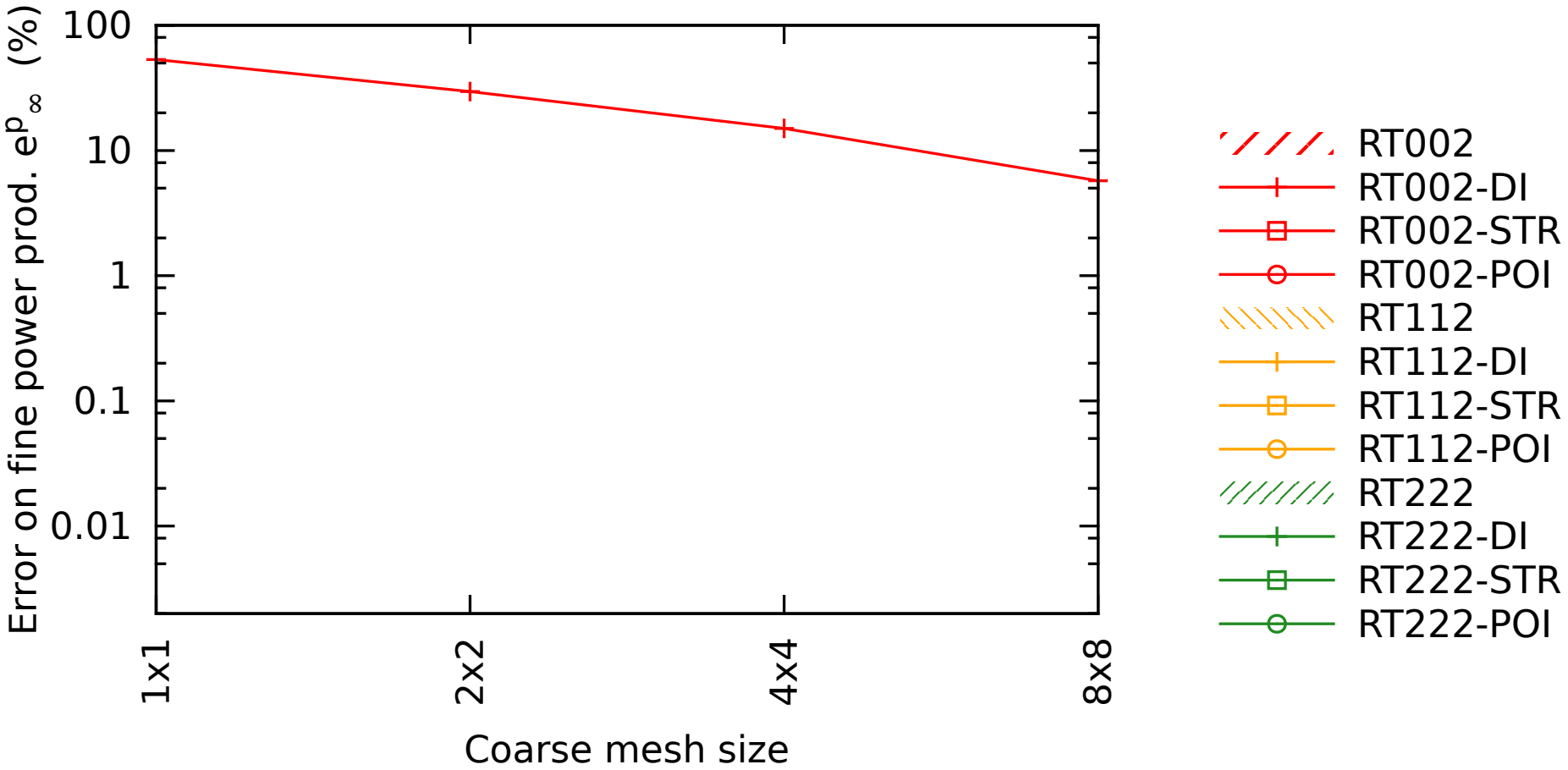
  2 datasets coming from real 900 MWe PWR loading patterns:
  - UOX only
  - UOX + MOX

- **Reference calculation**:
  - fine SP_1 RT_2 computation
  - quantity of interest: **fine power production rates**

$$P^f = \sum_g \kappa \, \Sigma_f(g) \, I^f(g) \qquad e^p_\infty = \frac{\| P^f - P^{f,ref} \|_\infty}{\| P^{ref} \|_\infty}$$

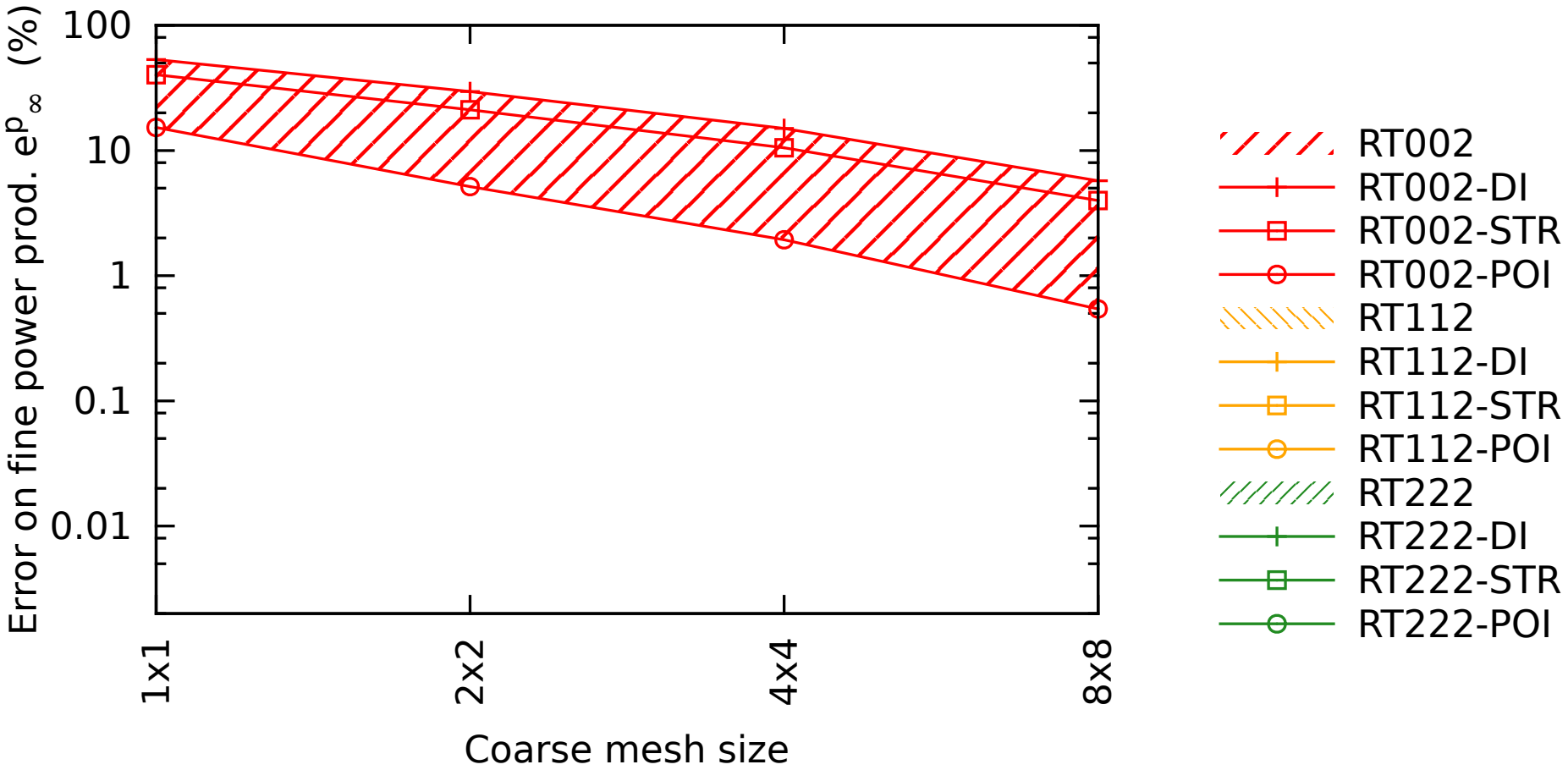
Real-World results – PWR with UOX loading pattern

