# „Pseudo Adjoints“

## Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Vitaliy Pasyuga

Sommerschule’06: “Automatisches Differenzieren“
14.8.2006, Universitätskolleg Bommerholz

# 1. Motivation:

- Finite Differenzen, Leapfrog Schemata
- Shallow Water Modell-Gleichungen
- Tranformation zu N-dimensionalen Zeit-abhängigen Systemen
- Kosten-Funktionen und Adjoints
- AD, Leapfrog-Schema und „Pseudo-Adjoints“

# 2. Theory:

- Parallele Berechnung von den dünnbesetzten „Timestep“-Jacobimatrizen
- Die Hoch-Niveau Adjungierte Rekursion:

  1. Leapfrog-Schema
  2. „One Step“-Schema
- Generalisierung des „Time-Stepping“ Operators
- Generalisierung der Kosten-Funktion

## Finite Differenzen, Leapfrog Schemata:

### Beispiel 1. ODEs:

Erste Ordnung: $x \in \mathbb{R}^1$; $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{y} = \mathbf{y}(x)$.

$$\frac{d\mathbf{y}}{dx} = \mathbf{f}(\mathbf{y}, x).$$

### Beispiel 2. ODEs:

Zweite Ordnung: $t \in \mathbb{R}^1$; $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{x} = \mathbf{x}(t)$.

$$\frac{d^2\mathbf{x}}{dt^2} = \mathbf{f}\left(\mathbf{x}, \frac{d\mathbf{x}}{dt}, t\right).$$

Alle ODEs $k$-ter Ordnung können in ein System der ODEs erster Ordnung umgewandelt werden, dabei steigt die Dimension des Systems:

### Beispiel 3. ODEs:

$$\frac{d^2\mathbf{x}}{dt^2} = f\left(x, \frac{dx}{dt}, t\right).$$

$$y_1 := x; y_2 := \frac{dx}{dt}.$$

$$\begin{cases} \frac{dy_1}{dt} = y_2 \\ \frac{dy_2}{dt} = f(y_1, y_2, t). \end{cases}$$

## Finite Differenzen, Leapfrog Schemata:

Vorherige Behauptung kann auf PDGs umgesetzt werden:
PDGs $k$-ter Ordnung werden in ein System der ODEs erster Ordnung umgewandelt, dabei steigt die Dimension des Systems.

Beispiel 4. Finite Differenze für die erste Ableitung:

$$u = u(x), h \ll 1.$$

Approximation der ersten Ordnung von $h$:

$$\frac{du}{dx}(x_0) \approx D_+u(x_0) := \frac{u(x_0+h)-u(x_0)}{h};$$
$$\frac{du}{dx}(x_0) \approx D_-u(x_0) := \frac{u(x_0)-u(x_0-h)}{h}.$$

Approximation der zweiten Ordnung (zentrierte Differenzen, werden für Leapfrog benutzt):

$$\frac{du}{dx}(x_0) \approx D_0u(x_0) := \frac{1}{2}(D_+u(x_0)+D_-u(x_0)) := \frac{u(x_0+h)-u(x_0-h)}{2h};$$

das Problem des ersten Schrittes.

Approximation der dritten Ordnung:

$$\frac{du}{dx}(x_0) \approx D_3u(x_0) := \frac{1}{6h}[2u(x_0+h)+3u(x_0)-6u(x_0-h)+u(x_0-2h)];$$

das Problem ersten zwei Schritten.

Dies lasst sich auf Ableitunge höherer Ordnung umsetzen (Superposition).

## Finite Differenzen, Leapfrog Schemata:

ODEs:

$$\frac{du}{dt} = f(u).$$

Finite Differenze auf gleichmäßiger Gitter mit dem Schritt $\Delta t$.

Startbedingungen (Anfangswerte):

AB1 - ein Schritt für den Start, die erste Ordnung der Approximation (nicht gut, es gibt Modifikationen):

$$u_0 = u(0),$$
$$u_{n+1} = u_n + \Delta t f(u_n).$$

AB2 - zwei Schritte für den Start:

$$u_{-1} = u(-\Delta t),$$
$$u_0 = u(0),$$
$$u_{n+1} = u_n + \Delta t \left(\frac{3}{2} f(u_n) - \frac{1}{2} f(u_{-1})\right).$$

Usw.

## Finite Differenzen, Leapfrog Schemata:

### Motivieren von AB1:

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(u(t))dt \simeq u(t_n) + \int_{t_n}^{t_{n+1}} I_p f(u(t))dt \Rightarrow u_{n+1} := u_n + \Delta t f(u_n).$$

### Motivieren von Leapfrog:

$$u(t_{n+1}) = u(t_{n-1}) + \int_{t_{n-1}}^{t_{n+1}} f(u(t))dt \simeq u(t_{n-1}) + 2\Delta t f(u(t_n)) \Rightarrow u_{n+1} := u_{n-1} + 2\Delta t f(u_n).$$

$$Z(0) := u(t_0), Z(1) := Z(0) + \Delta t f(Z(0)).$$

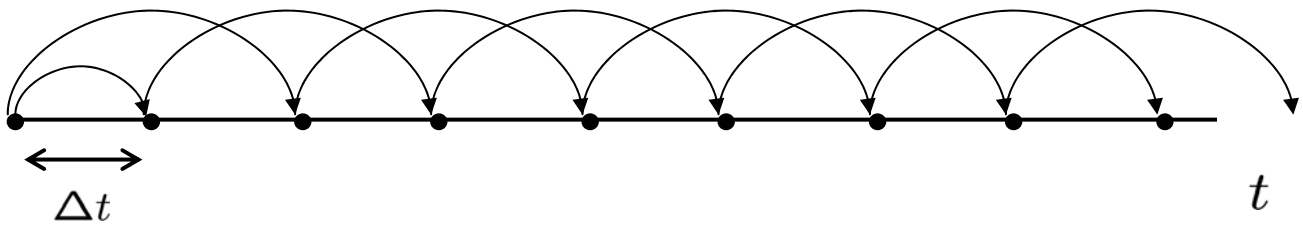
$$Z(k) := u(t_k);$$

$$H(Z(k), Z(k-1), \Delta t) := Z(k-1) + 2\Delta t f(Z(k)) :$$

$$Z(k+1) = H(Z(k), Z(k-1), \Delta t).$$

**Bemerkung:**

1. Dies kann direkt auf Syteme von ODEs, $u = (u_1, \ldots, u_N)$, umgesetzt werden.
2. Weitere Umsetzung folgt auf System von PDEs.
3. Leapfrog ist relativ gut für hyperbolische Gleichungen, instabil für parabolische Gleichungen.

