TOY2DAC Manual

Romain Brossier[1], Ludovic Métiver[2,1]
romain.brossier@univ-grenoble-alpes.fr, ludovic.metivier@univ-grenoble-alpes.fr
[1] ISTerre, Univ. Grenoble Alpes, Grenoble, France
[2]LJK, CNRS, Univ. Grenoble Alpes, Grenoble, France

code version 2.6 - SVN trunk 18658 - May 2019

SEISCOPE Consortium

`http://seiscope2.osug.fr`

Seismic imaging of complex structures
from multicomponent global offset data by full waveform inversion
from multiparameter imaging to rock physics properties

Contents

Legal statement

Acknowledgements:
TOY2DAC and other SEISCOPE project's codes have been developed
in the framework of SEISCOPE and SEISCOPE 2 consortia, and we thank
the sponsors of these projects
SEISCOPE : BP, CGG-VERITAS, ENI, EXXON-MOBIL, PETROBRAS, SAUDI ARAMCO,
SHELL, STATOIL and TOTAL;
SEISCOPE2 (2013-2015) : BP, CGG, CHEVRON, EXXON-MOBIL, JGI, PETROBRAS, SAUDI ARAMCO,
SCHLUMBERGER, SHELL, SINOPEC, STATOIL, TOTAL and WOODSIDE
SEISCOPE2 (2016--) : CGG, CHEVRON, EXXON-MOBIL, JGI, SHELL, SINOPEC, STATOIL, TOTAL
 and WOODSIDE
TOY2DAC have also been partialy funded by the French research agency
(Agence Nationale de la Recherche) by project ANR-05-NT05-2-42427
 and is supported by Univ. Joseph Fourier Grenoble and CNRS.

References:
The use of TOY2DAC should be associated to the appropriate references in
publications
#forward modeling in isotropic media
Hustedt, B., Operto, S., and Virieux, J. (2004).

Mixed-grid and staggered-grid finite difference methods for frequency
domain acoustic wave modelling. Geophysical Journal International, 157:1269--1296.

#forward modeling in TTI media
Operto, S., Virieux, J., Ribodetti, A., and Anderson, J.E. (2009).
Finite-difference frequency-domain modeling of visco-acoustic wave
propagation in two-dimensional TTI media. Geophysics, 74 (5):T75--T95.

#truncated-Newton
Metivier, L., Bretaudeau, F., Brossier, R., Operto, S., and Virieux, J. (2014).
Full waveform inversion and the truncated Newton method: quantitative imaging
of complex subsurface structures. Geophysical Prospecting, 62:1353--1375.

Metivier, L., Brossier, R., Virieux, J., and Operto, S. (2013).
Full Waveform Inversion and the truncated Newton method.
SIAM Journal On Scientific Computing, 35(2):B401--B437.

#optimization tool-box
Metivier, L. and Brossier, R. (2016).
The seiscope optimization toolbox: A large-scale nonlinear
optimization library based on reverse communication.
Geophysics 81:F11-F25, 2016

Acknowledgments

TOY2DAC and other SEISCOPE project's codes have been developed in the framework of SEISCOPE and SEISCOPE 2 consortia and we thank the sponsors of these projects

SEISCOPE : BP, CGG-VERITAS, ENI, EXXON-MOBIL, PETROBRAS, SAUDI ARAMCO, SHELL, STATOIL and TOTAL;

Many thanks to sponsors supporting the SEISCOPE consortium (`http://seiscope2.osug.fr`) devoted to high resolution seismic imaging using full waveform : AkerBP, BP, CGG, Chevron, ExxonMobil, JGI, Petrobras, Saudi Aramco, Shell, Schlumberger, Sinopec, Statoil, Total and Woodside.

2018 sponsors: AkerBP, CGG, Chevron, ExxonMobil, JGI, Petrobras, Schlumberger, Shell, Sinopec, Statoil and Total.

2019 sponsors: AkerBP, CGG, Chevron, Equinor, ExxonMobil, JGI, Petrobras, Schlumberger, Shell, Sinopec, Sisprobe and Total.

TOY2DAC have also been partialy funded by the French research agency (Agence Nationale de la Recherche) by project ANR-05-NT05-2-42427 and is supported by Univ. Joseph Fourier Grenoble and CNRS.

Access to the high performance computing facilities of the Mesocentre SIGAMM (Observatoire de la Côte d'Azur, France) (`http://crimson.oca.eu/`), the Géoazur lab (University of Nice Sophia-Antipolis-CNRS, France), the Mesocentre CIMENT (University Joseph Fourier, Grenoble, France)(`https://ciment.ujf-grenoble.fr/`) and the GENCI- [CINES/IDRIS] (`http://www.genci.fr/`,`http://www.idris.fr/`,`http://www.cines.fr/`) provided the required computer resources to develop these packages.

