Parallel, matrix-free framework for frequency-domain modelling, imaging and inversion

Tristan van Leeuwen

SLIM
University of British Columbia

Matlab framework

Wave-equation solver
model->wavefield

Modelling operator
model -> data

Migration/de-migration
(Jacobian)
model pert. <-> data pert.

objective function
(model,data)->(misfit,gradient,Hessian)

Misc. linear operators
(interpolation)
vector -> vector

Transforms
(Curvelets, Wavelets)
vector -> vector

Misc. non-linear functions
vector->(scalar,gradient)
vector -> (vector, Jacobian)

SPOT: Matrix-free linear operators
pSPOT: Parallel SPOT
function handles

gradient-based optimization
(L-BFGS, Gauss-Newton)
objective function->solution

linear solvers
(LSQR, SPGl1)
(Matrix, r.h.s)->solution

Full waveform inversion =
objective function +
non-linear optimization

Least-Squares migration =
Jacobian + LSQR

sparsity promoting migration =
Jacobian + SPGL1

Matlab framework

Modelling

```
[D,J] = F(m,Q,model)
```

`D` - data cube as Matlab distributed array

`J` - Jacobian as SPOT operator

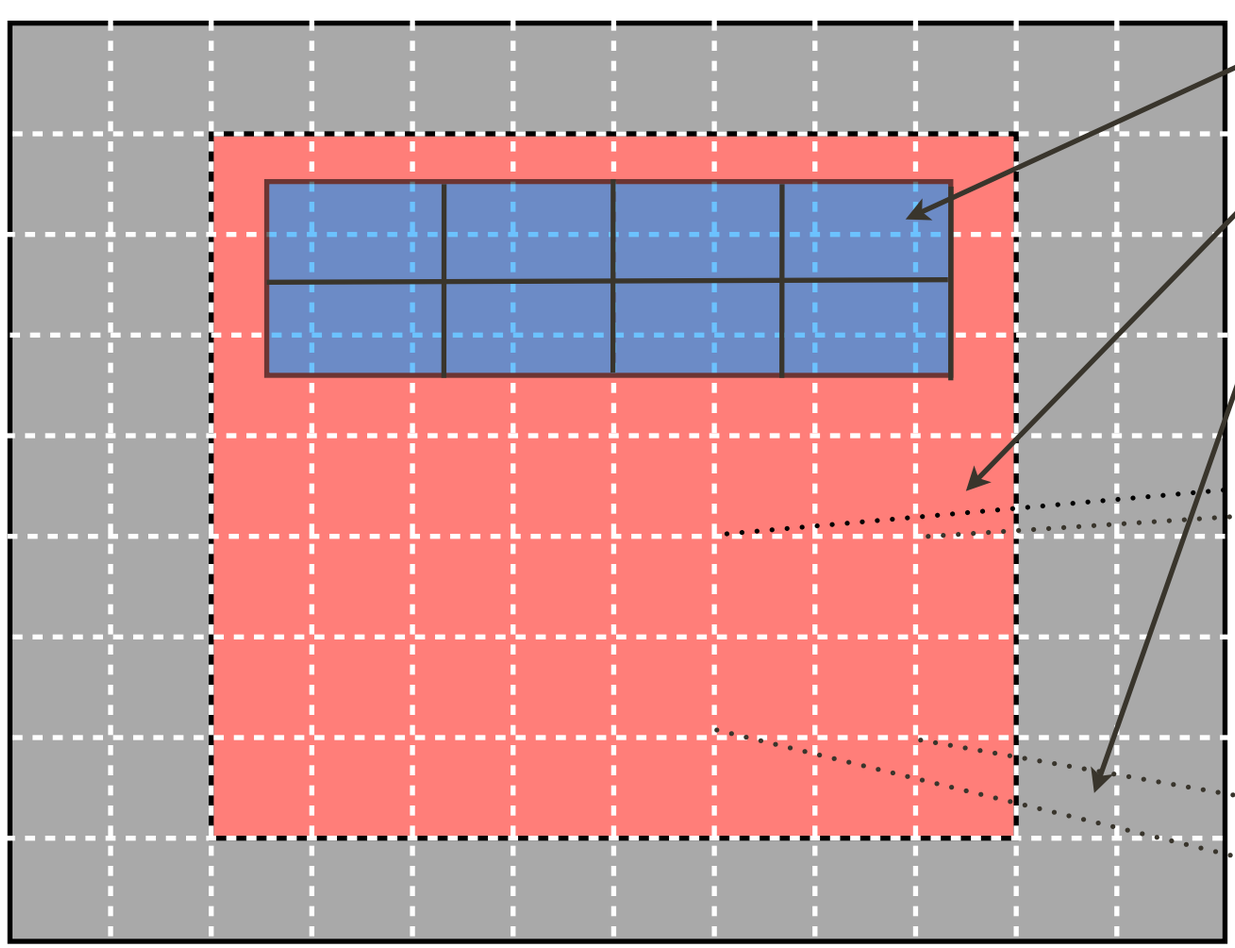
`m` - model as Matlab array

`Q` - source functions [q1, q2, ...]

`model` - struct containing acquisition setup etc.

Matlab framework

Discretization:



Matlab framework

- 9-point mixed-grid stencil [Jo et al '96]
- sponge boundary
- cubic interpolation for source injection and sampling
- Exact adjoint of discretized system
- parallel over frequencies

Matlab framework

1140 x 344 gridpoints, 500 sources, 64 frequencies

# of procs	time [s]	speed up	efficiency
2	36611	1.58	0.79
4	19544	2.96	0.74
8	9623	6.03	0.75
16	4525	12.82	0.80
32	2289	25.34	0.79

Matlab framework

Action of the Jacobian

```
D    = J*dm;
% calls J.multiply(dm,1)
dmt = J'*D;
% calls J.multiply(D,-1)
```

Matlab framework

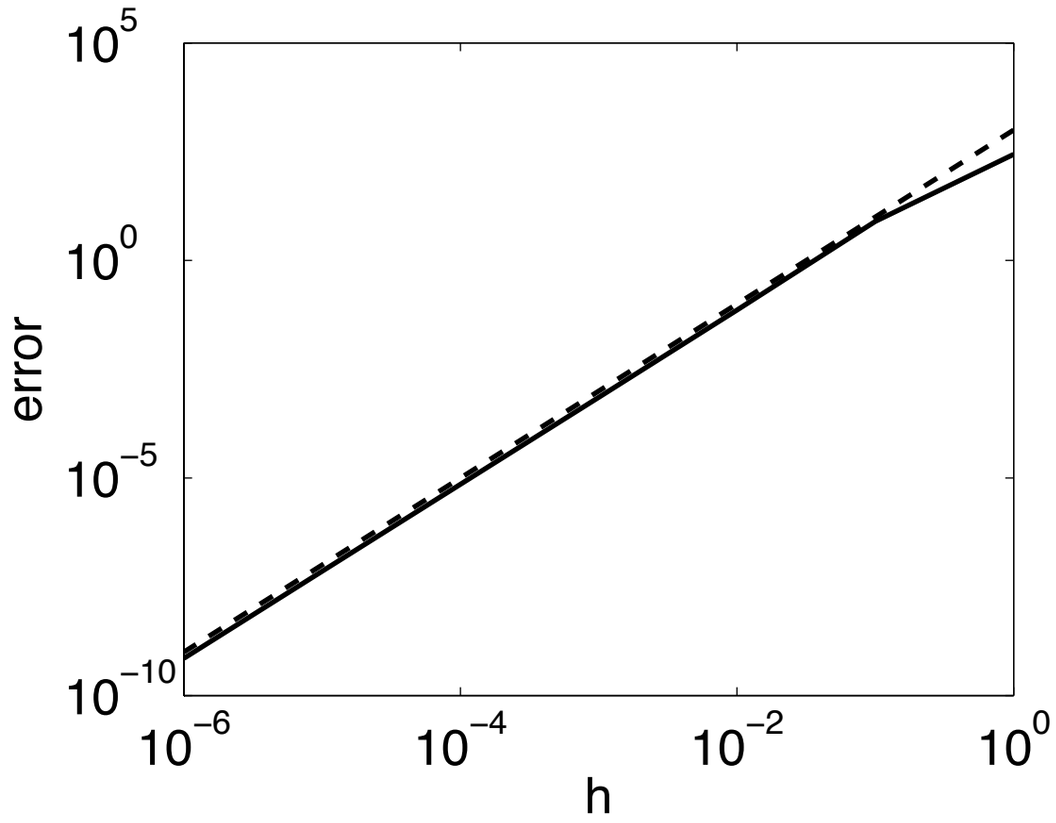
Testing

```
%test Jacobian
[D0,J0] = F(m0,Q,model);
D1      = F(m0+dm,Q,model);
dD      = J0*dm;
norm(D1 - D0 - dD)

%dottest: (Ax)'*b = x'*(A'*b)
(J*dm)'*dD
dm'(J'*dD)
```