Accuracy vs. discretization



Real-World results – PWR with UOX loading pattern

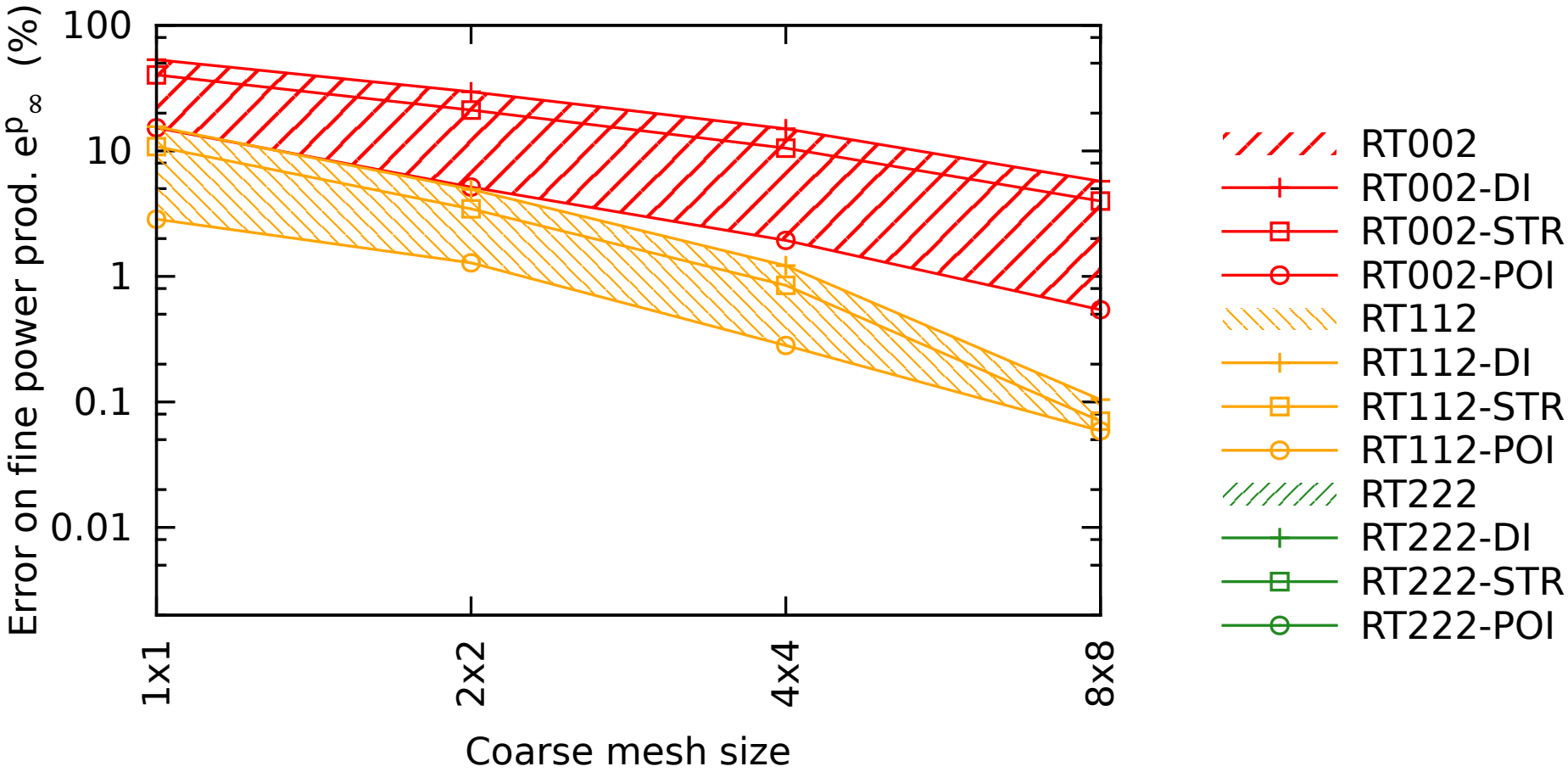
Accuracy vs. discretization



Poisson > StrawHat > Direct Integration

Real-World results – PWR with UOX loading pattern

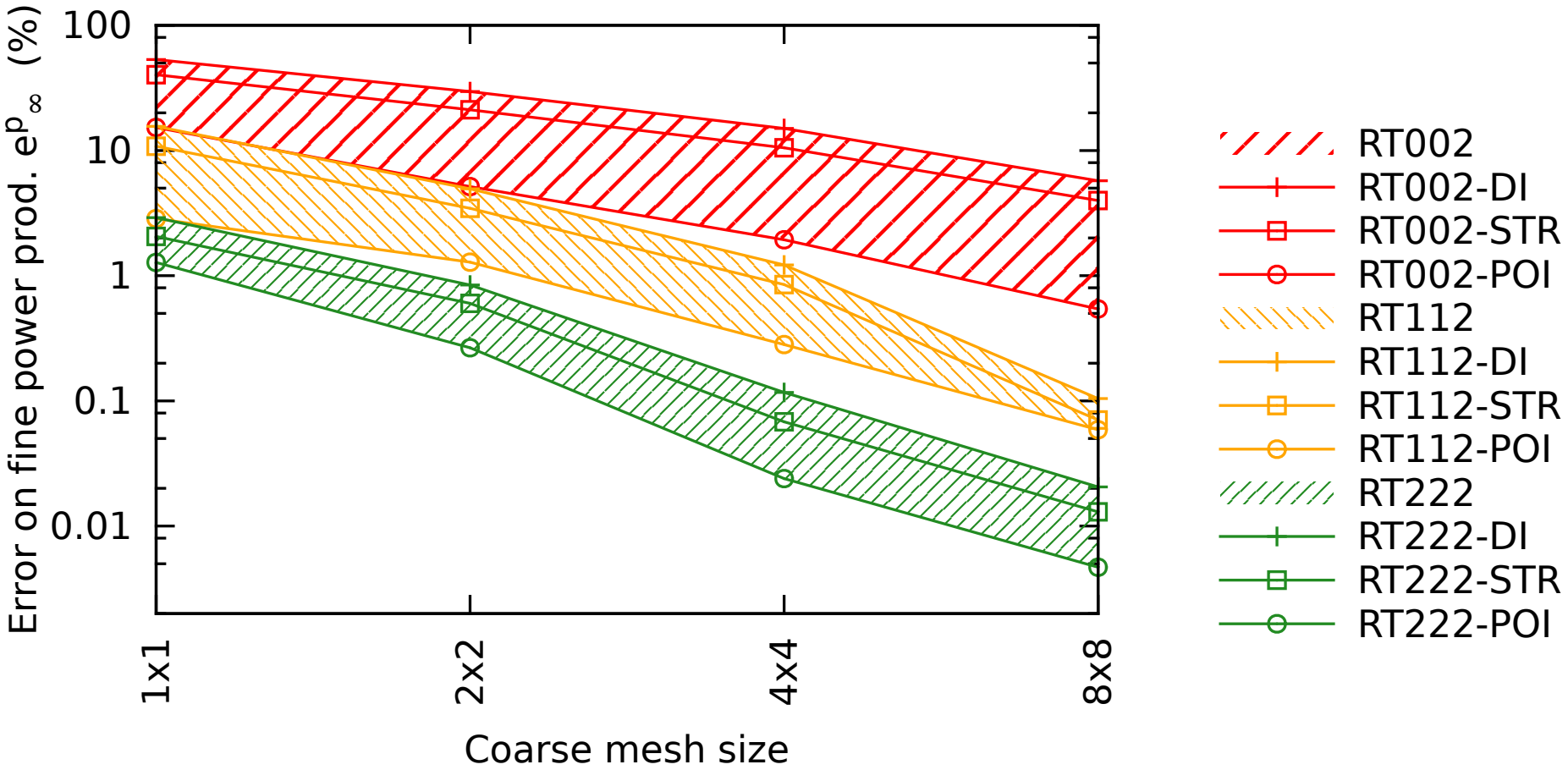
Accuracy vs. discretization



Poisson > StrawHat > Direct Integration

Real-World results – PWR with UOX loading pattern

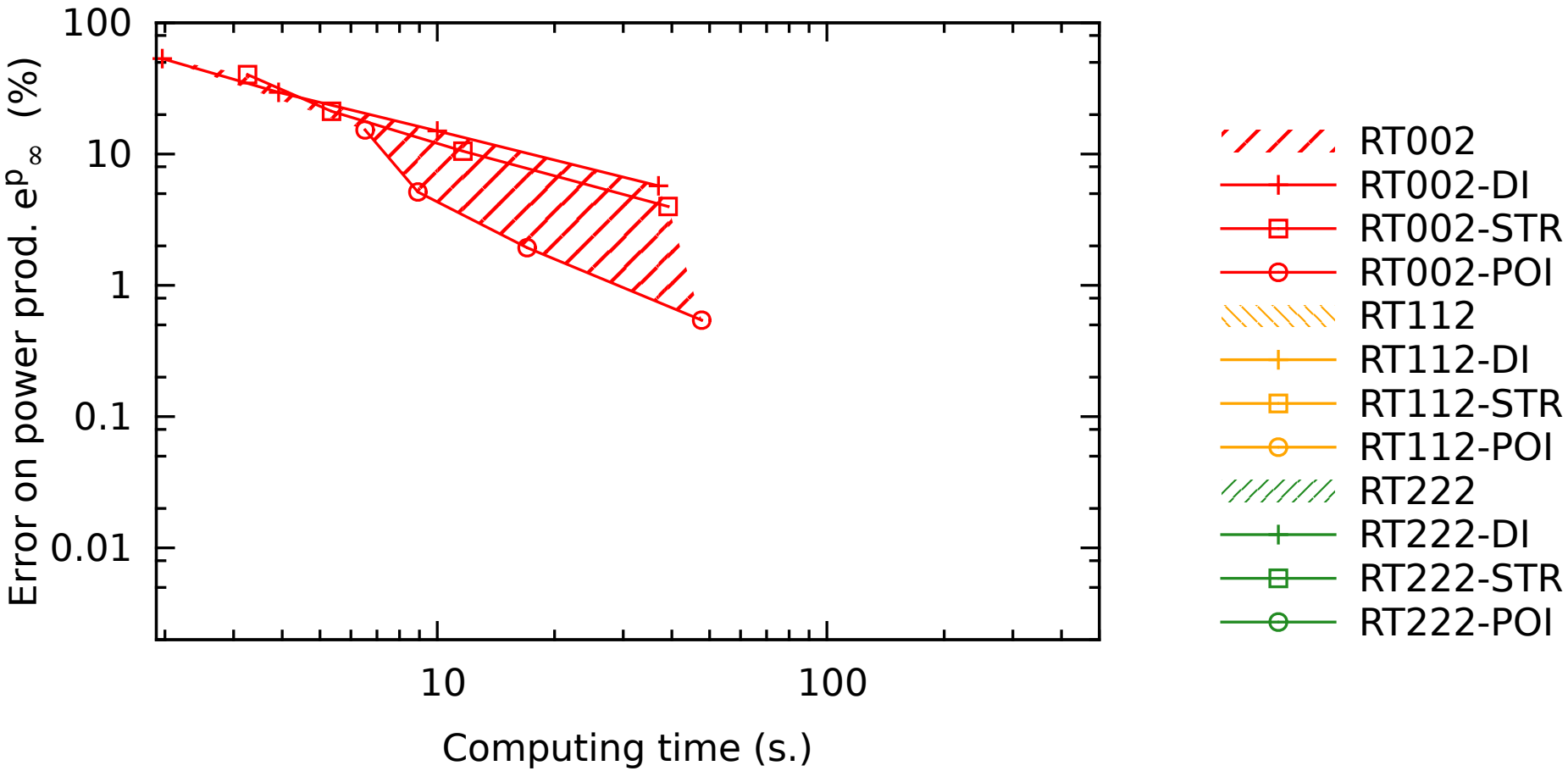
Accuracy vs. discretization



$$RT_2 > RT_1 > RT_0$$

Real-World results – PWR with UOX loading pattern

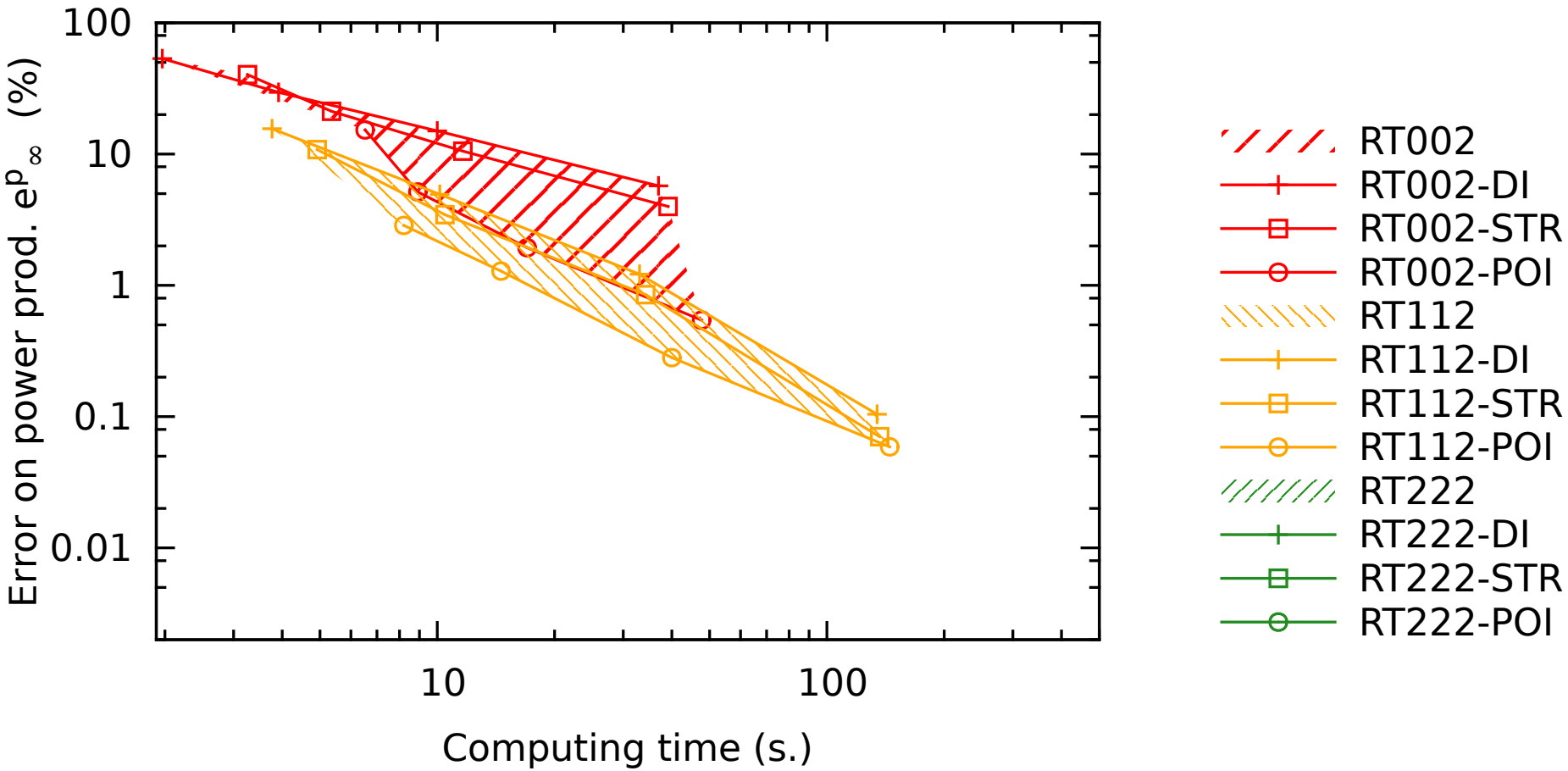
Accuracy vs. computing time



Poisson > StrawHat > Direct Integration

Real-World results – PWR with UOX loading pattern

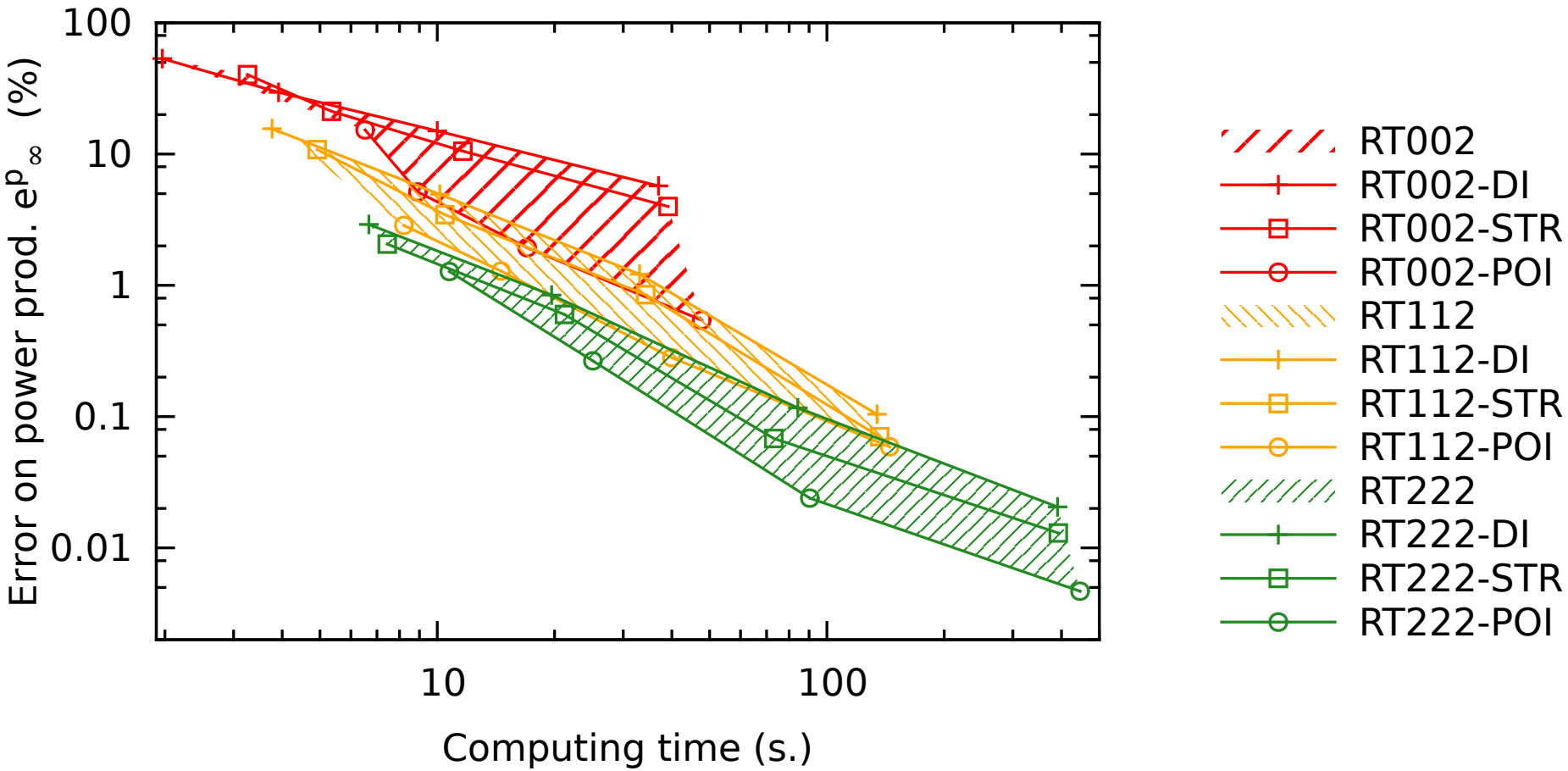
Accuracy vs. computing time



Poisson > StrawHat > Direct Integration

Real-World results – PWR with UOX loading pattern

Accuracy vs. computing time

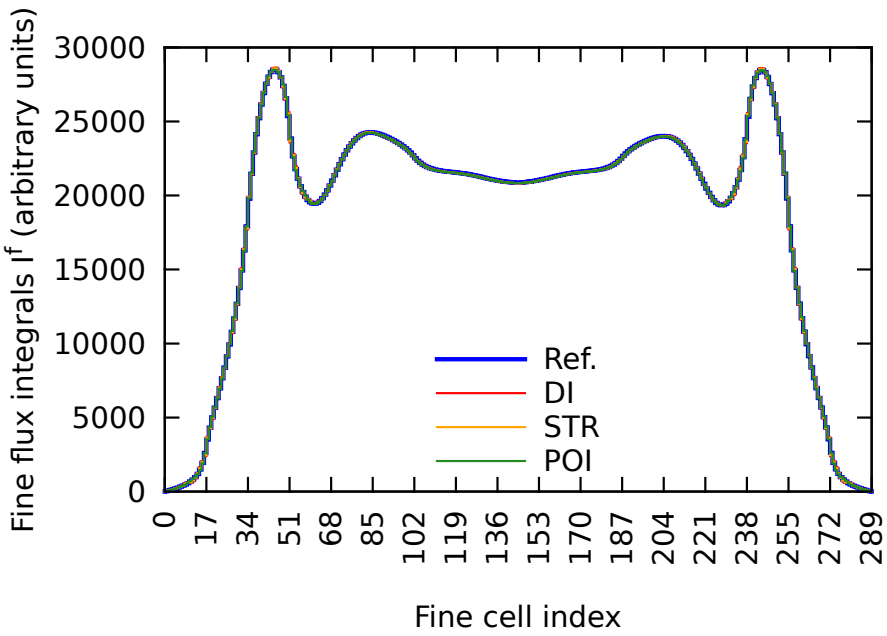


$$RT_2 > RT_1 > RT_0$$

Real-World results

1D flux cross-sections