1 Introduction

TOY2DAC package is a two-dimensional full waveform modeling and inversion code. It includes several modeling engine. In the current version only a frequency-domain visco-acoustic modeling/inversion in isotropic and VTI/HTI/TTI (in the current no tilted angle inversion) anisotropic media is implemented.

The frequency-domain modeling engine is based on a optimized mixed-grid finite-difference formulation of the second order visco-acoutsic wave equation. The user can refer to Hustedt et al. (2004); Operto et al. (2009) for details on the mathematical formulation and the implementation concerning the forward method used for visco-acoustic media in isotropic and VTI/TTI anisotropic media.

The full-waveform inversion scheme is based on an adjoint-formulation which uses the gradient of the misfit function as well as information on second-order derivatives to iteratively update the velocity models (or other parameters). The information on second-order derivatives is obtained through second-order adjoint techniques which yield an efficient algorithm to compute Hessian-vector products.

TOY2DAC makes use of the optimization routines provided in the SEISCOPE OPTIMIZATION TOOLBOX. Options in the fwi_input file allows to select a particular optimization scheme, from the steepest descent to the truncated Newton method.

2 TOY2DAC package content

The TOY2DAC package contains several directories and files that are described bellow :

- *bin* directory that contains the binary files after compilation.
- *doc* directory that contains this manual
- *include* directory that contains several files included during compilation. The user shouldn't have to modify this files.
- *src* directory that contains the source files of TOY2DAC . TOY2DAC is written in Fortran language (including also some C routines) with parallel MPI directives.
- *0LEGAL_STATEMENT* file that contains the licensing statement of the code, under a FreeBSD-like license, but with restrictions for diffusion.
- *0README* that contains basics how to run the code ...

3 Compilation/installation

3.1 Required environment and libraries

For compilation of TOY2DAC , an adapted informatics environment is required.

- Fortran and C compilers are required for compilation. TOY2DAC have already been tested with INTEL and PATHSCALE compilers.

TOY2DAC is compiled and linked with scientific libraries for solving large linear systems. Several libraries are therefore required :

- the MUMPS solver that perform LU decomposition of large sparse matrices, available at `http://graal.ens-lyon.fr/MUMPS/`. MUMPS required some ordering libraries as METIS and/or SCOTCH. We recommend to have the METIS library at least, available at `http://glaros.dtc.umn.edu/gkhome/metis/metis/download`. We recommend to use the last version of MUMPS. TOY2DAC should be linked (depending on compilation option) either with the complex simple precision or complex double precision version of MUMPS
- MUMPS solver requires to be linked with standard sequential and parallel linear algebra and communication libraries that are : SCALAPACK, BLACKS and LAPACK. The MKL library developed by INTEL includes all these libraries in a well compatible environment. The Netlib version (`http://www.netlib.org`) of these libraries can also be compiled, but a beginner user could have more problems to have a well compatible set of libraries.
- the BLAS library for matrix linear algebra operation done by MUMPS. A Netlib version can be used, available at `http://www.netlib.org/blas/`. However, this version is poorly slow. We recommend to use an optimized version of the BLAS library. Several implementations are freely available as ATLAS (`http://math-atlas.sourceforge.net/`). MKL includes also an efficient implementation of BLAS, well compatible with the other MKL libraries. Some proprietary implementations are also available as the IBM librairy (ESSL) on IBM machines.
- the METIS library, available at `http://glaros.dtc.umn.edu/gkhome/metis/metis/download`.

3.2 TOY2DAC compilation

The main part of TOY2DAC is located in *src*. A *Makefile* allows to compile the source. Before compiling, the user must put a *Makefile.inc* file at the same level than the *Makefile*. This *Makefile.inc* incudes the definition of the compilers, compilation options, libraries and includes locations. The variables of the *Makefile.inc* that MUST be filled are :

- *CC, FC, FL* are the definition of the C compiler, Fortran compiler and linking call, respectively. Example with INTEL compiler with OPENMPI wrappers:

```
CC =  mpicc
FC =  mpif90
FL =  mpif90
```

- *OPTF, OPTC, OPTL, OPTFF* are the options defined for Fortran compilation (Fortran 77 with fixed format), C compilation, linking and Fortran 90 compilation (free format), respectively. Please note that the variable *OPT_PRE* MUST be included in the *OPTF* and *OPTFF* lines. Example for INTEL :

```
OPTF =  -O3  -assume byterecl  $(OPT_PRE)
OPTC =  -O3
OPTL =  -O3  -assume byterecl
OPTFF = ${OPTF}
```

- *OPT_PRE* is the preprocessing flag. Currently, only one preprocessing flag is implemented in the code : `DOUBLEPRECISION`, allowing to compile a version of TOY2DAC coupled with the double precision version of MUMPS.

