Perfectly Matched Layer and TAPENADE

Yin Huang, Lei Fu

TRIP review meeting

May 6, 2014

Perfectly Matched Layer (PML)

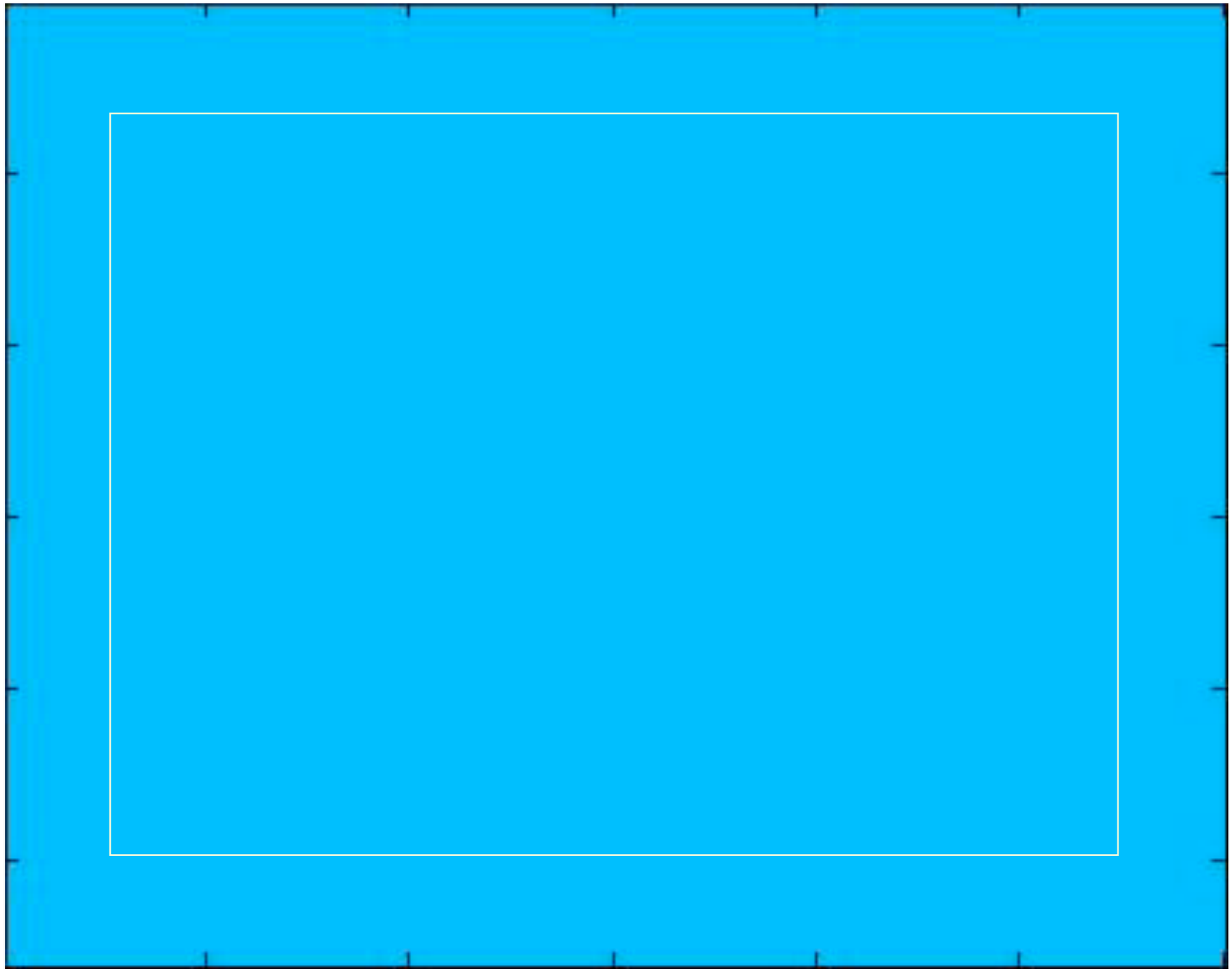
PML is an absorbing material boundary condition to approximate wave propagations in an infinite domain.

- Originally developed for Maxwells equations (Berenger, 1994)
- Most PML formatlations need first-order systems of wave equations (Habashy 2007, etc)
- Grote's PML for second order wave equation (Grote and Sim, 2010)

$$
\begin{aligned}
u_{tt} + (\zeta_0 + \zeta_1)u_t + \zeta_0\zeta_1 u &= c^2\Delta u + D_z\phi_0 + D_x\phi_1 \\
\phi_{0t} &= -\zeta_0\phi_0 + c^2(\phi_1 - \phi_0)D_x u \\
\phi_{1t} &= -\zeta_1\phi_1 + c^2(\phi_0 - \phi_1)D_z u
\end{aligned}
$$

with ζ_0 and ζ_1 two damping profiles in z and x directions respectively, and ϕ_0, ϕ_1 two auxiliary fields.

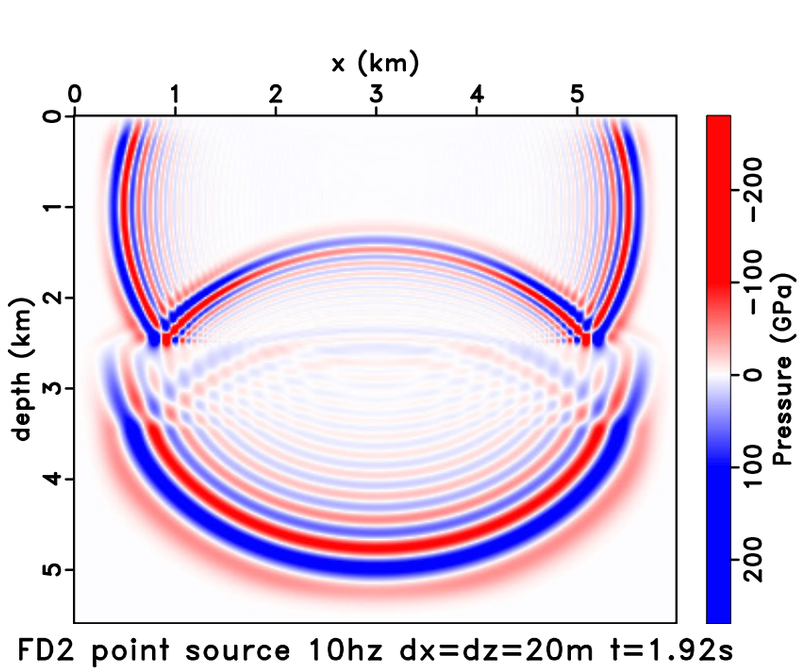
Movie with PML

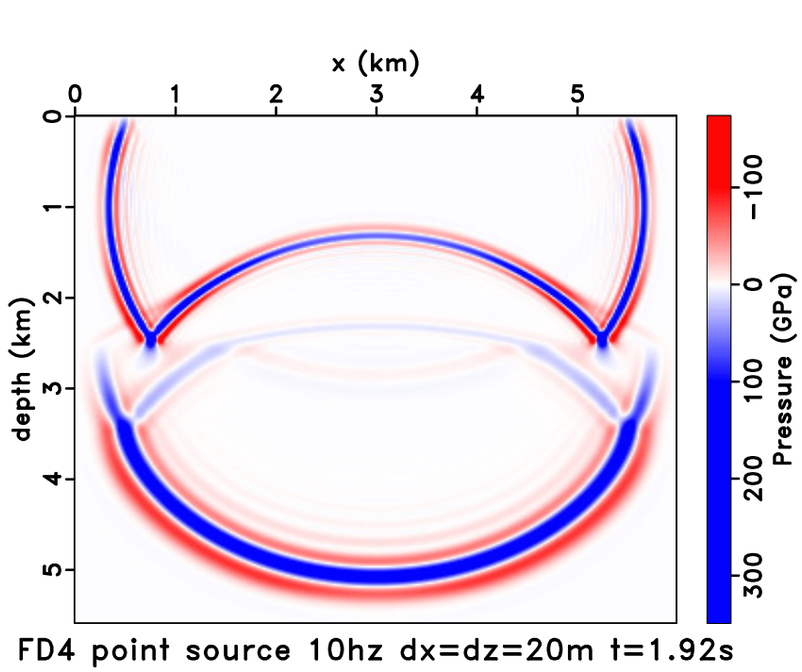


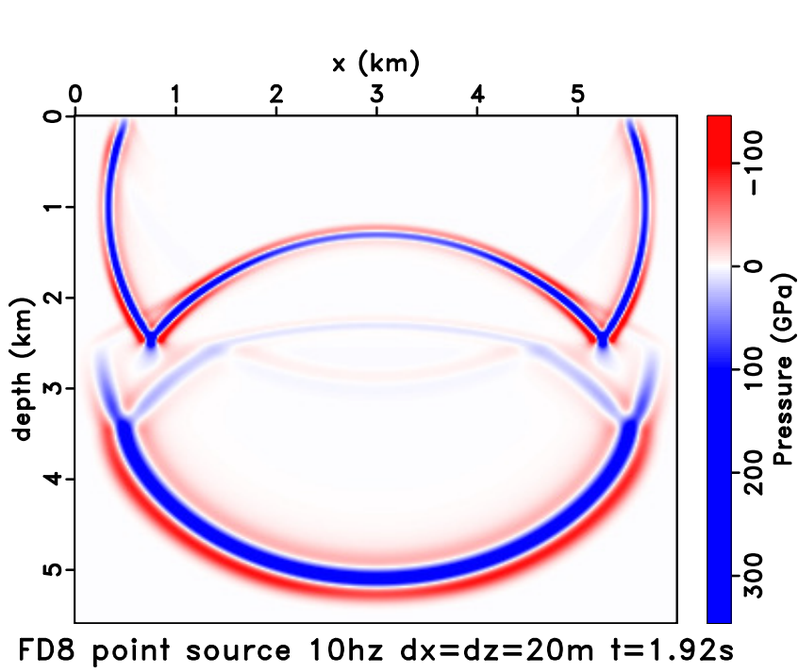
High order finite difference

Original paper uses 2nd order scheme for both time and space discretization.

Only Δu and u_{tt} are non-zero outside the PML layer. $\Rightarrow$ only discretize Δu using higher order finite difference will get higher order scheme.