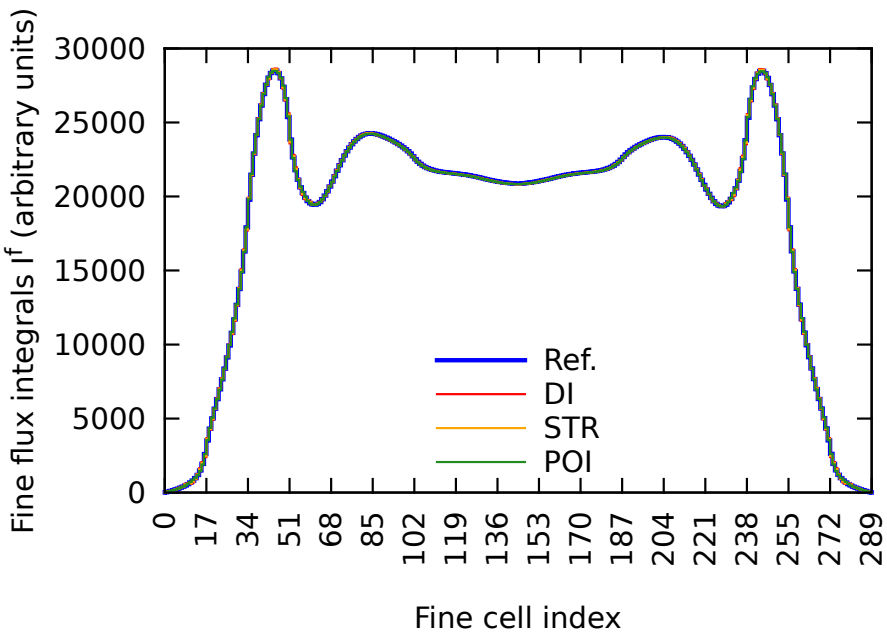
- UOX loading pattern



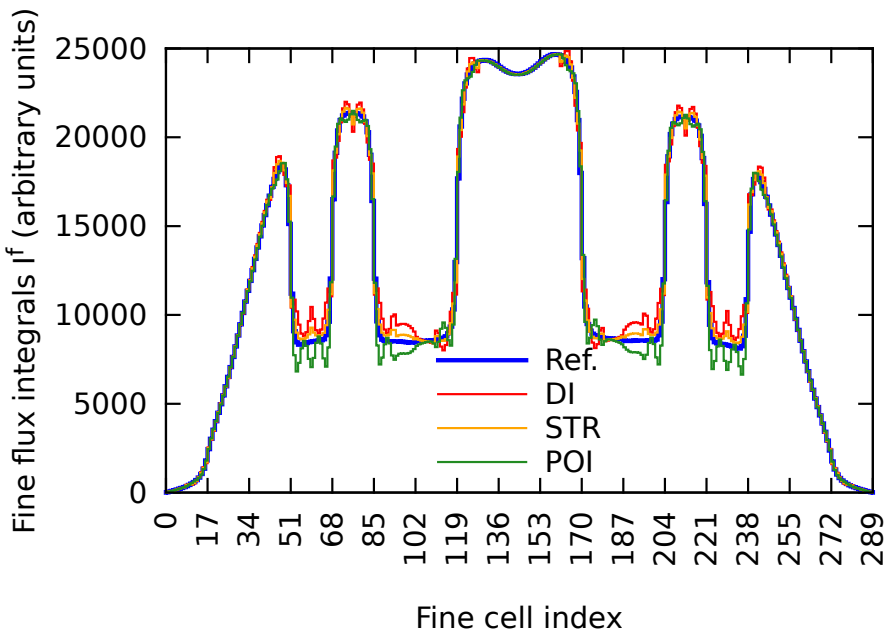
Real-World results

1D flux cross-sections

UOX loading pattern

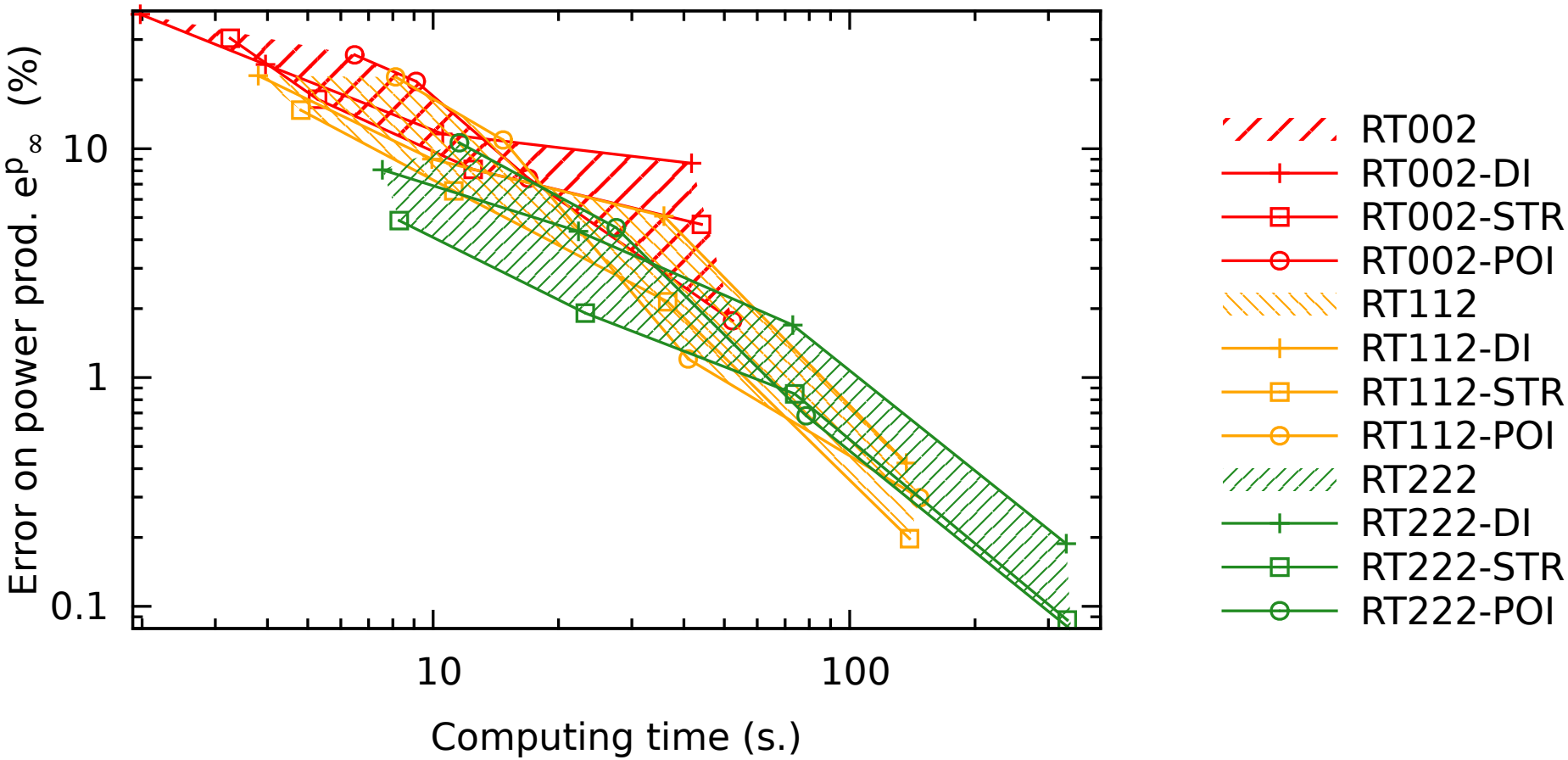


UOX-MOX loading pattern



Real-World results – PWR with UOX-MOX interfaces

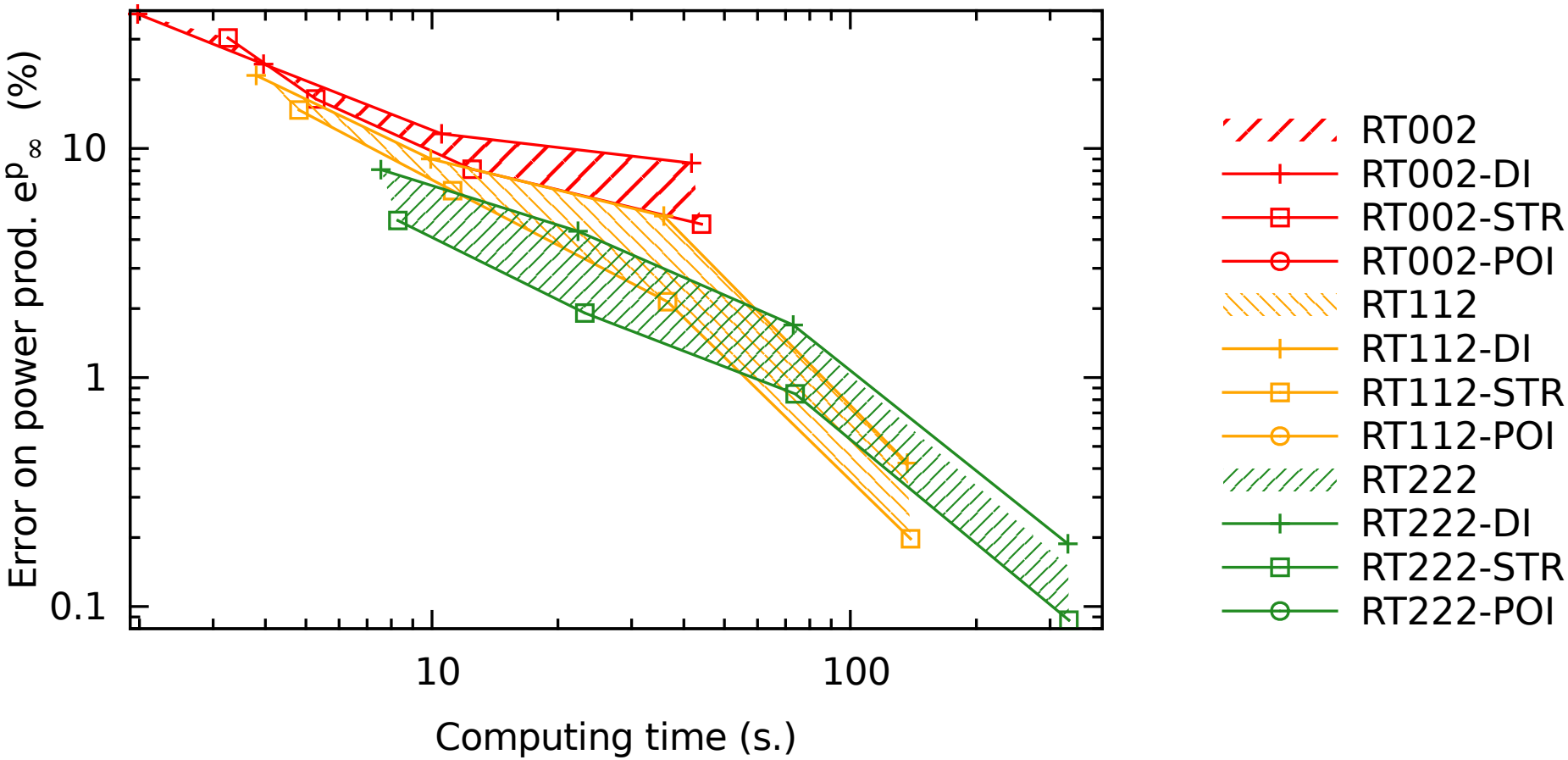
Accuracy vs. computing time



Poisson ?!? **StrawHat > Direct Integration – Not very clear...**

Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. computing time



Poisson out of the game:

StrawHat $>$ **Direct Integration** and $\mathbf{RT_2} > \mathbf{RT_1} > \mathbf{RT_0}$ (in our area of interest)

Conclusions – Perspectives

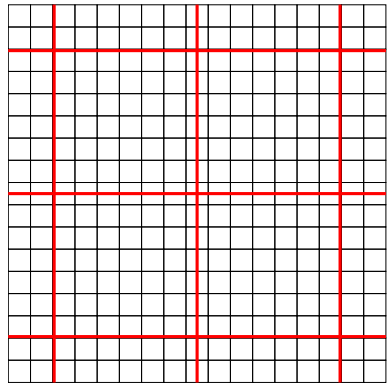
Conclusions

- 2 new methods for Fine Flux Integration in COCAGNE
  - based on fine current interpolation

- Assessment on 3-D PWR core calculations, from the fine flux viewpoint:
  - **higher RT_k orders are always better** (accuracy vs. time)
  - for smooth enough flux distributions:
    Poisson $>$ StrawHat $>$ Direct Integration
