Fine Flux Integration Methods for the SPN Solver in Cocagne

D. Couyras, F. Févotte, L. Plagne

M&C 2013, Sun Valley, ID, USA
2-step calculation scheme

assembly calculation

- 2D Assembly calculation
  - APOLLO2 - REL2005
  - 281 energy groups
  - very fine spatial discretization
  - $\Sigma$: 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 $M^c$
    - (1 × 1 to 4 × 4 meshes/ass.)
  - $k_{eff}$: multiplication factor
  - $\varphi^c$, $j^c$: coarse flux, current
2-step calculation scheme

assembly calculation → fine (pin by pin) reaction rates:
\[ \tau_i^f = \sum_i I_i^f A_i^f \]

2D Assembly calculation
- very fine spatial discretization

3D Core calculation
- coarse spatial mesh \( \mathcal{M}^c \)
  - \( 1 \times 1 \) to \( 4 \times 4 \) meshes/ass.
- \( \varphi^c, j^c \): coarse flux, current

post-treatment
- fine spatial mesh \( \mathcal{M}^f \)
  - \( I^f \): pin-by-pin-averaged flux
  - \( A^f \): fine shape function
  - \( \tau^f \): 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:**

\[
\begin{align*}
\frac{1}{D} j + \nabla \varphi &= 0, \\
\text{div}(j) + \Sigma \varphi &= s.
\end{align*}
\]

**Discretized system with RT\(_k\) FEM:**

\[
\begin{align*}
A_k J - B_k \Phi &= 0, \\
tB_k J + T_k \Phi &= S_k.
\end{align*}
\]

**RT\(_1\) DoFs in a mesh cell:**

- \(j\) DoF
- \(\varphi\) DoF

**RT\(_1\) flux:**

\(\varphi(x)\)

**RT\(_1\) current:**

\(j(x)\)
Direct flux integration method

Principle

- Direct integration of the coarse flux
  - Example of a 1D RT$_1$ coarse flux:

\[ \varphi^c(x) \]

\[ \int M f_i \varphi_c(x) \, dx = |M f_i| \int \varphi_c(x) \, dx \]

\[ \Rightarrow I f_i \text{ can be identified to an RT}_0 \text{ flux discretization on} \]

\[ \{ A f_0 J - B f_0 \Phi = 0, \]
\[ B f_0 J + T f_0 \Phi = S f_0 \}. \]
**Direct flux integration method**

**Principle**

- Direct integration of the coarse flux
  - Example of a 1D RT$_1$ coarse flux:
    \[
    I^f_i = \frac{1}{|M^f_i|} \int_{M^f_i} \varphi^c(x) \, dx
    \]

![Graph showing the relationship between \(\varphi^c(x)\) and \(I^f_i\)](image_url)
Direct flux integration method

Principle

- Direct integration of the coarse flux
  > Example of a 1D RT₁ coarse flux:

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

\[
= \frac{1}{|\mathcal{M}_i^f|} \int \phi^c(x) \mathbf{1}_{\mathcal{M}_i^f}(x) \, dx
\]

⇒ \(I_i^f\) can be identified to an RT₀ flux discretization on \(\mathcal{M}_i^f\)

Let us try and solve the following fine RT₀ SP₁ system:

\[
\begin{align*}
A_0^f J - B_0^f \Phi &= 0, \\
^t B_0^f J + T_0^f \Phi &= S_0^f.
\end{align*}
\]
Fine Flux Integration

1. Introduction

2. Fine Flux Integration
   Poisson
   StrawHat

3. Real-World Results

4. Conclusions – Perspectives
Fine system resolution

- Solve the fine $SP_1$ $RT_0$ system:

$$\begin{align*}
\mathbf{A}^f_0 \mathbf{J} - \mathbf{B}^f_0 \Phi &= 0, \\
t\mathbf{B}^f_0 \mathbf{J} + t\mathbf{T}^f_0 \Phi &= S^f_0.
\end{align*}$$

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

- project/interpolate $(\Phi^c, J^c)$ on $\mathcal{M}^f$ to initialize the fine solver
  → open the way to multi-level/multigrid methods; see the perspectives.

- solve only subset of the problem at the fine level
  → solutions explored in the following.
Current-based integration methods

Principle

\[
\begin{align*}
A_0^f J^f - B_0^f \Phi^f &= 0 \\
T_0^f \Phi^f + S_0^f &= 0
\end{align*}
\]

◮ Idea:
  ◯ project the current only, and treat it as a known source term;
  ◯ ignore the last set of equations:
    \[ B_0^f \Phi^f = A_0^f J^f_{proj} \]

◮ Advantage:
  ◯ \( B \) does not contain any physical data.

