Adjoint-Based Uncertainty Quantification and Sensitivity Analysis for Reactor Depletion Calculations

Hayes F. Stripling[1] Marvin L. Adams[1] Mihai Anitescu[2]

[1]Department of Nuclear Engineering, Texas A&M University

[2]Argonne National Laboratory

25 July 2013

DOE CSGF Program Review

We are interested in depletion perturbation calculations.

Overview of the depletion perturbation problem:

1) The forward problem:
   - Solve a transport equation: solve for neutron flux shape, ψ
   - Solve a material balance equation for densities, N
   - Compute a derived quantity of interest, QOI or Q

2) The adjoint problem:
   - Mathematically related to forward system
   - Solved backwards in time for adjoint variables $\psi^\dagger$ and $N^\dagger$

3) Perform uncertainty quantification calculations:
   - Sensitivity of QOI with respect to uncertain parameters, $\frac{dQ}{dp}$
   - Cost of obtaining $\frac{dQ}{dp}$ does not grow rapidly with $\texttt{length}(p)$

4) Target: large systems, *lots of p's*, and advanced architectures

We are interested in depletion perturbation calculations.

Overview of the depletion perturbation problem:

1) The forward problem:
   - Solve a transport equation: solve for neutron flux shape, ψ
   - Solve a material balance equation for densities, N
   - Compute a derived quantity of interest, QOI or Q

2) The adjoint problem:
   - Mathematically related to forward system
   - Solved backwards in time for adjoint variables $\psi^\dagger$ and $N^\dagger$

3) Perform uncertainty quantification calculations:
   - Sensitivity of QOI with respect to uncertain parameters, $\frac{dQ}{dp}$
   - Cost of obtaining $\frac{dQ}{dp}$ does not grow rapidly with $\texttt{length}(p)$

4) Target: large systems, *lots of p's*, and advanced architectures

We are interested in depletion perturbation calculations.

Overview of the depletion perturbation problem:

1) The forward problem:
   - Solve a transport equation: solve for neutron flux shape, ψ
   - Solve a material balance equation for densities, N
   - Compute a derived quantity of interest, QOI or Q

2) The adjoint problem:
   - Mathematically related to forward system
   - Solved backwards in time for adjoint variables $\psi^\dagger$ and $N^\dagger$

3) Perform uncertainty quantification calculations:
   - Sensitivity of QOI with respect to uncertain parameters, $\frac{dQ}{dp}$
   - Cost of obtaining $\frac{dQ}{dp}$ does not grow rapidly with $\texttt{length}(p)$

4) Target: large systems, *lots of p's*, and advanced architectures

We are interested in depletion perturbation calculations.

Overview of the depletion perturbation problem:

1) The forward problem:
   - Solve a transport equation: solve for neutron flux shape, ψ
   - Solve a material balance equation for densities, N
   - Compute a derived quantity of interest, QOI or Q

2) The adjoint problem:
   - Mathematically related to forward system
   - Solved backwards in time for adjoint variables $\psi^{\dagger}$ and $N^{\dagger}$

3) Perform uncertainty quantification calculations:
   - Sensitivity of QOI with respect to uncertain parameters, $\frac{dQ}{dp}$
   - Cost of obtaining $\frac{dQ}{dp}$ does not grow rapidly with $\texttt{length}(p)$

4) Target: large systems, *lots of p's*, and advanced architectures

Example: The source-driven forward depletion equations

Suppose we are using explicit time-stepping,

$$\begin{aligned}
\text{Material balance:} \quad & N_n = N_{n-1} + hB_{n-1}N_{n-1} \\
\text{Transport Eq.:} \quad & H_n\psi_n = S_0 \\
\text{Initial Condition:} \quad & N(t_0) = N_0 \\
\text{Time increment:} \quad & t_n = t_{n-1} + h
\end{aligned}$$

and we are interested in a QOI that depends only on the solution at $t = t_f$:

$$Q = \left\langle R\Big(N(t_f), \psi(t_f), p\Big) \right\rangle_{E,\mathcal{D},\Omega} \equiv \int dr \int dE \int d\Omega\, R(t_f).$$

Our goal is to compute $\frac{dQ}{dp}$ for *every* p.