  - for discontinuous flux distributions (e.g. UOX/MOX interfaces):
    Poisson not so good; **StrawHat looks like a good candidate**.

Perspectives

- Study of **non-uniform computation meshes**:
  - assessment of the methods on the $^{\star}4 \times 4^{\star}$ mesh;
  - mesh refinement around UOX/MOX interfaces for the Poisson method.



- Mid-term:
  - Extension to other **boundary conditions** (symmetry, ...)
  - Extension to **SP3/SP5**.
- Science fiction:
  - **Automatic selection** between Poisson and StrawHat.
  - Use the new flux/current projections for **multilevel/multigrid methods**.

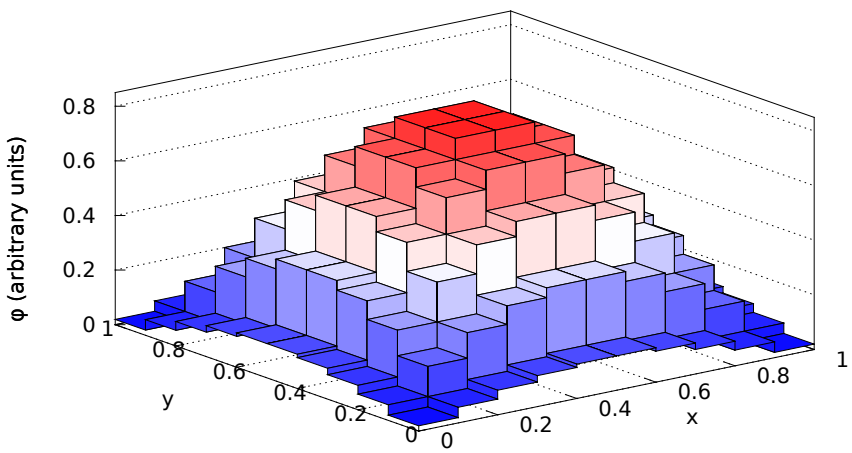
Thank You!

Questions?

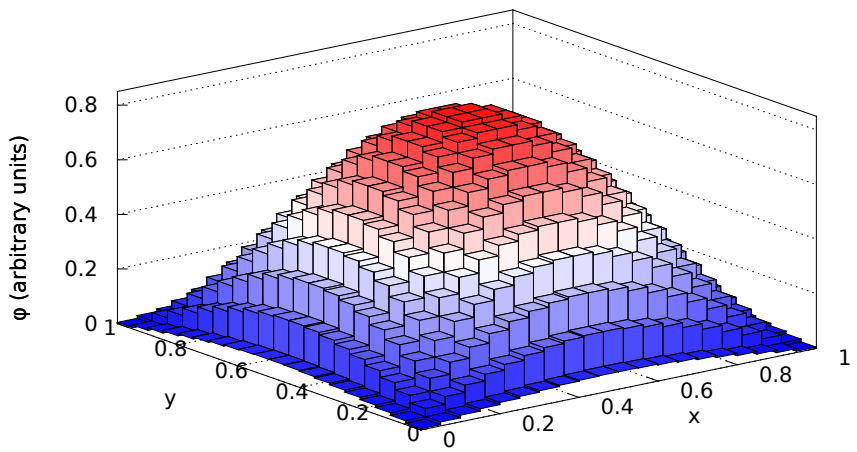
Numerical benchmark

- Running example for all methods in this talk:
  - Poisson's equation with sinusoidal source term
  - RT_0 discretization
  - $\mathcal{M}^c$: 10×10 mesh cells
  - $\mathcal{M}^f$: 20×20 mesh cells

- Coarse calculation results:

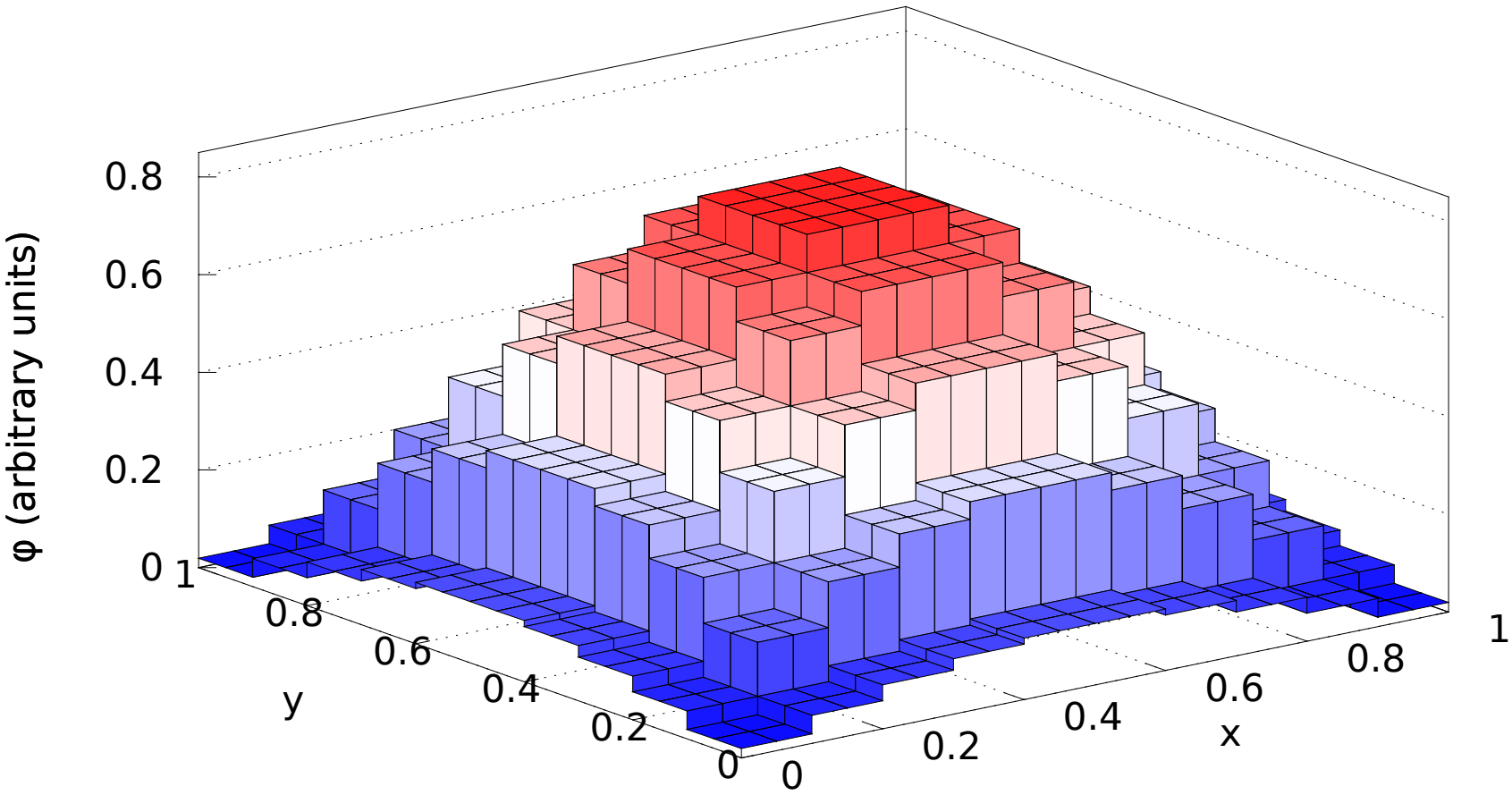


- Fine calculation results (reference):