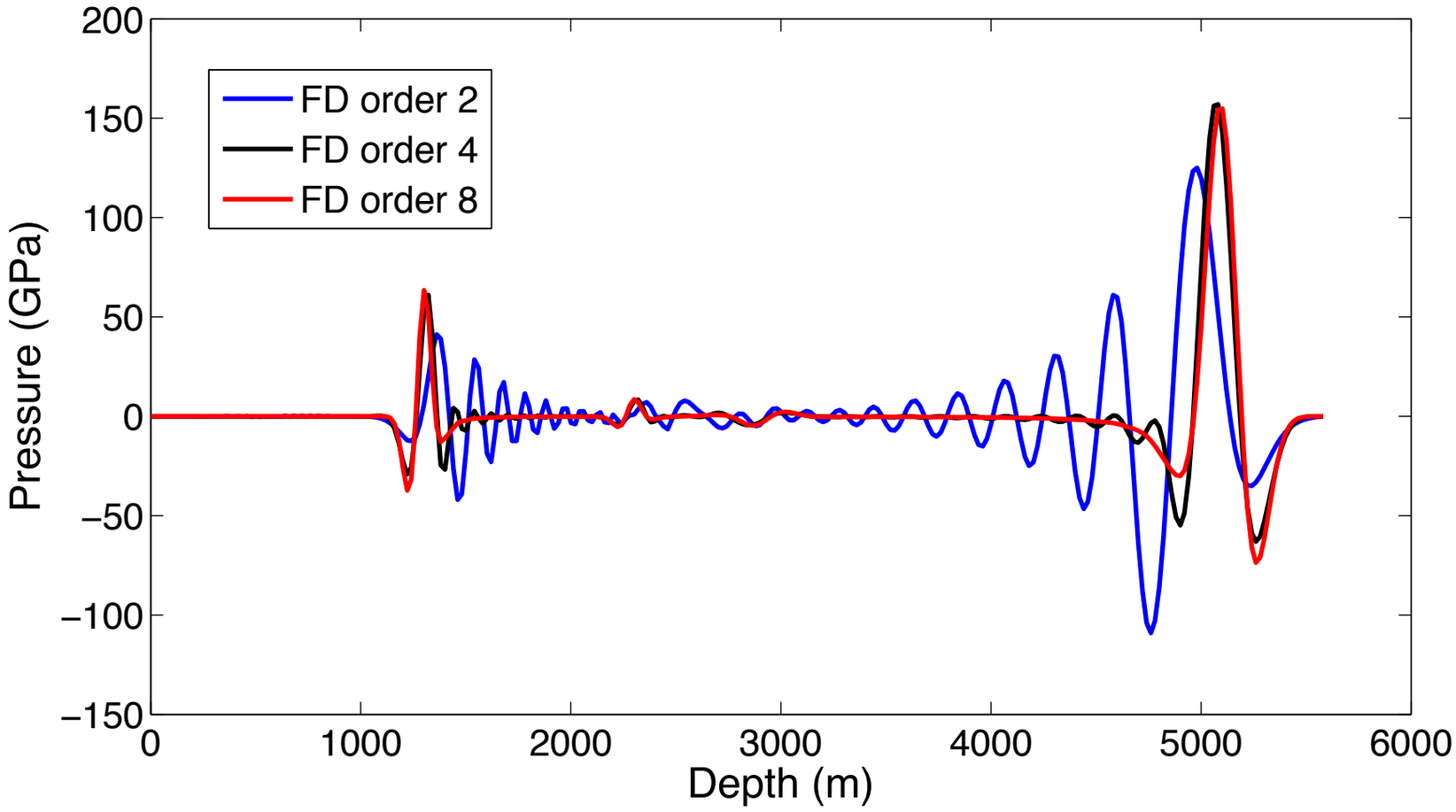




High order finite difference

Original paper uses 2nd order scheme for both time and space discretization.

Only Δu and u_{tt} are non-zero outside the PML layer. $\Rightarrow$ only discretize Δu using higher order finite difference will get higher order scheme.



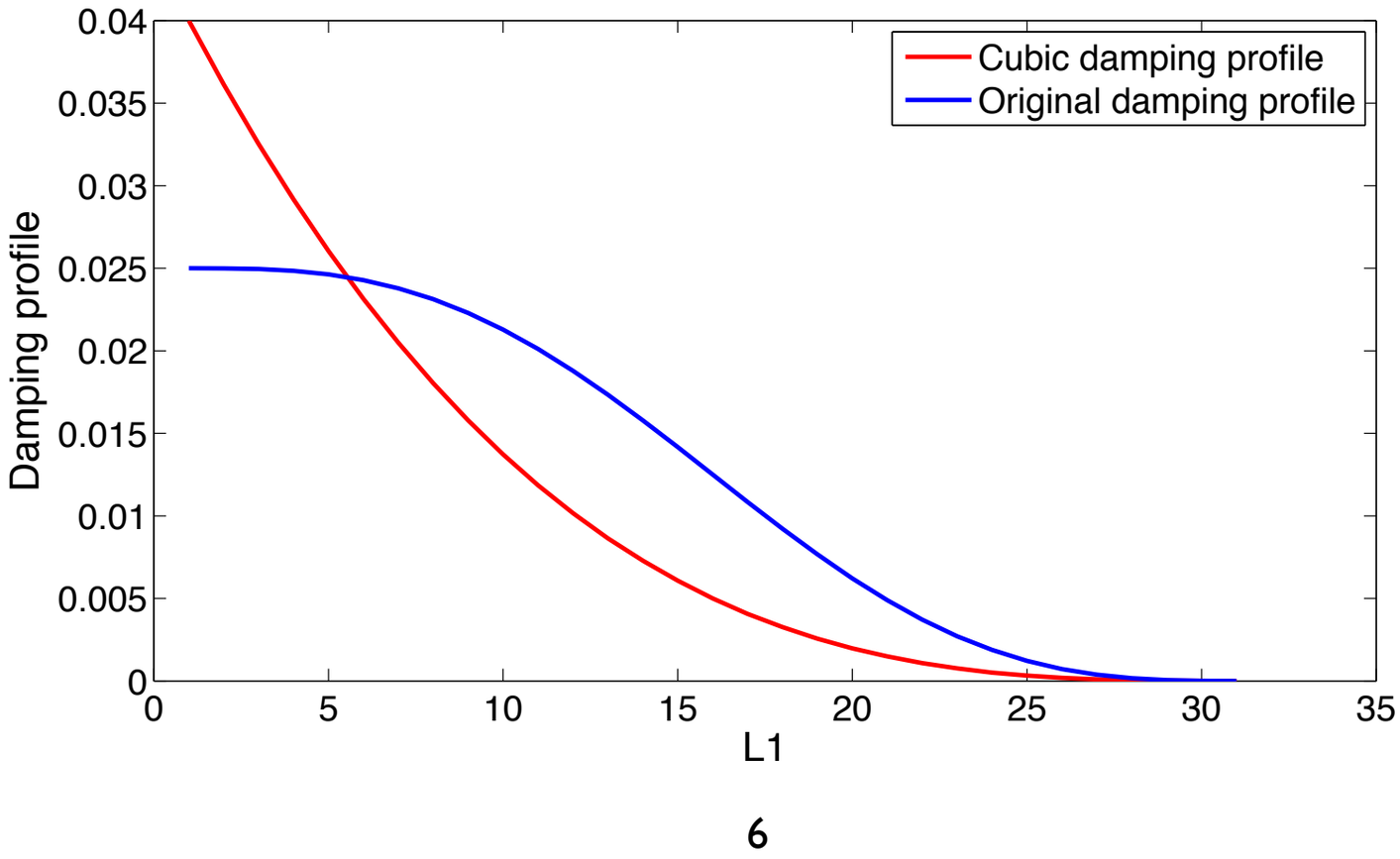
Different damping profiles

Original damping profile

$$\zeta_i(x_i) = \zeta_m \left(\frac{|x_i - L_i|}{L_i} - \frac{\sin(\frac{2\pi|x_i - Li|}{L_i})}{2\pi} \right), x_i \leq L_i$$

Cubic damping profile

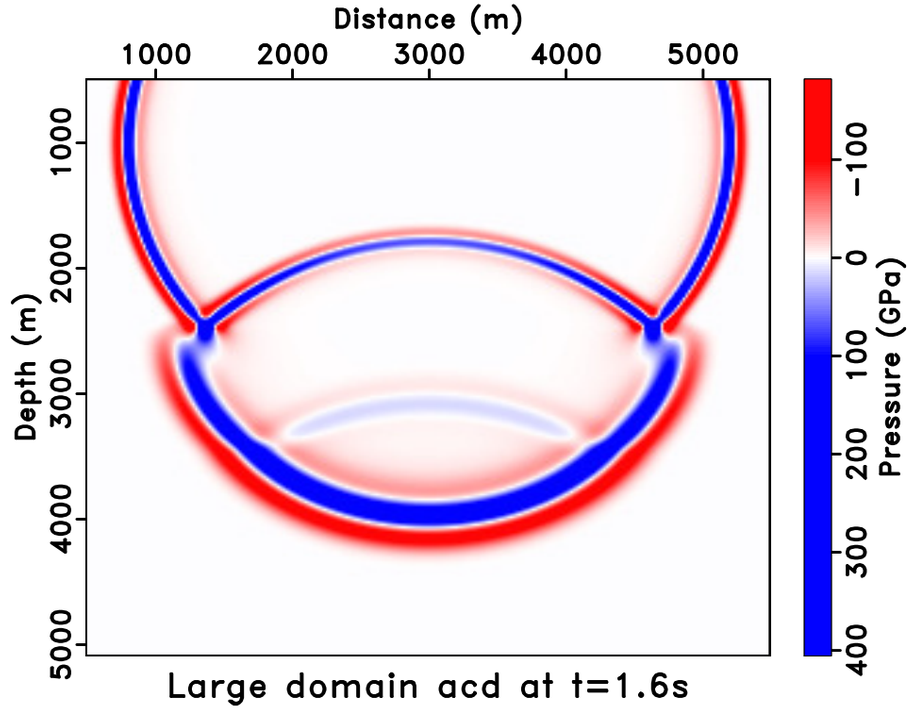
$$\zeta_i(x_i) = \zeta_m \left(\frac{|x_i - L_i|}{L_i} \right)^3, x_i \leq L_i$$

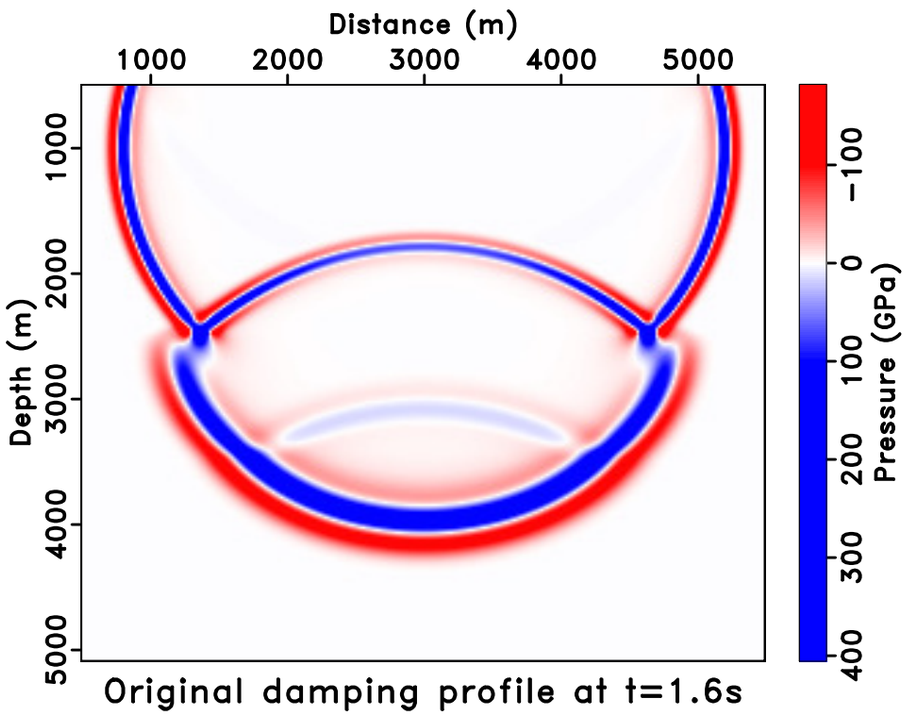


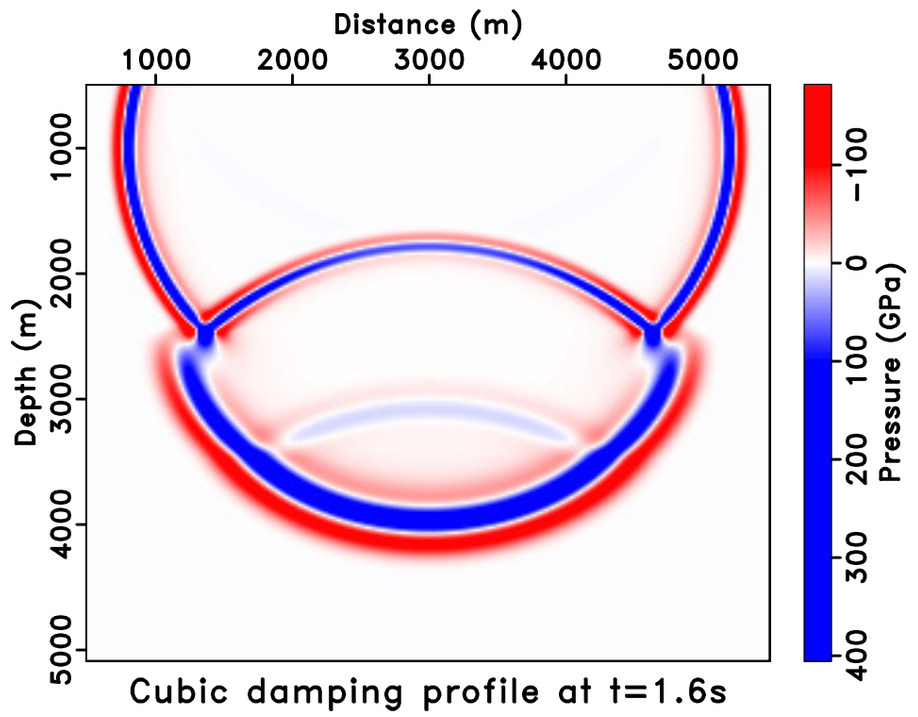
2 reflector model and frames

Frames at $t = 1.6$s with different damping profiles, and large domain simulation.

