

# Recent developments for checkpointing strategies

Andreas Kowarz, Andrea Walther

Institut für Wissenschaftliches Rechnen  
Technische Universität Dresden

**2nd European Workshop on Automatic Differentiation**  
17.11.2005

Granted by DFG (WA 1607/2-1)

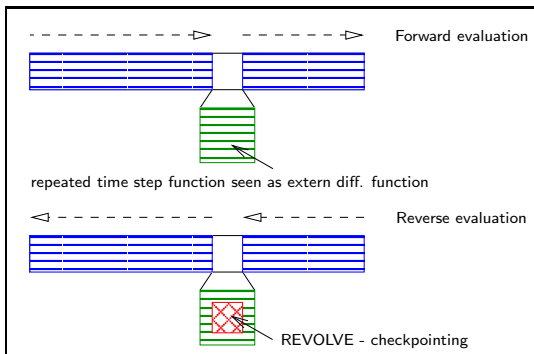
- 1 Binomial checkpointing in ADOL-C
  - Preliminary considerations
  - Demands on the user
  - Practical results
- 2 Multi-Stage checkpointing
  - Motivation
  - Ideas & observations
  - Mathematical proof
- 3 Summary & outlook

# Binomial checkpointing in ADOL-C

## Used base:

- REVOLVE (Andreas Griewank, Andrea Walther), distributed with ADOL-C
- checkpointing using the concept of extern diff. functions

## Underlying approach:



## User has to:

- provide time step function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  with properties
  - identical in and output vector
  - based on class `adouble` and on type `double` (two separate versions)

```
int timeStepFunction(int n, adouble *x);  
int d_timeStepFunction(int n, double *x);
```

- register the time step function using

```
CpInfos *reg_timestep_fct(timeStepFunction);
```

## User has to (2):

- provide the checkpointing characteristics by using a pre-defined structure

```
cpInfos->timeStepFunction_double=d_timeStepFunction;  
cpInfos->checkpoints=100;  
cpInfos->steps=20000;  
cpInfos->n=1000;  
cpInfos->adp_x=...;  
cpInfos->adp_y=...;
```

- use the checkpointing interface to initialize the checkpointing procedure while taping the global function

```
int result=checkpointing(cpInfos);
```

## Source code - Example:

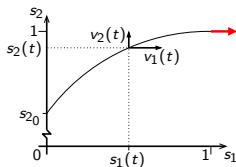
```
#include <checkpointing.h>
int timeStepFunction(int n, adouble *x);
int d_timeStepFunction(int n, double *x);

adouble *xa=new adouble[n], *ya=new adouble[m];
CpInfos *cpInfos;
...
trace_on(...);
for (int i=0; i<n; ++i) xa[i]<<=...;
cpInfos=reg_timestep_fct(timeStepFunction);
cpInfos->timeStepFunction_double=d_timeStepFunction;
cpInfos->checkpoints=100;
cpInfos->steps=20000;
...
int result=checkpointing(cpInfos);
...
for (int i=0; i<m; ++i) ya[i]>>=...;
trace_off();
...
zos_forward(...);
fos_reverse(...);
...
```

## Example: Trajectory in 2D

Initial value problem:

$$x_0 = \begin{pmatrix} 0 \\ s_{20} \\ v_{10} \\ v_{20} \end{pmatrix}, \quad \dot{x}(t) = f(x), \quad x_* = \begin{pmatrix} 1 \\ 1 \\ v_{1*} \\ 0 \end{pmatrix}$$



- discretization within time interval  $[0, T]$  with  $T = 1$  using  $l$  steps,  $l = 100, 1000, 10000$
- integration with explicit Euler

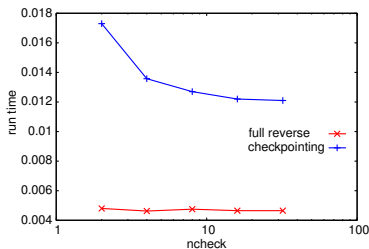
$$x_{k+1} = x_k + h * f(x_k) \quad \text{with} \quad x_0 = x(0), \quad h = \frac{1}{l}$$