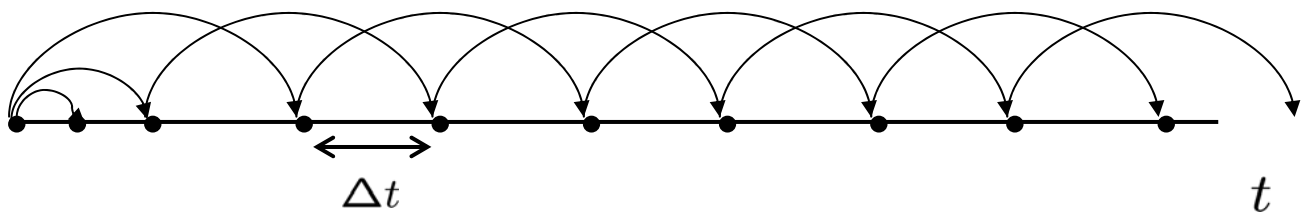



**Bemerkung:**

1. Ein Problem des Leapfrog-Schemas (und anderen Drei-Zeitniveau Schemas) ist, dass zwei Anfangswerte von $u$ werden gebraucht.
   Zusatzlich zum physischen Anfagswert $u^0$ braucht man auch Berechnungsanfangswert $u^1$.

2. Der Anfangswert $u^1$ kann aus des Leapfrog nicht bekommen werden, deshalb wird normalerweise ein einfaches nichtzentriertes Schritt benutzt:
   $u^1 = u^0 + \Delta t f(u^0)$.
   Aber dabei wird der Fehler der ersten Ordnung.

3. Eine Alternative ist den ersten Halbschritt zu benutzen, um den Fehler zu reduzieren:

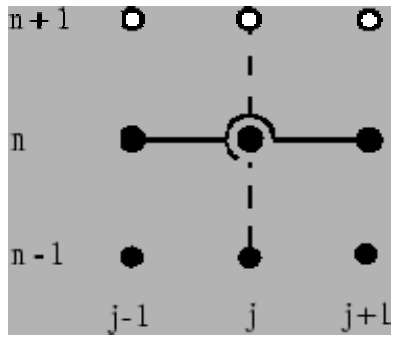


## Leapfrog Schemata von finiten Differenzen:

### Beispiel 5. Leapfrog Schema für die hyperbolische Gleichung.

$$\frac{\partial u}{\partial t} = -\frac{\partial J(u)}{\partial x}.$$

$$u_j^{n+1} = u_j^{n-1} - \frac{\Delta t}{\Delta x}[J_{j+1}^n - J_{j-1}^n].$$




$$Z(k) := \begin{pmatrix} u_1^k \\ u_2^k \\ \vdots \\ u_N^k \end{pmatrix}, W := [\Delta t, \Delta x],$$

$$Z_i(k+1) =$$

$$Z_i(k-1) + \frac{\Delta t}{\Delta x}\left[J(Z_{i+1}(k)) - J(Z_{i-1}(k))\right] =: H_i(Z(k), Z(k-1), \Delta t, \Delta x),$$

$$0 < i < N, N \gg 1.$$

Bemerkung:

1. Die dünnbesetzte Jacobimatrix: $\left[\frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right]$, hat maximal 5 Elemente in jeder Zeile.
2. Weitere Verschärfung ist das staggered Leapfrog (staggered Gitter) für vektorielles $u$. Dies kann auch in diese Form gebracht werden.

Shallow Water Modell-Gleichungen in den Karthesischen Koordinaten:
(inkompresible Flüssigkeit; die Tiefe ist, relativ dem horisontalen Durchmesser, gering)

$$\begin{cases} \frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} + fv - \frac{\partial \phi}{\partial x}, \\ \frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - fv - \frac{\partial \phi}{\partial y}, \\ \frac{\partial \phi}{\partial t} = -\frac{\partial (u\phi)}{\partial x} - \frac{\partial (v\phi)}{\partial y}. \end{cases}$$

- $u, v$ - Komponente der horisontalen Geschwindigkeit.
- $\phi = gh$ - das Geopotentialfeld.
- $f$ - ein Zeit-unabhängiger Modellparameter (der Coriolisfaktor).
  $f = \hat{f} + \beta\left(y - \frac{D}{2}\right)$, $0 \le x \le L$, $0 \le y \le D$.
- $U := (u, v, \phi)^T$ - der Zustandsvektor.

Benutze Leapfrog-Schema für die Zeit-Schritte, für räumliche Ableitunge werden zentrierte Differenzen benutzt:

Transformation zum Zeit-abhängigen N-dimensionalen System:

$$\begin{cases} \{u_{i,j}^{k+1} = u_{i,j}^{k-1} - \frac{\Delta t}{\Delta x} u_{i,j}^k (u_{i+1,j}^k - u_{i-1,j}^k) - \frac{\Delta t}{\Delta y} v_{i,j}^k (u_{i,j+1}^k - u_{i,j-1}^k) + 2\Delta t f_{i,j} v_{i,j}^k - \frac{\Delta t}{\Delta x}(\phi_{i+1,j}^k - \phi_{i-1,j}^k)\,, \\ \{v_{i,j}^{k+1} = v_{i,j}^{k-1} - \frac{\Delta t}{\Delta x} u_{i,j}^k (v_{i+1,j}^k - v_{i-1,j}^k) - \frac{\Delta t}{\Delta y} v_{i,j}^k (v_{i,j+1}^k - v_{i,j-1}^k) - 2\Delta t f_{i,j} v_{i,j}^k - \frac{\Delta t}{\Delta y}(\phi_{i,j+1}^k - \phi_{i,j-1}^k)\,, \\ \{\phi_{i,j}^{k+1} = \phi_{i,j}^{k-1} - \frac{\Delta t}{\Delta x}(u_{i+1,j}^k \phi_{i+1,j}^k - u_{i-1,j}^k \phi_{i-1,j}^k) - \frac{\Delta t}{\Delta y}(v_{i,j+1}^k \phi_{i,j+1}^k - v_{i,j-1}^k \phi_{i,j-1}^k)\,. \end{cases}$$

$i, j = 1, \ldots, n;\ k = 1, \ldots, T-1 \qquad f_{i,j} \approx \text{const} = f \qquad W := (f, \Delta x, \Delta y, \Delta t)$

- Bilde aus $\{u_{i,j}^k\}$, $\{v_{i,j}^k\}$, $\{\phi_{i,j}^k\}$ einen langen Zustandsvektor $Z(k)$ mit $\#Z(k) = 3 \times n \times n$.

