Advanced Algorithmic Differentiation: Memory-reduced Calculation of Adjoints

Andrea Walther and Sri Tadinada
Institut für Mathematik
Humboldt-Universität zu Berlin

CWI Autumn School 2025
Artificial Intelligence in Natural Sciences

Berlin Mathematics Research Center

Outline

Starting Point: Adjoints are Great!

Starting Point: Adjoints are Great!

BUT

$$\mathrm{MEM}(\bar{y}^\top F'(x)) \;\sim\; \mathrm{OPS}(F(x))$$

independent of chosen approach!

Starting Point: Adjoints are Great!

BUT

$$\mathrm{MEM}(\bar{y}^\top F'(x)) \;\sim\; \mathrm{OPS}(F(x))$$

independent of chosen approach!

Alternatives:

- no assumptions on structure of $F(x)$

  $\Rightarrow$ For AD tools: subroutine-oriented checkpointing
  see, e.g., Tapenade and OpenAD

Starting Point: Adjoints are Great!

BUT

$$\mathrm{MEM}(\bar{y}^\top F'(x)) \;\sim\; \mathrm{OPS}(F(x))$$

independent of chosen approach!

Alternatives:

- no assumptions on structure of $F(x)$

  $\Rightarrow$ For AD tools: subroutine-oriented checkpointing
  see, e.g., Tapenade and OpenAD
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
    $\Rightarrow$ checkpointing
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    $=$ fixed point iterations
    $\Rightarrow$ adapted derivative calculation

Starting Point: Adjoints are Great!

BUT

$$\mathrm{MEM}(\bar{y}^\top F'(x)) \;\sim\; \mathrm{OPS}(F(x))$$

independent of chosen approach!

Alternatives:

- no assumptions on structure of $F(x)$
  $\Rightarrow$ For AD tools: subroutine-oriented checkpointing
  see, e.g., Tapenade and OpenAD
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
    $\Rightarrow$ checkpointing
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    = fixed point iterations
    $\Rightarrow$ adapted derivative calculation
- $F(x)$ = iterative solve
  $\Rightarrow$ adapted derivative calculation

Berlin Mathematics Research Center

Starting Point: Adjoints are Great!

BUT

$$\text{MEM}(\bar{y}^\top F'(x)) \;\sim\; \text{OPS}(F(x))$$

independent of chosen approach!

Alternatives:

- no assumptions on structure of $F(x)$
  $\Rightarrow$ For AD tools: subroutine-oriented checkpointing
  see, e.g., Tapenade and OpenAD
- $F(x)$ = time-stepping procedure
  - transient procedure, i.e., real time-dependent process
    $\Rightarrow$ checkpointing
  - pseudo time-stepping, e.g., aerodynamics, spin-up
    $=$ fixed point iterations
    $\Rightarrow$ adapted derivative calculation
- $F(x)$ = iterative solve
  $\Rightarrow$ adapted derivative calculation

Gradient Calculation (Time Integration)

1. Forward integration: For given x_0

   $x_{i+1} = F_i(x_i, u_i), \; i = 0, \ldots, l-1$

   with x_i state, u_i control

   number of time steps l might be not known a priori

2. Evaluation of target function $J(x, u)$

3. Backward integration:

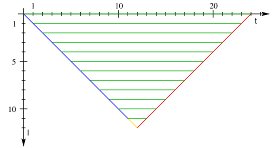
   $\bar{x}_{i-1} = \bar{F}_i(\bar{x}_i, x_{i-1}, u_{i-1}, u_i), \; i = l, \ldots, 1$

   AD, discretization of continuous adjoint, hand coded, ...