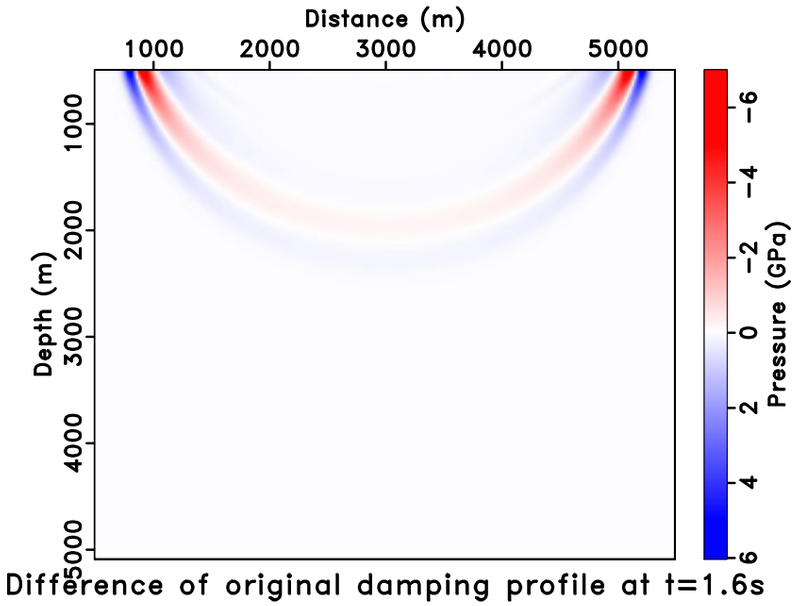
PML layers are not included.



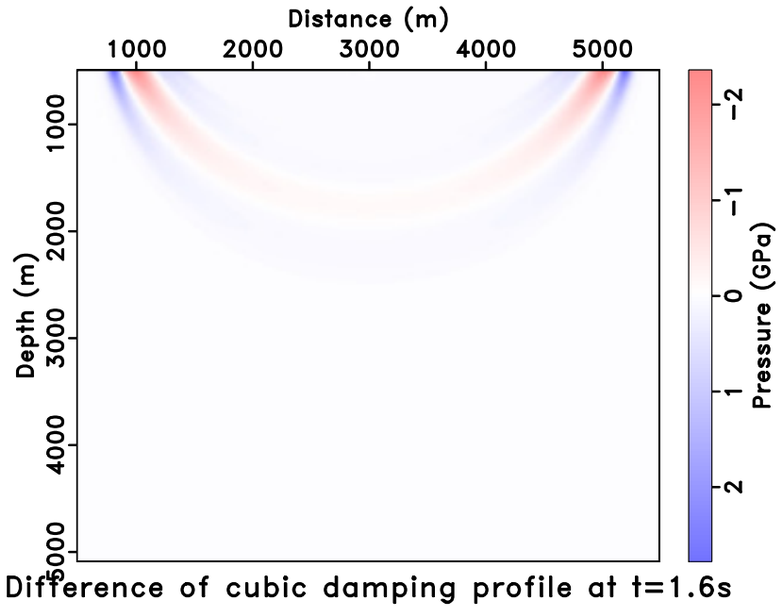




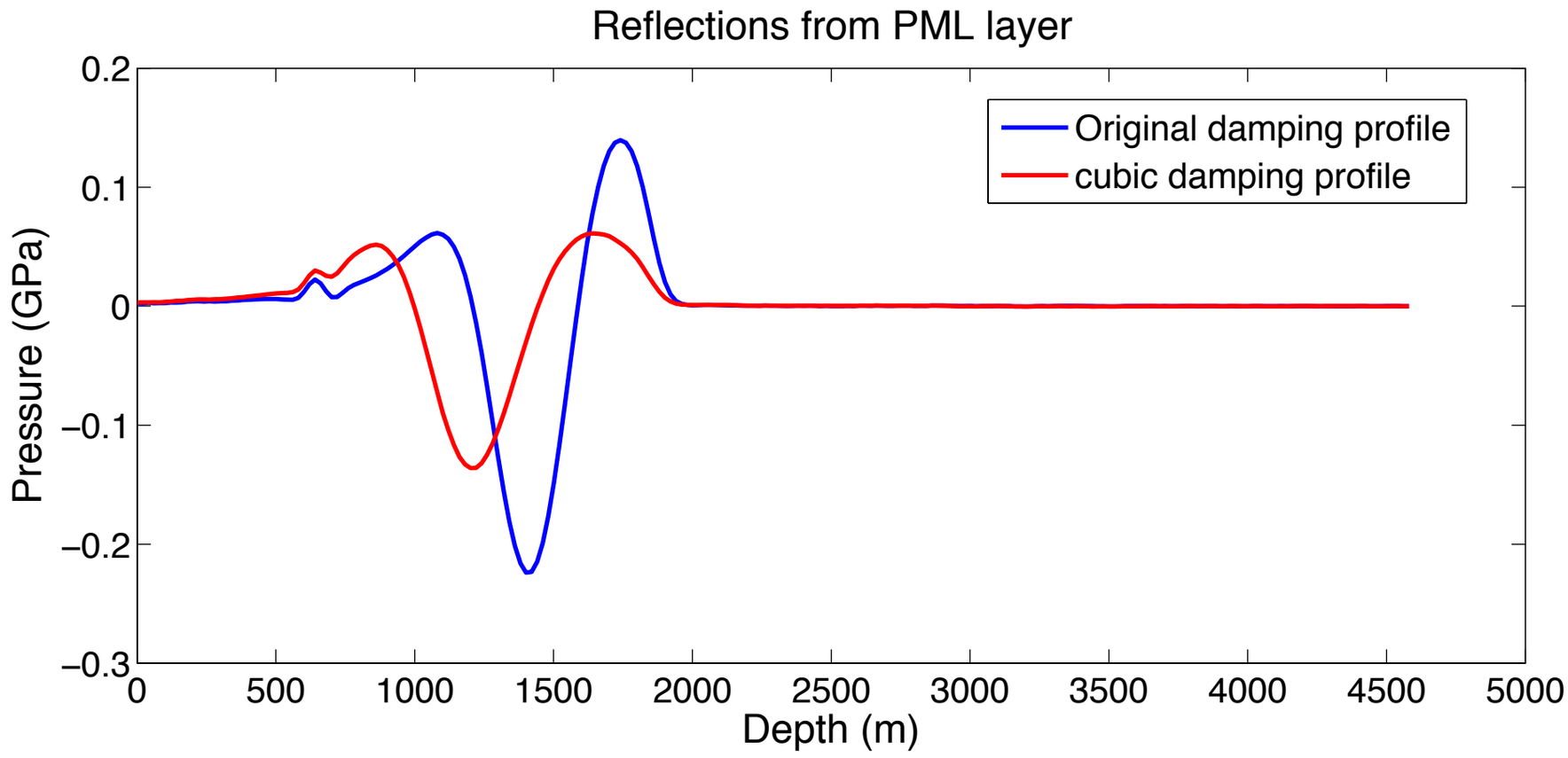
Frame differences: cubic is better



Difference of original damping profile at t=1.6s



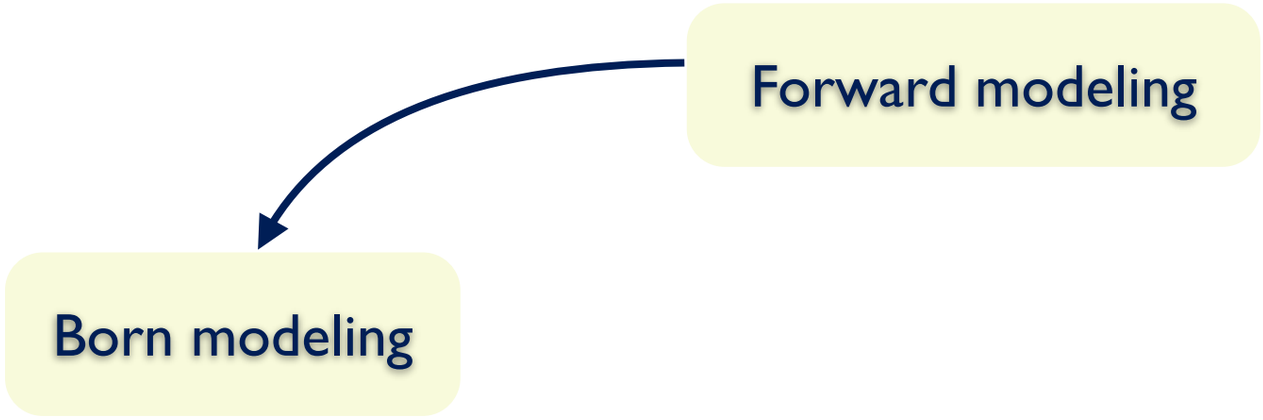
Difference of cubic damping profile at t=1.6s