  Example to compile the code with single precision : no flag:

  ```
  OPT_PRE = -cpp
  ```

  Example to compile the code with double precision : put the flag

  ```
  OPT_PRE = -cpp  -DDOUBLEPRECISION
  ```

- *INC* is all the include paths used by the code. It must include the path to *include* directory of the code, the include directory of MUMPS package, the include to the SEISCOPE tool-box and eventually (if you don't use MPI wrappers) the include directory of your MPI implementation.

- *LIB* defines all the libraries used at linking : BLAS, LAPACK, SCALAPACK, BLACS, METIS, MUMPS, (MPI). Example of both *INC* and *LIB* for INTEL MKL :

  ```
  #ROOT DIR OF MUMPS/METIS
  LADIR  = /home/brossier/LinAlg_MKL/

  #ROOT DIR OF TOOL BOX
  LTOOLS_BOX = /scratch/brossier/TOOLS_BOX/trunk

  #MKL LIBRARIES
  LMKL= -L ${mkl_DIR}/lib/intel64 -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
  -lmkl_blas95_lp64 -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_lapack95_lp64
  -lm -openmp -lpthread

  #MUMPS LIB AND INC
  LMUMPS = -L$(LADIR)/MUMPS/lib -lcmumps -lmumps_common
  IMUMPS = -I$(LADIR)/MUMPS/include

  #PORD LIB (INSIDE MUMPS)
  LPORDDIR = $(LADIR)/MUMPS/PORD/lib/
  LPORD    = -L$(LPORDDIR) -lpord

  #METIS LIB
  LMETISDIR = $(LADIR)/METIS/
  LMETIS    = -L$(LMETISDIR) -lmetis
  ```

```
#TOOL BOX LIB AND INC
LOPTIM = -L $(LTOOLS_BOX)/OPTIMIZATION/lib -lSEISCOPE_OPTIM
IOPTIM  = -I $(LTOOLS_BOX)/OPTIMIZATION/COMMON/include

#WE GATHER EVERYTHINGS
LIBPAR = $(LMUMPS) $(LPORD) $(LOPTIM)  $(LMETIS)  $(LMKL)
INCPAR = $(IMUMPS) $(IPORD) $(IOPTIM) -I../include

INC = $(INCPAR)
LIB = $(LIBPAR)
```

The *EXT* variable can also be filled optionally. That defines the extension that is placed at the end of the binary names. Example :

```
EXT= _intel03
```

Once the *Makefile.inc* is filled, the user can compile the codes with a simple make command :

```
make
```

The binary files created at compilation are automatically put in the *bin* directory.

4 Theory

The problem addressed by TOY2DAC is the following: starting from an initial guess m_0, solve the minimization problem

$$\min_m \frac{1}{2} \sum_{s=1}^{S} \|Ru_s(m) - d_s\|^2 \tag{4.1}$$

where $u_s(m)$ is the solution of the frequency-domain wave propagation problem

$$A(m)u_s = \varphi_s \tag{4.2}$$

and

- m denotes the subsurface parameters to be recovered
- S denotes the total number of data-sets
- d_s denotes the sth data-set
- R is a restriction operator mapping the wavefield $u_s(m)$ to the receivers location

The operator $A(m)$ stands for the 2D frequency-domain acoustic (an)isotropic wave propagation operator. In the isotropic case, (4.2) may be written as

$$\omega^2 u + \rho v_P^2 \mathrm{div}\left(\frac{1}{\rho}\nabla u\right) = \varphi_s \tag{4.3}$$

where ω is the circular frequency, $v_P(x)$ the pressure-wave velocity, $\rho(x)$ the density. In this case, the subsurface parameter m may be

- $v_P(x)$: mono-parameter inversion of the pressure-wave velocity
- $\rho(x)$: mono-parameter inversion of the density
- the collection $[v_P(x) \;\; \rho(x)]^T$: multi-parameter inversion of the pressure-wave velocity and the density

The first case is the more common. Note also that others parameters are also possible, as mentioned in the following.

4.1 Forward problem

The Forward problem consists in the solution of equation (4.2). In TOY2DAC , a fourth-order discretization scheme is implemented (more details in Hustedt et al., 2004; Operto et al., 2009, for the discretisation strategy relying on optimized mixed-grid stencil with anti-lumped mass for the isotropic and TTI cases), and the solution equation (4.2) amounts to the solution of a sparse large scale linear system. The direct solver MUMPS, which implements a multi-frontal LU decomposition of the resulting matrix, is used to solve this system. This is particularly interesting when S becomes large: only one LU decomposition is required, and the S linear systems can be solved through forward and backward substitution.

