Efficient Sensitivities for the Spin-up Phase

Thomas Kaminski, Ralf Giering,
and Michael Voßbeck

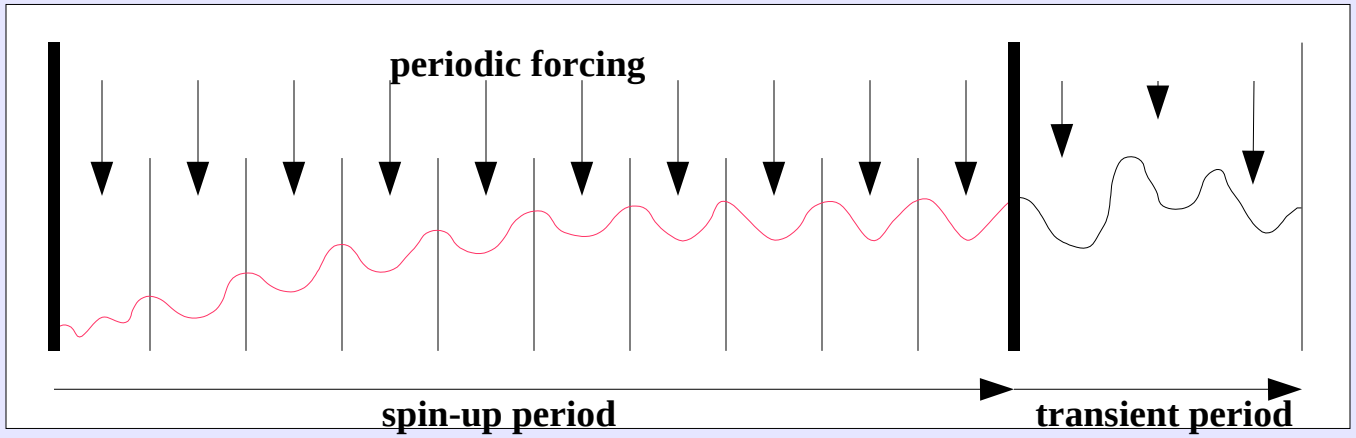
Thanks to *Srikanth Akkaram*

Copy of presentation at http://www.FastOpt.com

Outline

- Spin-up
- Strategies for spin-up sensitivities
- Performance for test code
- Heuristic formula for performance
- Outlook
- Summary

Spinup



Examples:

- **Simulating the global atmospheric CO2 concentration with a tracer model; Forcing: CO2 surface fluxes**
- **Simulating the global ocean circulation (including the deep ocean); Forcing: wind, freshwater, solar radiation ...**
- **Simulating CO2 fluxes from terrestrial biosphere (including decompositon of organic carbon in soils) to atmosphere; Forcing: temperature, rain, solar radiation...**
- **Simulating the steady aerodynamic flow around an airfoil Forcing: flow at surface and far-field (angle of attack, Mach #)?**

