Exploiting Intermediate Sparsity in Computing Derivatives for a Leapfrog Scheme

Roland Schäfer

IGPM — RWTH Aachen

2006-08-15

Outline

Outline

1. What is Leapfrog
2. Black-Box Approach
3. Intermediate Sparsity (IS) Approach, Compressed Jacobian
4. 2d-Example Shallow Water
   - Simple Update
   - Complex Update
5. Conclusion

What is Leapfrog

Target: Calculate $Z(T)$ from $Z(0)$ (initial value) and W (parameter)

What is Leapfrog

Target: Calculate $Z(T)$ from $Z(0)$ (initial value) and W (parameter)

Leapfrog: $Z(t+1) = H\big(Z(t),\, Z(t-1),\, W\big)$

What is Leapfrog

Target: Calculate $Z(T)$ from $Z(0)$ (initial value) and W (parameter)

Leapfrog: $Z(t+1) = H\big(Z(t),\ Z(t-1),\ W\big)$

Leapfrog Scheme (LS)

Initialize $Z(0)$ and W

Compute $Z(1)$

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

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

end do

Outline

Calculate Derivatives

Let X be a subset of s elements from the $n + p$ sized $[Z(0), W]$.

We want:

$$\frac{dZ(T)}{dX}$$

Calculate Derivatives

Let X be a subset of s elements from the $n+p$ sized $[Z(0), W]$.

We want:

$$\frac{dZ(T)}{dX}$$

Black-Box Approach (BB)

Initialize $[Z(0), \frac{dZ(0)}{dX}]$ and $[W, \frac{dW}{dX}]$

Compute $[Z(1), \frac{dZ(1)}{dX}]$

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

$$\left[Z(t+1), \frac{dZ(t+1)}{dX}\right] = \hat{H}\left(Z(t), \frac{dZ(t)}{dX}, Z(t-1), \frac{dZ(t-1)}{dX}, W, \frac{dW}{dX}\right)$$

end do

Complexity of BB

Operations:

- Computation of H takes f_H flops

Complexity of BB

Operations:

- Computation of H takes f_H flops
- Computation of $Z(T)$ takes $O(f_H T)$ flops

Complexity of BB

Operations:

- Computation of H takes f_H flops
- Computation of $Z(T)$ takes $O(f_H T)$ flops
- Computation of $[Z(T), \frac{dZ(T)}{dX}]$ takes $O(s \cdot f_H T)$ flops

Complexity of BB

Operations:

- Computation of H takes f_H flops
- Computation of $Z(T)$ takes $O(f_H T)$ flops
- Computation of $[Z(T), \frac{dZ(T)}{dX}]$ takes $O(s \cdot f_H T)$ flops

Memory:

- need to save two timesteps

Complexity of BB

Operations:

- Computation of H takes f_H flops
- Computation of $Z(T)$ takes $O(f_H T)$ flops
- Computation of $[Z(T), \frac{dZ(T)}{dX}]$ takes $O(s \cdot f_H T)$ flops

Memory:

- need to save two timesteps
- $Z \in \mathbb{R}^n$, $W \in \mathbb{R}^p$
  therefore computation of H takes $O(2n + p)$ words of storage

Complexity of BB

Operations:

- Computation of H takes f_H flops
- Computation of $Z(T)$ takes $O(f_H T)$ flops
- Computation of $[Z(T), \frac{dZ(T)}{dX}]$ takes $O(s \cdot f_H T)$ flops

Memory:

- need to save two timesteps
- $Z \in \mathbb{R}^n$, $W \in \mathbb{R}^p$
  therefore computation of H takes $O(2n + p)$ words of storage
- BB takes $O(s \cdot (2n + p))$ words of storage

Outline

1. What is Leapfrog
2. Black-Box Approach
3. Intermediate Sparsity (IS) Approach, Compressed Jacobian
4. 2d-Example Shallow Water
   - Simple Update
   - Complex Update
5. Conclusion

Motivation

Leapfrog Scheme:

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

Motivation

Leapfrog Scheme:

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

Differentiate w.r.t. X:

$$\frac{dZ(t+1)}{dX} = \frac{\partial H}{\partial Z(t)} \cdot \frac{dZ(t)}{dX} + \frac{\partial H}{\partial Z(t-1)} \cdot \frac{dZ(t-1)}{dX} + \frac{\partial H}{\partial W} \cdot \frac{dW}{dX}$$

Motivation

Leapfrog Scheme:

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

Differentiate w.r.t. X:

$$\frac{dZ(t+1)}{dX} = \frac{\partial H}{\partial Z(t)} \cdot \frac{dZ(t)}{dX} + \frac{\partial H}{\partial Z(t-1)} \cdot \frac{dZ(t-1)}{dX} + \frac{\partial H}{\partial W} \cdot \frac{dW}{dX}$$

Fact

$$\frac{\partial H}{\partial \ldots} \textit{ is sparse for PDE problems}$$

Motivation

Leapfrog Scheme:

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

Differentiate w.r.t. X:

$$\frac{dZ(t+1)}{dX} = \frac{\partial H}{\partial Z(t)} \cdot \frac{dZ(t)}{dX} + \frac{\partial H}{\partial Z(t-1)} \cdot \frac{dZ(t-1)}{dX} + \frac{\partial H}{\partial W} \cdot \frac{dW}{dX}$$

Fact

$$\frac{\partial H}{\partial \ldots} \text{ *is sparse for PDE problems*}$$

Exploit this fact with cheap "sparse matrix – matrix" multiplications

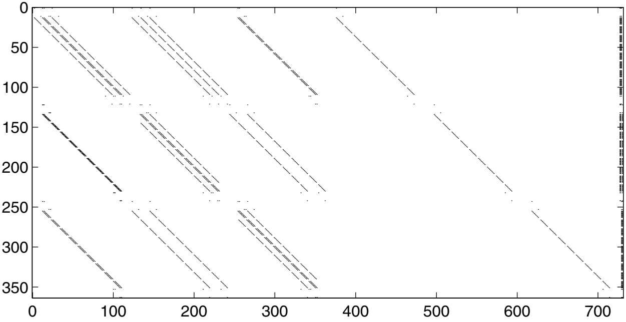
Why is the matrix sparse

H typically comes from a stencil, which depends only on a few, neighbored cells

Why is the matrix sparse

H typically comes from a stencil, which depends only on a few, neighbored cells

In forthcoming example:
max. 13 non-zero entries of $[\frac{\partial H}{\partial Z(t)}, \frac{\partial H}{\partial Z(t-1)}, \frac{\partial H}{\partial W}]$



Assume: “sparse matrix – matrix” multiplications are cheap

Intermediate Sparsity Approach (IS)

Assume: "sparse matrix – matrix" multiplications are cheap

Intermediate Sparsity Approach (IS)

Initialize $[Z(0), \frac{dZ(0)}{dX}]$ and $[W, \frac{dW}{dX}]$.

Compute $[Z(1), \frac{dZ(1)}{dX}]$.

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

  Step 1: Compute $Z(t+1)$ and $\frac{\partial H}{\partial Z(t)}, \frac{\partial H}{\partial Z(t-1)}, \frac{\partial H}{\partial W}$

  Step 2: Compute* $\frac{dZ(t+1)}{dX}$ via matrix-matrix multiplication

end do

* via $$\frac{dZ(t+1)}{dX} = \frac{\partial H}{\partial Z(t)} \cdot \frac{dZ(t)}{dX} + \frac{\partial H}{\partial Z(t-1)} \cdot \frac{dZ(t-1)}{dX} + \frac{\partial H}{\partial W} \cdot \frac{dW}{dX}$$

Complexity for IS-SL

Assume: "matrix-matrix" multiplications are optimized for sparse matrices: sparse linear algebra (SL)

Complexity for IS-SL

Assume: "matrix-matrix" multiplications are optimized for sparse matrices: sparse linear algebra (SL)

Operations:

- Stencil size is $O(\kappa)$

Complexity for IS-SL

Assume: "matrix-matrix" multiplications are optimized for sparse matrices: sparse linear algebra (SL)

Operations:

- Stencil size is $O(\kappa)$
- Step 1 needs a total of $O(\kappa f_H T)$ flops (instead of $O(s f_H T)$)

Complexity for IS-SL

Assume: "matrix-matrix" multiplications are optimized for sparse matrices: sparse linear algebra (SL)

Operations:

- Stencil size is $O(\kappa)$
- Step 1 needs a total of $O(\kappa f_H T)$ flops (instead of $O(s f_H T)$)
- matrix-matrix multiplication needs $O(snT)$

Complexity for IS-SL

Assume: "matrix-matrix" multiplications are optimized for sparse matrices: sparse linear algebra (SL)

Operations:

- Stencil size is $O(\kappa)$
- Step 1 needs a total of $O(\kappa f_H T)$ flops (instead of $O(sf_H T)$)
- matrix-matrix multiplication needs $O(snT)$

Memory:

- Step 1 needs $O(\kappa(2n+p))$ words of storage

Complexity for IS-SL

Assume: "matrix-matrix" multiplications are optimized for sparse matrices: sparse linear algebra (SL)