  $N := 3 \times n \times n$, $p := \#W$, typisch: $N \gg p$.

- Gleichungen ergeben uns einen nichtlinearen "Time-Stepping"-Operator:

  $Z(k+1) = H(Z(k), Z(k-1), W)$ für $k = 1, \ldots, T-1$.

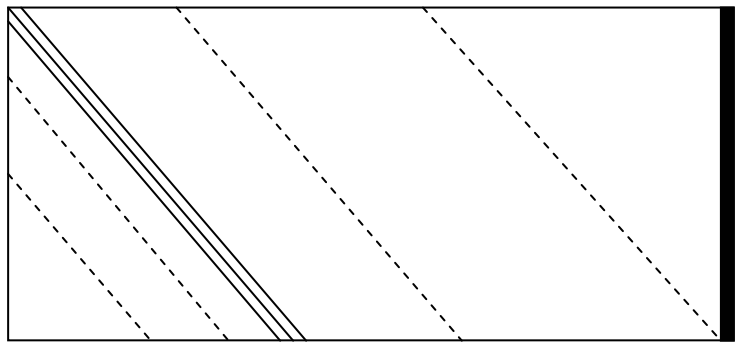
- Die l-te Komponente $H_l(Z(k), Z(k-1), W)$ hängt maximal von:

  1 Element von $Z(k-1)$, 8 Elementen von $Z(k)$ und 4 Elementen von $W$;
  insgesamt 13 Variablen ab.

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

1. Die dünnbestzte "Timestep"-Jakobimatrix: $\left[\frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right]$, maximal 13 Elemente in jeder Zeile.

2. $\left[\frac{\partial H}{\partial W}\right]$ ist i.a. Fall dicht ($p$ Elemente in jeder Zeile). Aber für $N \gg p$ wird der Einflüss gering.



Schematische Darstellung des "Leapfrog"-Schema:

Initialize $Z(0)$ and $W$.
Compute $Z(1)$ ("One-Step"-Schema).
**for** $k = 1$ **to** $T - 1$ **do**
    $Z(k+1) = H(Z(k), Z(k-1), W)$
**end do**

## Kosten-Funktion und Adjoints:

### Steurungsproblem in der Metheorologie und Okeanologie:

- Zwecks Variational Data Assimilation (VDA), $[0, T]$ - Assimilationsfenster;
- Sensitivitätsanalyse: wie wird die Veränderung der Anfangswerten (bzw. Modellparametern) eine Modellfunktion (Kosten-Funktion) $r : U(T) \mapsto r(U(T))$ bzw. $Z(T) \mapsto r(Z(T)) \in \mathbb{R}^1$ beeinflüssen.

### Typische Kosten-Funktionen:

$$r(U(T)) = J(U(0), W) = \frac{1}{2} \int_{\Omega} w(x) \left[U(T) - U(T)_{\text{obs}}\right]^2 dx;$$

$$r(Z(T)) = J(Z(0), W) = \frac{1}{2} \langle \widehat{w}(CZ(T) - Z(T)_{\text{obs}}), CZ(T) - Z(T)_{\text{obs}} \rangle;$$

$$Z(T) \in \mathbb{R}^N, Z(T)_{\text{obs}} \in \mathbb{R}^M, C : \mathbb{R}^N \mapsto \mathbb{R}^M;$$

$$J(Z(0), W) = \frac{1}{2} \int_0^T \langle \widehat{w}(CZ(t) - Z(t)_{\text{obs}}), CZ(t) - Z(t)_{\text{obs}} \rangle dt;$$

$$J(Z(0), W) = \frac{1}{2} \sum_{k=0}^{T} \langle \widehat{w}(k)(CZ(k) - Z(k)_{\text{obs}}), CZ(k) - Z(k)_{\text{obs}} \rangle dt;$$

$$J(Z(0), W) = \frac{1}{2} \int_0^T \left\| \widehat{H}X - X_{\text{obs}} \right\|^2 dt, \widehat{H} \text{ - observation matrix.}$$

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Ziel: minimiere $J(U(0), W)$ bzw. $J(Z(0), W)$