◮ Problem:
  ◯ this system is not square!
Current-based integration methods

Principle

**Idea:**
- project the current only, and treat it as a known source term;
- ignore the last set of equations:
  \[ B_0^f \Phi^f = A_0^f J_{proj}^f \]

**Advantage:**
- \( B \) does not contain any physical data.

**Problem:**
- this system is not square!
Current-based integration methods

Poisson method

**Idea:** left-multiply the equation by \( t^\top B \):

\[
B \Phi = A J_{proj}
\]
Current-based integration methods

Poisson method

- **Idea:** left-multiply the equation by $t B$:

$$tB B \Phi_{poisson} = tB A J_{proj}$$

- **Advantages:**
  - $tB B$ is the classical finite-differences discretization of the Laplace operator;
  - there exist very efficient methods to solve it.

For example, in 1D:

$$B = \begin{pmatrix} 1 & & & \\ -1 & \ddots & & \\ & \ddots & 1 & \\ & & -1 & \end{pmatrix} \quad tB B = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$
Current-based integration methods
“StrawHat” method

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

\[
\mathbf{B} \, \Phi = \mathbf{A} \, J_{proj}
\]

\[
\mathbf{B} = \begin{pmatrix}
1 \\
-1 \\
\vdots \\
\vdots \\
\vdots \\
1 \\
-1
\end{pmatrix}
\]
Current-based integration methods
“StrawHat” method

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

\[ \tilde{\mathbf{B}} \Phi_{\text{strawhat}} = \tilde{\mathbf{A}} \mathbf{J}_{\text{proj}} \]

- **Advantage:**
  - \( \tilde{\mathbf{B}} \) is lower triangular.

\[
\mathbf{B} = \begin{pmatrix}
1 \\
-1 \\
\vdots \\
\vdots \\
\vdots \\
-1
\end{pmatrix}
\quad
\tilde{\mathbf{B}} = \begin{pmatrix}
1 \\
-1 \\
\vdots \\
\vdots \\
\vdots \\
-1 & 1
\end{pmatrix}
\]
Real-World Results

1. Introduction
2. Fine Flux Integration
3. Real-World Results
   UOX cases
   UOX/MOX interfaces
4. Conclusions – Perspectives
Real-World results: 3D PWR core computation

Experimental setup

- **Spatial discretization:**

<table>
<thead>
<tr>
<th>Mesh Type</th>
<th>Spatial Discretization</th>
<th>Number of Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine mesh $\mathcal{M}^f$</td>
<td>$(x, y)$ $z$</td>
<td>$17 \times 17$ cells/ass. $40$ cells</td>
</tr>
<tr>
<td>Computation mesh $\mathcal{M}^c$</td>
<td>$(1 \times 1)$ $\rightarrow$ $(8 \times 8)$</td>
<td>$40$ cells</td>
</tr>
<tr>
<td>Discretization order</td>
<td>$RT_0 \rightarrow RT_2$</td>
<td>$RT_2$</td>
</tr>
</tbody>
</table>

- **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

\[
P^f = \sum_g \kappa \Sigma_f(g) I^f(g)
\]

\[
e^p_\infty = \frac{\|P^f - P^{f,\text{ref}}\|_\infty}{\|P^{\text{ref}}\|_\infty}
\]
Real-World results – PWR with UOX loading pattern

Accuracy vs. discretization

![Graph showing error on fine power production vs. coarse mesh size for different cases.]

- RT002
- RT002-DI
- RT002-STR
- RT002-POI
- RT112
- RT112-DI
- RT112-STR
- RT112-POI
- RT222
- RT222-DI
- RT222-STR
- RT222-POI
Real-World results – PWR with UOX loading pattern

Accuracy vs. discretization

Error on fine power prod. $e_p^\infty$ (%) vs. coarse mesh size

Poisson > StrawHat > Direct Integration
Real-World results – PWR with UOX loading pattern
Accuracy vs. discretization

Error on fine power prod. $e_{\infty}^p$ (%)

Coarse mesh size

RT002
RT002-DI
RT002-STR
RT002-POI
RT112
RT112-DI
RT112-STR
RT112-POI
RT222
RT222-DI
RT222-STR
RT222-POI

Poisson $>$ StrawHat $>$ Direct Integration
Real-World results – PWR with UOX loading pattern
Accuracy vs. discretization

Error on fine power prod. e_p^∞ (%)
Coarse mesh size

RT_2 > RT_1 > RT_0
Real-World results – PWR with UOX loading pattern

Accuracy vs. computing time

Error on power prod. $e_{p_{\infty}}$ (%) vs. Computing time (s.)