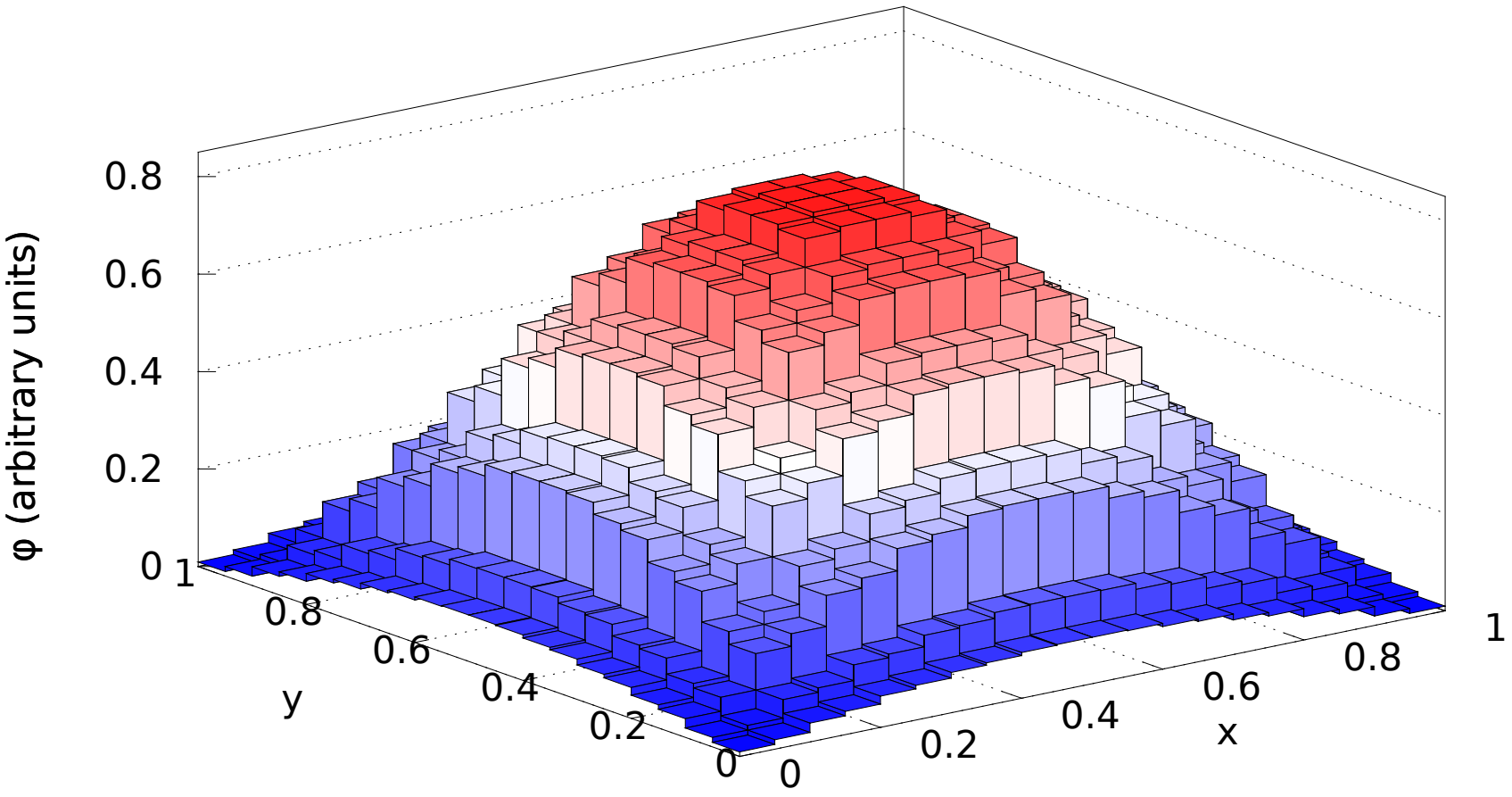
Numerical Benchmark

Direct Integration



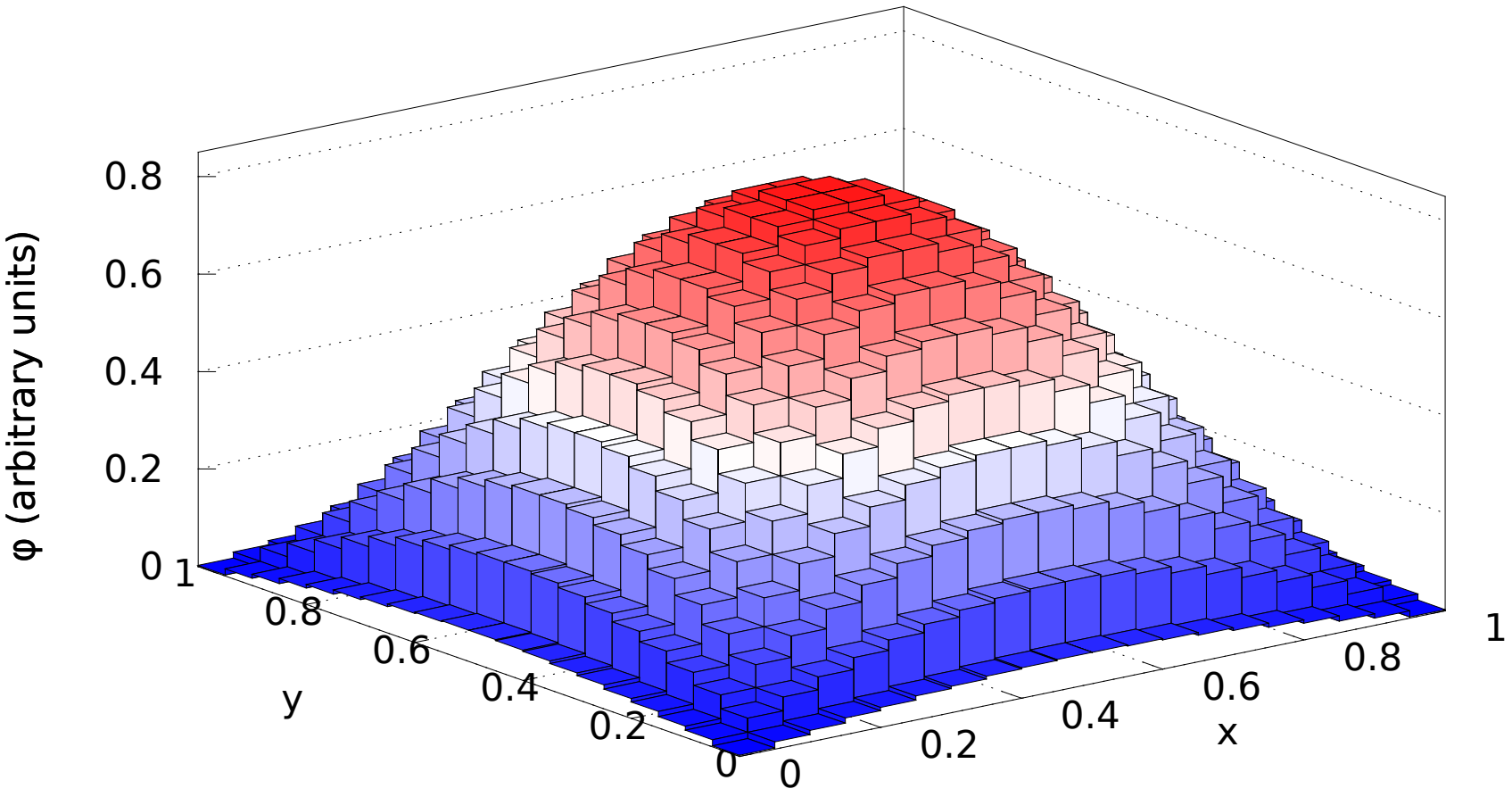
Numerical Benchmark

StrawHat Method



Numerical Benchmark

Poisson Method

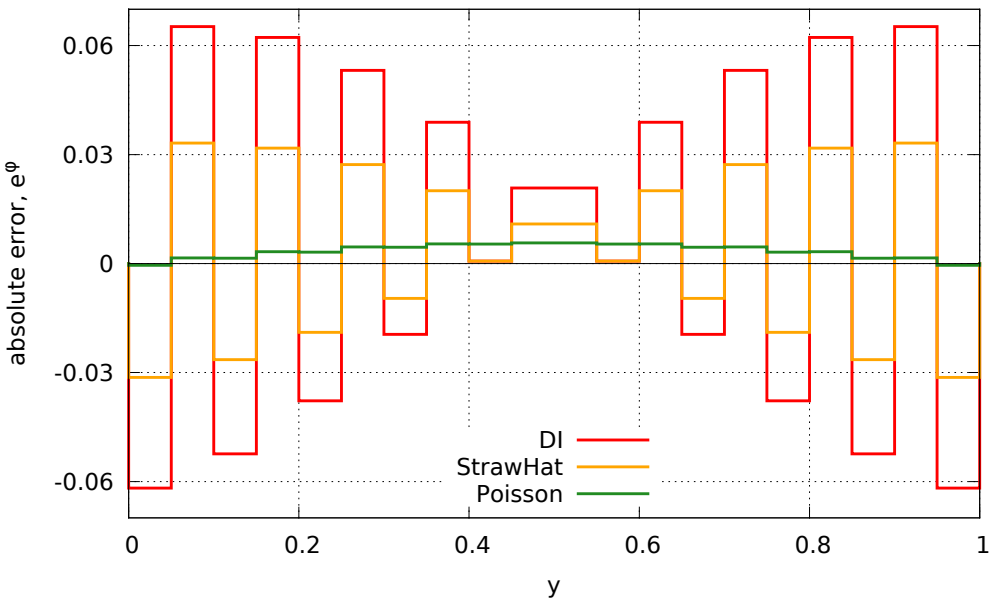


Methods comparison

Analytical benchmark

- Errors w.r.t. the reference method (fine problem resolution):

	DI	*StrawHat*	*Poisson*
$\|\|e^{\varphi}\|\|_{rel,\infty}$	8.2%	4.3%	0.69%
$\|\|e^{\varphi}\|\|_{rel,2}$	11%	5.6%	0.68%

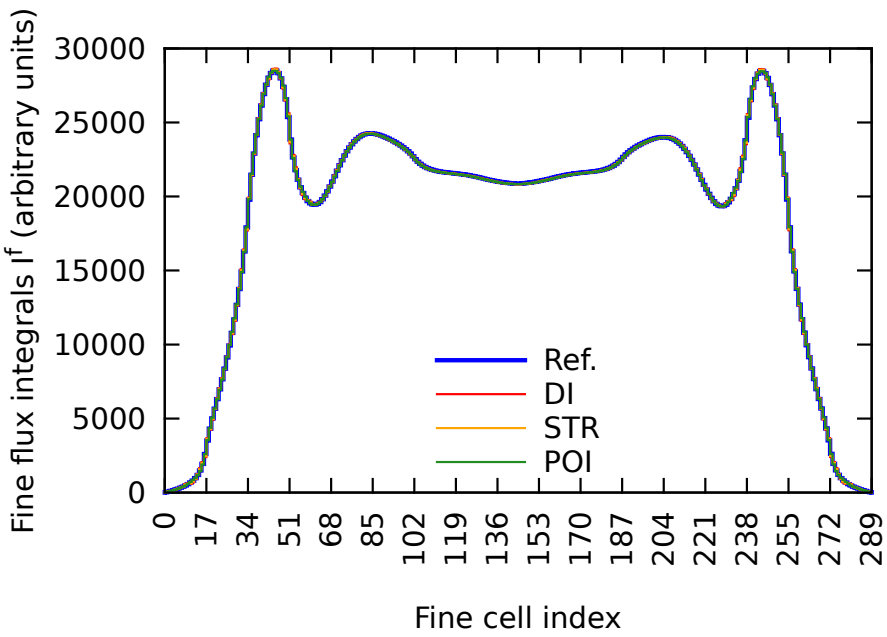


- Results:
  - **Poisson** > **StrawHat** > **DI**
  - **DI and StrawHat** produce oscillatory errors
    → **interpolation problems**
  - **Poisson** produces a smooth error
    → **balance error**.