4.2 Inverse Problem

The solution of the inverse problem (4.1) is based on local descent algorithms. From an initial guess m_0, a sequence of subsurface models m_k is built such that

$$m_{k+1} = m_k + \alpha_k \Delta m_k \tag{4.4}$$

where $\alpha_k \in \mathcal{R}$ is a scaling parameter computed through a linesearch process and Δm_k is a descent direction. This descent direction depends on the choice made for the optimization algorithm. The most basic choice is

$$\Delta m_k = -\nabla f(m_k) \tag{4.5}$$

which corresponds to the steepest-descent direction. More elaborated algorithms, such as the ones proposed in the SEISCOPE OPTIMIZATION TOOLBOX, approximate the influence of the inverse Hessian matrix $\nabla^2 f(m)$ through an estimation Q_k and use descent directions of the form

$$\Delta m_k = -Q_k \nabla f(m_k) \tag{4.6}$$

More details can be found in the SEISCOPE OPTIMIZATION TOOLBOX documentation (Métivier and Brossier, 2016). TOY2DAC implements the truncated-Newton method (Métivier et al., 2013, 2014)

4.3 Frequency-selection strategy

CPU-efficient frequency-domain FWI is generally carried out by successive inversions of single frequencies, by proceeding from the low frequencies to the higher ones (Pratt and Worthington, 1990; Sirgue and Pratt, 2004). This defines a multiresolution framework that mitigates of the nonlinearity of the inverse problem associated with high frequency cycle-skipping artefacts. CPU-efficient algorithms can be designed by selecting a few coarsely sampled frequencies, such that the wavenumber redundancy, which results from dense sampling of frequencies and aperture angles, is decimated. This strategy has proven to be effective for several applications of acoustic FWI (Ravaut et al., 2004; Operto et al., 2006; Brenders and Pratt, 2007). TOY2DAC included a simple framework that run only one group of frequencies. If multiple groups are required, TOY2DAC must be launched several times.

5 Input files for frequency-domain finite-difference modeling engine

5.1 All applications

5.1.1 Medium parameters files

The medium parameters files are BINARY files (direct access simple precision real numbers) that includes the physical properties of the medium. It is the medium for simulation in modeling mode or the starting models for inversion. The names of the files should be put in the input files (described in the following). The required parameters are

- P-wave velocity, P-wave quality factor, Density for isotropic visco-acoustic simulations
- P-wave velocity, P-wave quality factor, Density, Thomsen parameter ϵ, Thomsen parameter δ, Titt angle θ for anisotropic visco-acoustic simulations

These file should contain $nz \times nx$ values.

5.1.2 Acquisition file

The acquisition files used in TOY2DAC have the same format that the other SEISCOPE 2D codes. This is ASCII files built as follows :

```
300      300      0        0        0
300      1700     0        0        1
350      1700     0        0        1
350      300      0        0        0
300      1700     0        0        1
350      1700     0        0        1
 ......
```

In each line, the first value is the z coordinate of the source or the receiver, the second value is the x coordinate, the third value is the first arrival time or the 0 value for the receivers (always 0 for sources). The 4^{th} value is not used in TOY2DAC , but should include a value (real), and the 5^{th} value is the code (integer) that specifies a source (0) or a receiver (1). The acquisition file as the following structure :

```
line for source 1
line for receiver 1 of source 1
line for receiver 2 of source 1
line for receiver 3 of source 1
line for source 2
line for receiver 1 of source 2
line for receiver 2 of source 2
line for source 3
 ......
```

The number of receivers per source is not required to be constant. However, if a receiver is common to several sources, it should be written after each associated source.

5.2 *mumps_input* file

The *mumps_input* allows to manage some tuning option for the MUMPS solver used in TOY2DAC . For historical reasons, this files includes 4 parameters. The *mumps_input* is built as follows:

```
7         ! icntl_7= choice of ordering (default 7->automatic choice)
60        ! icntl_14=percentage of increasing of estimated workspace for facto (default=20
760       ! icntl_23=MEMORY USED PER PROC
8         ! keep_84 =  blocking factor for Multiple RHS
```

- The *icntl_7* parameter allows to choose the ordering algorithm used by MUMPS. We strongly recommend the user to refer to the MUMPS documentation for details on MUMPS options. By default, we recommend to use the 7 value that most of the time chooses the METIS algorithm for our matrices.

- The *icntl_14* parameter allows to tune the relaxation factor for MUMPS intern tables. We strongly recommend the user to refer to the MUMPS documentation for details on MUMPS options. This option allows that MUMPS allocates a bigger table that estimated during analysis (roughly: size = $(1 + icntl\_14/100)\times$ estimated size). If MUMPS crashes and asks to increase the value of *icntl(14)*, the user should increase the value of this parameter. The standard values found in our applications are in the range [40;300]. ATTENTION, no more really used since MUMPS version 4.9.X

- The *icntl_23* parameter gives the allowed memory in Mb per MPI process during factorization (used since MUMPS version 4.9.X). This flag is really important as it determine the size of the table allowed on each MPI process to store the LU factors.