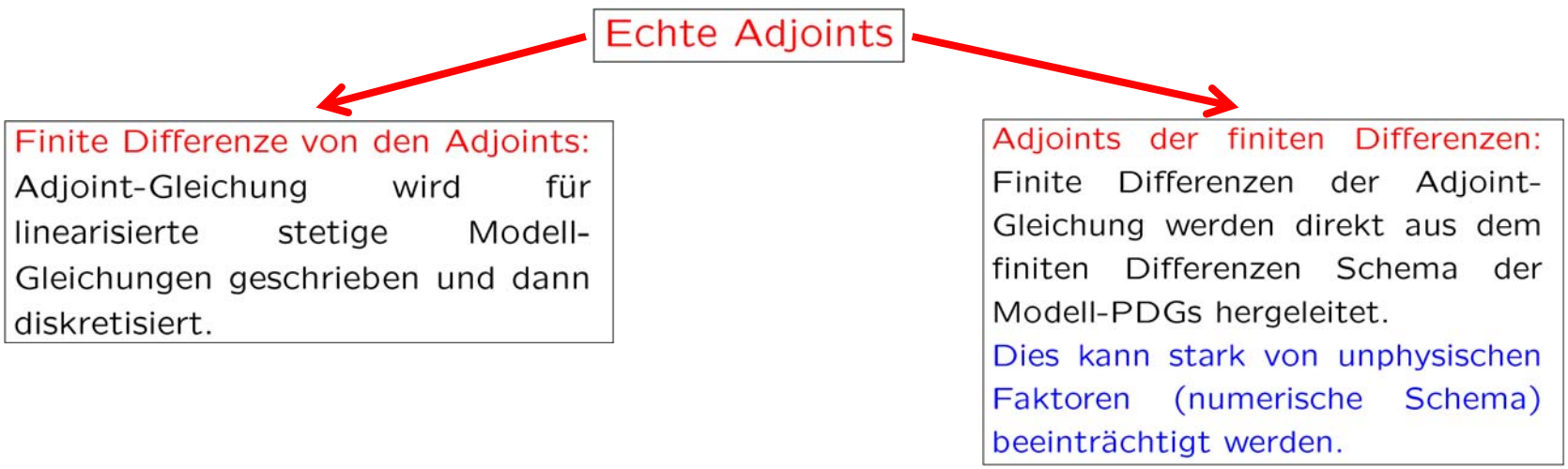
$$\Downarrow$$

$$\frac{dr}{d[Z(0), W]} = \frac{dJ(Z(0), W)}{d[Z(0), W]}$$

Sensitivität, Adjoint, "Long Gradient".

Adjoint-Ansätze werden angewendet (echte Adjoints):

- Stetige Adjoints:
  werden mathematisch (analytisch) aus der Definition des entsprechenden Operators bekommen.
- Diskrete Adjoints:
  werden durch den Ansatz des reversen Modes von AD auf den Computer-Code bekommen.



Beispiel 8. Adjoint Newton Algorithm (for distributed parameter systems):

$$\frac{dZ(t)}{dt} = F(Z(t)),$$

$$Z(0) = U.$$

$$Z \longrightarrow Z + \hat{Z}, U \longrightarrow U + U'.$$

$$\Downarrow$$

$$\frac{d\hat{Z}(t)}{dt} \approx \frac{\partial F(Z)}{\partial Z}\hat{Z}(t),$$

$$\hat{Z}(0) = U'.$$

$$\min_U J(U) = \min_U \left\{ \frac{1}{2} \int_0^T \langle \hat{w}(CZ - Z_{\text{obs}}), CZ - Z_{\text{obs}} \rangle dt \right\}.$$

$$\Downarrow$$

$$\delta J = [\nabla_U J, U'] = \int_0^T \langle \hat{w}(CZ - Z_{\text{obs}}), C\hat{Z} \rangle dt.$$

Um die lineare Abhängigkeit von $U'$ zu benutzen und $\nabla_U J$ zu berechnen wird die adjungierte Variable $P$ eingeführt.

$$\Downarrow$$

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

$$\int_0^T \left\langle -P, \frac{d\hat{Z}}{dt} \right\rangle dt = \int_0^T \left\langle -P, \frac{\partial F}{\partial Z}\hat{Z} \right\rangle dt,$$

$$\Downarrow$$

$$\langle -P(T), \hat{Z}(T)\rangle + \langle P(0), \hat{Z}(0)\rangle = \int_0^T \left\langle \hat{Z}, -\frac{dP}{dt} - \left(\frac{\partial F}{\partial Z}\right)^* P \right\rangle dt.$$

Gleichungssystem ist unvollständig. Füre zusätzliche Einschränkung auf $P$ ein (adjoint Gleichung):

$$-\frac{dP}{dt} = \left(\frac{\partial F}{\partial Z}\right)^* P + C^* \hat{w}(CZ - Z_{\text{obs}}),$$

$$P(T) = 0.$$

$$\Downarrow$$

$$\delta J(U, U') = \langle \nabla_U J, U' \rangle = \langle U', P(0) \rangle,$$

$$\nabla_U J = P(0).$$

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Schematische Darstellung des "Leapfrog"-Schema:

Initialize $Z(0)$ and $W$.
Compute $Z(1)$ ("One-Step"-Schema).
**for** $k = 1$ **to** $T - 1$ **do**
    $Z(k+1) = H(Z(k), Z(k-1), W)$
**end do**

Ziel: $\frac{dr(Z(T))}{d[Z(0),W]}$.

Der neue Ansatz: Pseudo-Adjoints.

Ingredienzien:

- Nutzung der Dünnbesetzung (für "stencil-based" Berechnungen) der Jacobimatrizen: $\left[\frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right]$.
- Gleichzeitige (parallele) Berechnung der Ableitungen von $H$ für verschiedene $k$.
- "Reverse-Mode Harness", die effektiv akkumuliert Ableitunge von verschiedenen Zeit-Schritten.

Parallele Berechnung der dünnbesetzten "Timestep"-Jacobimatrizen:

$$\frac{dZ(k+1)}{d[Z(0),W]} = \frac{\partial H(Z(k),Z(k-1),W)}{\partial Z(k)} \cdot \frac{dZ(k)}{d[Z(0),W]} + \frac{\partial H(Z(k),Z(k-1),W)}{\partial Z(k-1)} \cdot \frac{dZ(k-1)}{d[Z(0),W]} + \frac{\partial H(Z(k),Z(k-1),W)}{\partial W} \cdot \frac{dW}{d[Z(0),W]}, 1 \leq k \leq T-1.$$

a. Für die 2-D shallow water Modell, $n = 11$, $N = 3 \times n \times n = 363$, sind $\frac{\partial H}{\partial Z(k)}$ und $\frac{\partial H}{\partial Z(k-1)}$ dünn besetzte $N \times N$ Matrize.

b. Die $N \times p$ Matrix $\frac{\partial H}{\partial W}$ ist dicht besetzt.

c. Jede Zeile der Matrix $\left[\frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right]$ besitzt maximal 13 (nicht Null) Elemente.

d. $\frac{dW}{d[Z(0),W]} = [0_{p\times N}, I_{p\times p}]$.

Idee:

- Benutze die Dünnbesetzung von $\frac{\partial H}{\partial Z(k)}$ und $\frac{\partial H}{\partial Z(k-1)}$ zwecks Effektivität bei der Berechnung von $\frac{dZ(T)}{d[Z(0),W]}$ mit der AD-Tool.