Matlab framework

Jacobian test

$$F(\mathbf{x} + h\delta\mathbf{x}) = F(\mathbf{x}) + h\nabla F(\mathbf{x})\delta\mathbf{x} + \mathcal{O}(h^2).$$



Matlab framework

Dottest:

$\langle A\mathbf{x}, \mathbf{y} \rangle$	$\langle \mathbf{x}, A^*\mathbf{y} \rangle$
-9.3405e+08+4.1677e+08i	-9.3405e+08+4.1677e+08i
8.8128e+08-1.6759e+08i	8.8128e+08-1.6759e+08i
3.7126e+08+2.1559e+07i	3.7126e+08+2.1559e+07i
-1.7585e+08-1.7851e+08i	-1.7585e+08-1.7851e+08i

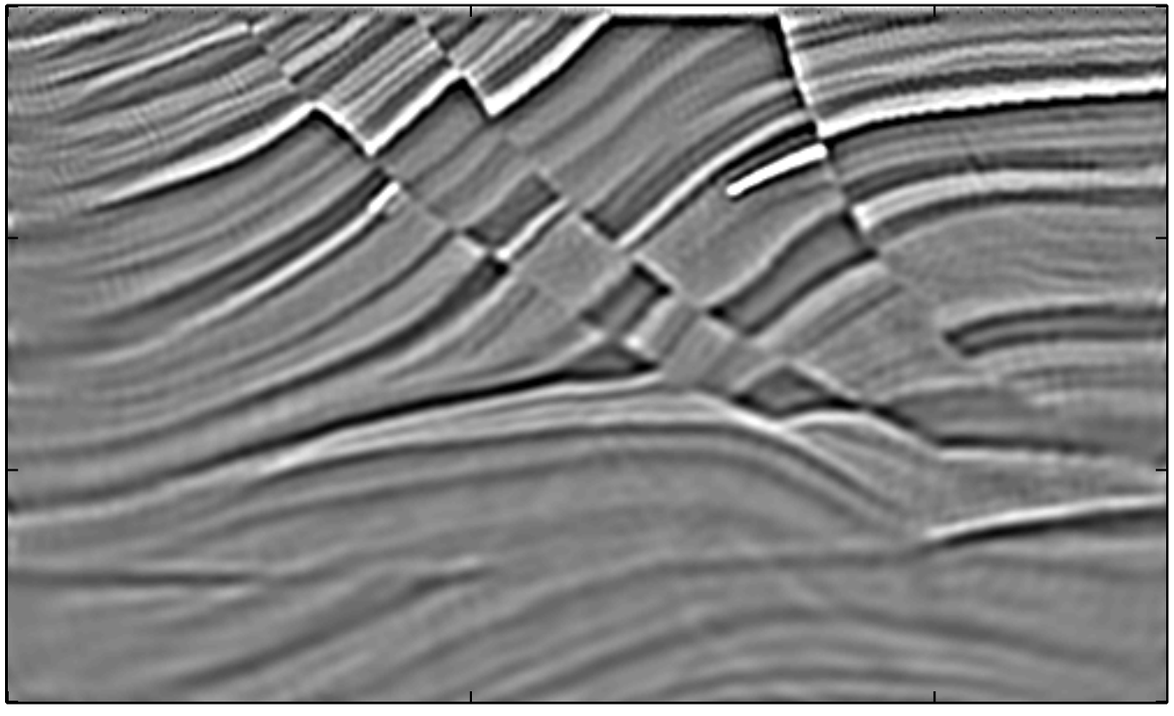
Matlab framework

Least-squares migration

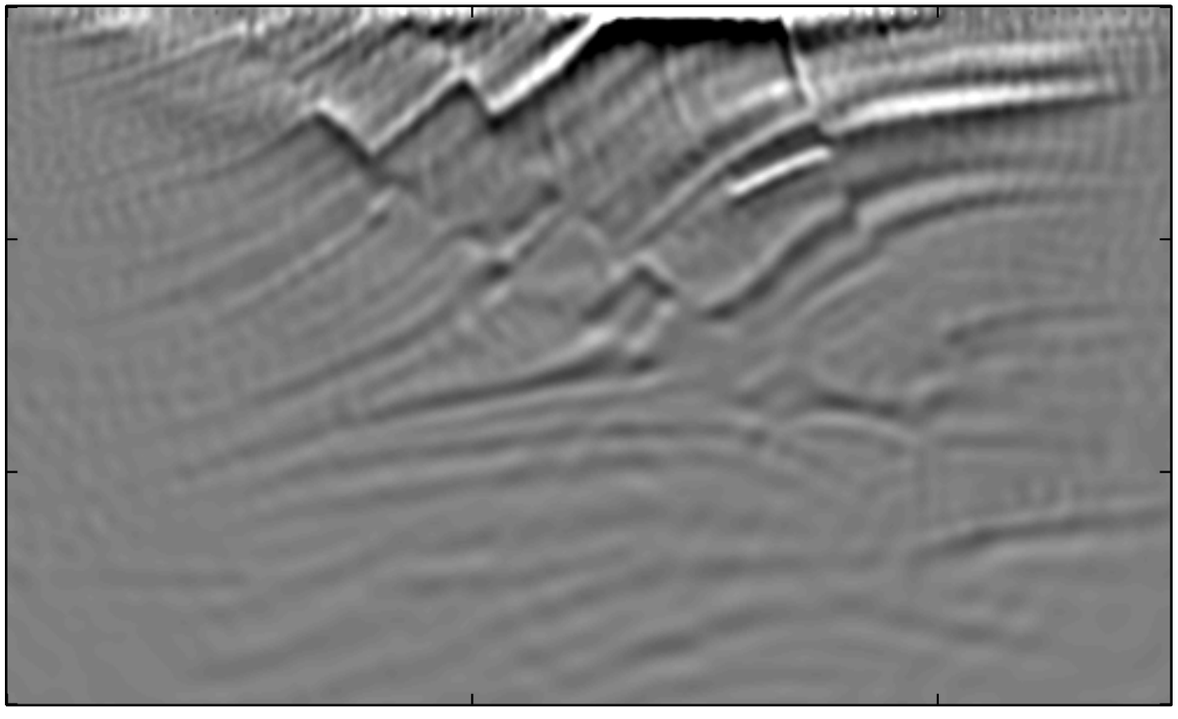
```
% model m0, data D

J   = opDF(m0,Q,model);
dm = LSQR(J,D);
```

