SLIM FWI framework documentation

Curt Da Silva

Part I
Introduction

This is the accompanying documentation for the 2D/3D FWI framework that I wrote. It is a work in progress, so if there are features/aspects missing that should be addressed, please let me know.

Part II
Code organization

This codebase uses a modular approach to solving FWI problems which is outlined as follows, from lowest in the hierarchiy to highest.

- The most basic function/class pairing is discrete_helmholtz.m, which takes in as input a model vector on the computational domain, appends a pml to it, and outputs an opHelmholtz operator. The opHelmholtz operator is an abstract representation of a Helmholtz matrix, which knows how to multiply itself to a vector and divide itself to a vector, using either a specified linear solver/preconditioner for 3D systems or a specified direct solver for 2D systems.
- PDEfunc.m generates the quantities needed by FWI (objective value/gradient, forward modelled data, demigration/migration, hessian, gauss-newton hessian matrix vector products) depending on the specified user input. This function uses opHelmholtz to solve the required PDEs and assembles all the necessary 'components' in the proper way. This is a purely serial function, which receives a cell array of source/frequency indices to compute.
- PDEfunc_dist.m is responsible for distributing the computation of the desired PDEfunc quantity parallelized over joint (sx,sy, freq) coordinates in 3D or over freq in 2D.
- misfit_setup creates a function handle for a least-squares objective function that corresponds to user-selected sources/frequencies, using calls to PDEfunc_dist, and is suitable for use with a black-box optimization solver.

Part III
Necessary packages from the software release

- algorithms/CommonFreqModeling
- algorithms/2DFreqModeling (for 2D problems)
- algorithms/3DFreqModeling (for 3D problems)

- algorithms/WRI (for 2D WRI problems)
- functions/misc
- operators/misc
- solvers/Krylov
- solvers/Multigrid

Part IV
Basic usage

There are a few main components, all of which have components that are detailed below - model struct - contains all of the geometry information of the problem - params struct .pdefunopts - a PDEopts object .lsopts - a LinSolveOpts object

Part V
Model struct organization

The model struct describes the geometry of the FWI problem. It consists of the following parameters

- {o,d,n} - 1 x 3 vectors, describing the o,d,n parameters in the x-,y-,z- directions, respectively
- unit - model parameter unit, either 'm/s' (velocity), 's2/m2' (slowness squared), or 's2/km2' (seconds^2 / km^2)
- {xsrc,ysrc,zsrc} - coordinates describing the source grid for 3D, or {zsrc, xsrc} for 2D (note the z-direction is the first dimension)
- {xrec,yrec,zrec} - coordinates describing the receiver grid for 3D, or {zrec, xrec} for 2D
- t0 - time shift of the source wavelet (in seconds)
- f0 - Ricker wavelet peak frequency (in Hz)
- freq - vector of frequencies to use, in Hz

Note: for 2D FWI, due to legacy reasons, the grid ordering is (z,x) (including the various parameters in the model struct, model vectors, etc.), whereas for 3D FWI it is (x,y,z). Make sure that your code follows these conventions or you will generate incorrect results.

Part VI
Available options

These are the options that you can specify for a given FWI problem, as well as the functions in which they are used.

1 Primary options

- **pdefunopts** : PDEopts object, specifies parameters used in PDEfunc3D. See PDEopts.m for more details.
  - **numcompsrc** - number of sources to process at the same time (default: *1*)

- **zeroboundary** - if true, zeros the gradient/hessian-vector product at the boundary nodes (default: *false*)
- **window_source_grad** - if true, zeros the gradient/hessian-vector product in a neighbourhood of the sources (default: *false*)

- **src_interp** - source interpolation method, one of
  - PDEopts.SRC_INTERP_LIN - linear interpolation
  - PDEopts.SRC_INTERP_SINC - sinc interpolation (*default*)
- **rec_interp** - receiver interpolation method, one of
  - PDEopts.REC_INTERP_LIN - linear interpolation
  - PDEopts.REC_INTERP_SINC - sinc interpolation (*default*)
- **debug_mode** - if true, perform some basic debugging checks + excessive PDE solve outputs (default: *false*)