- Benutze Zeit-parallele Berechnung der dünnnbesetzten "timestep"-Jacobimatrizen mit dem forward-mode basierten AD-Tool, weil um $\left[\frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right]$ zu berechnen werden Ableitunge $\frac{dZ(i)}{d[Z(0),W]}$ nicht gebraucht.

- Benutze rückwärts Rekursion (Multiplizieren der Vektoren auf dünnbesetzte Matrize), um $\frac{dr}{d[Z(0),W]} = \frac{dr}{dZ(T)} \cdot \frac{dZ(T)}{d[Z(0),W]}$ zu berechnen.

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Neues Schema, "Pseudo Adjoints":

Schematic for computing timestep Jacobians using $P$ processors:

Initialize $Z(0)$ and $W$.
Compute and save $Z(1)$.

**for** $k = 1$ **to** $T - 1$ **do**

  $Z(k+1) = H(Z(k), Z(k-1), W)$

  Save $Z(k+1)$.

**end do**

**for** $k = 0$ **to** $T - 1$ **do**

  **if** $(\mathrm{mod}(k, P) = \mathrm{my\_id})$ **then**

    **if** $(k = 1)$ **then**

      Compute and store $\left[\frac{dZ(1)}{dZ(0)}, \frac{dZ(1)}{dW}\right]$.

    **else**

      Compute $Z(k+1)$ as well as $\left[\frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right]$ via

      an invocation of $\widehat{H}$ geared toward exploiting sparsity.

      Store $\left(k, \frac{\partial H}{\partial Z(k)}, \frac{\partial H}{\partial Z(k-1)}, \frac{\partial H}{\partial W}\right)$.

    **end if**

  **end if**

**end do**

Die Komplexität ist proportional der Komplexität für $H$, die ist von $N$ unabhängig.

## Die Hoch-Niveau Adjoint Rekursion.

Nun wurden alle timestep Jakobians berechnet, dann werden Adjoints rekursiv berechnet.

### Leapfrog Schema.

$$\frac{dr}{d[Z(0),W]} = \frac{dr}{dZ(T)} \cdot \frac{dZ(T)}{d[Z(0),W]}.$$

$$\begin{aligned}\frac{dr}{dZ(T)} \cdot \frac{dZ(T)}{d[Z(0),W]} = (x^T)_{1\times N} \cdot \frac{dZ(T-1)}{d[Z(0),W]} \\ +(y^T)_{1\times N} \cdot \frac{dZ(T-2)}{d[Z(0),W]} \\ +(w^T)_{1\times(N+p)}, T \geq 2;\end{aligned}$$

$$(x^T)_{1\times N} = \frac{dr}{dZ(T)_{1\times N}} \cdot \frac{\partial H(Z(T-1),Z(T-2),W)}{\partial Z(T-1)_{N\times N}},$$

$$(y^T)_{1\times N} = \frac{dr}{dZ(T)_{1\times N}} \cdot \frac{\partial H(Z(T-1),Z(T-2),W)}{\partial Z(T-2)_{N\times N}},$$

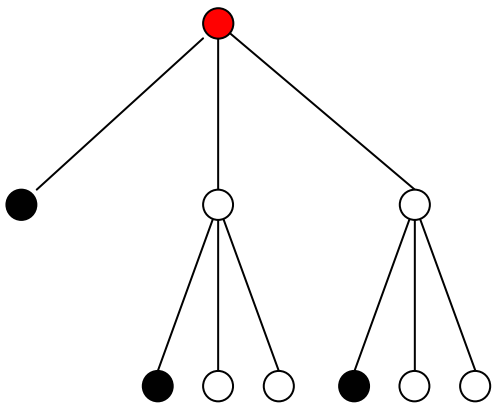
$$(w^T)_{1\times(N+p)} = \frac{dr}{dZ(T)_{1\times N}} \cdot \frac{\partial H(Z(T-1),Z(T-2),W)}{\partial W_{N\times p}} \cdot [0_{p\times N}, I_{p\times p}].$$

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

$$T = 1: \ \frac{dr}{dZ(1)} \cdot \frac{dZ(1)}{d[Z(0),W]} = \frac{dr}{dZ(1)} \cdot \left[\frac{dZ(1)}{dZ(0)}, \frac{dZ(1)}{dW}\right],$$

$$\frac{dW}{d[Z(0),W]} = [0_{p\times N}, I_{p\times p}].$$

$$T = 0: \ \frac{dr}{dZ(0)} \cdot \frac{dZ(0)}{d[Z(0),W]} = \frac{dr}{dZ(0)} \cdot \left[I_{N\times N}, 0_{N\times p}\right].$$



$$\frac{dr}{d[Z(0),W]} = \text{leapfrog\_adjoint}\left(\frac{dr}{dZ(T)}, T\right).$$

Kombinatorischer Fehler!
Der binäre Baum ist für sparse Matrize auch schief.
Es gibt mehrere Multiplikationen verschiedener Vektoren auf gleiche Matrize.
"One Step" Schema besser sammelt diese Vektore und reduziert die Anzahl der Multiplikationen.

$T - 1$ Rekursionenniveau.