The adjoint problem that leads to $\frac{dQ}{dp}$

Adjoint material balance: $N^{\dagger}_{n-1} = N^{\dagger}_{n} - h \left\langle \psi^{\dagger}_{n}, \frac{\partial H_n \psi_n}{\partial N} \right\rangle_{E,\mathcal{D},\Omega} - B^{\dagger}_{n} N^{\dagger}_{n}$

Adjoint transport Eq.: $H^{\dagger}_{n-1} \psi^{\dagger}_{n-1} = N^{\dagger}_{n-1} \frac{\partial B_{n-1} N_{n-1}}{\partial \psi}$

Terminal condition: $N^{\dagger}(t_f) = \mathcal{L}(N, \psi)$

Checkpointing the forward solution

At each time step, we must have access to the **forward** solution in order to compute the terms in the **adjoint** equations.

The adjoint problem that leads to $\frac{dQ}{dp}$

Adjoint material balance: $N^\dagger_{n-1} = N^\dagger_n - h\left\langle \psi^\dagger_n, \frac{\partial H_n \psi_n}{\partial N} \right\rangle_{E,\mathcal{D},\Omega} - B^\dagger_n N^\dagger_n$

Adjoint transport Eq.: $H^\dagger_{n-1}\psi^\dagger_{n-1} = N^\dagger_{n-1}\frac{\partial B_{n-1}N_{n-1}}{\partial \psi}$

Terminal condition: $N^\dagger(t_f) = \mathcal{L}(N,\psi)$

Checkpointing the forward solution

At each time step, we must have access to the **forward** solution in order to compute the terms in the **adjoint** equations.

It's simple to imagine a high-fidelity problem that quickly overruns RAM capacity.

Using DOE's Sequoia as a model: 100k nodes with 16 cores/node and 16 GB RAM/node. A high-fidelity reactor problem might have (per node)

- 200 energy groups
- 500 angles
- 1000 spatial cells
- 4 elements per cell (linear FEM)

That's 400M unknowns, or 3.2GB per snapshot of ψ per node!

The future does not bode well for memory-intensive algorithms. We're headed towards

a) Extreme cpu-counts (high FLOP rates)

b) Decreasing RAM availability (per cpu)

c) Expensive I/O (relative to FLOPs)

It's simple to imagine a high-fidelity problem that quickly overruns RAM capacity.

Using DOE's Sequoia as a model: 100k nodes with 16 cores/node and 16 GB RAM/node. A high-fidelity reactor problem might have (per node)

- 200 energy groups
- 500 angles
- 1000 spatial cells
- 4 elements per cell (linear FEM)

That's 400M unknowns, or 3.2GB per snapshot of ψ per node!

The future does not bode well for memory-intensive algorithms. We're headed towards

a) Extreme cpu-counts (high FLOP rates)

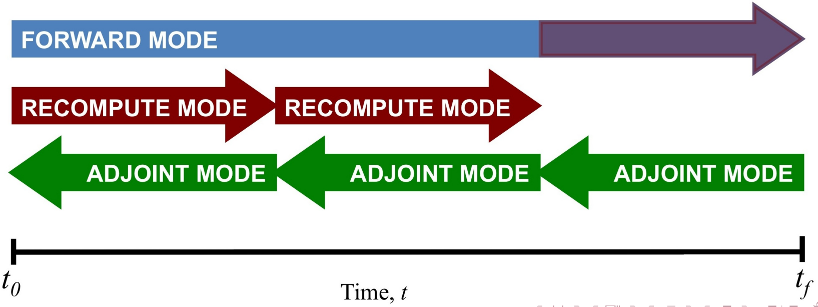
b) Decreasing RAM availability (per cpu)

c) Expensive I/O (relative to FLOPs)

The general checkpointing strategy

Overall Idea:

1) Progress through forward problem, checkpointing snapshots of forward solution at intervals
2) Enter adjoint mode
3) Recompute "chunks" of forward solution as required



We developed algorithms that leverage a lower-order representation of the angular flux solution.

