cu$Q$-RTM Manual

Author Yufeng Wang
Date 2017-12-01
Link https://github.com/Super-Messiah/cuQRTM

1 Overview of cu$Q$-RTM package

`cu-QRTM` is a CUDA-based code package that implements Q-RTM based on a set of stable and efficient strategies, such as streamed CUFFT, checkpointing-assisted time-reversal reconstruction (CATRC) and adaptive stabilization scheme. This package is provided for accelerating conventional CPU-based Q-RTM, and mimicking how a geophysicist writes down a seismic processing modules such as modeling, imaging and inversion in the framework of the `CPU-GPU` heterogeneous computing platform. We provide two package versions: `cuQRTM-Express` for quick execution with 4 shots, which can be done within 3 min under single GPU card (GTX 760); `cuQRTM-Standard` for standard excution with 64 shots, which will take 10 min under 4 GPU cards (Tesla K10).

2 The architecture of cu$Q$-RTM package

- `input`: accurate velocity and Q model for Q-RTM:
  - `acc_vp.txt`: quasi-Marmousi velocity model;
  - `acc_Qp.txt`: quasi-Marmousi Q model;
  - `ascii2bin.m`: converting ASCII files to Binary files.
- `output`: generated results such as seismograms, images of each shot and final stacked images. What we are most interested in is final images, wich includes:
  - `Final_image_cor_type0.dat`: image from acoustic RTM;
  - `Final_image_cor_type1.dat`: image from viscoacoustic RTM with compensation;
  - `Final_image_cor_type2.dat`: image from Q-RTM using low-pass filtering;
  - `Final_image_cor_type3.dat`: image from Q-RTM using adaptive stabilization;
- `plot`: scripts for plotting figures, which includes:
  - `/madagascar/SConstruct`: plot images, velocity and Q models;

  - `/matlab/martrace`: plot extracted trace from final migrated images for comparison.

- `Myfunctions.h`: header file;
- `CUDAQRTM.cu`: cuda code file;
- `QRTM.cpp`: c++ code file, there are serveal important flags and parameters to control performance of Q-RTM, which includes:
  - `RTMtype`: you can change this flag to generate different migrated images.

```
int RTMtype=0;   // RTMtype=0 for acoustic RTM
   // RTMtype=1 for viscoacoustic RTM without compensation
   // RTMtype=2 for QRTM using low-pass filtering
   // RTMtype=3 for QRTM using adaptive stabilization scheme
```

  - `GPU_N` you can set GPU_N=n, where n denotes the number of GPU cards you will use.

```
int i,GPU_N;     // GPU_N stands for the total number of GPUs per node
getdevice(&GPU_N); // Obtain the number of GPUs per node
printf("The available Device number is %d on %s\n",GPU_N,processor_name);
```

  - `kx_cut` and `kz_cut`: these parameters are defined for low-pass filtering.

```
float kx_cut=3.0*2*PI*f0/vp_max; // be careful to change this parameter
float kz_cut=3.0*2*PI*f0/vp_max;
float kx_cut_inv=3.0*2*PI*f0/vp_max;
float kz_cut_inv=3.0*2*PI*f0/vp_max;
float taper_ratio=0.2;        // taper ratio for turkey window filter
```

  - `sigma` and `Order`: these two parameters are defined for adaptive stabilization,

```
float sigma=2.5e-3; // be careful to change this parameter
int Order=1;     // defult
```

- `Makefile`: excution script.

3 Prerequisites

`cu-QRTM` package is developed under `Linux` system, which should be equipped with the following environments:

- CUDA environment (for example, `-I/usr/local/cuda-8.0/include -L/usr/local/cuda-8.0/lib64`);
- MPI environment (for example, `-I/home/wyf/intel/impi/5.0.1.035/intel64/include -L/home/wyf/intel/impi/5.0.1.035/intel64/lib/`);
- matlab;
- madagascar.

4 How to run this package

If you want to quick test the package, please use fast version `cuQRTM-Express`; `cuQRTM-Standard` should be excuted on cluster with multi-GPUs (or you can excute on a sigle GPU card within an hour).

- Step 1: Run the matlab file `ascii2bin.m` in `./input` to convert the ASCII data into binary data;
- Step 2: Confirm the environment in `Makefile`, and replace the folder path with your own enviroment path;

```
#! /bin/sh
# compiler
CC=nvcc
# set CUDA and MPI environment path
INC=-I/usr/local/cuda-8.0/include -I/home/wyf/intel/impi/5.0.1.035/intel64/include
LIB=-L/usr/local/cuda-8.0/lib64 -L/home/wyf/intel/impi/5.0.1.035/intel64/lib/
# set CUDA and MPI Dynamic link library
LINK= -lcudart -lcufft -lm -lmpich -lpthread -lrt -DMPICH_IGNORE_CXX_SEEK -
     DMPICH_SKIP_MPICXX
# CUDA and C++ source codes
SOURCES=CUDAQRTM.cu QRTM.cpp
EXECNAME=QRTM
# Execution
all:
$(CC) -v -o $(EXECNAME) $(SOURCES) $(INC) $(LIB) $(LINK)
rm -f *.o
nohup mpirun -np 1 ./QRTM &
```