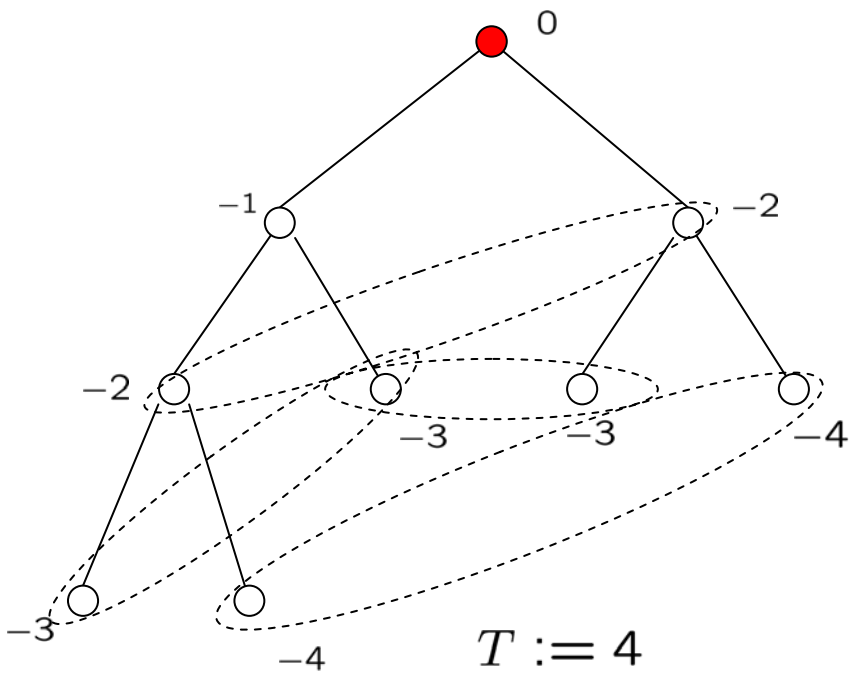
$\sim 2^T - 2$ linken Multiplikationen der $1 \times N$-Vektoren auf dünnbesetzte $N \times N$-Matrize.

$\sim 2^{T-1} - 1$ linken Multiplikationen der $1 \times N$-Vektoren auf dichte $N \times p$ Matrize,
Submatrize der $N \times (N + p)$ Matrizen.

$N \gg p \Rightarrow$ Kosten von den Multiplikationen (und Berechnungen) auf $\frac{\partial H}{\partial W}$ sind relativ gering.

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

$$\frac{dr}{d[Z(0), W]} = \text{leapfrog\_adjoint}\left(\frac{dr}{dZ(T)}, T\right).$$




$T := 4$

| $T$ | $u_T$ |
|---|---|
| 2 | 2 |
| 3 | 4 |
| 4 | 8 |
| 5 | 14 |
| 6 | 24 |
| 7 | 40 |
| ... | |

$$u_T = 2 + u_{T-1} + u_{T-2}$$

$T - 1$ Rekursionenniveau.

$T \gg 1 \Rightarrow \sim 2^{T-1}$ linken Multiplikationen der $1 \times N$-Vektoren auf sparse $N \times N$-Matrize.

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Recursive evaluation of adjoint for a leapfrog scheme:

$u =$ **function** leapfrog_adjoint $(v,k)$

**if** $(k \geq 2)$ **then**

$x^T = v \cdot \frac{\partial H}{\partial Z(k-1)}$

$y^T = v \cdot \frac{\partial H}{\partial Z(k-2)}$

$w^T = v \cdot \frac{\partial H}{\partial W} \cdot [0_{p\times N}, I_{p\times p}]$

$u =$ leapfrog_adjoint $(x^T, k-1)$ + leapfrog_adjoint $(y^T, k-2) + w^T$

**else**

$u = v \cdot \left[\frac{dZ(1)}{dZ(0)}, \frac{dZ(1)}{dW}\right]$ **if** $(k = 1)$

$u = v \cdot [I_{N\times N}, 0_{N\times p}]$ **if** $(k = 0)$

**end if**

**return** $(u)$

**end function**

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

"One-Step" Schema:

(weniger Matrix-Vektor Multiplikationen bzw. Zweigen, aber höhere Dimension)

$\overline{Z(k)} = [Z(k)^T, Z(k-1)^T]^T \in \mathbb{R}^{2N}$. Der erweiterte Zustandsvektor

$$\overline{Z(k+1)} = \bar{H}(\overline{Z(k)}, W), 1 \leq k \leq T-1,$$

$$\bar{H}(\overline{Z(k)}, W) = \begin{bmatrix} H(Z(k), Z(k-1), W) \\ Z(k) \end{bmatrix}.$$ Der "neue" update Operator

$$\frac{d\overline{Z(k+1)}}{d[Z(0), W]} = \frac{\partial \bar{H}}{\partial \overline{Z(k)}} \cdot \frac{d\overline{Z(k)}}{d[Z(0), W]} + \frac{\partial \bar{H}}{\partial W} \cdot \frac{dW}{d[Z(0), W]}, \; k \geq 1.$$ Rekursion

$\bar{r} : \mathbb{R}^{2N} \to R^N, \; \bar{r} : \overline{Z(T)} \mapsto r(Z(T))$. Kosten-Funktion des erweiterten Zustandes

$$\frac{d\bar{r}}{d[Z(0), W]} = \frac{d\bar{r}}{d\overline{Z(T)}} \cdot \frac{d\overline{Z(T)}}{d[Z(0), W]}.$$

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

$$\left[\frac{dr}{dZ(T)}, 0_{1\times N}\right]_{1\times 2N} \cdot \frac{d\overline{Z(T)}}{d[Z(0),W]} = (\bar{x}^T)_{1\times 2N} \cdot \frac{d\overline{Z(T-1)}}{d[Z(0),W]} + (\bar{w}^T)_{1\times(N+p)}, \quad T \geq 2,$$

$$(\bar{x}^T)_{1\times 2N} = \left[\frac{dr}{dZ(T)}, 0_{1\times N}\right]_{1\times 2N} \cdot \begin{bmatrix} \frac{\partial H}{\partial Z(T-1)} & \frac{\partial H}{\partial Z(T-2)} \\ I_{N\times N} & 0_{N\times N} \end{bmatrix}_{2N\times 2N},$$

$$(\bar{w}^T)_{1\times(N+p)} = \left[\frac{dr}{dZ(T)}, 0_{1\times N}\right]_{1\times 2N} \cdot \begin{bmatrix} \frac{\partial H}{\partial W} \\ 0_{N\times p} \end{bmatrix}_{2N\times p} \cdot [0_{p\times N}, I_{p\times p}].$$

$$\frac{d\overline{Z(1)}}{d[Z(0),W]} = \begin{bmatrix} \frac{dZ(1)}{dZ(0)} & \frac{dZ(1)}{dW} \\ \frac{dZ(0)}{dZ(0)} & \frac{dZ(0)}{dW} \end{bmatrix} = \begin{bmatrix} \frac{dZ(1)}{dZ(0)} & \frac{dZ(1)}{dW} \\ I_{N\times N} & 0_{N\times p} \end{bmatrix}.$$

Der "neue" sparse timestep Jakobian wird wie der "Alte", nach demselben Algorithmus berechnet. Die Komplexität ist zu der Komplexität für $H$ proportional und von $N$ unabhängig.