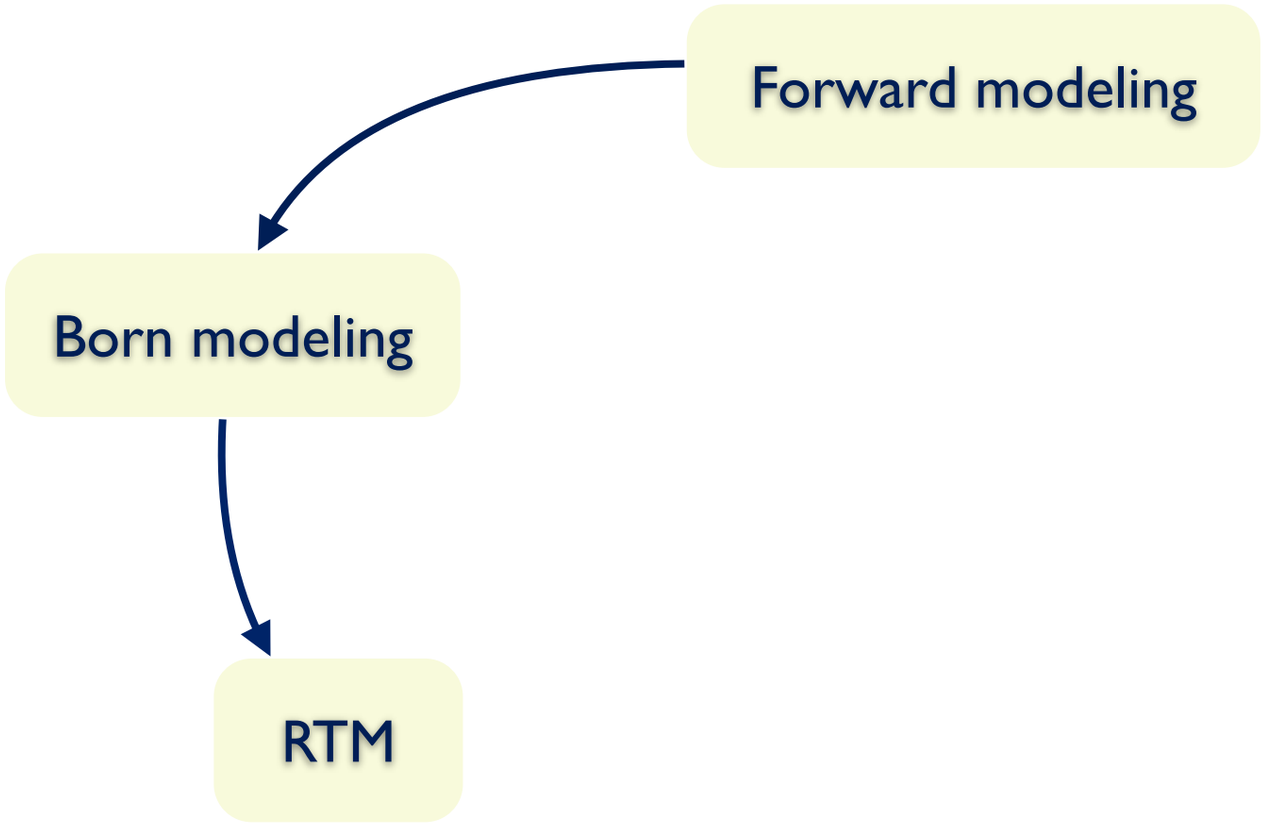
Typical steps after forward modeling

Forward modeling

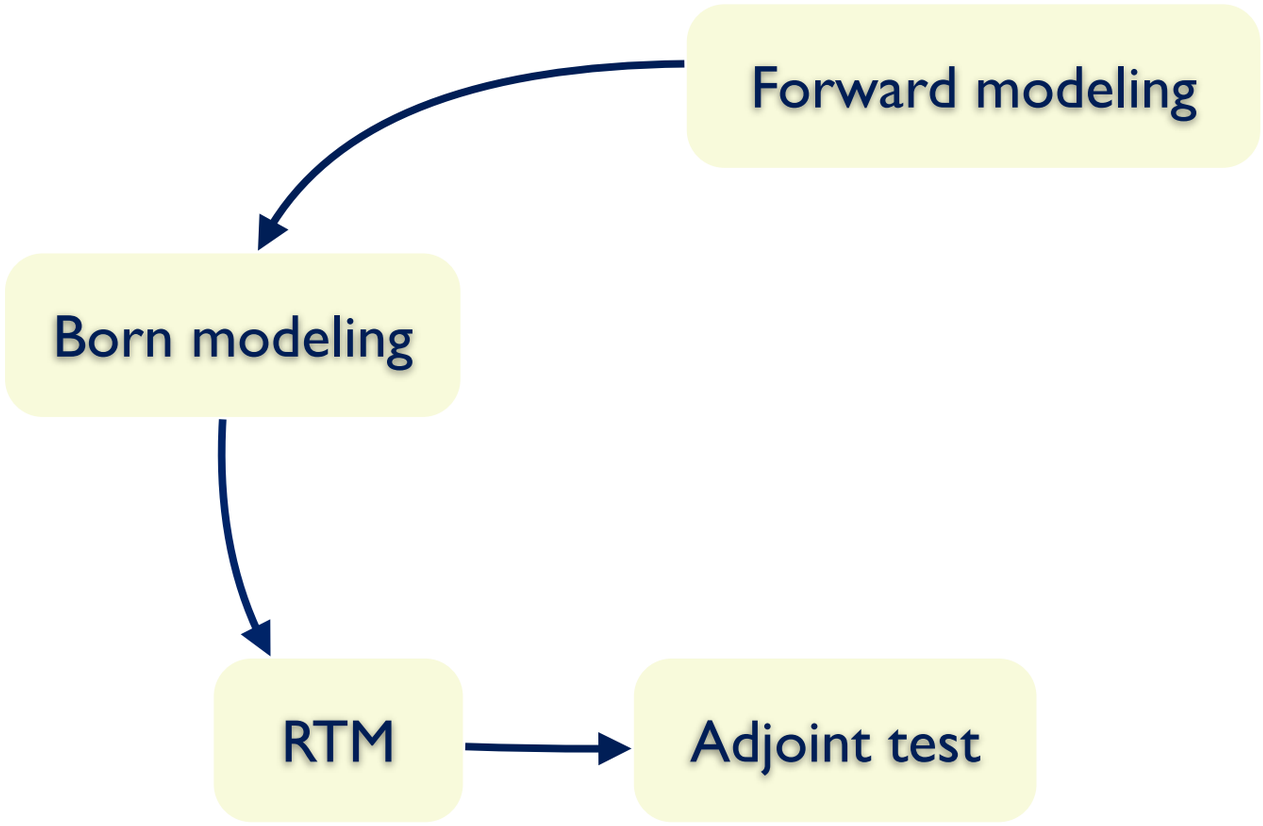
Typical steps after forward modeling



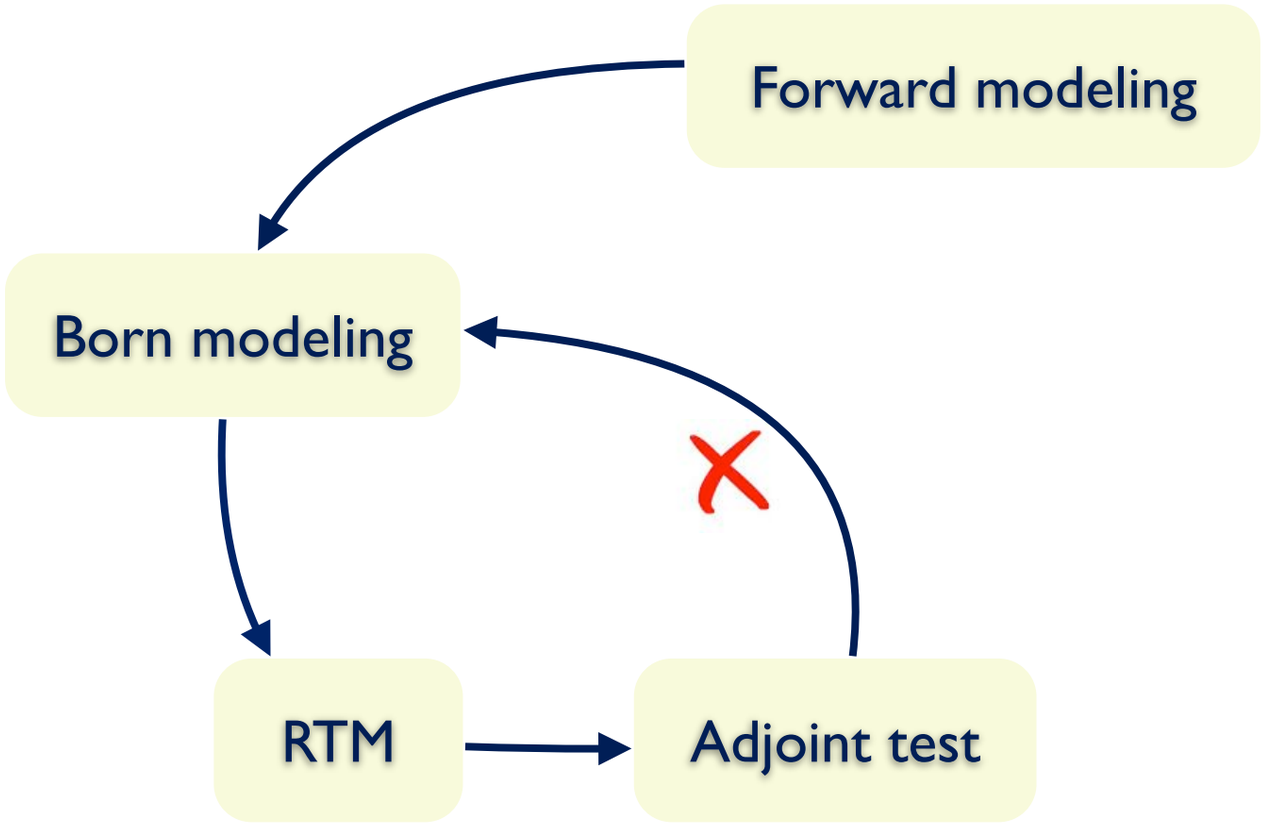
Typical steps after forward modeling



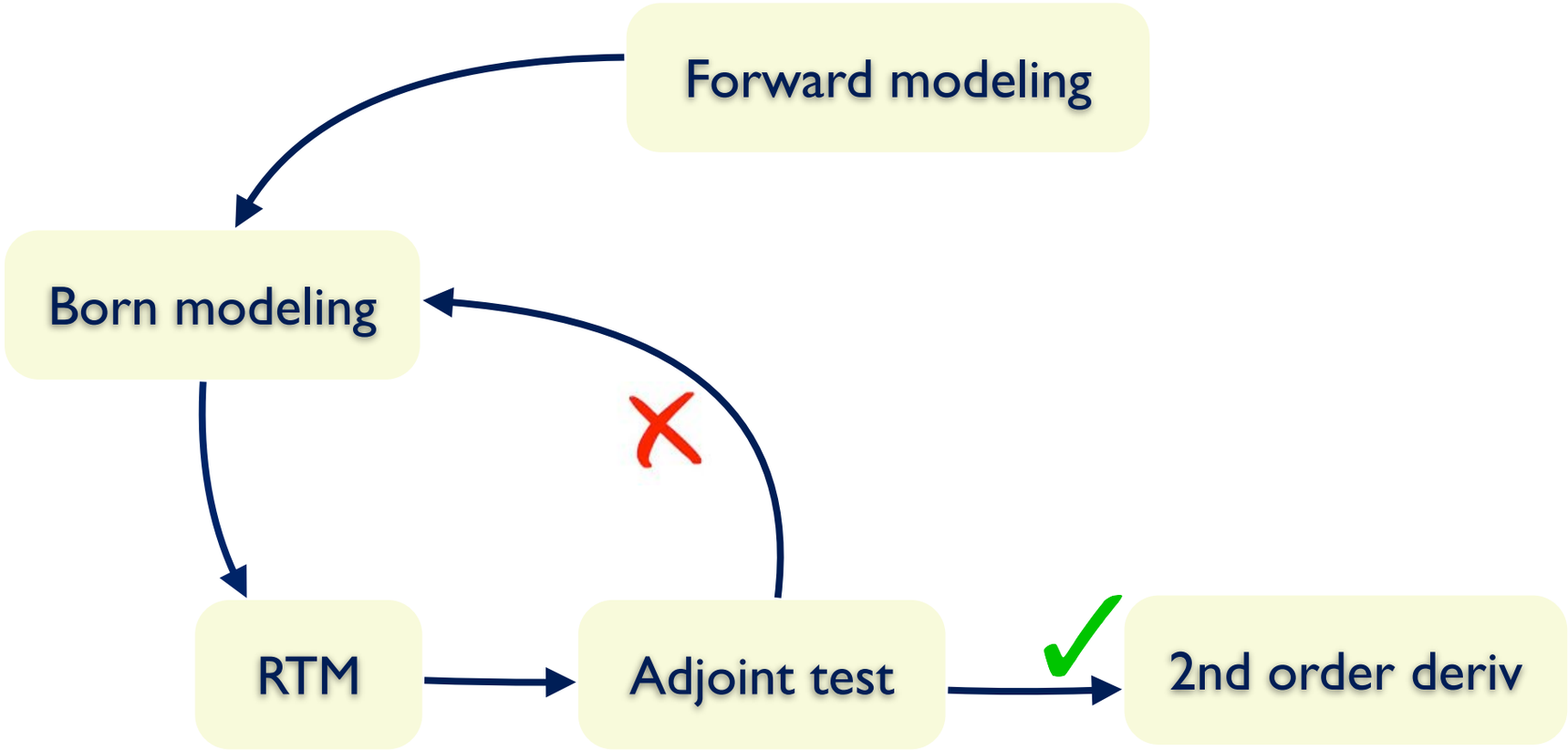
Typical steps after forward modeling



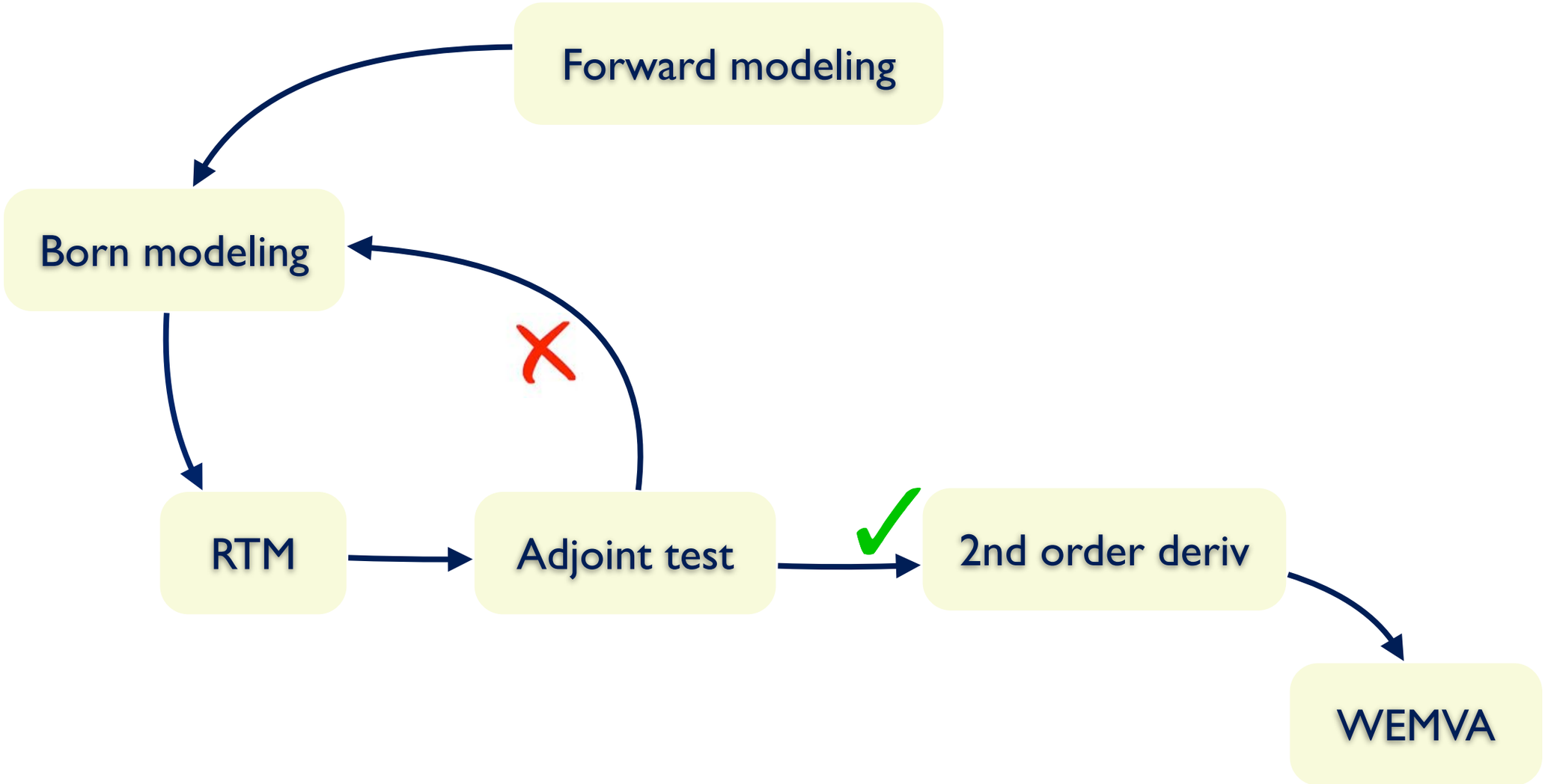
Typical steps after forward modeling



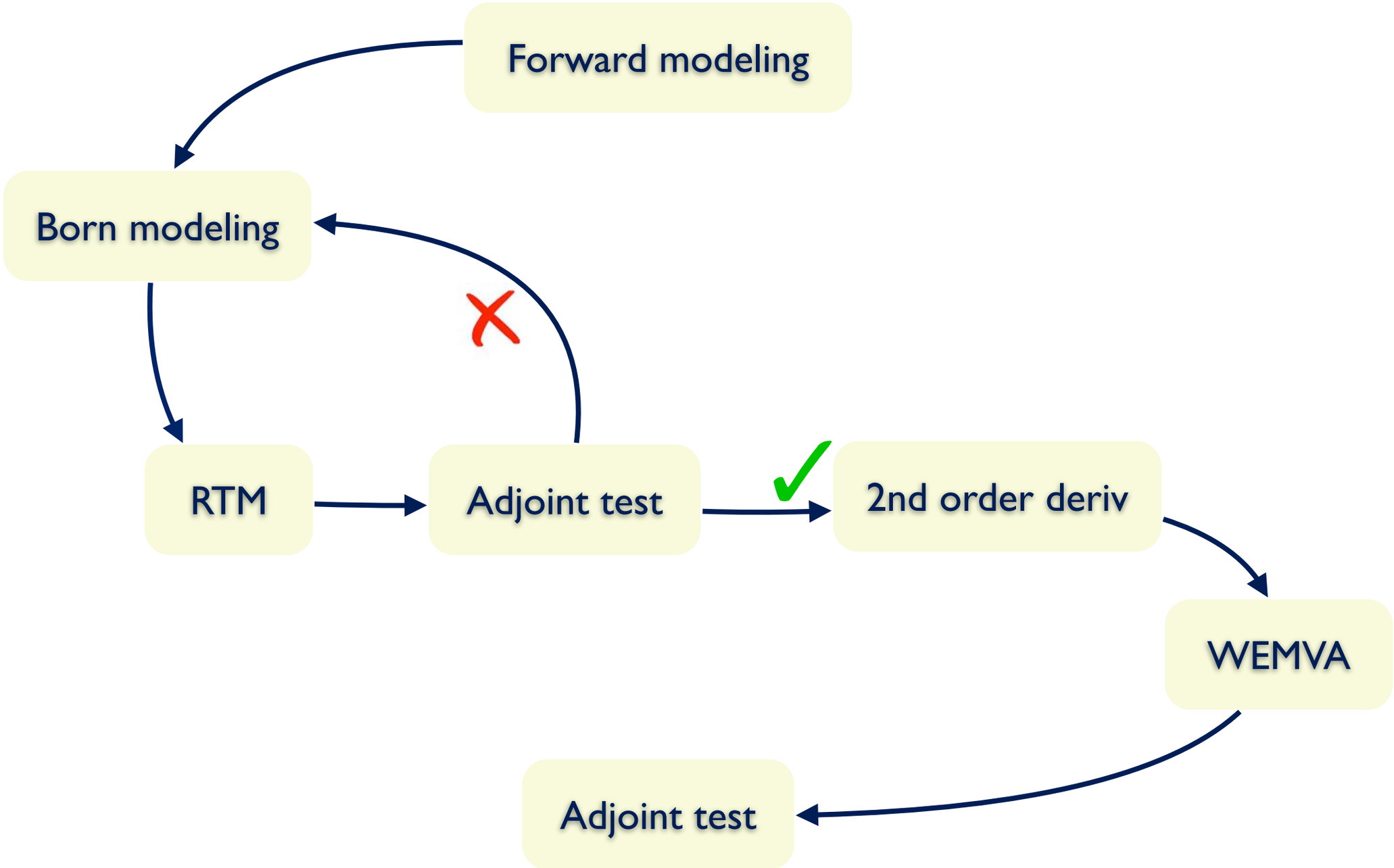
Typical steps after forward modeling



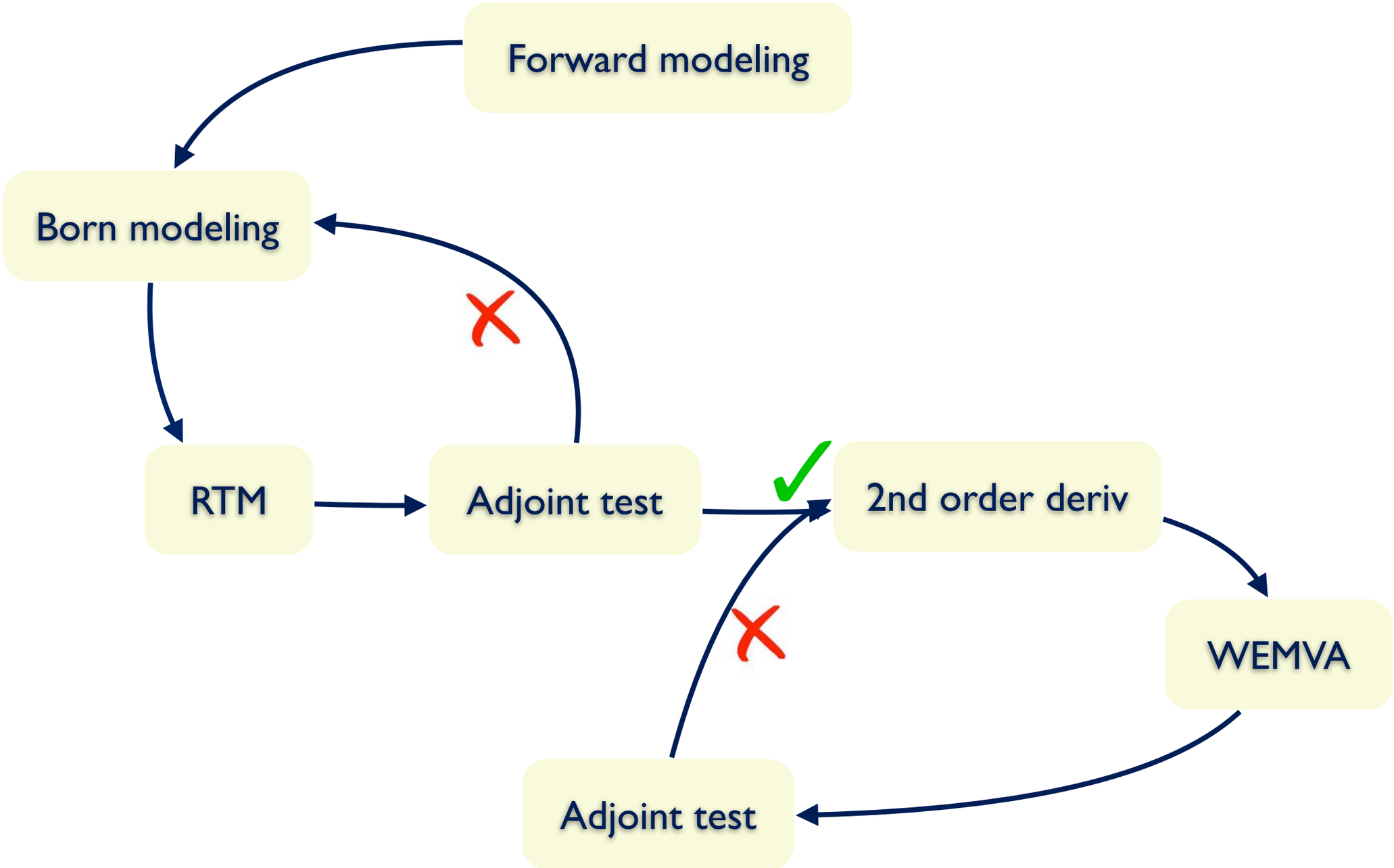
Typical steps after forward modeling



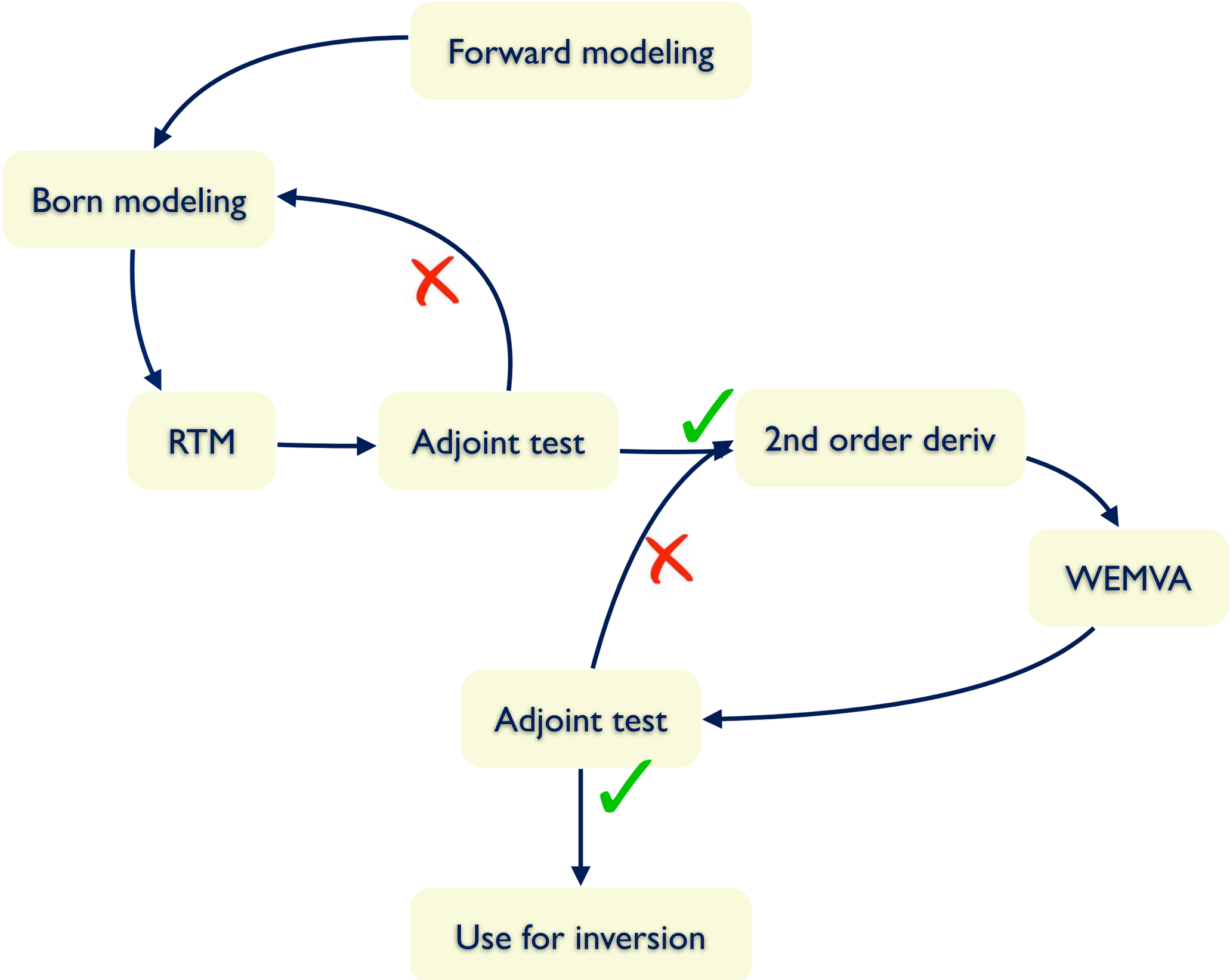
Typical steps after forward modeling



Typical steps after forward modeling

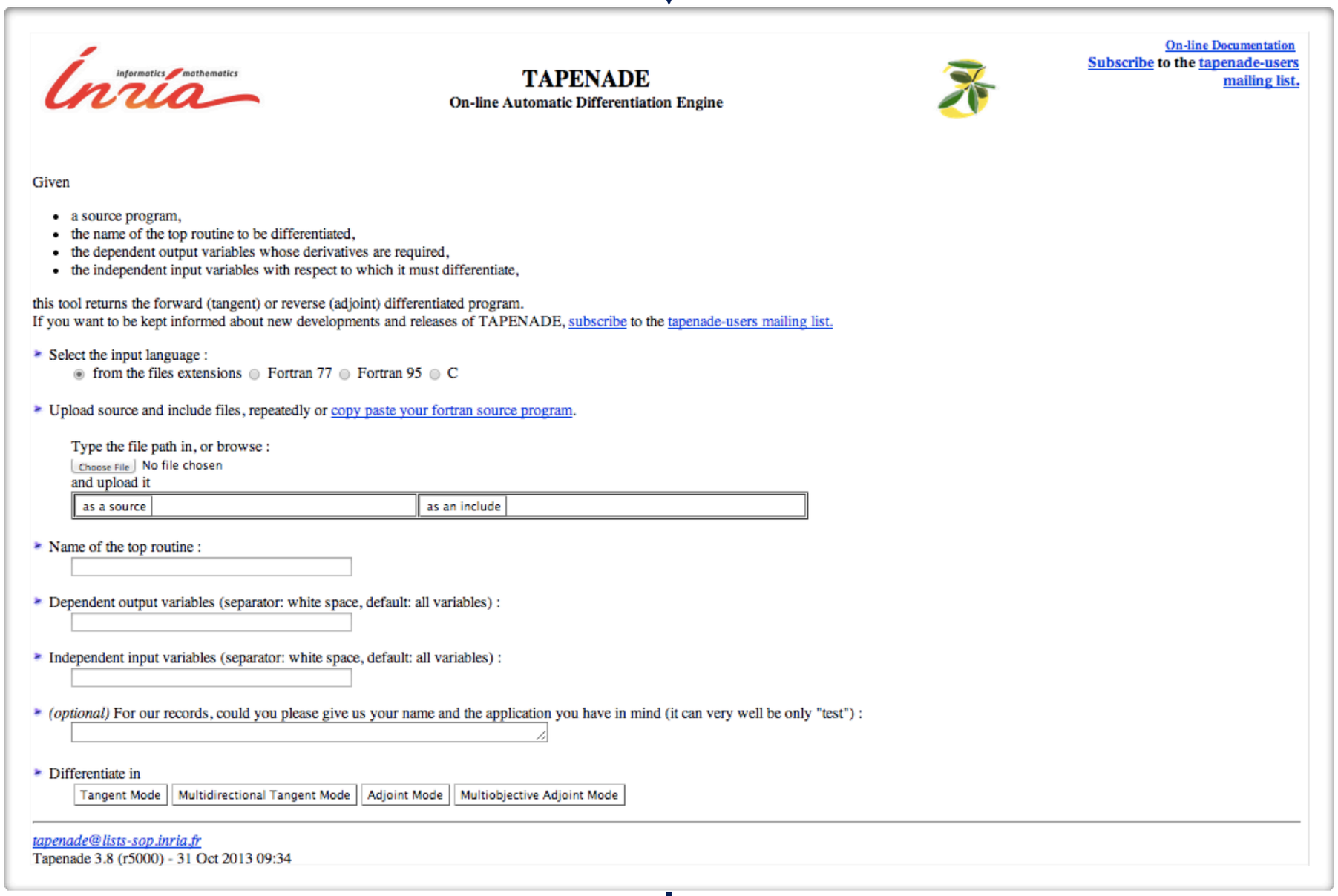


Typical steps after forward modeling



Use TAPENADE

Forward modeling



TAPENADE
On-line Automatic Differentiation Engine

On-line Documentation
Subscribe to the tapenade-users mailing list.

Given

- a source program,
- the name of the top routine to be differentiated,
- the dependent output variables whose derivatives are required,
- the independent input variables with respect to which it must differentiate,