4. Gradient calculation

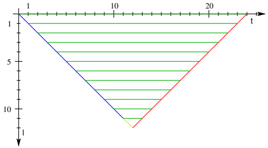
Idea of Checkpointing

Store-Everything Approach:

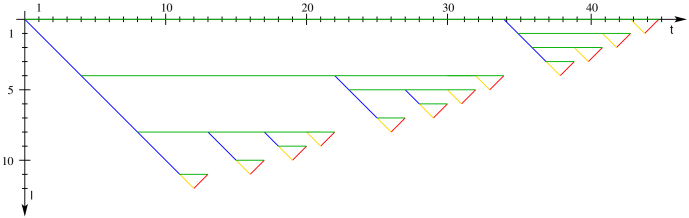


Idea of Checkpointing

Store-Everything Approach:



Checkpointing:



Checkpointing Theory

Goal: Minimal number of recomputations for c checkpoints

Available results:

- l known, constant step costs
  (Griewank 1992), (Griewank, Walther 2000), (Kowarz, Walther 2007)
- l known, variable step costs
  (Walther 2000), (Hinze, Sternberg 2005)

Checkpointing Theory

Goal: Minimal number of recomputations for c checkpoints

Available results:

- l known, constant step costs
  (Griewank 1992), (Griewank, Walther 2000), (Kowarz, Walther 2007)
- l known, variable step costs
  (Walther 2000), (Hinze, Sternberg 2005)
- l unknown, constant step costs
  (Hinze, Walther, Sternberg 2006), (Stumm, Walther 2010), (Moin, Wang 2010)

Checkpointing Theory

Goal: Minimal number of recomputations for c checkpoints

Available results:

- l known, constant step costs
  (Griewank 1992), (Griewank, Walther 2000), (Kowarz, Walther 2007)
- l known, variable step costs
  (Walther 2000), (Hinze, Sternberg 2005)
- l unknown, constant step costs
  (Hinze, Walther, Sternberg 2006), (Stumm, Walther 2010), (Moin, Wang 2010)
- l known, variable access cost
  (Stumm, Walther 2009), (Aupy, Hovland et al. 2015), (Schanen, Marin et al. 2015)

Checkpointing Theory

Goal: Minimal number of recomputations for c checkpoints

Available results:

- l known, constant step costs
  (Griewank 1992), (Griewank, Walther 2000), (Kowarz, Walther 2007)
- l known, variable step costs
  (Walther 2000), (Hinze, Sternberg 2005)
- l unknown, constant step costs
  (Hinze, Walther, Sternberg 2006), (Stumm, Walther 2010), (Moin, Wang 2010)
- l known, variable access cost
  (Stumm, Walther 2009), (Aupy, Hovland et al. 2015), (Schanen, Marin et al. 2015)

Also applicable for continuous adjoints!

Checkpointing Theory

Goal: Minimal number of recomputations for c checkpoints

Available results:

- l known, constant step costs
  (Griewank 1992), (Griewank, Walther 2000), (Kowarz, Walther 2007)
- l known, variable step costs
  (Walther 2000), (Hinze, Sternberg 2005)
- l unknown, constant step costs
  (Hinze, Walther, Sternberg 2006), (Stumm, Walther 2010), (Moin, Wang 2010)
- l known, variable access cost
  (Stumm, Walther 2009), (Aupy, Hovland et al. 2015), (Schanen, Marin et al. 2015)

Also applicable for continuous adjoints!

Berlin Mathematics Research Center

Theory for Binomial Checkpointing I

Theorem (Complexity Result I)

Given

$$\begin{array}{lcl} l & = & \textit{total number of time steps to be reversed} \\ c & = & \textit{number of checkpoints that can be used} \\ r & = & \textit{repetition number, i.e., unique integer satisfying} \end{array}$$

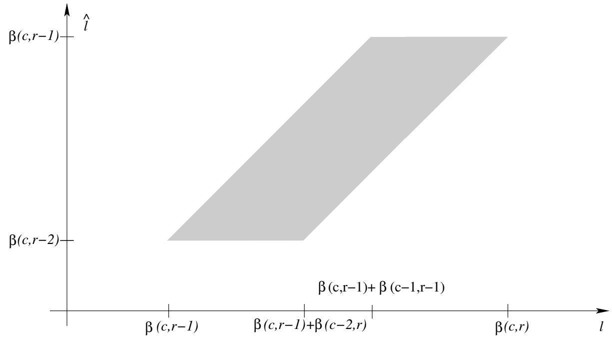
$$\beta(c, r-1) \; < \; l \; \leq \; \beta(c, r) \; \equiv \; \binom{c+r}{c}$$