- $x(T)$  object to  $\min_{x_0} J(x(T))$  with

$$J(x(T)) = \frac{1}{2} * \left( (s_1(T) - 1)^2 + (s_2(T) - 1)^2 + v_2(T)^2 \right)$$

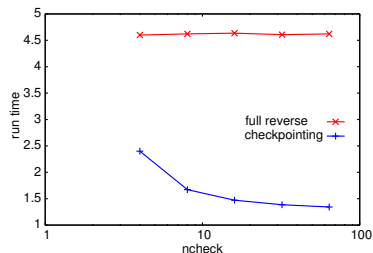
## Run times:

•  $l = 100$



• complies with theoretical results

•  $l = 10000$



• too high memory requirements for full reverse → hard disk access

⇒ **Keep an eye on the memory access costs!**

# Multi-Stage checkpointing

## Assumption for the binomial checkpointing:

- negligible costs for reading/writing checkpoints

## Special cases in practice:

- main memory too small  
due to too many or too large checkpoints
- ⇒ different checkpoint access costs

## Example:

- large application with parallel I/O
    - several levels of checkpoint costs based on memory hierarchy (and I/O ?)
- ⇒ planned cooperation with Prof. Heuveline

## Base:

- well known and proven statements for binomial checkpointing with negligible checkpoint costs

## Base:

- well known and proven statements for binomial checkpointing with negligible checkpoint costs

## Prerequisites:

- checkpoint costs (reads, writes) are known
- checkpoint type can be chosen by the user (cheap  $\leftrightarrow$  expensive)

## Base:

- well known and proven statements for binomial checkpointing with negligible checkpoint costs

## Prerequisites:

- checkpoint costs (reads, writes) are known
- checkpoint type can be chosen by the user (cheap  $\leftrightarrow$  expensive)

## Assumption:

- earlier checkpoints are set less frequently than the later ones
- assigning expensive storage locations to earlier checkpoints minimizes overall costs

## Base:

- well known and proven statements for binomial checkpointing with negligible checkpoint costs

## Prerequisites:

- checkpoint costs (reads, writes) are known
- checkpoint type can be chosen by the user (cheap  $\leftrightarrow$  expensive)

## Assumption:

- earlier checkpoints are set less frequently than the later ones
- assigning expensive storage locations to earlier checkpoints minimizes overall costs

## Goal:

- mathematically proven cost function



## Revolve output analysis:

l=33, c=4	global r is 3	c0: 1W , 3R	c1: 3W , 6R	c2: 5W , 9R	c3: 9W , 14R
l=34, c=4	global r is 3	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 9W , 14R
l=35, c=4	global r is 3	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 15R
l=36, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 16R
l=37, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 17R
l=38, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 18R
l=39, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 11R	c3: 10W , 18R
l=40, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 11R	c3: 10W , 19R
l=41, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 11R	c3: 10W , 20R
l=42, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 12R	c3: 10W , 20R
l=43, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 12R	c3: 10W , 21R

## Revolve output analysis:

l=33, c=4	global r is 3	c0: 1W , 3R	c1: 3W , 6R	c2: 5W , 9R	c3: 9W , 14R
l=34, c=4	global r is 3	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 9W , 14R
l=35, c=4	global r is 3	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 15R
l=36, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 16R
l=37, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 17R
l=38, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 10R	c3: 10W , 18R
l=39, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 11R	c3: 10W , 18R
l=40, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 11R	c3: 10W , 19R
l=41, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 11R	c3: 10W , 20R
l=42, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 12R	c3: 10W , 20R
l=43, c=4	global r is 4	c0: 1W , 3R	c1: 3W , 6R	c2: 6W , 12R	c3: 10W , 21R