this tool returns the forward (tangent) or reverse (adjoint) differentiated program.
If you want to be kept informed about new developments and releases of TAPENADE, subscribe to the tapenade-users mailing list.

Select the input language :
from the files extensions Fortran 77 Fortran 95 C

Upload source and include files, repeatedly or copy paste your fortran source program.

Type the file path in, or browse :
Choose File No file chosen
and upload it
as a source | as an include

Name of the top routine :

Dependent output variables (separator: white space, default: all variables) :

Independent input variables (separator: white space, default: all variables) :

(optional) For our records, could you please give us your name and the application you have in mind (it can very well be only "test") :

Differentiate in
Tangent Mode | Multidirectional Tangent Mode | Adjoint Mode | Multiobjective Adjoint Mode

tapenade@lists-sop.inria.fr
Tapenade 3.8 (r5000) - 31 Oct 2013 09:34

Use for inversion

Time step function and TAPENADE command

Time step function of $u_{tt} - c^2 \Delta u = 0$

```
1  // acd_2d.c
   void acd_2d(float **uc, float **up, float **csq,
3            int *s, int *e, float c0, float *c1) {
     int i0, i1;
5    for (i1=s[1]; i1<=e[1]; i1++) {
       for (i0=s[0]; i0<=e[0]; i0++) {
7        up[i1][i0] = 2.0*uc[i1][i0] - up[i1][i0] + csq[i1][i0]
          *(c0*uc[i1][i0]+c1[0]*(uc[i1  ][i0+1]+uc[i1  ][i0-1])
9                        +c1[1]*(uc[i1+1][i0  ]+uc[i1-1][i0  ]));
       }
11   }
```

Use TAPENADE to generate time step functions for Born modeling and RTM

```
1  tapenade -tangent -head acd_2d -vars "uc up csq" -outvars "uc up"
       acd_2d.c
   tapenade -reverse -head acd_2d -vars "uc up csq" -outvars "uc up"
       acd_2d.c
```

Meanings of TAPENADE command

Use TAPENADE to generate time step functions for Born modeling and RTM

```
tapenade -tangent -head acd_2d -vars "uc up csq" -outvars "uc up"
    acd_2d.c
tapenade -reverse -head acd_2d -vars "uc up csq" -outvars "uc up"
    acd_2d.c
```

- *-tangent* generate derivative code.
- *-reverse* generate adjoint code of the derivative.
- *-head* the name of the C function.
- *acd_2d.c* the name of the source file.
- *-vars* input variables with which the differentiation is taken.
- *-outvars* output variables whose derivatives are wanted.