- The *keep_84* parameter allows to choose the number of shots (right hand side terms of the linear system) solved simultaneously with MUMPS. MultiRHS solving allows to take benefit of BLAS3 routines, more efficient that BLAS2 due to cache optimization. However, big values required more memory consumption (only in fwt2d_elas code). We recommend to use a maximum value of 16, which often works fine. Smaller values of 8 or 4 can also be used.

5.3 *toy2dac_input* file

The *toy2dac_input* is the main file for tuning TOY2DAC . An example is provided bellow, before explaining each parameter :

```
1          ! mode of the code
1          ! forward modeling tool 1->fdfd iso 2-> fdfd aniso
acqui_tomo ! acquisition file name
```

1. *mode of the code* allows to choose between frequency modeling (0), FWI (1), RTM (not-working) (2), MVA (not-working) (3) or time-domain modeling (not-working) (4)
2. *forward modeling tool* allows to choose the type of modleing engine. Currently, mode 1 provide a frequency-domain finite difference in isotropic media, mode 2 provide a frequency-domain finite difference in TTI anisotropic media.
3. *acquisition* is the name of acquisition file

5.3.1 *fdfd_input* file

The *fdfd_input* is the main file for tuning the frequency domain finite-difference modeling engine of TOY2DAC . An example is provided bellow, before explaining each parameter :

```
101 101                                ! nz,nx
20.                                    ! h
vp_homo qp rho epsilon0 delta theta    ! files name
90. 10                                 ! pml coef & npml
0                                      ! Hicks interpolation (0 NO, 1 YES)
0                                      ! free surface (0 NO, 1 YES)
0 0      ! itypes (0:explosion/1: vertical force) ityperec (0: hydrophone/1: vertical geop
0.      ! slaplace (imaginray part of frequency)
```

1. *nz,nx* are the number of points of the FD grid in z and x directions
2. h is step-size of the FD grid (common in both directions)
3. *files name* are the name of input model files in this order for isotropic

```
Vp, QP, rho
```

and for anisotropic

```
Vp, QP, rho, epsilon, delta, theta
```

4. *pml tunning* with the value of the pml amplitude (90 is a good pragmatical value) and the number of FD grid point in PML (10 is a good pragmatical value)
5. *Hicks interpolation* is an additional parameter to activate the interpolation from Hicks (Hicks, 2002) when sources and/or receivers are not located on grid points.
6. *free surface* is an additional parameter to impose a free-surface boundary condition on the top side of the medium
7. *itypes* is the type of sources: explosion (0) or vertical force (1). *ityper* is the type of receivers: hydrophone (0) (or vertical geophone (1), but not yet implemented). This option is possible only when Hicks interpolation is used.
8. *slaplace* is the Laplace constant, i.e., the imaginary part of frequency used to exponentially damp the seismic wavefield with time (Shin and Cha, 2008; Shin and Ha, 2008).

5.3.2 *freq_management* file

The file *freq_management* allows to manage the frequencies used during modeling or inversion. An example is provided bellow, before explaining each parameter :

```
4
2.56598234  3.05474091 3.54349947 4.03225803
```

1. the first line indicates the number of frequencies involved in the simulation or the FWI frequency group (recall : TOY2DAC work with only one frequency group)
2. the second line indicates the list of frequencies, in Hz

5.4 Specific to FWI

5.4.1 Bathymetry file

The bathymetry file is a direct access binary file (size nx) that contains the depth, in meters, of the sea floor. All the grid point above this limit are not involved during inversion. For land data, this bathymetry file can be used to describe a fixed zone during inversion

5.5 Data weighting file

The data weighting file is a ascii file. It is a 1D amplitude file that describes the weight to apply to the data, according to the source-receiver distance. The size and step length (assumed regular) are specified in `fwi_input`). According to real source-receiver distance, the value of weighting is interpolated. If the file is not long enough, according to the source-receiver distance, the last value of the file is considered

5.5.1 *fwi_input* file

The *fwi_input* is the main file for tuning the full waveform inversion of TOY2DAC . An example is provided bellow, before explaining each parameter :

```
data !name of obs data
1          ! family
2 1 2      ! npar & inverted para 1->vp 2->rho 3->qp 4->eps 5->delta 11->log(vp) 12->log(rh
0          ! src estimation 0->no 1->1 mean value 2->one per gather
fbathy     ! bathy file
0.         ! deadzone
5          ! optimization method 1:STD  2:PSTD  3:NLCG  4:PNLCG  5:LBFBGS   6:PLBFGS  7:TGN
1e-4       ! convergence criterion
1e-4       ! convergence criterion model
5          ! maximum number of nonlinear iterations
20         ! memory parameter for l-BFGS
3          ! maximum number of inner CG iterations for truncated Newton
1e-2       ! threshold parameter for the preconditioner
0          ! debug option for the optimization routines
0. 0.      ! lambda for Tikhonov regularization
1. 0.5     ! lambda_x, lambda_z (directional weight for Tikhonov regularization)
0.         ! prior information regularization weights
1          ! bound constraints
4000.      ! upper bound
1000.      ! lower bound
1.         ! tolerance for bound constraints
data_weight !file for data weighting
2 2000.    ! number of sample and space step of the weighting file
```