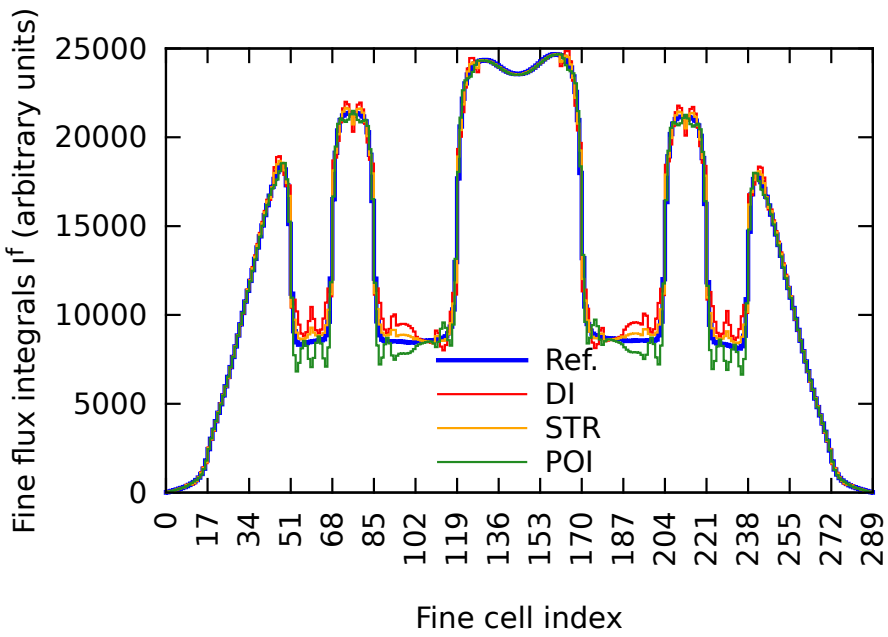
Real-World results

1D flux cross-sections

UOX loading pattern



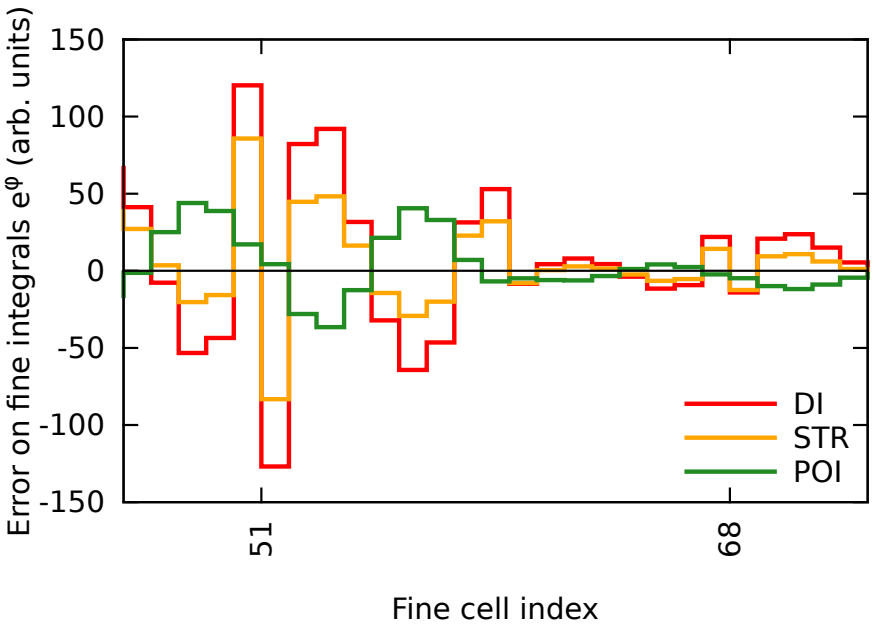
UOX-MOX loading pattern



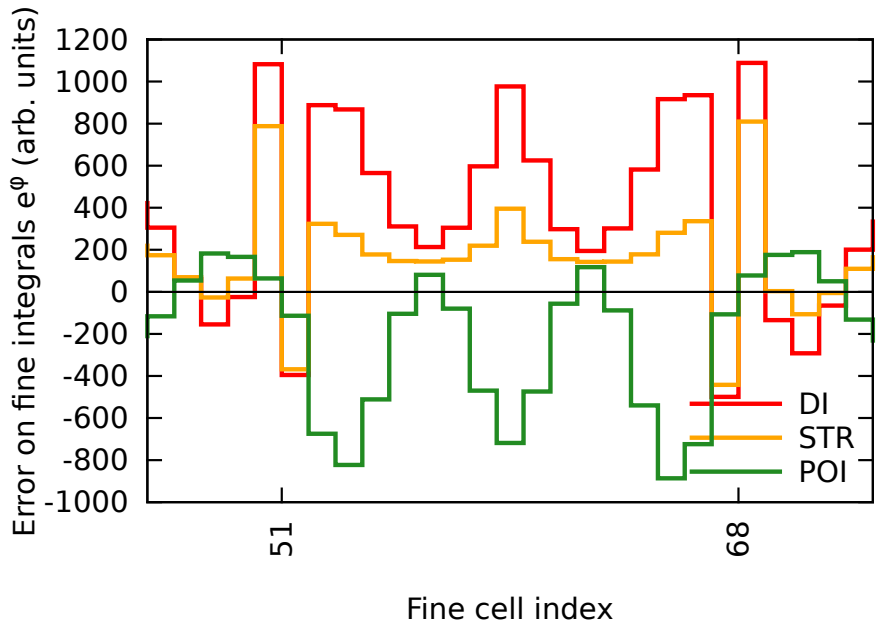
Real-World results

1D flux cross-sections

UOX loading pattern

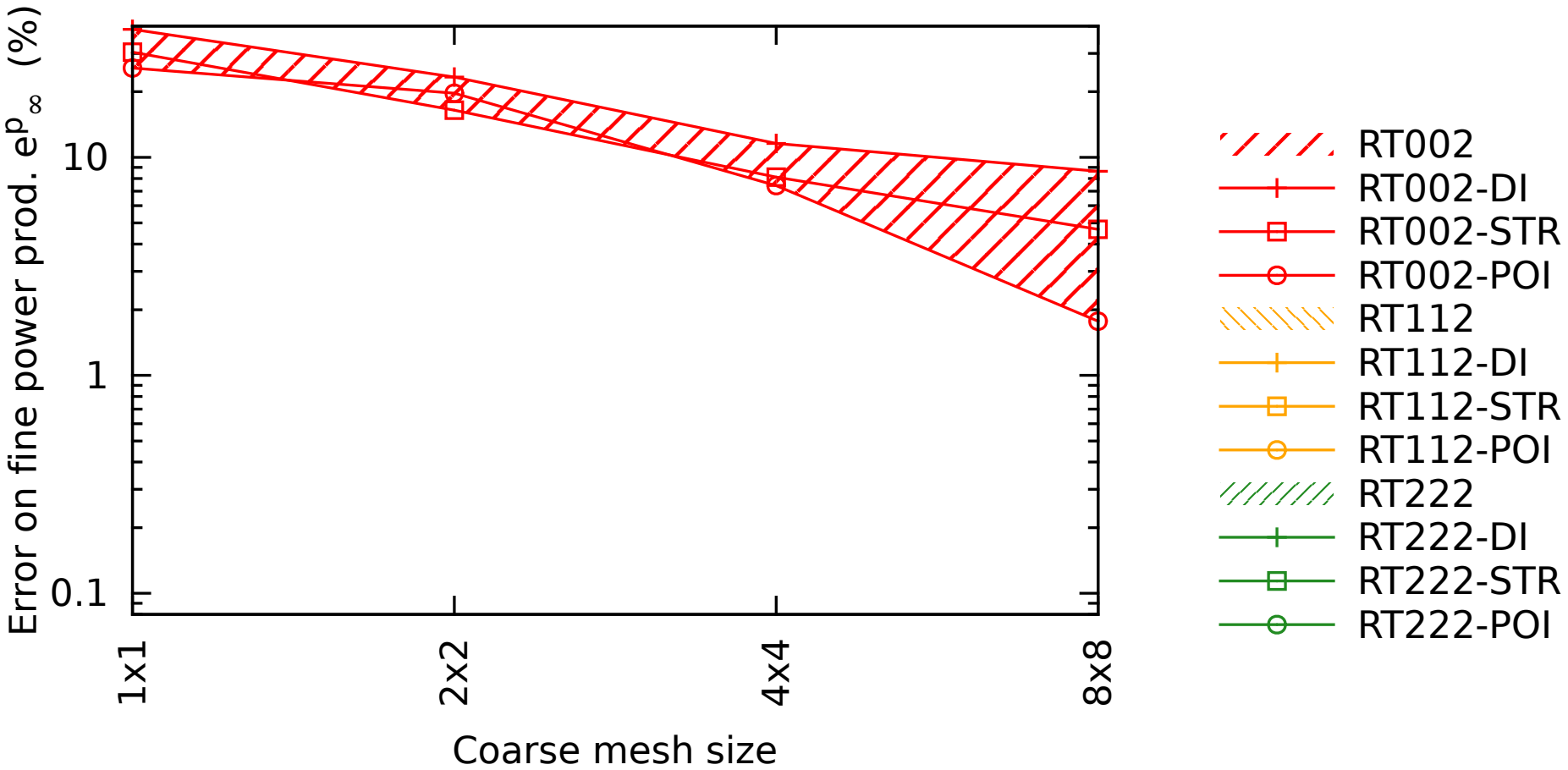


UOX-MOX loading pattern



Real-World results – PWR with UOX-MOX interfaces

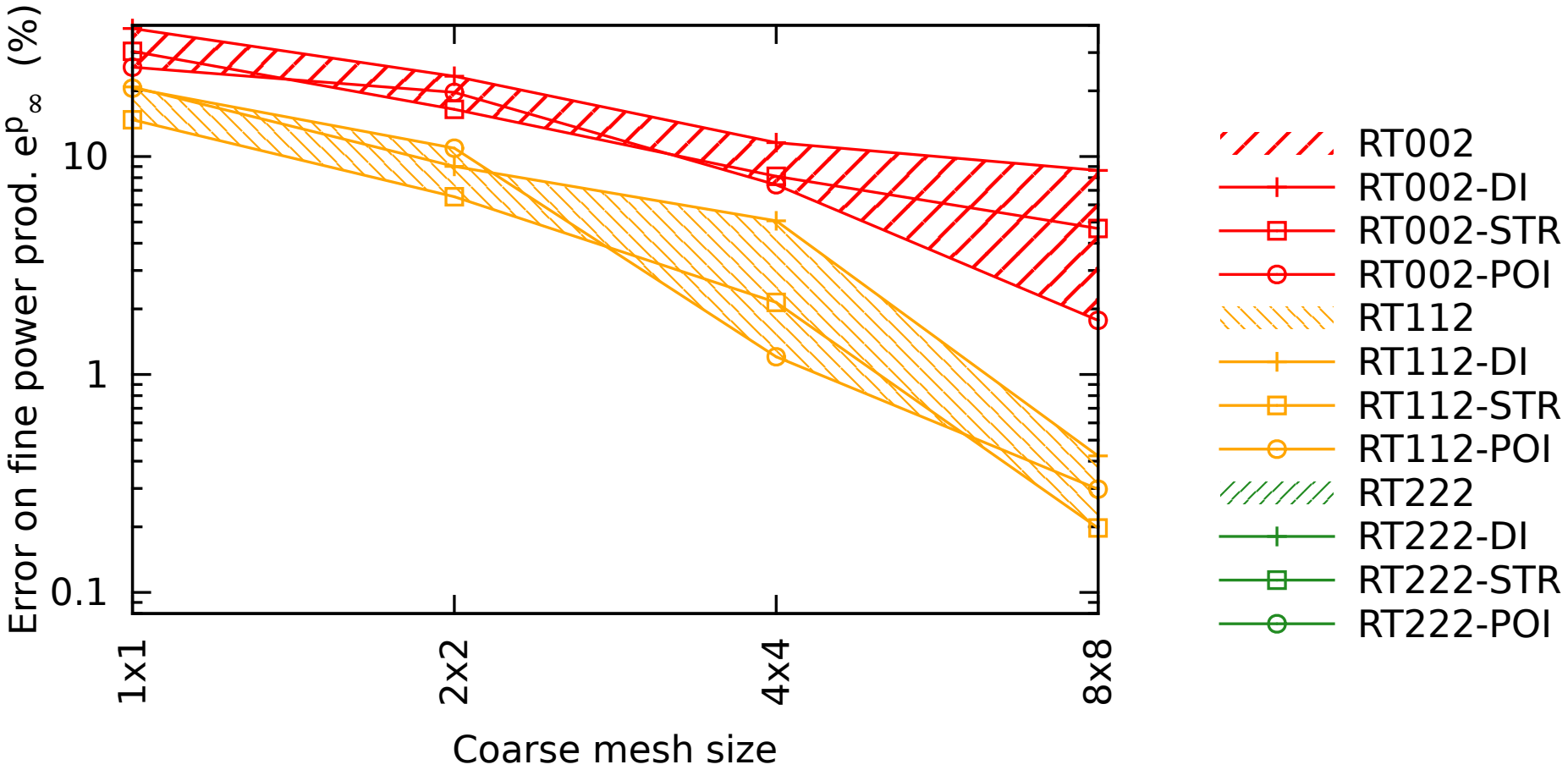
Accuracy vs. discretization



Poisson: not robust...