- Our transport solvers iterate to converge the flux solution:

$$\Omega \cdot \nabla \psi^{(\ell+1)} + \Sigma_t \psi^{(\ell+1)} = S(\psi^{(\ell)}).$$

- Each update is called a "sweep."
- The angular dependence of the source term is represented as a truncated polynomial expansion. For example, the scattering source:

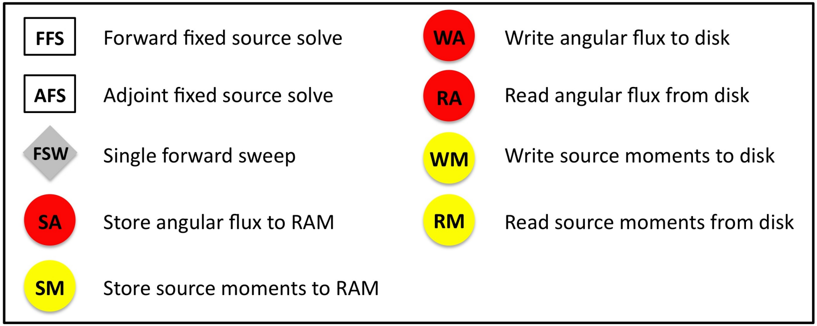
$$\begin{aligned} S_S(\psi^{(\ell)}) &= \int_0^\infty dE' \int_{4\pi} d\Omega' \psi^{(\ell)}(E', r, \Omega') \Sigma_s(E' \to E, \Omega' \to \Omega) \\ &\approx \int_0^\infty dE' \sum_{k=0}^{\mathcal{M}} C_k \Sigma_{s,k}(E' \to E) Y_k(\Omega) \int_{4\pi} d\Omega' Y_k(\Omega') \psi^{(\ell)}(E', r, \Omega') \end{aligned}$$

- The number of moments, $\mathcal{M}$ is at most equal to the number of discrete ordinates, but typically it's much less.

The new schemes will checkpoint only the converged source moments.

- This reduces RAM footprint and file I/O loads.
- When ψ is needed at a particular time step, the cost is a *single* sweep.
    - This simply re-performs the last iterate of the source-iteration scheme.
- These schemes mimic the evolution of advanced computer architectures.

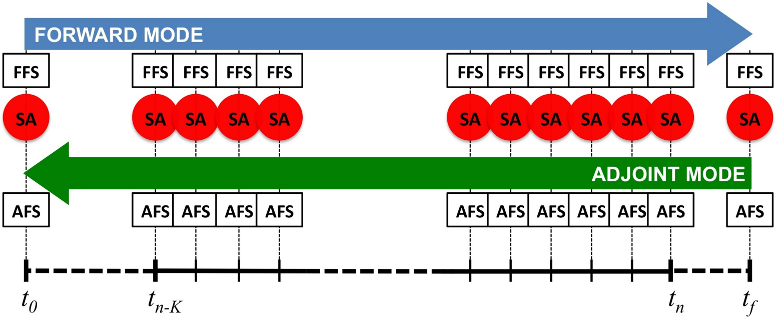
I will use schematics to describe and analze the schemes. Here is the legend:



FFS	Forward fixed source solve	WA	Write angular flux to disk
AFS	Adjoint fixed source solve	RA	Read angular flux from disk
FSW	Single forward sweep	WM	Write source moments to disk
SA	Store angular flux to RAM	RM	Read source moments from disk
SM	Store source moments to RAM		

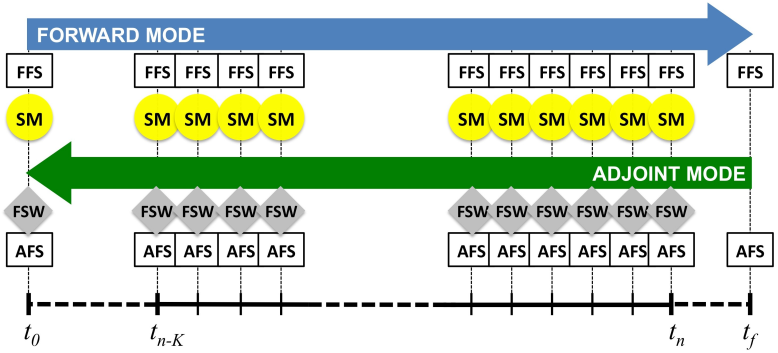
Checkpointing algorithms: STOR_ALL mode.