Matlab framework

50 LSQR iterations



correct adjoint



wrong adjoint

Matlab example

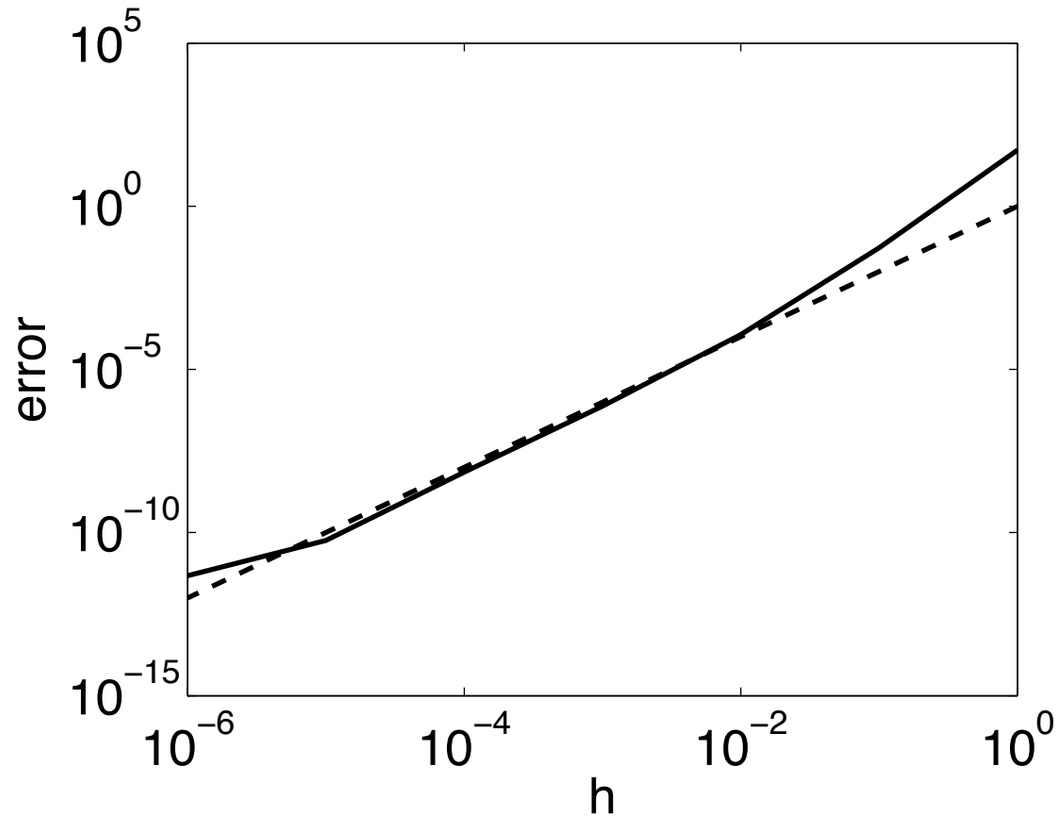
misfit

```
function [f,g] = Phi(m,Q,D,model)

[Dt,Jt]  = F(m,Q,model);
[f,df]   = penalty(Dt-D);
g        = Jt'*df;
```

slide title

gradient test

$$f(\mathbf{x} + h\delta\mathbf{x}) = f(\mathbf{x}) + h\langle \nabla f(\mathbf{x}), \delta\mathbf{x} \rangle + \mathcal{O}(h^2).$$