Operations:

- Stencil size is $O(\kappa)$
- Step 1 needs a total of $O(\kappa f_H T)$ flops (instead of $O(sf_H T)$)
- matrix-matrix multiplication needs $O(snT)$

Memory:

- Step 1 needs $O(\kappa(2n+p))$ words of storage
- Step 2 needs $O(sn)$ words of storage

Another way: Compressed Jacobians (IS-CJ)

Let S^1, S^2 be suitable chosen seed matrices with λ_1, λ_2 columns for $\frac{dZ(t)}{dX}$ and $\frac{dZ(t-1)}{dX}$

Another way: Compressed Jacobians (IS-CJ)

Let S^1, S^2 be suitable chosen seed matrices with λ_1, λ_2 columns for $\frac{dZ(t)}{dX}$ and $\frac{dZ(t-1)}{dX}$

Obtain a compressed version of $\frac{\partial H}{\partial \ldots}$

Another way: Compressed Jacobians (IS-CJ)

Let S^1, S^2 be suitable chosen seed matrices with λ_1, λ_2 columns for $\frac{dZ(t)}{dX}$ and $\frac{dZ(t-1)}{dX}$

Obtain a compressed version of $\frac{\partial H}{\partial \ldots}$

Complexity: $(\lambda = \lambda_1 + \lambda_2 + p)$

- Computation: $O(\lambda f_H T) + O(snT)$
- Memory: $O(\lambda(2n + p)) + O(sn)$

Complexity overview

	BB	IS–SL	IS–CJ
Computation	$O(sf_H T)$	$O(\kappa f_H T) + O(snT)$	$O(\lambda f_H T) + O(snT)$
Storage	$O(s(2n+p))$	$O(\kappa(2n+p)) + O(sn)$	$O(\lambda(2n+p)) + O(sn)$

Complexity overview

	BB	IS–SL	IS–CJ
Computation	$O(sf_H T)$	$O(\kappa f_H T) + O(snT)$	$O(\lambda f_H T) + O(snT)$
Storage	$O(s(2n+p))$	$O(\kappa(2n+p)) + O(sn)$	$O(\lambda(2n+p)) + O(sn)$

When is IS faster then BB?

Complexity overview

	BB	IS–SL	IS–CJ
Computation	$O(sf_H T)$	$O(\kappa f_H T) + O(snT)$	$O(\lambda f_H T) + O(snT)$
Storage	$O(s(2n+p))$	$O(\kappa(2n+p)) + O(sn)$	$O(\lambda(2n+p)) + O(sn)$

When is IS faster then BB?

Assume: $\kappa, \lambda \ll s$

Complexity overview

	BB	IS–SL	IS–CJ
Computation	$O(sf_H T)$	$O(\kappa f_H T) + O(snT)$	$O(\lambda f_H T) + O(snT)$
Storage	$O(s(2n+p))$	$O(\kappa(2n+p)) + O(sn)$	$O(\lambda(2n+p)) + O(sn)$

When is IS faster then BB?

Assume: $\kappa, \lambda \ll s$

IS is faster then BB if $O(\kappa f_H T) \gg O(snT)$

Outline

1. What is Leapfrog
2. Black-Box Approach
3. Intermediate Sparsity (IS) Approach, Compressed Jacobian
4. 2d-Example Shallow Water
   - Simple Update
   - Complex Update
5. Conclusion

Example Shallow Water in 2d

Shallow Water used to simulate water flow, where vertical dimension is much smaller than horizontal (shallow).

E.g. rivers, lakes, costal flow

Variables: water height, x-momentum, y-momentum (2d)

Example Shallow Water in 2d

Shallow Water used to simulate water flow, where vertical dimension is much smaller than horizontal (shallow).

E.g. rivers, lakes, costal flow

Variables: water height, x-momentum, y-momentum (2d)

We calculate $s = n + p$ derivatives

Grid size	n	p	$s = n + p$
11×11	$3 \cdot 11 \cdot 11 = 363$	4	367
16×16	$3 \cdot 16 \cdot 16 = 768$	4	772
21×21	$3 \cdot 21 \cdot 21 = 1323$	4	1327

Memory & . . .