- Store the full ψ vector at each time step during forward mode
- No re-compute required during adjoint mode



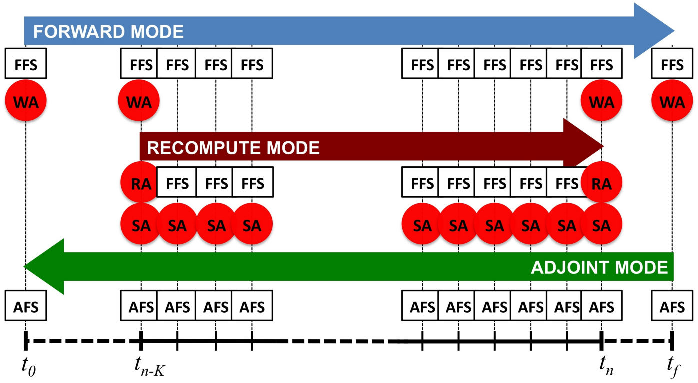
Checkpointing algorithms: STOR_MOM mode

- Store only the converged source moments during forward mode
- A single sweep is required to recover ψ before each adjoint solve



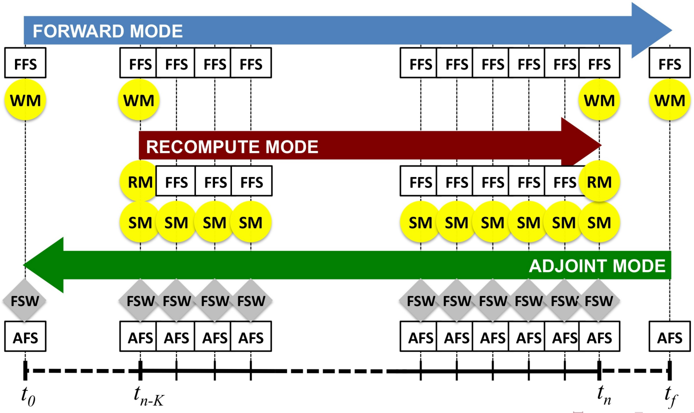
Checkpointing algorithms: CKPT_ALL mode

- Write to-file full ψ vector every K time-steps during forward mode
- Re-compute and store the full ψ during recompute mode
- No re-compute required during adjoint mode



Checkpointing algorithms: CKPT_MOM mode

- Write to-file source moments every K time-steps during forward mode
- Re-compute and store the source moments during recompute mode
- Single forward sweep required before each adjoint solve



Predictions of the fixed-source cost and RAM footprint of each algorithm.

Legend:

- N_R = number of re-compute segments
- K = number of stages per re-compute segment
- M_ψ = RAM footprint of ψ vector
- M_S = RAM footprint of source moments vector

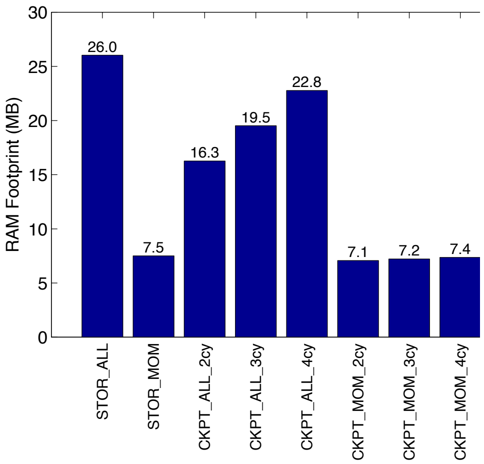
Scheme	Recompute Fixed Source Solves	RAM Footprint
STOR_ALL	0	$M_\psi(K \cdot N_R + 1)$
STOR_MOM	0	$2M_\psi + M_S(K \cdot N_R)$
CKPT_ALL	$(N_R - 1)(K - 1)$	$M_\psi(K + 3)$
CKPT_MOM	$(N_R - 1)(K - 1)$	$2M_\psi + M_S(K + 2)$