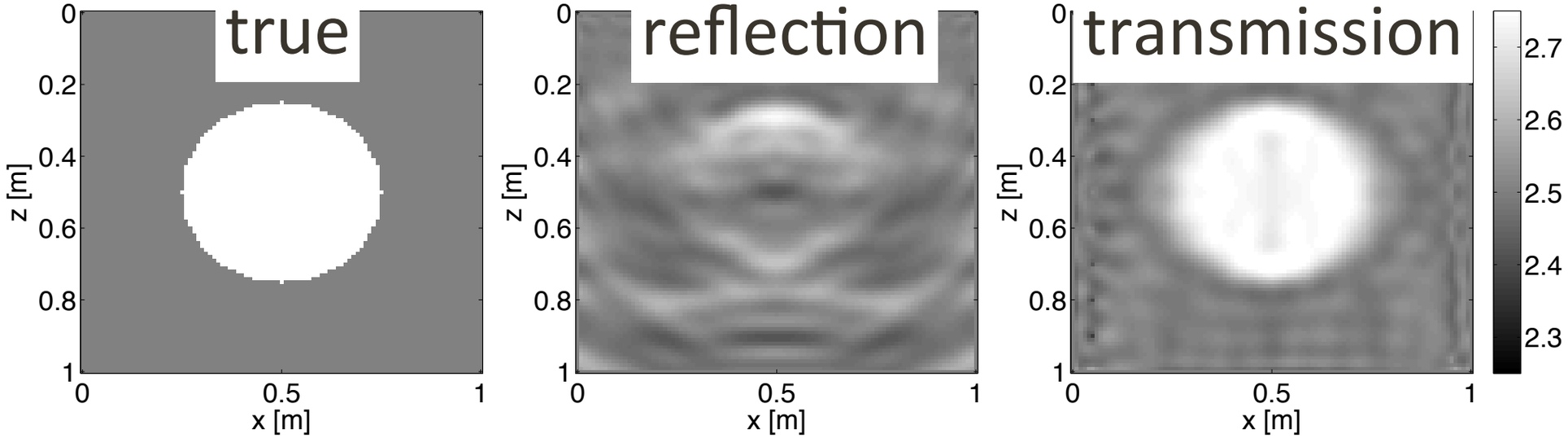


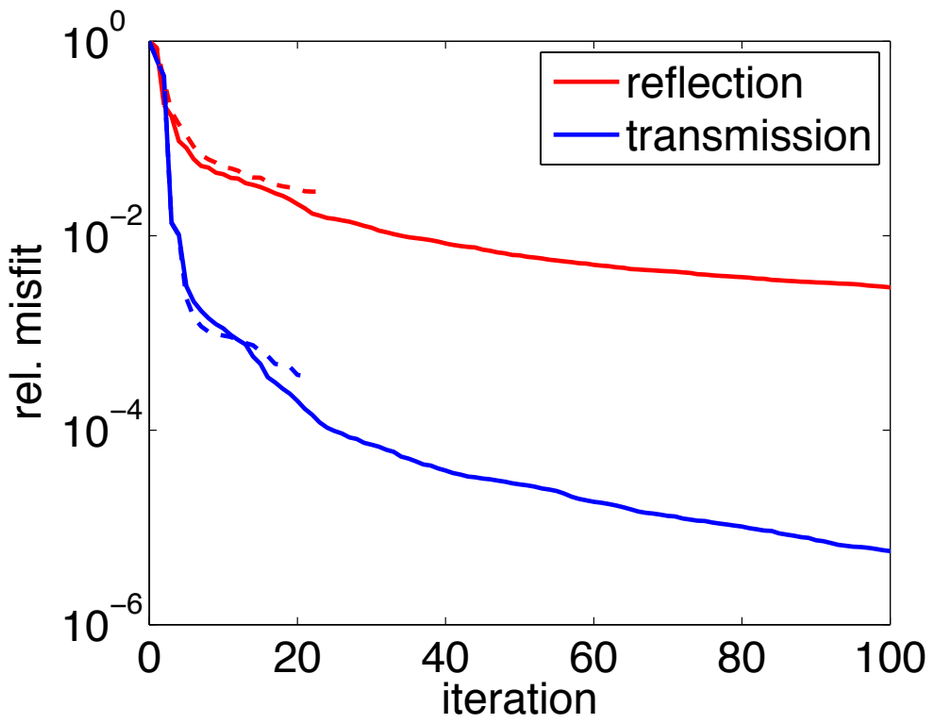
Matlab framework

Optimization

```
% read data D
% define model,Q,m0

% function handle
fh = @(m)misfit(m,Q,D,model);

% optimization
mn = mylbfgs(fh,m0);
```

Matlab framework





Summary

- Flexible and scalable framework
- easy prototyping
- correct Jacobian & adjoint is important
- *download now:*

https://www.slim.eos.ubc.ca/releases

SLIM

Download now

Acknowledgements

This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, BP, Chevron, ConocoPhillips, Petrobras, PGS, Total SA, and WesternGeco.