Spinup: Formalisation

parameters,
boundary conditions

$$y = f(b, x)$$

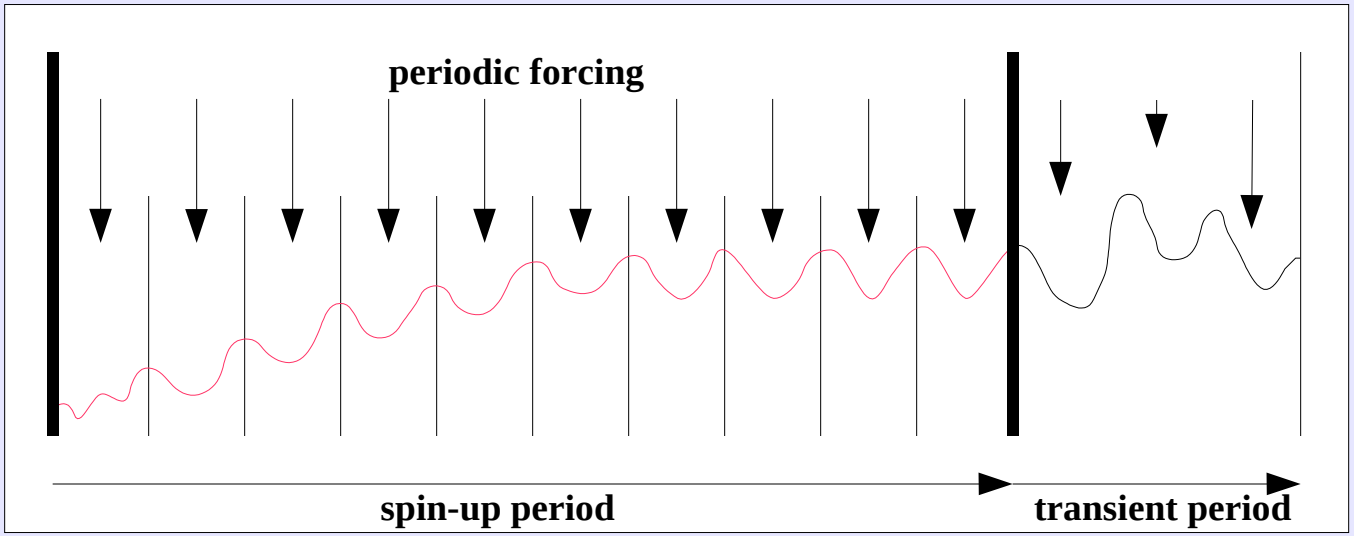
state
after iteration k

state before
iteration k

- Spinup terminated when fixed-point x_f is reached:

$$x_f = f(b, x_f)$$

Spinup Sensitivities



Interested in sensitivities e.g. of

- spun-up state w.r.t. forcing or model-parameters
- transient simulation w.r.t. parameters.
  Includes sensitivity of spun-up state.

Spinup Sensitivities
straight forward

Iterate both function and derivative simultaneously until convergence:

$$y = f(b, x)$$

$$dy/db = \partial f/\partial b + \partial f/\partial x * dx/db$$

Straight forward AD would generate code to run these iterations.

Command line for Fortran subroutine implementation:

```
      subroutine f(b,x,y)
      real b(p),x(n),y(n)
```

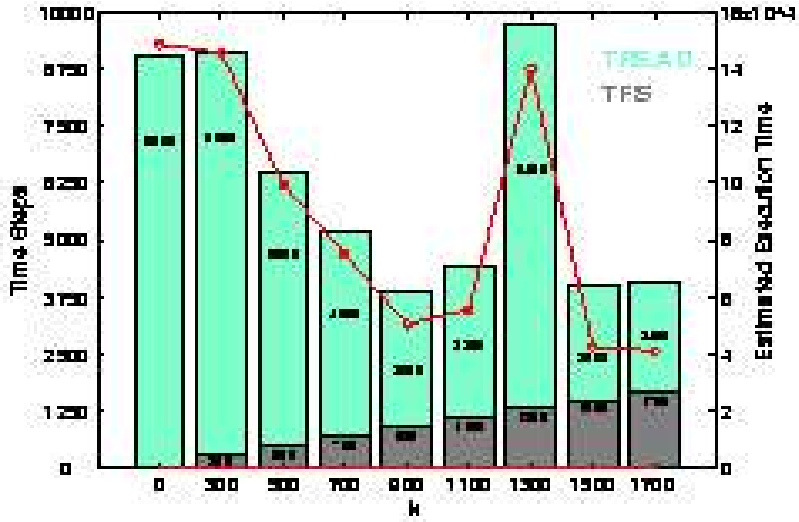
taf -input b,x -output y -toplevel f -forward

Spinup Sensitivities
Delayed Propagation

Advanced strategies for sensitivities:

- **Griewank's (2000) book:**
  Only turn on derivative iteration,
  once convergence of function iteration is „regular“
  (see also work of
  Bischof, Carle, Forth, Green, ...)

- **---------->**
  Applied by Buecker et al. (2003)
  to an airfoil simulation
  with the CFD code TFS
  and use ADIFOR to evaluate the
  derivative of the steady state
  w.r.t. 8 design parameters



Thanks to MB !

Figure 2. Performance of the approach delaying the derivative computations by k time steps.

Spinup Sensitivities
Reverse mode

Advanced strategies for sensitivities:

- **In reverse mode, straight forward AD would provide the required variables (including the state) of each iteration**
- **Christianson (1994, 1998) derives an efficient alternative version of the adjoint: Provided that iteration has converged, it is sufficient to use required variables from final iteration.**
  Saves disk/memory space or recomputations.
- **FastOpt's AD-tool TAF implements the Christianson-recipe.**
  Generation of alternative adjoint is triggered by TAF loop directive.