2nd order derivative and WEMVA

The following C function declarations are generated by Tapenade previous commands.

```
void acd_2d_d(float **uc,  float **ucd,
              float **up,  float **upd,
              float **csq, float **csqd,
              int *s, int *e, float c0, float *c1);
void acd_2d_b(float **uc,  float **ucb,
              float **up,  float **upb,
              float **csq, float **csqb,
              int *s, int *e, float c0, float *c1);
```

Use TAPENADE one more time to get the time step function for 2nd order derivative and WEMVA

```
tapenade -tangent -head acd_2d_d -vars "uc up csq ucd upd" -outvars
    "uc up ucd upd" acd_2d_d.c
tapenade -reverse -head acd_2d_d -vars "uc up csq ucd upd" -outvars
    "uc up ucd upd" acd_2d_d.c
```

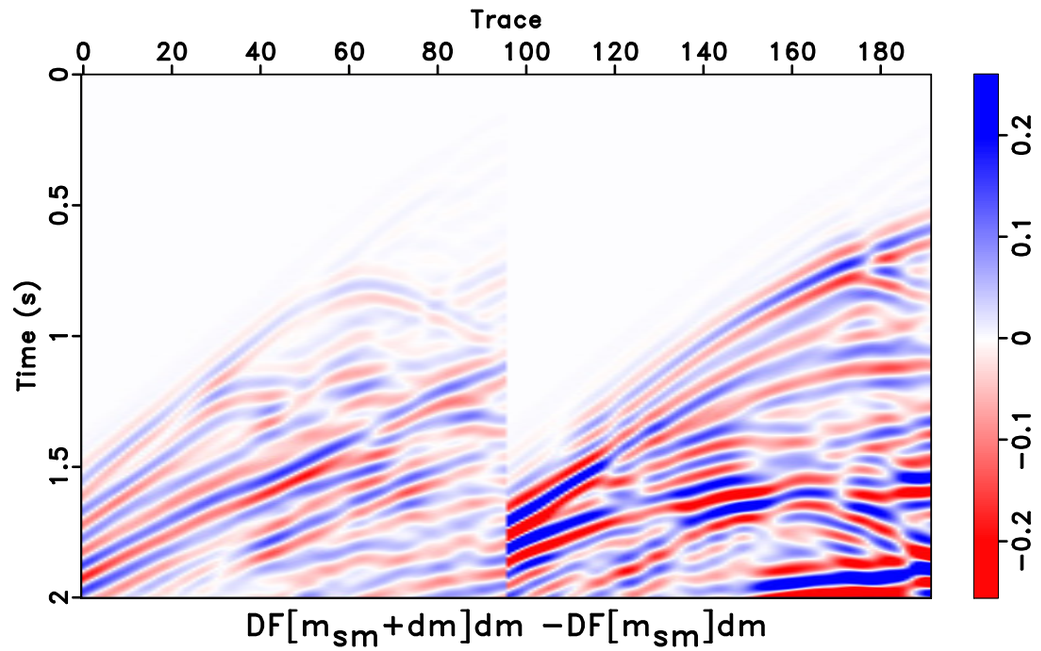
2nd order derivative accuracy test

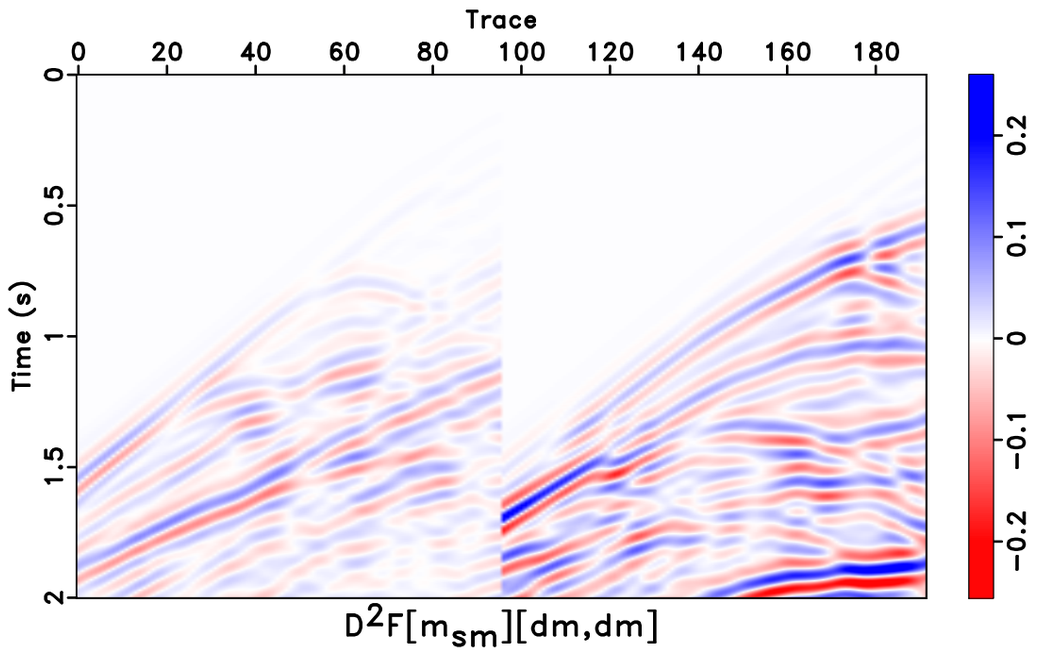
Comparison of

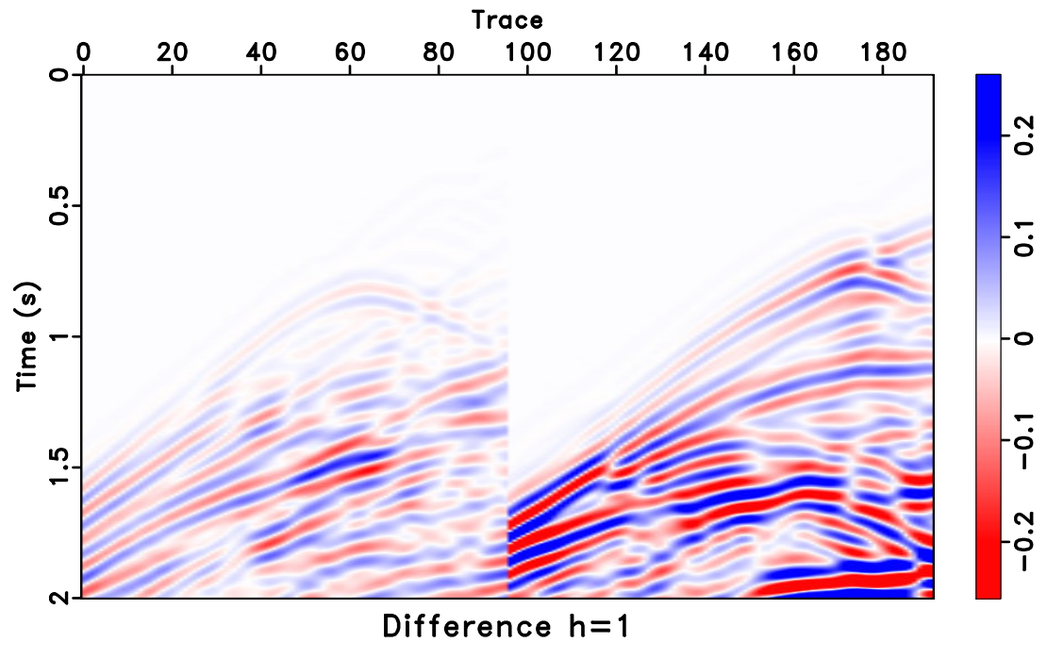
$$D^2F[m][dm, dm]$$

with

$$\frac{DF[m]dm - DF[m + h * dm]dm}{h}$$







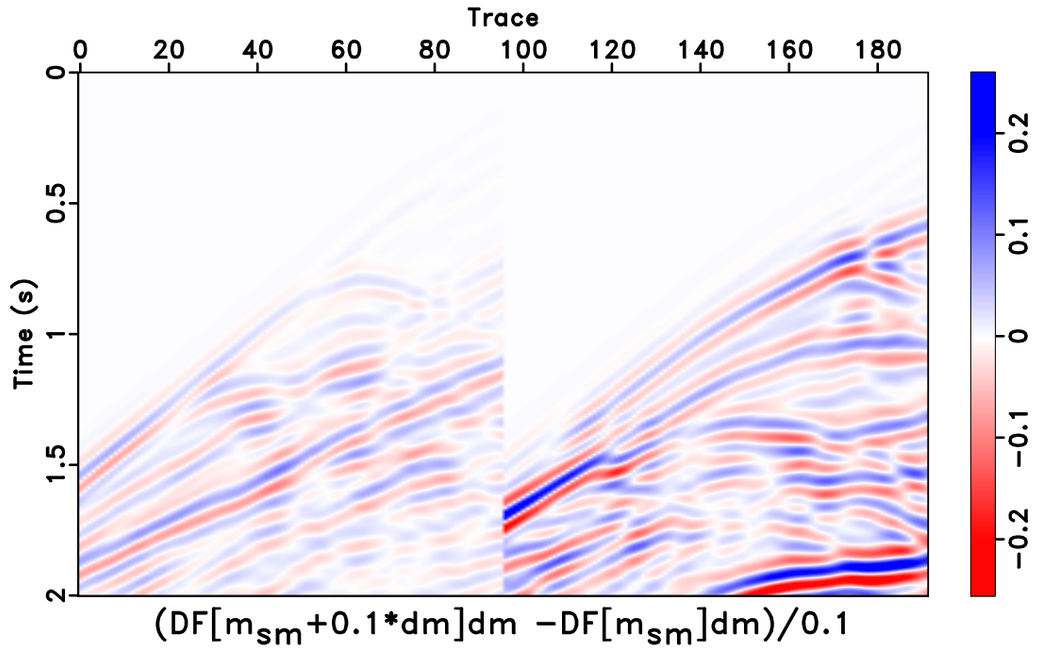
2nd order derivative accuracy test

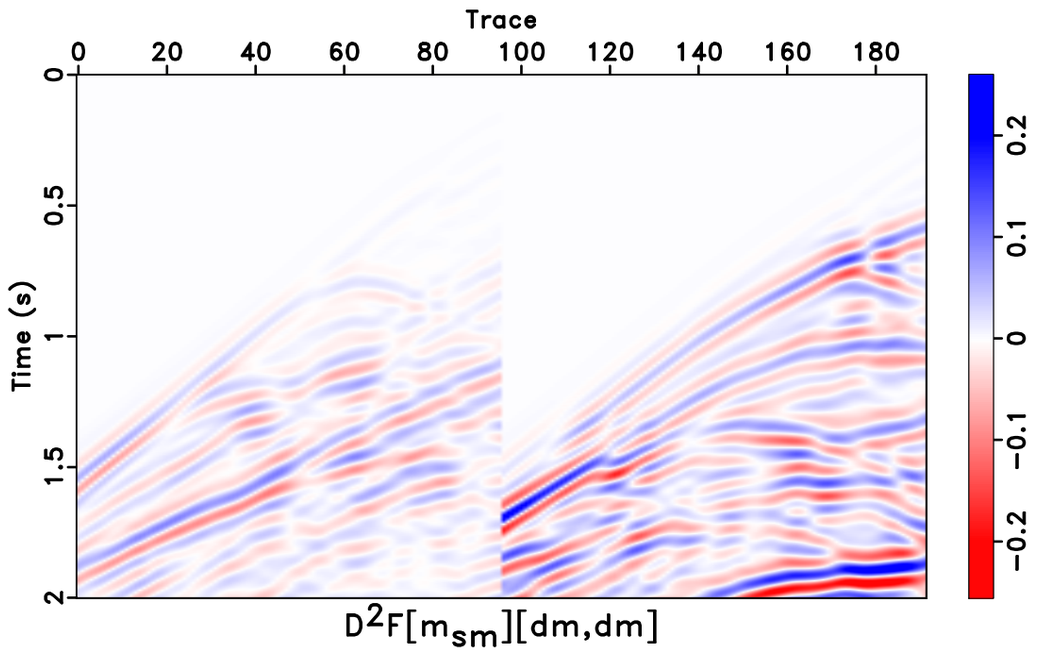
Comparison of

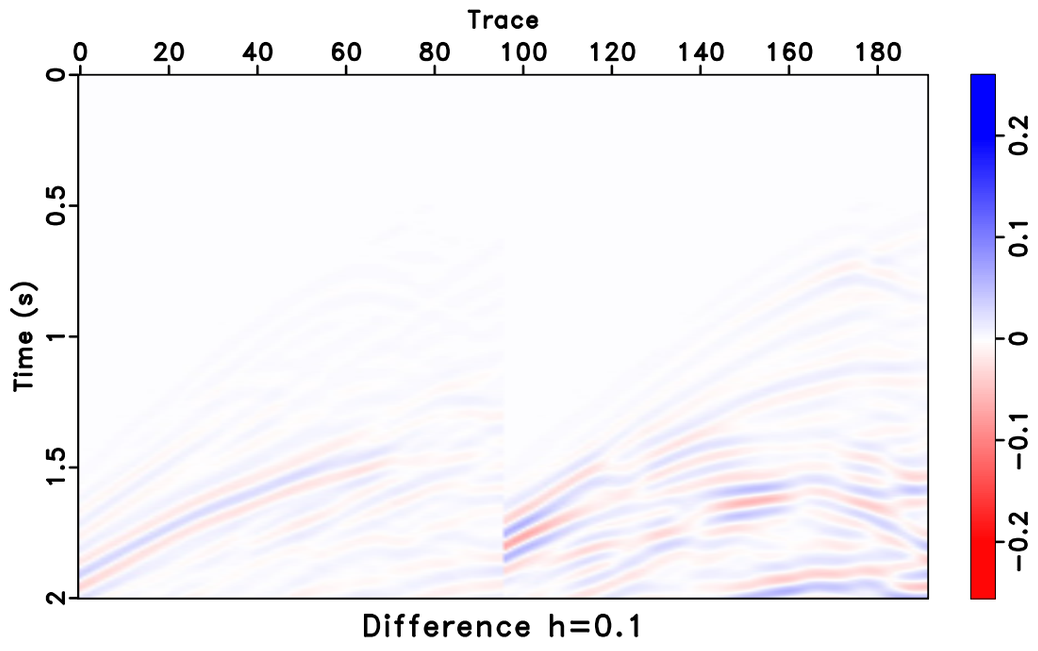
$$D^2F[m][dm, dm]$$

with

$$\frac{DF[m]dm - DF[m + h * dm]dm}{h}$$







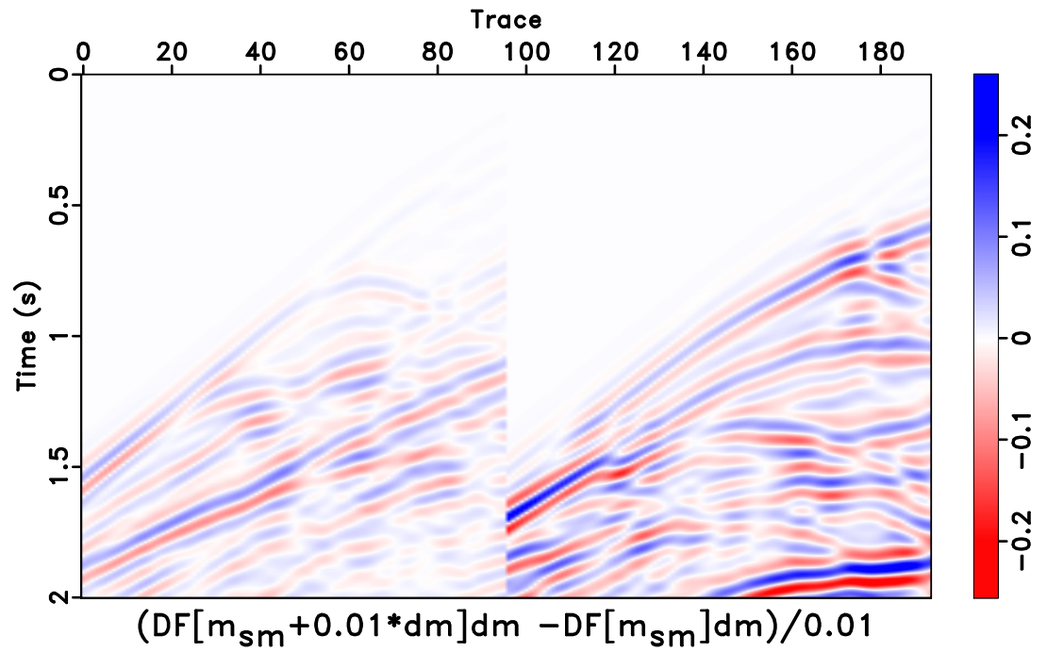
2nd order derivative accuracy test

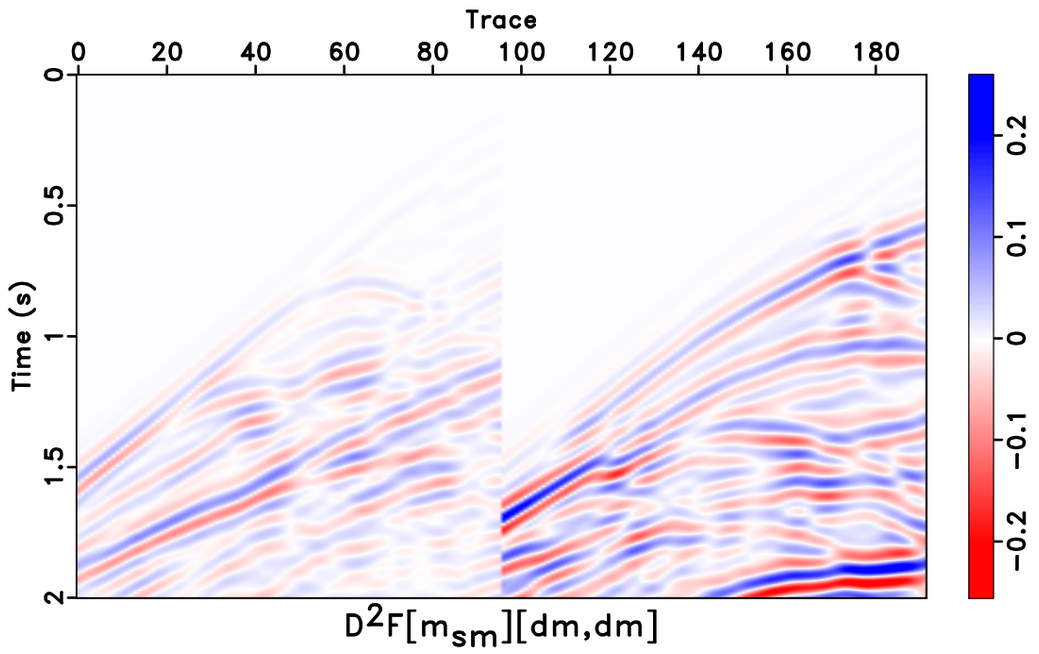
Comparison of

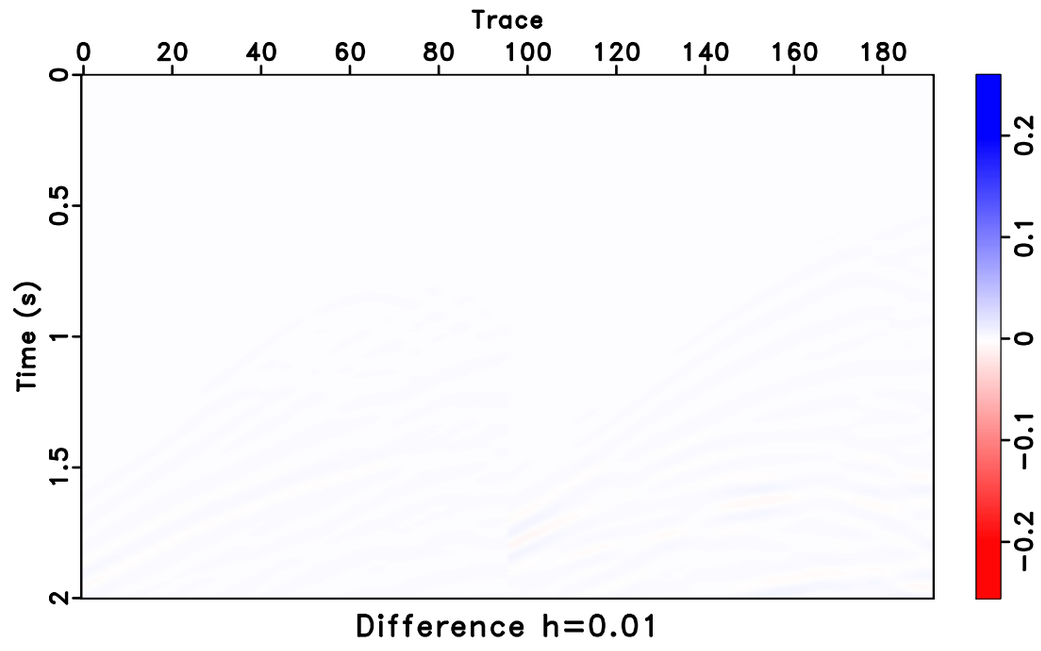