1. *name of obs data* is the name of the observed data file
2. *family* is the parametrization family choice. Currently only the choice 1 is available.
3. *npar and inverted para* number of inverted parameters and associated inverted parameter. For family 1, we have the following choices
   - 1 for P-wave velocity V_p
   - 2 for density ρ
   - 3 for quality factor Q_p
   - 4 for Thomsen parameter ϵ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
   - 5 for Thomsen parameter δ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
   - 11 for $\log(V_p)$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
   - 12 for $\log(\rho)$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)

- 31 for $\log(V_p/V_{p0})$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
- 32 for $\log(\rho/\rho_0)$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
- 33 for $\log(Q_p/Q_{P0})$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
- 34 for $\log(1+\epsilon)$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)
- 35 for $\log(1+\delta)$ (attention, this parameter can be reconstructed only by non-preconditionned optimization methods)

4. *src estimation.* 0-no estimation of the wavelet, 1-one estimation for all the gathers, 2-one estimation per shot gather

5. *bathy file* : bathymetry file name (*cf* part 5.4.1)

6. *dead zone* is a distance, considered bellow the bathymetry depth, where the properties of the media are not involved during inversion, but the properties can changed due to regularization. It is a kind of dead zone.

7. *optimization method.* Choice of the optimization routine for the inversion. 1: steepest descent, 2: preconditioned steepest-descent, 3: nonlinear conjugate gradient, 4: preconditioned nonlinear conjugate gradient, 5: l-BFGS, 6: preconditioned l-BFGS, 7: truncated Gauss-Newton, 8: truncated Newton, 9: precision truncated Gauss-Newton, 10: preconditioned truncated Newton. The preconditioning currently implemented is the inverse of the diagonal pseudo-Hessian approximation (Shin et al., 2001). This preconditioning is only adapted to surface acquisition.

8. *convergence criterion.* Convergence criterion of the optimization routines. The iteration stop as soon as $f(m)/f(m_0)$ becomes lower than this value (m_0 being the initial model).

9. *nonlinear iterations.* Maximum number of nonlinear iterations. The iterations stop as soon as this number is reached, even if the convergence criterion is not satisfied.

10. *maximum linesearch iterations.* Maximum number of iterations for the linesearch. The linesearch is declared to have failed when no appropriate steplength in the current descent direction has been found after performing this maximum number of linesearch iterations.

11. *maximum number of inner CG iterations.* This option concerns only the truncated Newton methods. This is the maximum number of inner CG iterations that can be performed at each nonlinear iteration. When this number is reached, the resolution of the linear system associated with the computation of the Newton descent direction stops, and the current descent direction is returned.

12. *threshold parameter for the preconditioner.* Threshold value for the definition of the preconditioner through the pseudo-Hessian approximation. Practical values range from 10^{-1} to 10^{-5}. The smallest this value is, the stronger impact the preconditioner has on the inversion. Too small values may yield instabilities. A good average choice is (often) 10^{-2}. More details are available in Métivier et al. (2013, 2014).

13. *debug option for the optimization routines.* If this option is set to 1, additional printings in the output files give information on the linesearch process. In addition, for the truncated Newton methods only, the decrease of the local quadratic approximations through the conjugate gradient procedure is checked at each nonlinear iteration. This modifies the output files iterate_*_CG.dat. This requires additional computations and may slow down slightly the truncated Newton algorithms.

14. *lambda for Tikhonov regularization.* Two functionnalities beside this parameter. If negative value, that activates the smoothing of the gradient. If positive value, this is the absolute regularization weights per parameter class for first order derivatives Tikhonov regularization. A vector of the size of the number of parameter classes inverted is required in this case. Each element give the weight associated to the Tikhonov regularization for a given parameter class, respecting the order of the parameters in the chosen family.

15. *lambda_x, lambda_z.* Directional weight for regularization/smoothing. If the previous *lambda for Tikhonov regularization* is negative, *lambda_x, lambda_z* are fraction of the local wavelength used to define the correlation length of the Laplacian function used to smooth the gradient. Typical value are between 0.2 (almost no smoothing) and 1.2-2 (big smoothing). If the previous *lambda for Tikhonov regularization* is positive, *lambda_x, lambda_z* are regularization weights for the Tikhonov regularization. In both case, *lambda_x* acts on the x direction and *lambda_z* acts on the z direction. If *lambda_x*=*lambda_z*, the regularization/smoothing is isotropic.

