FWI applications in pseudocode

Valesca Peereboom

April 3, 2020

This document describes the FWI_Preprocess, FWI_UnifiedProcess and Postprocess applications in Pseudocode and how to execute those applications from the terminal.

Algorithm 1 PreProcess: ./FWI_PreProcess {run_folder}

GenericInputCardReader: Read input from run_folder/input, gInput (GenericInput)
Start clock (CpuClock)
procedure GENERATEREFERENCEPRESSUREFIELDFROMCHI( )
    Initialize grid (Grid2D), chi (dataGrid2D),
    sources (Sources), receivers (Receivers), frequencies (FrequenciesGroup)
    procedure CREATE_MODEL(grid, sources, receivers, frequencies, gInput)
                                ▷ IntegralForwardModel from parent ForwardModelInterface
        Initialize IntegralForwardModel(grid, sources, receivers, frequencies),
        _Greens (Greens_rect_2D_cpu), _p0 (complexDataGrid2D),
        _ptot (complexDataGrid2D), _Kappa (complexDataGrid2D),
        _fmInput (IntegralForwardModelInput).
        return model
    model computes PTOT(chi), KAPPA( ) and PDATA(chi, referencePressureData (Vector⟨Complex⟨double⟩⟩))
    Write referencePressureData to file InvertedChiToPressure.text
Stop clock

Algorithm 2 Process: ./FWI_UnifiedProcess {run_folder} {inversion_model} {forward_model}

GenericInputCardReader: Read input from run_folder/input, gInput (GenericInput)
Start clock (CpuClock)
procedure PREFORMINVERSION(...)
  Initialize grid (Grid2D), sources (Sources), receivers (Receivers), frequencies (FrequenciesGroup)
  FACTORY::CREATEFORWARDMODEL(...,forward_model) **return** forwardmodel
    ▷ Creates a forwardmodel from parent ForwardModelInterface
  FACTORY::CREATEINVERSION(..., inversion_model, forwardmodel) **return** inversionmodel
    ▷ Creates an inverse model from parent InversionInterface
  procedure INVERSIONMODEL → RECONSTRUCT( )
    Initialize all variables
    Open residual.log file to store the residuals
    for all iterations until N_max **do**
      Calculate kappa with forwardmodel → CALCULATEKAPPA( ) ▷ $Pdata = Kappa * Chi$
      Calculate residuals with forwardmodel → CALCULATERESIDUAL(chiEstimate, pDataReference))
      Compute the next chiEstimate step according to the inversion model
      Update chiEstimate
      Compute new residuals and write to residual.log
      if residuals < tolerance **then**
        break;
  Write chiEstimate to file chi_est_{runName}.txt
Stop clock
WRITEPLOTINPUT(...) ▷ Creates {runName}.pythonIn and lastRunName.text, needed for postprocess

Algorithm 3 PostProcess: cp parallelized-fwi/PythonScripts/PostProcessing-python3.py FWIInstall, python3 PostProcessing-python3.py {run_folder} {run_number}

Read runName from first line in lastRunName.txt in outputfolder
Read variables nxt, nzt, nxt_original and nzt_orginal from {runName}.pythonIn in output folder
Read chi1 from chi_ref_{runName}.txt in outputfolder
Read chi2 from chi_est_{runName}.txt in outputfolder
Resize the chi of smallest size to the largest size
Compute mean square error and average relative error
Read execution time, virtual memory and physical memmory from {runName}.pythonIn in outputfolder
Create image Results.png with new, old and differences of chi values
Create image residuals.png with residuals over iterations
procedure OUTPUTLOGGER(...) ▷ from parallelized-fwi/pythonScripts/classes/OutputLogger.py
    procedure COMPLETE_OUTPUT_LOG( )
        Read kind of image (as temple or dog) used in the procces
        from parallelized-fwi/inputFiles/default/input/GenericInput.json
        Also ***read*** grid resolutions (ngrid and ngrid_original) and hardwarespecifics from GenericInput.json
        Read description of all methods used and path from parallelized-fwi/results/description{run_number}.txt
        and afterward ***remove*** this file
        Get input file FWIInstall/{path}/input/{inversionmethod}Input.json
    procedure SAVE_OUTPUT_LOG( )
        if parallelized-fwi/results does not exists **then**
            create directory
        write log file parallelized-fwi/results/log_{datetime}.json