Grid size	IS–SL	IS–CJ	BB
11×11	3.72	3.85	4.70
16×16	13.61	13.84	18.82
21×21	37.82	38.16	53.31

Memory requirements in megabytes

. . . & Runtime

Grid size (platform)	SL	CJ	MM	IS–SL	IS–CJ	BB
11×11 (IBM)	4.90	1.93	8.03	12.93	9.96	4.24
16×16 (IBM)	17.77	8.70	38.66	56.43	47.36	36.68
21×21 (IBM)	42.98	21.51	119.32	162.30	140.83	71.98
11×11 (Sun)	12.26	6.55	19.24	31.50	25.79	26.63

Runtime in seconds

- SL: calculate sparse Jacobians
- CJ: calculate compressed Jacobians
- MM: matrix-matrix multiplications
- SL+MM = IS-SL, CJ+MM = IS-CJ

Complexity overview

	BB	IS–SL	IS–CJ
Computation	$O(sf_H T)$	$O(\kappa f_H T) + O(snT)$	$O(\lambda f_H T) + O(snT)$
Storage	$O(s(2n+p))$	$O(\kappa(2n+p)) + O(sn)$	$O(\lambda(2n+p)) + O(sn)$

When is IS faster then BB?

Assume: $\kappa, \lambda \ll s$

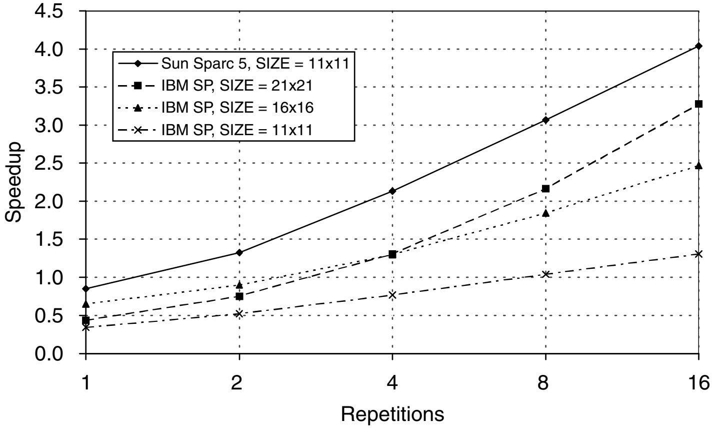
IS is faster then BB if $O(\kappa f_H T) \gg O(snT)$

Expensive Update

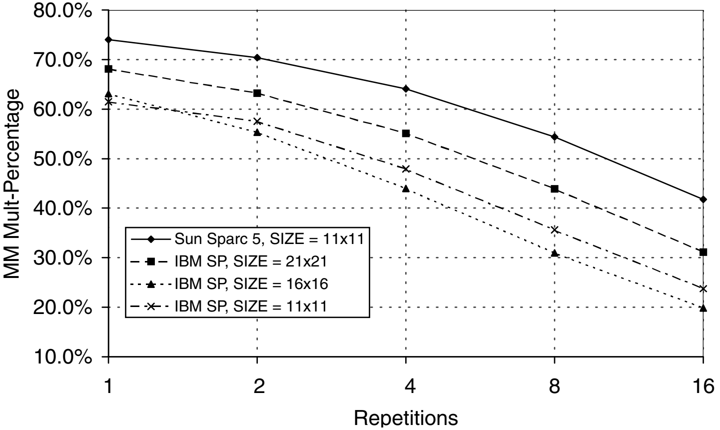
Lets evaluate H not only one time, but up to 16 times to emulate a function H, which is more expensive to calculate

Expensive Update

Lets evaluate H not only one time, but up to 16 times to emulate a function H, which is more expensive to calculate (IS-SL)



MM Multiplication Percentage



Outline

1. What is Leapfrog
2. Black-Box Approach
3. Intermediate Sparsity (IS) Approach, Compressed Jacobian
4. 2d-Example Shallow Water
   - Simple Update
   - Complex Update
5. Conclusion

Conclusion

- Exploiting Sparsity works, if $O(\kappa f_H T) \gg O(snT)$

Conclusion

- Exploiting Sparsity works, if $O(\kappa f_H T) \gg O(snT)$
- this means: evaluation of H must be expensive

Conclusion

- Exploiting Sparsity works, if $O(\kappa f_H T) \gg O(snT)$
- this means: evaluation of H must be expensive
- for example: High-Order code

Conclusion

- Exploiting Sparsity works, if $O(\kappa f_H T) \gg O(snT)$
- this means: evaluation of H must be expensive
- for example: High-Order code
- substantial speedup

Conclusion

- Exploiting Sparsity works, if $O(\kappa f_H T) \gg O(snT)$
- this means: evaluation of H must be expensive
- for example: High-Order code
- substantial speedup

Lit.: C. Bischof, M. Bücker, P. Wu:
Exploiting Intermediate Sparsity in Computing Derivatives for a Leapfrog Scheme,
Comp. Opt. Appl. 24, 117–133, 2003