Spinup Sensitivities
Full Jacobian Approach

- The above approaches are based on the implicit function theorem applied to the equation for the final state:

$$x_f = f(b, x_f)$$

- Under certain regularity conditions on f and non-singular $\partial f/\partial x$ x_f can be expressed as a function of $x_f(b)$, and for dx_f/db:

$$0 = \partial f/\partial b(b, x_f(b)) + (\partial f/\partial x(b, x_f(b)) - Id)dx/db$$

The above strategies are 'matrix free' approaches to solve for dx/db

- Question: How efficient is full Jacobian approach,
  i.e. iterate function to convergence,
  then evaluate the Jacobians $\partial f/\partial b$ and $\partial f/\partial x$,
  and then solve for dx_f/db?

. t

Test Code: Boxmod

We investigate the spin-up of a test-code, Boxmod:

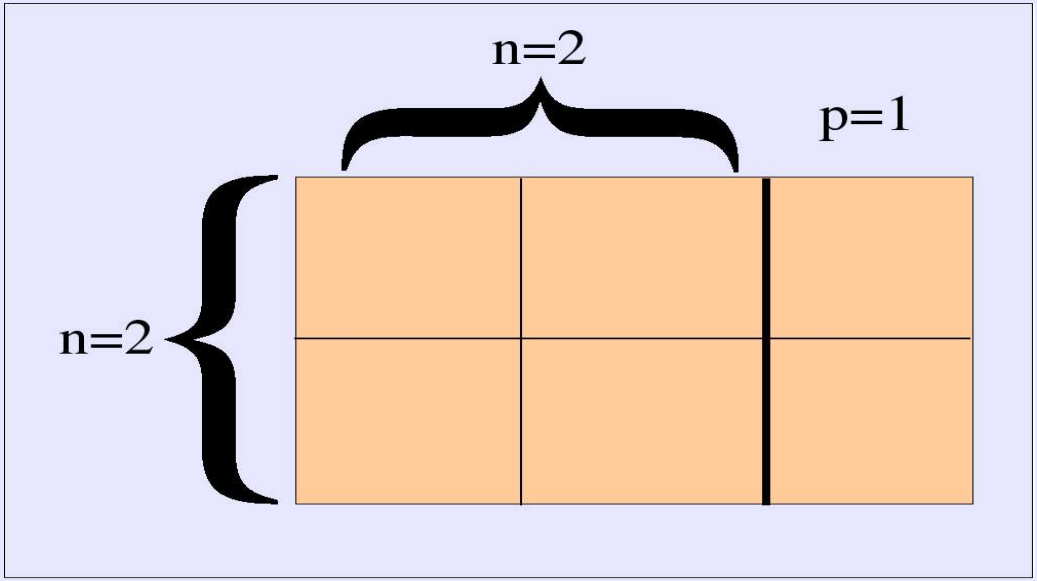
- Simple two-box model of the global atmospheric transport
- Setup for simulation of methylchloroform concentration (Rayner et al., 2000)
- State (x) : Tracer concentration in each box
- Parameter (b) : Interhemispheric mixing rate: 1/year
- Surface source field: Repeating 1978 estimate by Prinn et al (1992)
- Uniform sink term corresponding to an inverse lifetime of 1/4.7 a (Houweling et al., 1998)
- Boxmod used in educational studies of Tans (1997) and its adjoint in Rayner et al. (2000). Check those for details on the model.
- 100,000 time steps per year, to mimic a 'big model'

AD of Boxmod

- Generated derivative code by TAF in `pure` mode (no function eval)
- Evaluated derivative dx_f/db by
  - *standard* approach and
  - *full Jacobian* approach
- Convergence criterion:
  Rel. difference in both components of $x < 10^{-7}$: abs. difference $< 10^{-5}$
  k=49 iterations (years)
- Tested performance on Intel Pentium M 1400 Mhz,
  using Lahey-Fujitsu `lf95`, with option `-dbl` for double precision
- dx_f/db from both approaches has rel. difference $< 10^{-9}$
- *Full Jacobian* approach is 21 times faster
- Why is that?