Dank der linearen Rekursion (und dem Vektorsammeln) wird es nur $2(T-1)$ multiplikationen mit sparsen $N \times N$-Matrizen durchgeführt.

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

$$\frac{dr}{d[Z(0),W]} = \text{one\_step\_adjoint}\left(\left[\frac{dr}{dZ(T)}, 0_{1\times N}\right], T\right).$$

Recursive evaluation of adjoint for a one-step scheme:

$\bar{u} =$ **function** one_step_adjoint $(\bar{v}, k)$

**if** $(k \geq 2)$ **then**

$$\bar{x}^T = \bar{v} \cdot \begin{bmatrix} \frac{\partial H}{\partial Z(k-1)} & \frac{\partial H}{\partial Z(k-2)} \\ I_{N\times N} & 0_{N\times N} \end{bmatrix}$$

$$\bar{w}^T = \bar{v} \cdot \begin{bmatrix} \frac{\partial H}{\partial W} \\ 0_{N\times p} \end{bmatrix} \cdot [0_{p\times N}, I_{p\times p}]$$

$$\bar{u} = \text{one\_step\_adjoint}\,(\bar{x}^T, k-1) + \bar{w}^T$$

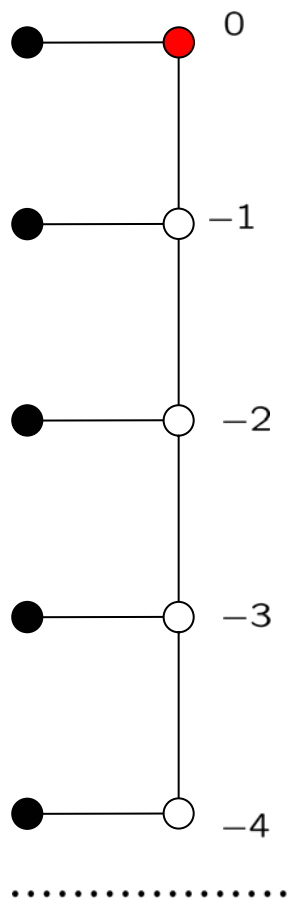
**else**

$$\bar{u} = \bar{v} \cdot \begin{bmatrix} \frac{dZ(1)}{dZ(0)} & \frac{dZ(1)}{dW} \\ I_{N\times N} & 0_{N\times p} \end{bmatrix}$$

**end if**

**return** $(\bar{u})$

**end function**



Generalisierung des "Time-Stepping" Operators:

$$Z(k+1) = H(Z(k), Z(k-1), \ldots, Z(k-l+1), W), \;\; k \geq l-1.$$

Alte Kosten-Funktion:

$$\frac{dr}{d[Z(0), W]} = \frac{dr}{dZ(T)} \cdot \frac{dZ(T)}{d[Z(0), W]}.$$

Rekursion:

$$\frac{dZ(T)}{d[Z(0), W]} = \sum_{\tau=T-l}^{T-1} \frac{\partial H(Z(T-1), \ldots, Z(T-l), W)}{\partial Z(\tau)} \cdot \frac{dZ(\tau)}{d[Z(0), W]} + \frac{\partial H(Z(T-1), \ldots, Z(T-l), W)}{\partial W} \cdot \frac{dW}{d[Z(0), W]}, \;\; T \geq l.$$

$$\frac{dr}{dZ(T)_{1\times N}} \cdot \frac{dZ(T)}{d[Z(0), W]_{N\times(N+p)}} = \sum_{\tau=T-l}^{T-1} (x_\tau^T)_{1\times N} \cdot \frac{dZ(\tau)}{d[Z(0), W]} + (w^T)_{1\times(N+p)}, \;\; T \geq l,$$

$$(x_\tau^T)_{1\times N} = \frac{dr}{dZ(T)_{1\times N}} \cdot \frac{\partial H(Z(T-1), \ldots, Z(T-l), W)}{\partial Z(\tau)_{N\times N}},$$

$$(w^T)_{1\times(N+p)} = \frac{dr}{dZ(T)_{1\times N}} \cdot \frac{\partial H(Z(T-1), \ldots, Z(T-l), W)}{\partial W_{N\times p}} \cdot [0_{p\times N}, I_{p\times p}].$$

## $l$ Anfangsbedingungen:

$$\frac{dr}{dZ(\tau)} \cdot \frac{dZ(\tau)}{d[Z(0), W]} = \frac{dr}{dZ(\tau)} \cdot \left[\frac{dZ(\tau)}{dZ(0)}, \frac{dZ(\tau)}{dW}\right], \ \tau = 1, 2, \ldots, l-1,$$

$$\frac{dr}{dZ(0)} \cdot \frac{dZ(0)}{d[Z(0), W]} = \frac{dr}{dZ(0)} \cdot \left[I_{N\times N}, 0_{N\times p}\right].$$

$l^{T-l+1}$.

## Transformation in die "One Step" Schema (lineare Rekursion):

$$\widehat{Z(k)} = [Z(k)^T, Z(k-1)^T, \ldots, Z(k-l+1)^T] \in \mathbb{R}^{lN}.$$

$$\widehat{Z(k+1)} = \widehat{H}(\widehat{Z(k)}, W), \ k \geq l-1.$$

$$\widehat{H}(\widehat{Z(k)}, W) = \begin{bmatrix} H(Z(k), Z(k-1), \ldots, Z(k-l+1), W) \\ Z(k) \\ Z(k-1) \\ \vdots \\ Z(k-l+2) \end{bmatrix}.$$

$$\widehat{r} : \mathbb{R}^{lN} \to \mathbb{R}; \ \widehat{r} : \widehat{Z(T)} \mapsto r(Z(T)).$$