Then: The minimal number of time step evaluations required equals

$$t(c, l) \; = \; rl - \beta(c+1, r-1) + 1$$

Theory for Binomial Checkpointing I

Place of next checkpoint $\hat{l}$:



Berlin Mathematics Research Center MATH+

Theory for Binomial Checkpointing II

Theorem (Complexity Result II)

Given

$$\begin{array}{lcl} l & = & \textit{total number of time steps to be reversed} \\ c & = & \textit{number of checkpoints that can be used} \\ r & = & \textit{repetition number, i.e., unique integer satisfying} \end{array}$$

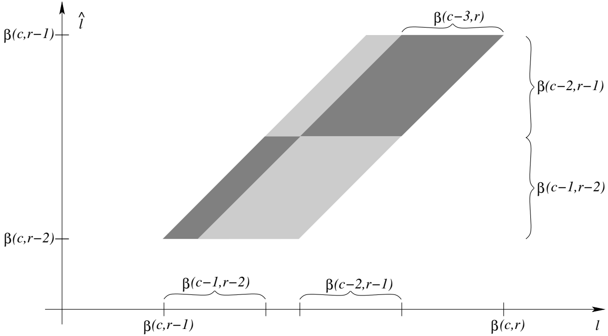
$$\beta(c, r-1) \;<\; l \;\leq\; \beta(c, r) \;\equiv\; \binom{c+r}{c}$$

Then: Among all checkpoint schedules satisfying the previous theorem the minimal number of times a current state is saved onto the stack of checkpoints is given by

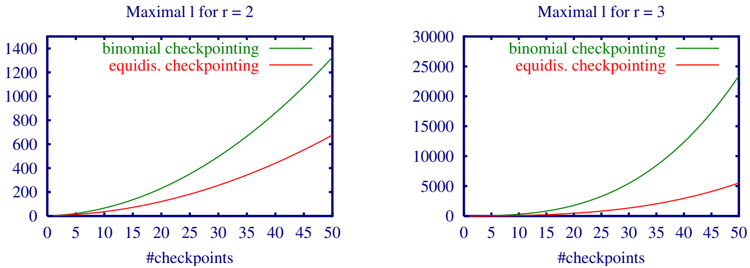
$$q(l, r) = \begin{cases} \beta(c-1, r-1) & \text{if } l \leq \beta(c, r-1) + \beta(c-1, r-1) \\ l - \beta(c, r-1) & \text{if } l \geq \beta(c, r-1) + \beta(c-1, r-1) \end{cases}$$

Theory for Binomial Checkpointing II

Place of next checkpoint $\hat{l}$:



Repetition Number r vs Time Step Number l



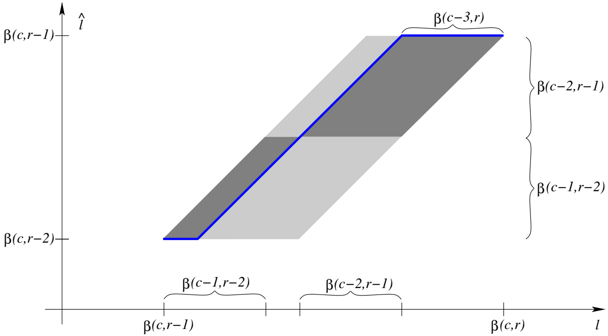
[Walther, Griewank 2004]