Real-World results – PWR with UOX-MOX interfaces

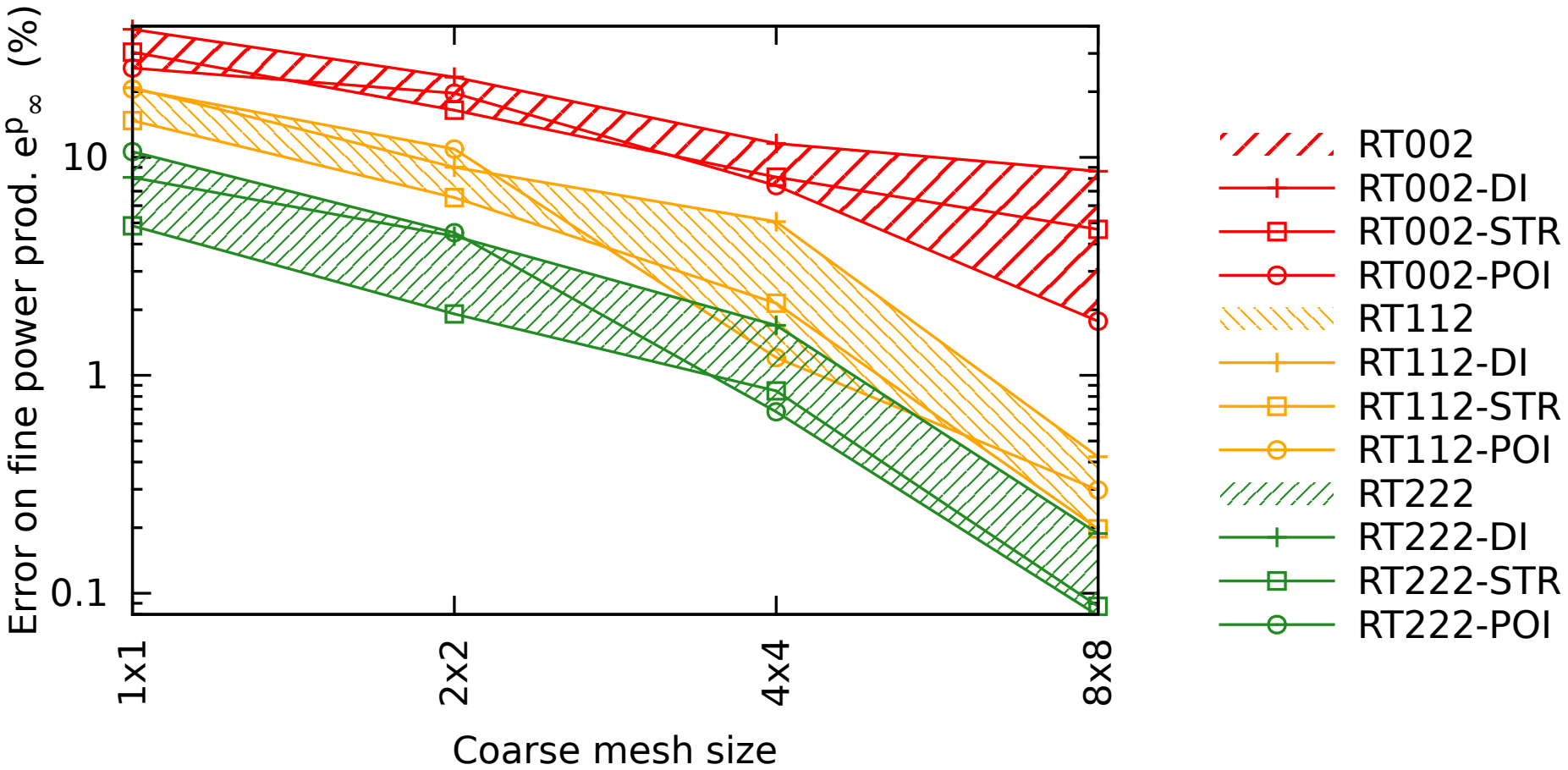
Accuracy vs. discretization



Poisson: not robust...

Real-World results – PWR with UOX-MOX interfaces

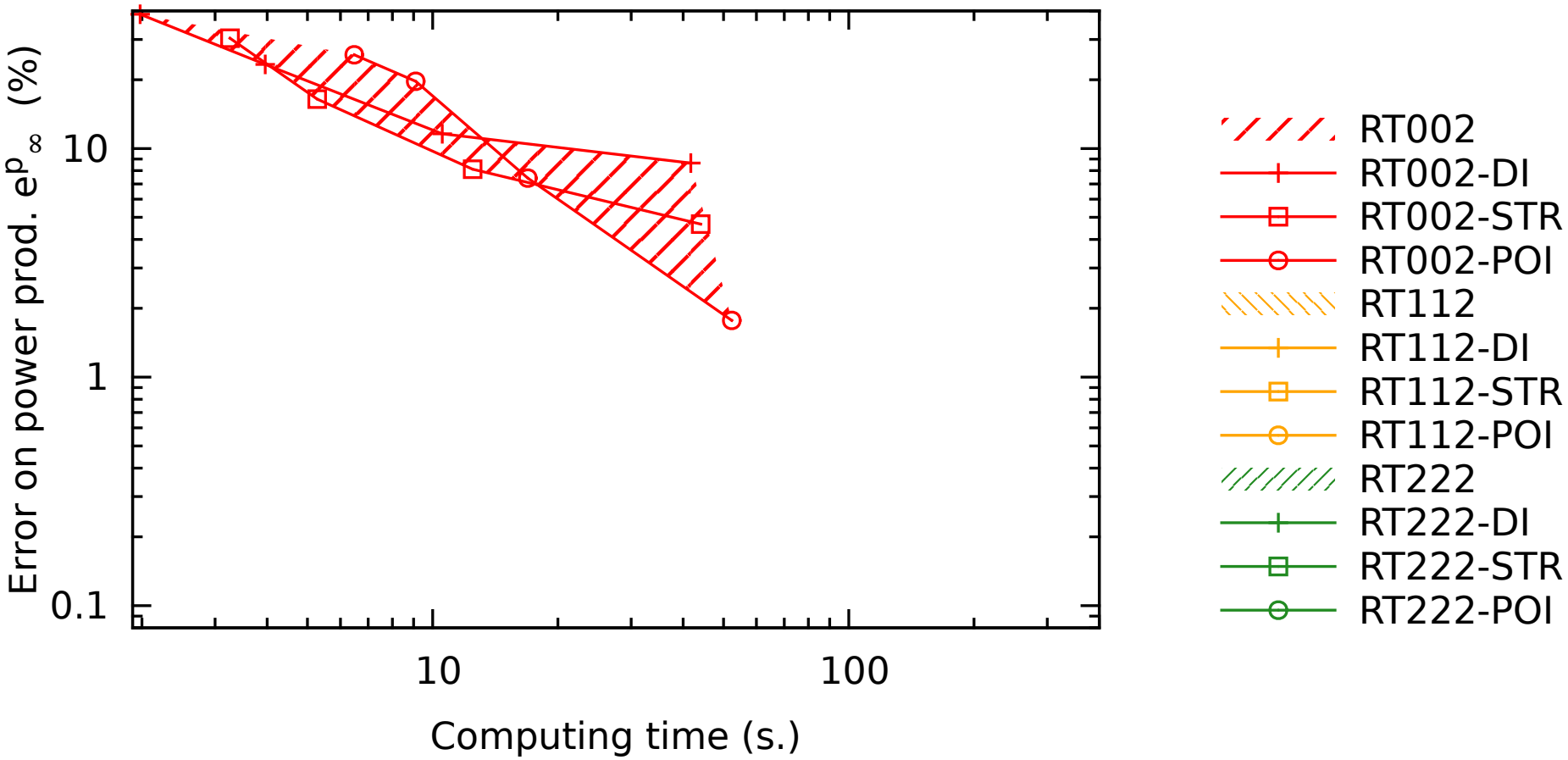
Accuracy vs. discretization



$$RT_2 > RT_1 > RT_0$$

Real-World results – PWR with UOX-MOX interfaces

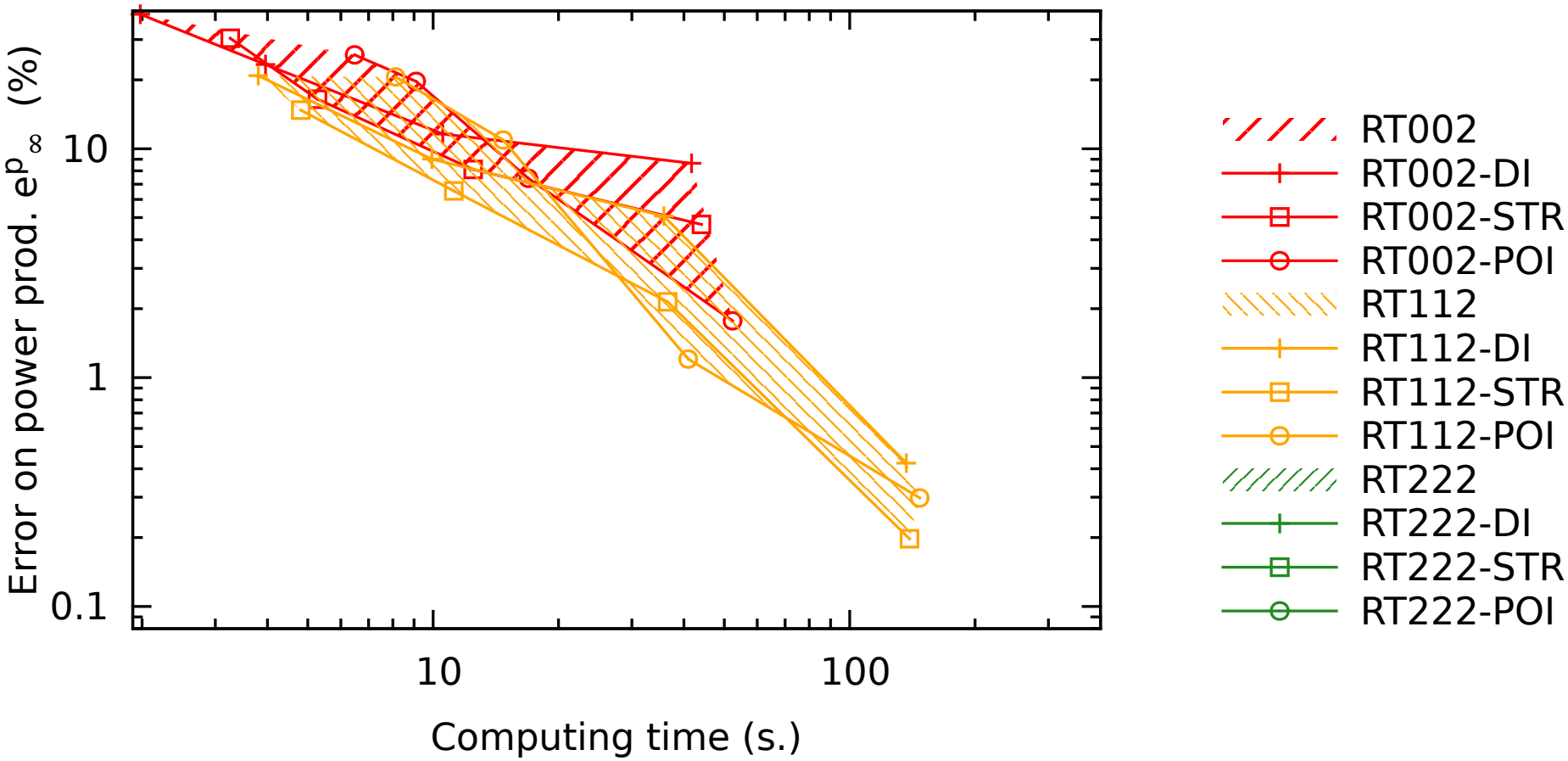
Accuracy vs. computing time



Poisson $\simeq$ StrawHat $>$ Direct Integration – Even less clear...