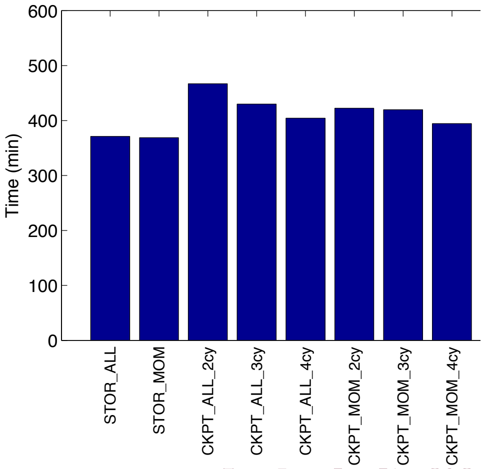
Note: $K * N_R$=total # timesteps

We scaled the methods by increasing both the number of processors and the number of unknowns per processor.

- Schemes:
  1. STOR_ALL
  2. STOR_MOM
  3. CKPT_ALL_2, CKPT_ALL_3, CKPT_ALL_4
  4. CKPT_MOM_2, CKPT_MOM_3, CKPT_MOM_4
- Processor Counts: 1024, 2048, 4096
- Problem sizes (unk. per cpu): 200k, 400k, 800k

Memory footprint and time to solution (400k unknowns/proc, 2048 processros)





Our checkpointing schemes improve the tractability of high-fidelity depletion perturbation calculations.

We eliminate the need to store multiple copies of ψ by checkpointing converged source moments.

- This strategy reduces the memory footprint and I/O load at the cost of extra FLOPs, and
- mimics the evolution of machine architectures.

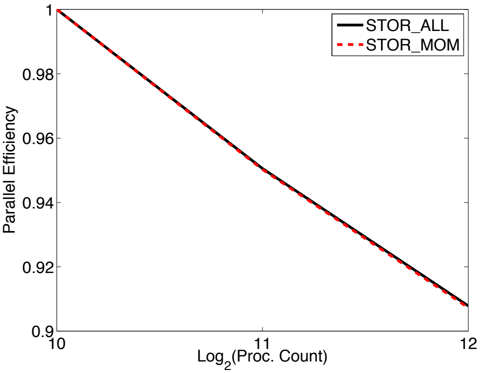
Scaling results show that our new schemes greatly reduce the memory footprint and in many cases reduce time to solution.

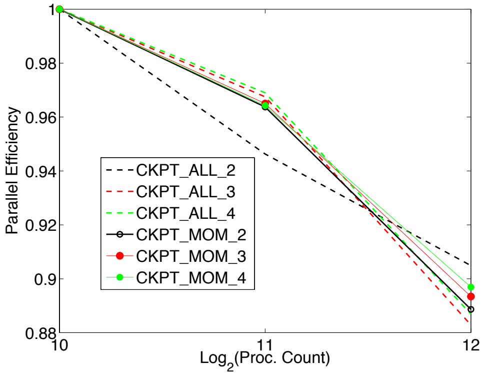
- We are still working to characterize and tune schemes at larger core counts and on larger problems.
- Variants of these schemes incur even more FLOP costs in order to further reduce memory and I/O loads.

Questions and Discussion

- Many, many thanks to the DOE CSGF program for the opportunities and funding that it provides.
- Some work was funded by the Center for Exascale Simulations of Advanced Reactors (CESAR), a DOE exascale co-design center.

Weak Scaling Results*** (400k unk/proc)


Parallel Efficiency
Log2(Proc. Count)
STOR_ALL
STOR_MOM


Parallel Efficiency
Log2(Proc. Count)
CKPT_ALL_2
CKPT_ALL_3
CKPT_ALL_4
CKPT_MOM_2
CKPT_MOM_3
CKPT_MOM_4