The Checkpointing Routine revolve()

```
...
do
    whatodo = revolve()
    switch(whatodo)
        case advance:  for old_state < i ≤ next_state    forward(x)
        case takeshot: store(x, xstore, current_checkpoint)
        case firsturn:  init(bar_x, bar_u)
                        reverse(bar_x, bar_u)
        case youturn:   reverse(bar_x, bar_u)
        case restore:   restore(x, xstore, current_check)
while(whatodo <> terminate)
```

Current Strategy Chosen by revolve()

Place of next checkpoint $\hat{l}$:



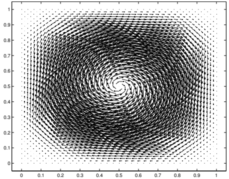
Test Case: Cavity Flow

joint work with Michael Hinze

Cost function of tracking typ, initial state:

$$y_0(x) = e \begin{bmatrix} (\cos 2\pi x_1 - 1) \sin 2\pi x_2 \\ -(\cos 2\pi x_2 - 1) \sin 2\pi x_1 \end{bmatrix},$$

$\Omega := (0,1) \times (0,1)$, $T = 1$, $\mathrm{Re} = 10$, $e =$ Euler number



Test Case: Cavity Flow

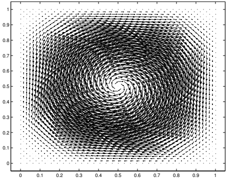
joint work with Michael Hinze

Cost function of tracking typ, initial state:

$$y_0(x) = e \begin{bmatrix} (\cos 2\pi x_1 - 1)\sin 2\pi x_2 \\ -(\cos 2\pi x_2 - 1)\sin 2\pi x_1 \end{bmatrix},$$

$\Omega := (0,1) \times (0,1)$, $T = 1$, $\mathrm{Re} = 10$, $e =$ Euler number

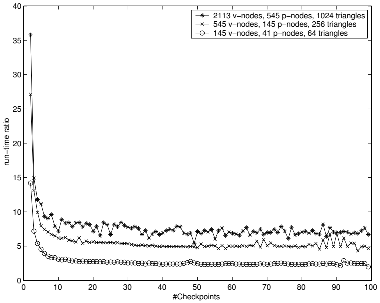
Desired state:



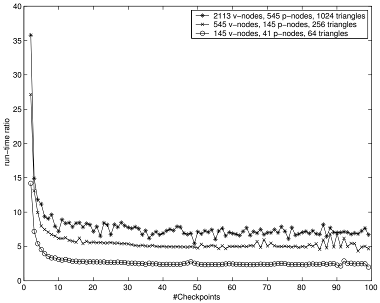
$$\begin{aligned} z(x,t) &= \begin{bmatrix} \varphi_{x_2}(x,t) \\ -\varphi_{x_1}(x,t) \end{bmatrix}, \\ \varphi(x,t) &= \theta(x_1,t)\theta(x_2,t), \\ \theta(y,t) &= (1-y)^2(1-\cos kt), \\ y &\in [0,1]. \end{aligned}$$

Berlin Mathematics Research Center

Runtime Ratios, 1000 Time Steps



Runtime Ratios, 1000 Time Steps



- Run-time ratio between 2 and 7.
- Only a very small number of checkpoints required!

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!
- Time-dependent problem:
  Implicit time-stepping ? Unknown number of time steps?
  Apply adapted checkpointing strategy

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!
- Time-dependent problem:
  Implicit time-stepping ? Unknown number of time steps?
  Apply adapted checkpointing strategy
- Pseudo time-stepping:
  Two-phase approach "easy" to apply
  One-shot approach more efficient but also more complicated

Summary

- Adjoint computation may result in emormous memory requirement
- Exploit structure!
- Time-dependent problem:
  Implicit time-stepping ? Unknown number of time steps?
  Apply adapted checkpointing strategy
- Pseudo time-stepping:
  Two-phase approach “easy” to apply
  One-shot approach more efficient but also more complicated
- Differentiation of Newton-like iterations OK
- Be careful when differentiating solvers like CG, GMRES, . . .

Berlin Mathematics Research Center