**Theorem:**

- number of writes for checkpoint  $i$ :

$$w_i = \binom{i+r-2}{r-2} \quad i = 0, \dots, c-1$$

for:  $\beta(c, r-1) < l \leq \beta(c, r-1) + \beta(c-1, r-1)$

with:  $\beta(c, r) = \binom{c+r}{c}$

## Theorem:

- number of writes for checkpoint  $i$ :

$$w_i = \binom{i+r-2}{r-2} \quad i = 0, \dots, c-1$$

for:  $\beta(c, r-1) < l \leq \beta(c, r-1) + \beta(c-1, r-1)$

with:  $\beta(c, r) = \binom{c+r}{c}$

## Sketch of Proof:

- start of induction:
  - for  $c = 1, l \in \mathbf{N}$  : checkpoint is set exactly once
  - for  $c \in \mathbf{N}, l = c + 1$  : each checkpoint is set exactly once



## Sketch of Proof (2):

- proposition of induction:
  - choose unique integer  $r$  for given  $l$  and  $c$  satisfying

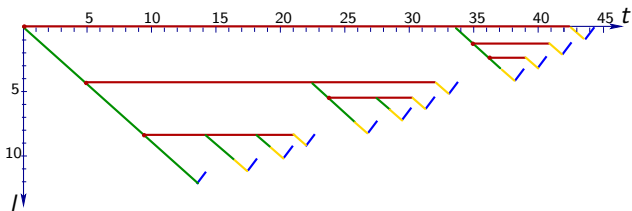
$$\beta(c, r - 1) < l \leq \beta(c, r - 1) + \beta(c - 1, r - 1)$$

- then the number of writes  $w_i$  for each checkpoint is given by

$$w_i = \binom{i + r - 2}{r - 2} \quad i = 0, \dots, c - 1$$

## Sketch of Proof (3):

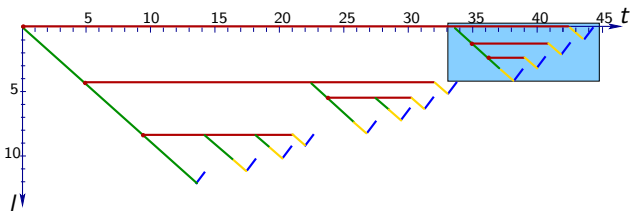
- induction step:





## Sketch of Proof (3):

- induction step:



Storing next checkpoint on position  $\hat{l}$  results in:

- lower part with  $\hat{c} = c - 1, \hat{r} = r, l - \hat{l}$
- upper part with  $\hat{c} = c, \hat{r} = r - 1, \hat{l}$

## Open questions:

- formula for  $\beta(c, r - 1) + \beta(c - 1, r - 1) < l \leq \beta(c, r)$  ?
  - number of checkpoint writes not constant
  - new idea based on a different REVOLVE strategy

## Open questions:

- formula for  $\beta(c, r - 1) + \beta(c - 1, r - 1) < l \leq \beta(c, r)$  ?
  - number of checkpoint writes not constant
  - new idea based on a different REVOLVE strategy
- what about the read counts?
  - ignored by REVOLVE so far!

## Open questions:

- formula for  $\beta(c, r - 1) + \beta(c - 1, r - 1) < l \leq \beta(c, r)$  ?
  - number of checkpoint writes not constant
  - new idea based on a different REVOLVE strategy
- what about the read counts?
  - ignored by REVOLVE so far!
- revolve algorithm optimal for Multi-Stage checkpointing?

## Summary:

- binomial checkpointing integrated into ADOL-C
- Multi-Stage checkpointing introduced with partial proof

## Summary:

- binomial checkpointing integrated into ADOL-C
- Multi-Stage checkpointing introduced with partial proof

## Outlook:

- update ADOL-C towards nested tape evaluation
- complete Multi-Stage checkpointing proof, implement and test it

**Thanks to the audience!**