Poisson > StrawHat > Direct Integration
Real-World results – PWR with UOX loading pattern

Accuracy vs. computing time

Error on power prod. $e_{p\infty}^\rho$ (%)

Computing time (s.)

Poisson > StrawHat > Direct Integration
Real-World results – PWR with UOX loading pattern

Accuracy vs. computing time

\[ \text{Error on power prod. } e_p^\infty \ (\%) \]

\[ \text{Computing time (s.)} \]

\[ \begin{align*}
\text{RT002} & > \text{RT112} & > \text{RT222}
\end{align*} \]

Fine Flux Integration for EDF’s SPN Solver
Real-World results
1D flux cross-sections

UOX loading pattern

Fine flux integrals $f^I$ (arbitrary units)

Fine cell index

Ref.
DI
STR
POI

Fine Flux Integration for EDF’s SPN Solver
Real-World results
1D flux cross-sections

- UOX loading pattern
- UOX-MOX loading pattern

Fine Flux Integration for EDF’s SPN Solver
Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. computing time

Error on power prod. $e_{p,8}^\infty$ (%) vs. Computing time (s.)

- RT002
- RT002-DI
- RT002-STR
- RT002-POI
- RT112
- RT112-DI
- RT112-STR
- RT112-POI
- RT222
- RT222-DI
- RT222-STR
- RT222-POI

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 $RT_2 > RT_1 > RT_0$ (in our area of interest)
Conclusions – Perspectives

1. Introduction

2. Fine Flux Integration

3. Real-World Results

4. 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 \(4 \times 4\) 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
  - RT₀ discretization
  - \( M^c \): 10 × 10 mesh cells
  - \( M^f \): 20 × 20 mesh cells

- Coarse calculation results:
- Fine calculation results (reference):
Numerical Benchmark

Direct Integration
Numerical Benchmark

StrawHat Method

Fine Flux Integration for EDF’s SPN Solver
Numerical Benchmark
Poisson Method

Fine Flux Integration for EDF’s SPN Solver
Methods comparison
Analytical benchmark

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

<table>
<thead>
<tr>
<th></th>
<th>DI</th>
<th>StrawHat</th>
<th>Poisson</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|e^\varphi|_{rel,\infty}$</td>
<td>8.2%</td>
<td>4.3%</td>
<td>0.69%</td>
</tr>
<tr>
<td>$|e^\varphi|_{rel,2}$</td>
<td>11%</td>
<td>5.6%</td>
<td>0.68%</td>
</tr>
</tbody>
</table>

Results:
- Poisson $>$ StrawHat $>$ DI
- DI and StrawHat produce oscillatory errors $\rightarrow$ interpolation problems
- Poisson produces a smooth error $\rightarrow$ 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

![Graph of UOX loading pattern](image1)

![Graph of UOX-MOX loading pattern](image2)
Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. discretization

Coarse mesh size

Error on fine power prod. $e_p^8$ (%)

Poisson: not robust...
Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. discretization

Poisson: not robust...

Fine Flux Integration for EDF's SPN Solver
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 $\approx$ StrawHat $>\$ Direct Integration – Even less clear...
Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. computing time

Error on power prod. $e^{p,8}$ (%)

Computing time (s.)

Poisson $\approx$ StrawHat $>\text{Direct Integration} – \text{Even less clear...}$
Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. computing time

\[ \text{Error on power prod. } e_{\text{p}}^\infty (\%) \]

\[ \text{Computing time (s.)} \]

\[ \text{RT} \]

\[ \text{RT002} \]
\[ \text{RT002-DI} \]
\[ \text{RT002-STR} \]
\[ \text{RT002-POI} \]
\[ \text{RT112} \]
\[ \text{RT112-DI} \]
\[ \text{RT112-STR} \]
\[ \text{RT112-POI} \]
\[ \text{RT222} \]
\[ \text{RT222-DI} \]
\[ \text{RT222-STR} \]
\[ \text{RT222-POI} \]

\[ \text{RT}_2 \geq \text{RT}_1 \geq \text{RT}_0 \]
Real-World results – PWR with UOX-MOX interfaces

Accuracy vs. computing time

Poisson out of the game:

StrawHat > Direct Integration and RT₂ > RT₁ > RT₀ (almost always)
Numerical Benchmark

StrawHat Method
Numerical Benchmark
StrawHat Method