Real-World results – PWR with UOX-MOX interfaces

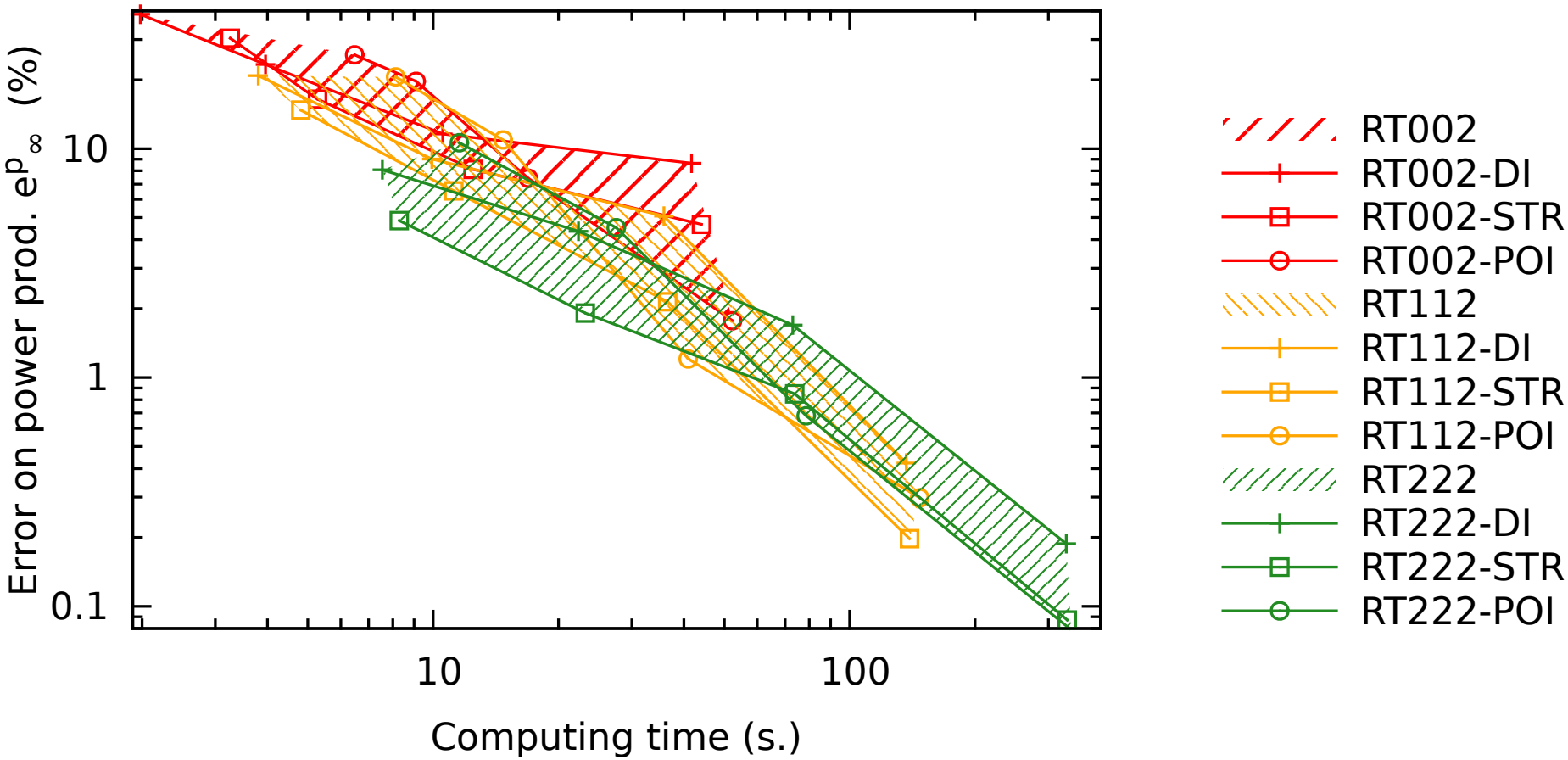
Accuracy vs. computing time



Poisson $\simeq$ StrawHat $>$ Direct Integration – Even less clear...

Real-World results – PWR with UOX-MOX interfaces

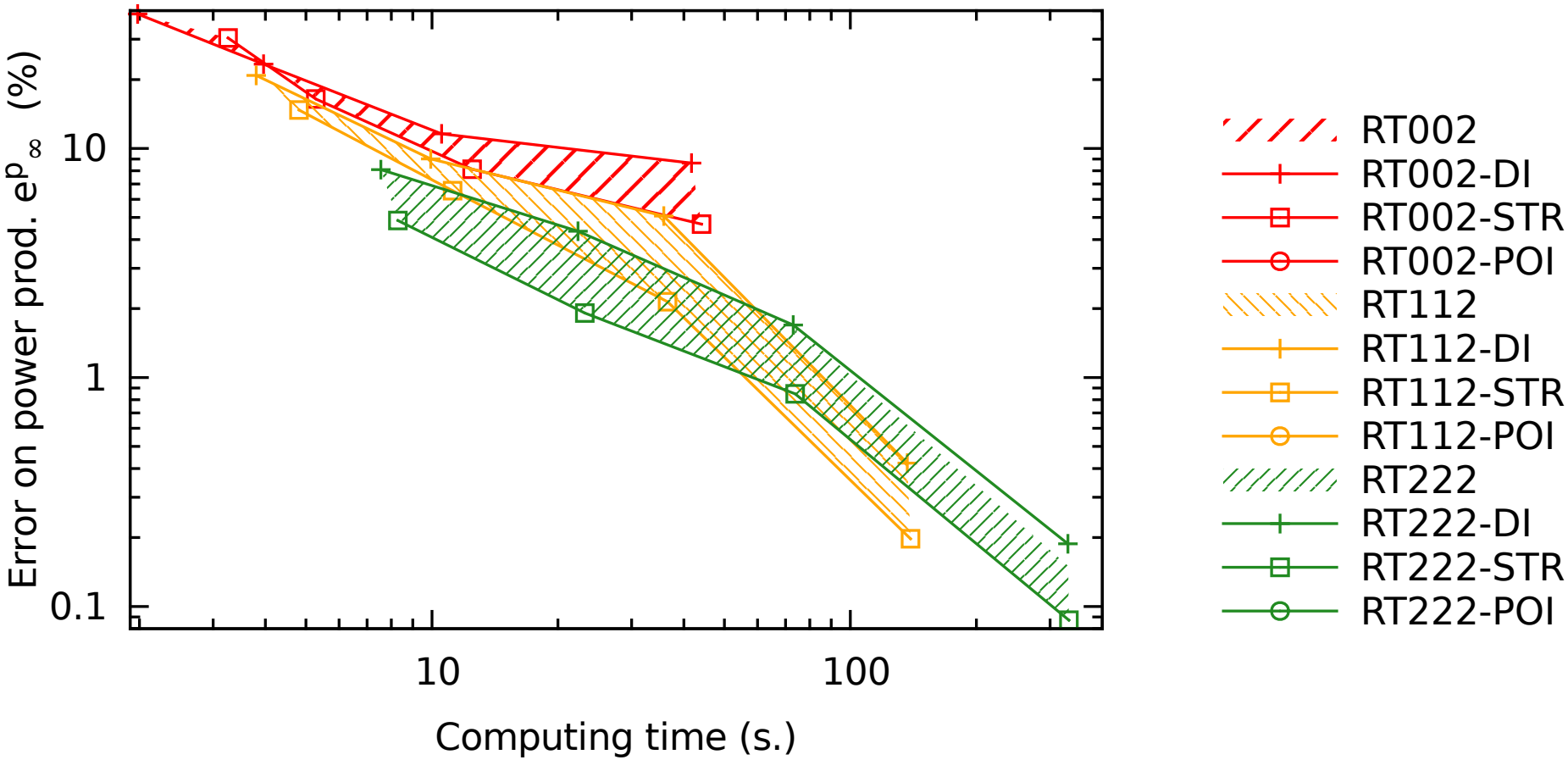
Accuracy vs. computing time



$$RT_2 > RT_1 > RT_0$$

Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. computing time

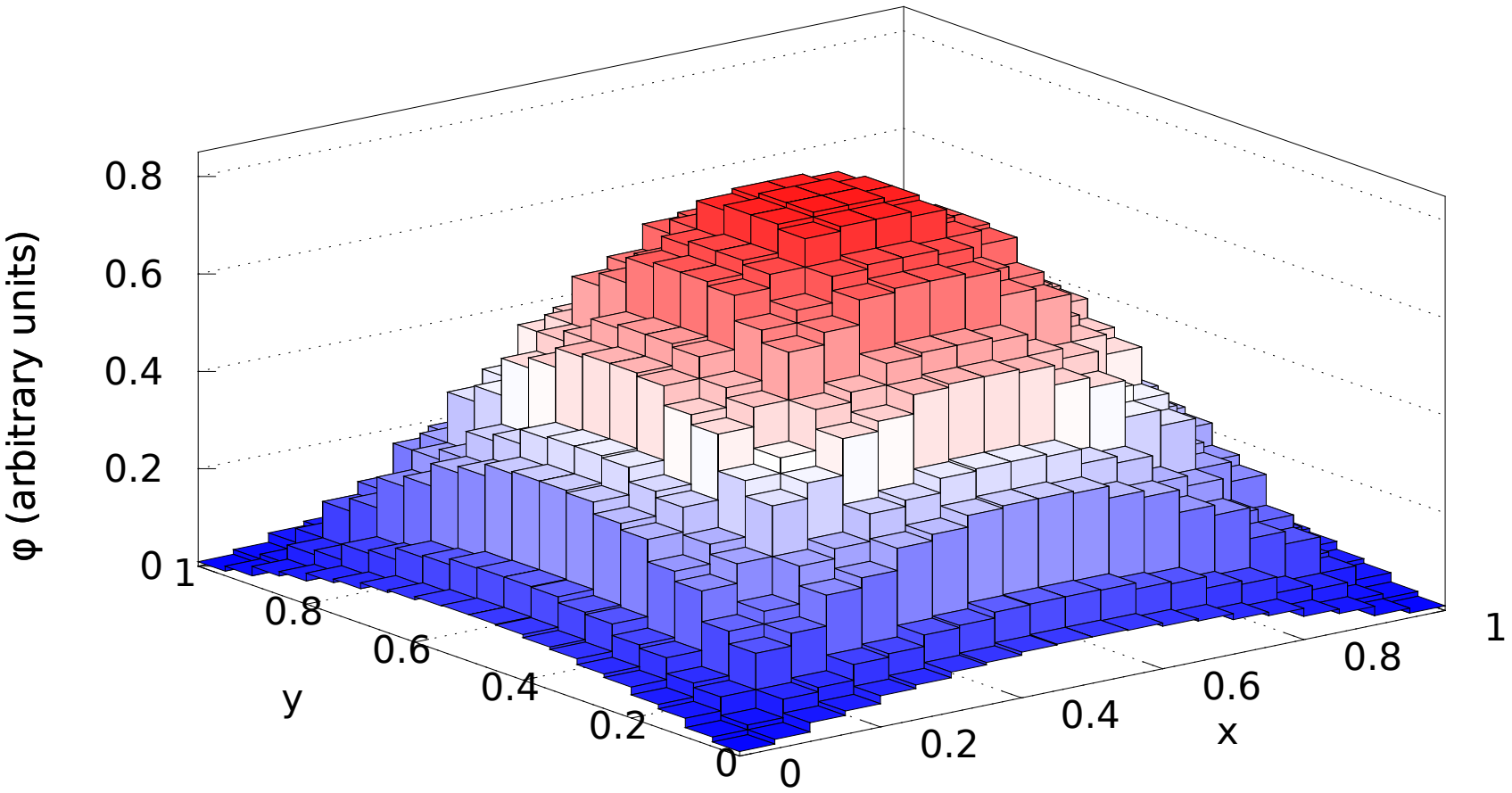


Poisson out of the game:

StrawHat $>$ **Direct Integration** and $\mathbf{RT_2} > \mathbf{RT_1} > \mathbf{RT_0}$ (almost always)

Numerical Benchmark

StrawHat Method



Numerical Benchmark

StrawHat Method