16. *prior information weight* : not implemented, put 0

17. *bound constraints* : flag to activate (1) of not (0) the bounds in the parameters

18. *upper bound* value if previous flag is 1

19. *lower bound* value if previous flag is 1

20. *tolerance for bound constraints* in the unit of the reconstructed parameter.

21. `file name of the data weighting` : name of the file for weighting (`cf` part 5.5)

22. `number of point and sampling of the data weighting file` : size of the data weighting file and the associated step size.

6 Running TOY2DAC

6.1 Output files in modeling mode

The main output files are data at receivers (complex-value frequency-domain).

The data files in the frequency-domain at receivers positions is always called : *data_modeling.* This is a direct access BINARY files. This files is ordered as follows :

```
data_bloc_frequency_1 data_bloc_frequency_2 ... data_bloc_frequency_N
```

where the N number is driven by the number of frequency put the user in file *freq_management*
For each frequency bloc *data_bloc_frequency_i*, the data are ordered as follows :

```
data_bloc_source_1 data_bloc_source_2 ... data_bloc_source_L
```

Finally, for each source bloc, the data are ordered as follows :

```
data_receiver_1 data_receiver_2 ... data_receiver_K
```

In case of commom number of receiver per source, the file can be looked with ximage

```
ximage < data_modeling n1=2*nrec
```

where nrec=number of receivers.

6.2 Output files in FWI mode

The output files are listed bellow

1. output log on standard output that contains several information, in particular in case of problem...
2. *invparinter* contains all the intermediate models during inversion. Note that the model are stored sequentialy, in the parametrization used for inversion
3. *invparfinal* contains the final models, in the parametrization used for inversion
4. *param_XX_inter* contains all the intermediate models stored sequentialy, for parameter XX. In isotropic, the XX are *vp*, *rho* and *qp*, in anisotropic the XX are *vp*, *rho*, *qp*, *epsilon*, *delta* and *theta*.
5. *param_XX_final* contains the final models for parameter XX. In isotropic, the XX are *vp*, *rho* and *qp*, in anisotropic the XX are *vp*, *rho*, *qp*, *epsilon*, *delta* and *theta*.
6. *iterate_*.dat*: the optimization routine output log that contains information on the convergence.

The files *invparinter*, *invparfinal*, *param_XX_inter* and *param_XX_final* can be looked with a ximage command as:

```
ximage < param_vp_final n1=101
```

where 101 is the nz size.

7 Template examples

This part presents several template available in the TOY2DAC package. These template consider both FWI and modeling problems.

7.1 *Gaussian perturbation model*

This very small template is a good way to test the code on your platform because it can be run in only few minutes, and requires a very small amount of memory. It also allows to understand how FWI is working on a simple boxcard-like configuration for which the result can be anticipated. This template in composed of 1 directory : *run_ball_template*.

run_ball_template contains the required files to run TOY2DAC in mode *0* in order to compute frequency-domain data in an homogeneous background with 1 circular anomaly in V_P. Once TOY2DAC is compiled, the template can be run by following the different operations :

1. make a local copy (in order to keep the template directory clean...).
2. the description of the medium are files : vp_ball, qp and rho. The user can plot them with ximage. For example :

   ```
   ximage < vp_ball n1=101 d1=20 d2=20 label1='depth in m' label2='distance in m' &
   ```

3. after a look in input files, checking that the input model files are *'vp_ball qp rho'* in file *fdfd_input* and mode=0 in file *toy2dac_input*, the user can then run the template. Example :

   ```
   mpirun -n 4 ../bin/toy2dac
   ```

4. after job end (on 4 processors, the elapsed time is few seconds), the "observed data" are stored in files *data_modeling*, and can be used for inversion.

The user can then run FWI.

1. the cartesian description of the medium are files : vp_homogeneous,qp and rho. The user can plot them with ximage. For example :

   ```
   ximage < vp\_homogeneous n1=101 d1=20 d2=20 label1='depth in m' label2='distance in m
   ```

2. after a look in input files, checking that the input model files are *'vp_homogeneous qp rho'* in file *fdfd_input*, the observed data file set to *'data_modeling'* at the first line of *fwi_input* and mode=1 in file *toy2dac_input*, the user can then run the template. Example :

   ```
   mpirun -n 4 ../bin/toy2dac
   ```

   item after job end (on 4 processors, the elapsed time is few minutes), the user can look at the final model. Example :

   ```
   ximage < param_vp_final n1=101 d1=20 d2=20 label1='depth in m' \\
   label2='distance in m' &
   ```

   the other binary files (intermediate etc...) can also be looked at, and the convergence in file *iterateXXX.dat*, where XXX depends on the optimization method chosen.