$$D^2F[m][dm, dm]$$

with

$$\frac{DF[m]dm - DF[m + h * dm]dm}{h}$$







Adjoint relation tests

The adjoint relation is satisfied when

$$\frac{|\langle Ax, y\rangle - \langle x, A^T y\rangle|}{\|Ax\|\|y\|} < 100 * macheps,$$

- $A = DF[m]$ for derivative order 1
- $A = D^2F[m][dm, \cdot]$ for derivative order 2
- $m = c^2$, $F[m] = u$ with u the solution of wave equations with or without PML.

Adjoint relation tests

Table : Adjoint relation results from running 100 time steps and *macheps*=1.19209290e-05. ACD means the naive acoustic constant density code. PML means the ACD code with PML boundary conditions

Derivative order	finite difference order	adjoint relation
ACD deriv order 1	2	5.96588334e-09
	4	2.86454878e-08
	8	2.43833931e-08
ACD deriv order 2	2	3.45298581e-08
	4	1.53316293e-08
	8	4.37524861e-08
PML deriv order 1	2	4.25377422e-09
	4	5.25341948e-09
	8	8.69006467e-09
PML deriv order 2	2	1.13406227e-08
	4	3.20660654e-09
	8	6.90543400e-09

These adjoint tests are running using IWAVE framework and RVL operator interface.

Conclusion

- Perfectly matched layer for second order wave equation;
- Improvement to Grote's PML method;
- How to use automatic differentiation tool: TAPENADE;
- Second order derivative code is correct;
- TAPENADE generated codes satisfy adjoint relation.

Acknowledgements

Great thanks to

- Wonderful audience;
- Current and former TRIP team members;
- Sponsors of The Rice Inversion Project.