- **helm_scheme** - helmholtz discretization scheme, one of
  - PDEopts.HELM3D_OPERTO27 - 27 pt stencil based on Operto et al. (2007) (*default for 3D*)
  - PDEopts.HELM3D_STD7 - 7 pt standard finite difference stencil
  - PDEopts.HELM2D_CHEN9P - optimal 9 point stencil of Chen at al. (2009) (*default for 2D*)
- **helm_cut_pml** - if *true* (default), removes the contribution of the velocity model in the PML region, if false, uses the true adjoint of PML extension to return to the computational domain
- **helm_free_surface** - if true, uses a free surface on the top of the computational domain, i.e., no pml (default: *false*)
- **helm_pml_max** - if specified, maximum number of pml points to add to the computational domain (default: *inf*)
- **helm_dt** - grid spacing for the computational domain, if the user is subsampling the model herself (default: *[], use model.d*)
- **helm_pml** - either a scalar or a length 3 vector, specifying the number of pml points in the x-y-z coordinates (default: *[], chosen by the modeling code*)
- **helm_mat_free** - if true, generates an implicit matrix-vector product, so no matrix entries are actually formed, if false, generates matrix explicitly (default: true for 3D, false for 2D)
- **src_est_mode** - source estimation parameters, one of
  - PDEopts.SRC_EST_NONE - no source estimation (*default*)
  - PDEopts.SRC_EST_RECOMPUTE - source estimation with recomputation of wavefields (saves memory, costs more time)
  - PDEopts.SRC_EST_NORECOMPUTE - source estimation without recomputation of wavefields (uses more memory, saves more time)
- **misfit_func** - function handle for the data misfit, should output objective, gradient, hessian (default: **?**)
- This class also defines the following variables for specifying the mode of PDEfunc3D / PDEfunc3D_dist
  - .OBJ - least squares objective/gradient
  - .FORW_MODEL - forward modeling
  - .JACOB_FORW - demigration-vector product
  - .JACOB_ADJ - migration-vector product

        * .HESS_GN - GN Hessian-vector product
        * .HESS - full Hessian-vector product

- **linsolveopts**: LinSolveOpts object, specifies parameters used for solving the Helmholtz equations. See LinSolveOpts.m for more details.
  - **tol** - relative residual tolerance (default: *1e-6*)
  - **maxit** - maximum (outer) iterations (default: *10000*)
  - **maxinnerit** - maximum (inner) iterations, for certain solvers (default: *10*)
  - **solver** - linear solver to use, one of
    * LinSolveOpts.SOLVE_CGMN - CGMN
    * LinSolveOpts.SOLVE_CRMN - CRMN
    * LinSolveOpts.SOLVE_GMRES - GMRES
    * LinSolveOpts.SOLVE_FGMRES - GMRES w/ a flexible preconditioning option
    * LinSolveOpts.SOLVE_LU - pivoted sparse LU decomposition, only suitable for 2D problems
    * LinSolveOpts.SOLVE_BACKSLASH - Matlab's backslash solver, only suitable for 2D problems
  - **precond** - preconditioner to use, one of
    * a SPOT operator
    * function handle
  - LinSolveOpts object, which uses the solver parameters as specified
    * OR a predefined preconditioner
      · LinSolveOpts.PREC_KACZSWP - kaczmarz sweeps (*default for CGMN, CRMN*)
      · LinSolveOpts.PREC_IDENTITY - identity preconditioner
      · LinSolveOpts.PREC_MLCRMN - multigrid CRMN preconditioner
      · LinSolveOpts.PREC_MLGMRES - multigrid GMRES preconditioner (default)

2 Secondary options

Used in : misfit_setup

- **srcfreqmask** - a nsrc x nfreq binary matrix, an entry of 1 in the (i,j)th position indicates that this code should compute the desired quantity associated to the ith source and jth frequency, 0 indicates that it should be omitted (default: *ones(nsrc,nfreq), all sources/freqs computed*)
  - Also used in PDEfunc_dist
- **batch_mode** - if true, function handle accepts model vector, source/freq indices as input (default: *false*)
- **subsample_mode** - if true, subsamples the initial model to a coarser grid determined by the PDE stencil (default: *false*)
- **hessian** - hessian spot operator to output
  - PDEopts.HESS_GN - Gauss Newton Hessian (*default*)
  - PDEopts.HESS - Full Hessian

Used in : PDEfunc_dist, PDEfunc

- **disp_progress** - if true, display a progress indicator + estimated time to completion of this operation