7.2 *Marmousi model*

This template is a simple geophysical configuration and runs also in few minutes. This template in composed of 1 directory : *run_marmousi_template*.

run_marmousi_template contains the required files to run TOY2DAC in mode *0* in order to compute frequency-domain data in the true Marmousi model. Once TOY2DAC is compiled, the template can be run by following the different operations :

1. make a local copy (in order to keep the template directory clean...).
2. the description of the medium are files : vp_Marmousi_exact, qp and rho. The user can plot them with ximage. For example :

   ```
   ximage < vp_Marmousi_exact n1=141 d1=25 d2=25 label1='depth in m' label2='distance in
   ```

3. after a look in input files, checking that the input model files are *'vp_Marmousi_exact qp rho'* in file *fdfd_input* and mode=0 in file *toy2dac_input*, the user can then run the template. Example :

   ```
   mpirun -n 4 ../bin/toy2dac
   ```

4. after job end (on 4 processors here), the "observed data" are stored in files *data_modeling*, and can be used for inversion.

The user can then run FWI.

1. the cartesian description of the medium are files : vp_Marmousi_init,qp and rho. The user can plot them with ximage. For example :

   ```
   ximage < vp_Marmousi_init n1=141 d1=25 d2=25 label1='depth in m' label2='distance in
   ```

2. after a look in input files, checking that the input model files are *'vp_Marmousi_init qp rho'* in file *fdfd_input*, the observed data file set to *'data_modeling'* at the first line of *fwi_input* and mode=1 in file *toy2dac_input*, the user can then run the template. Example :

   ```
   mpirun -n 4 ../bin/toy2dac
   ```

   item after job end (on 4 processors, the elapsed time is few minutes), the user can look at the final model. Example :

   ```
   ximage < param_vp_final n1=141 d1=25 d2=25 label1='depth in m' \\
   label2='distance in m' &
   ```

   the other binary files (intermediate etc...) can also be looked at, and the convergence in file *iterateXXX.dat*, where XXX depends on the optimization method chosen.

References

Brenders, A. J. and Pratt, R. G. (2007). Efficient waveform tomography for lithospheric imaging: implications for realistic 2D acquisition geometries and low frequency data. *Geophysical Journal International*, 168:152–170.

Hicks, G. J. (2002). Arbitrary source and receiver positioning in finite-difference schemes using Kaiser windowed sinc functions. *Geophysics*, 67:156–166.

Hustedt, B., Operto, S., and Virieux, J. (2004). Mixed-grid and staggered-grid finite difference methods for frequency domain acoustic wave modelling. *Geophysical Journal International*, 157:1269–1296.

Métivier, L., Bretaudeau, F., Brossier, R., Operto, S., and Virieux, J. (2014). Full waveform inversion and the truncated Newton method: quantitative imaging of complex subsurface structures. *Geophysical Prospecting*, 62:1353–1375.

Métivier, L. and Brossier, R. (2016). The SEISCOPE optimization toolbox: A large-scale nonlinear optimization library based on reverse communication. *Geophysics*, 81(2):F11–F25.

Métivier, L., Brossier, R., Virieux, J., and Operto, S. (2013). Full Waveform Inversion and the truncated Newton method. *SIAM Journal On Scientific Computing*, 35(2):B401–B437.

Operto, S., Virieux, J., Dessa, J. X., and Pascal, G. (2006). Crustal imaging from multifold ocean bottom seismometers data by frequency-domain full-waveform tomography: application to the eastern Nankai trough. *Journal of Geophysical Research*, 111(B09306):doi:10.1029/2005JB003835.

Operto, S., Virieux, J., Ribodetti, A., and Anderson, J. E. (2009). Finite-difference frequency-domain modeling of visco-acoustic wave propagation in two-dimensional TTI media. *Geophysics*, 74 (5):T75–T95.

Pratt, R. G. and Worthington, M. H. (1990). Inverse theory applied to multi-source cross-hole tomography. Part I: acoustic wave-equation method. *Geophysical Prospecting*, 38:287–310.

Ravaut, C., Operto, S., Improta, L., Virieux, J., Herrero, A., and dell'Aversana, P. (2004). Multi-scale imaging of complex structures from multi-fold wide-aperture seismic data by frequency-domain full-wavefield inversions: application to a thrust belt. *Geophysical Journal International*, 159:1032–1056.

Shin, C. and Cha, Y. H. (2008). Waveform inversion in the Laplace domain. *Geophysical Journal International*, 173(3):922–931.

Shin, C. and Ha, W. (2008). A comparison between the behavior of objective functions for waveform inversion in the frequency and laplace domains. *Geophysics*, 73(5):VE119–VE133.

Shin, C., Jang, S., and Min, D. J. (2001). Improved amplitude preservation for prestack depth migration by inverse scattering theory. *Geophysical Prospecting*, 49:592–606.

Sirgue, L. and Pratt, R. G. (2004). Efficient waveform inversion and imaging : a strategy for selecting temporal frequencies. *Geophysics*, 69(1):231–248.