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Rekursion:

$$\left[\frac{dr}{dZ(T)}, 0_{1\times N}, \ldots, 0_{1\times N}\right]_{1\times lN} \cdot \frac{d\overbrace{Z(T)}}{d[Z(0), W]_{lN\times(N+p)}} = (x^T)_{1\times lN} \cdot \frac{d\overbrace{Z(T-1)}}{d[Z(0), W]_{lN\times N+p}} + (w^T)_{1\times(N+p)}, \quad T \geq l, \qquad l(T-l+1).$$

$$(x^T)_{1\times lN} = \left[\frac{dr}{dZ(T)_{1\times N}}, 0_{1\times N}, \ldots, 0_{1\times N}\right]_{1\times lN} \cdot \begin{bmatrix} \frac{\partial H}{\partial Z(T-1)} & \frac{\partial H}{\partial Z(T-2)} & \cdots & \frac{\partial H}{\partial Z(T-l)} \\ I_{(l-1)N\times(l-1)N} & & & 0_{(l-1)N\times N} \end{bmatrix},$$

$$(w^T)_{1\times(N+p)} = \left[\frac{dr}{dZ(T)_{1\times N}}, 0_{1\times N}, \ldots, 0_{1\times N}\right]_{1\times lN} \cdot \begin{bmatrix} \frac{\partial H}{\partial W_{N\times p}} \\ 0_{N\times p} \\ \vdots \\ 0_{N\times p} \end{bmatrix}_{lN\times p} \cdot [0_{p\times N}, I_{p\times p}].$$

Anfangsbedingung:

$$\left[\frac{dr}{dZ(T)}, 0_{1\times N}, \ldots, 0_{1\times N}\right]_{1\times lN} \cdot \frac{d\overbrace{Z(l-1)}}{d[Z(0), W]} = \left[\frac{dr}{dZ(l-1)}, 0_{1\times N}, \ldots, 0_{1\times N}\right] \cdot \begin{bmatrix} \frac{dZ(l-1)}{dZ(0)} & \frac{dZ(l-1)}{dW} \\ \vdots & \vdots \\ \frac{dZ(1)}{dZ(0)} & \frac{dZ(1)}{dW} \\ I_{N\times N} & 0_{N\times p} \end{bmatrix}.$$

Generalisierung der Kosten-Funktion für die Leapfrog Schema:

$$Z(k) \in \mathbb{R}^N, \;\; Z(k+1) = H(Z(k), Z(k-1), W), \text{ für } k = 1, \ldots, T-1.$$

$$\tilde{r} : \mathbb{R}^{2N} \to \mathbb{R}, \quad \tilde{r} : (Z(T)^T, Z(T-1)^T)^T \mapsto \tilde{r}(Z(T), Z(T-1)).$$

Rekursion:

$$\frac{d\tilde{r}}{d[Z(0), W]} = \frac{\partial \tilde{r}}{\partial Z(T)} \cdot \frac{dZ(T)}{d[Z(0), W]} + \frac{\partial \tilde{r}}{\partial Z(T-1)} \cdot \frac{dZ(T-1)}{d[Z(0), W]}$$

$$=: g\left(\frac{\partial \tilde{r}}{\partial Z(T)}, \frac{\partial \tilde{r}}{\partial Z(T-1)}, T\right),$$

$$g(p, q, T) = g(w^T, x^T, T-1) + g(y^T, z^T, T-2) + v^T, \quad T \geq 2.$$

$(p)_{1\times N}$, $(q)_{1\times N}$,
$(w^T)_{1\times N}$, $(x^T)_{1\times N}$, $(y^T)_{1\times N}$,
$(v^T)_{1\times(N+p)}$

$\sim 4^{T-1}$, binäre Rekursion.

# Time-parallel computation of pseudo-adjoints for a leapfrog scheme

Reduziere die Anzahl sparser Matrix-Vektor Multiplikationen ("One Step" Schema):

$$\overline{Z(k)} = [Z(k)^T, Z(k-1)^T]^T \in \mathbb{R}^{2N}.$$

$$\left[\frac{\partial \tilde{r}}{\partial Z(T)}, \frac{\partial \tilde{r}}{\partial Z(T-1)}\right]_{1\times 2N} \cdot \frac{d\overline{Z(T)}}{d[Z(0), W]_{2N\times(N+p)}} = (x^T)_{1\times 2N} \cdot \frac{d\overline{Z(T-1)}}{d[Z(0), W]_{2N\times N+p}} + (w^T)_{1\times(N+p)}, \quad T \geq 2,$$

$$(x^T)_{1\times 2N} = \left[\frac{\partial \tilde{r}}{\partial Z(T)}, \frac{\partial \tilde{r}}{\partial Z(T-1)}\right]_{1\times 2N} \cdot \begin{bmatrix} \frac{\partial H}{\partial Z(T-1)} & \frac{\partial H}{\partial Z(T-2)} \\ I_{N\times N} & 0_{N\times N} \end{bmatrix}_{2N\times 2N},$$

$$(w^T)_{1\times(N+p)} = \left[\frac{\partial \tilde{r}}{\partial Z(T)}, \frac{\partial \tilde{r}}{\partial Z(T-1)}\right]_{1\times 2N} \cdot \begin{bmatrix} \frac{\partial H}{\partial W} \\ 0_{N\times p} \end{bmatrix}_{2N\times p} \cdot [0_{p\times N}, I_{p\times p}].$$

$$\sim 2(T-1)$$

[1 ] Christian H. Bischof and H. Martin Bücker, Po-Ting Wu, Time-parallel computation of pseudo-adjoints for a leapfrog scheme,
International Journal of High Speed Computing, V. 12, No. 1 (2004), 1-27.

[2 ] C. H. Bischof, H.M. Bücker and P.-T. Wu, Exploiting intermediate sparsity in computing derivatives for a leapfrog scheme, *Computational Optimization and Applications* **24** (1) (2003) 117-133.

[3 ] Z. Wang, Variational data assimilation with 2-D shallow water equations and 3-D FSU global spectral models, Technical Report FSU-SCRI-93T-149, Florida State University, Department of Mathematics, December 1993.

[4 ] Z. Wang, I. M. Navon, X. Zou and F. LeDimet, A truncated Newton optimization algorithm in meteorology applications with analytic Hessian/vector products, *Comput. Opt. Appl.* **4** (1995) 241-262.