AD of Boxmod
Performance

- Cost *standard* : 49 times Jacobian * (1 vector)
- Cost *full Jacobian* : 1 time Jacobian * (3 vectors)
  (provided that cost of solver can be neglected)



- k=49 reduces to k=6, if spin-up accuracy of 1 permil is sufficient

Full Jacobian
Efficiency

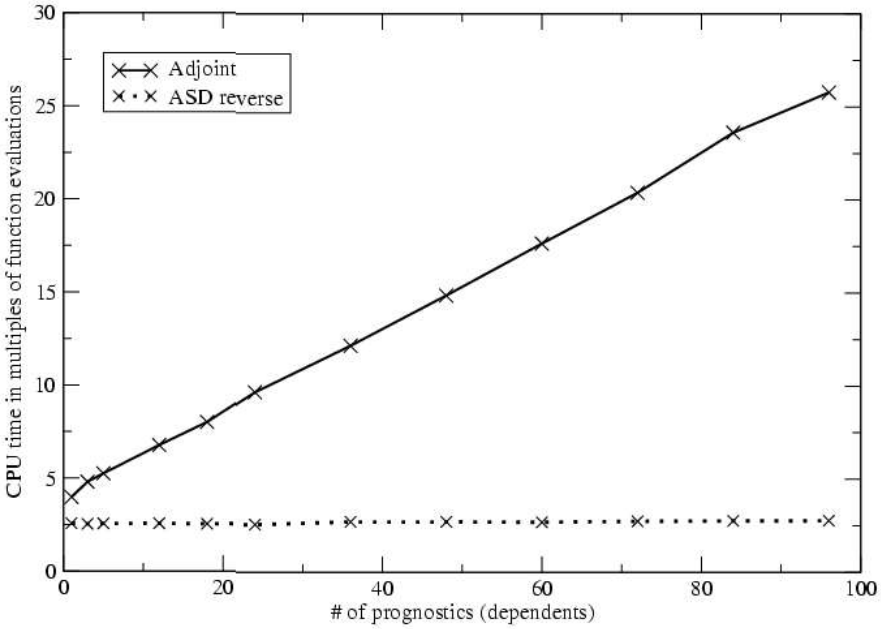
Performance r(m) of Jacobian times m vectors:

- **Flop count:**

$$r(m) = r(1) + s \cdot (m - 1)$$

First vector most expensive, includes cost of providing required variables;
! Can run out of memory.

- **CPU time (in mult. of function): Jacobian of biosphere model BETHY (Knorr, 1997) reverse: s about 0.25**
  ---------->
  forward: r(1) = 1.5; r(58) = 12 s about 0.2



Full Jacobian
Efficiency

CPU time *standard* | CPU time *full Jacobian*

$$k \cdot (r(1) + s \cdot (p-1)) > r(1) + s \cdot (p+n-1)$$

$$(k-1) \cdot (r(1) + p - 1)/s > n \quad .$$

# of iterations

CPU-time 1 vector

# of parameters

dimension of state

Assumption:

- Cost of inverting Jacobian neglected

Remarks:

- number of dependent variables q << p:
  Use “reverse mode r(q) on lhs”, and possibly even r(q+n) for rhs

Outlook

- Carbon Cycle Data Assimilation System (http://CCDAS.org) is built around terrestrial biosphere model BETHY
- First step of CCDAS consists of parameter estimation (calibration)
- In current setup cannot afford to spin-up of slow decomposing soil-carbon pool -> some 1000 years compared to current assimilation period of 26 years
- Current setup uses global 2x2 degree global grid, grid-boxes are *independent* of each other -> Jacobian of initial state to final state *diagonal block structure*
- Spin-up sensitivities (hopefully!) much cheaper than spin-up itself
- More details on CCDAS on poster by Widmann et al.

Summary

- "*Full Jacobian*" approach for spin-up sensitivities
- Investigated performance-gain for simple example owing to efficiency of (TAF-generated) Jacobian evaluation
- Derived heuristic performance-formula
- Approach may be favourable for a class of sensitivity evaluations

More Info:

- Posters on CCDAS, TAC++, and TAF applications
- http://CCDAS.org
- http://FastOpt.com