- Step 3: Run the `Makefile` by the command line: `make`;
- Step 4: View generated files in the folder `./ouput`;

```
make
vi nohup
cd output/
ximage n1=234 < Final_image_cor_type0.dat hbox=300 &
ximage n1=234 < Final_image_cor_type1.dat hbox=300 &
ximage n1=234 < Final_image_cor_type2.dat hbox=300 &
ximage n1=234 < Final_image_cor_type3.dat hbox=300 &
```

- Step 5: Plot figures by run `/plot/madagascar/SConstruct` and `/plot/matlab/martrace.m`.

```
scons view
cd Fig/
vpconvert format=pdf *.vpl
```

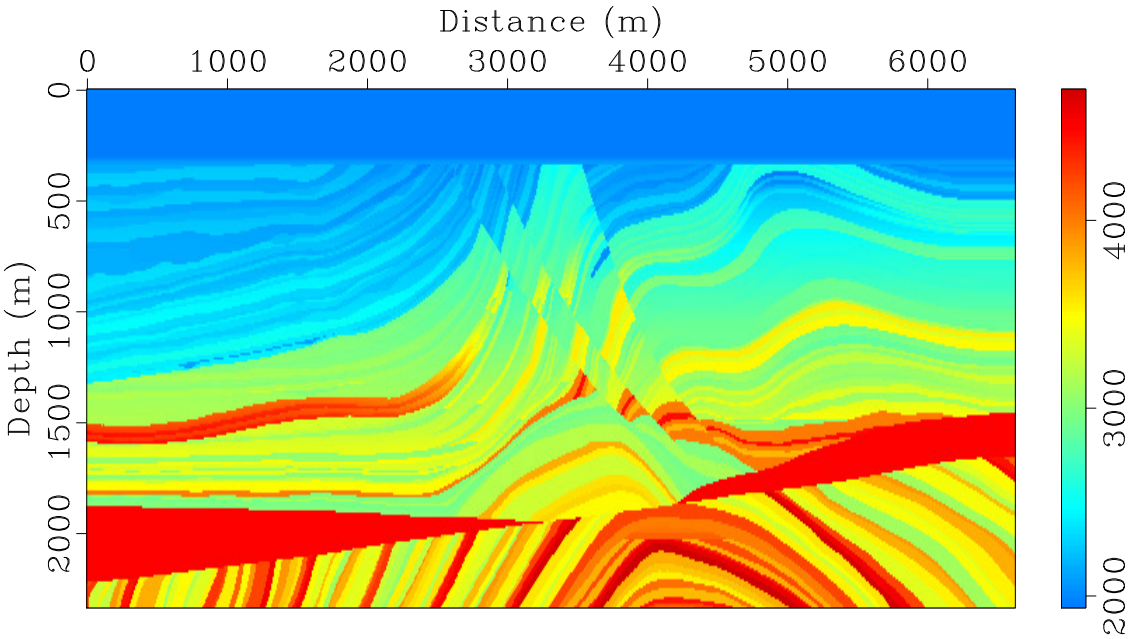

图 1: quasi-Marmousi velocity model

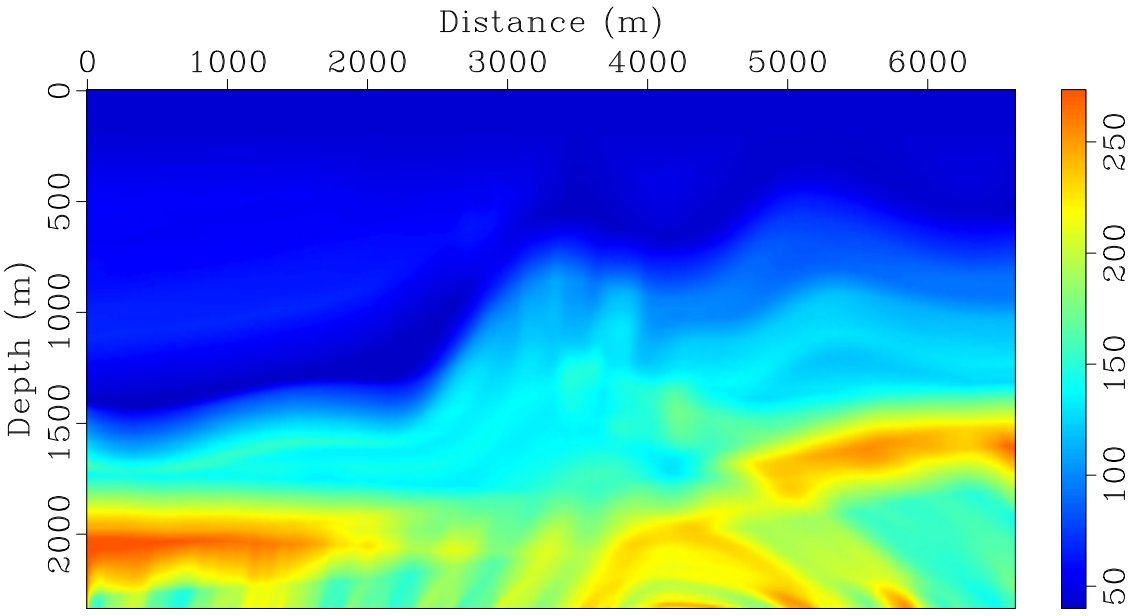

图 2: quasi-Marmousi Q model

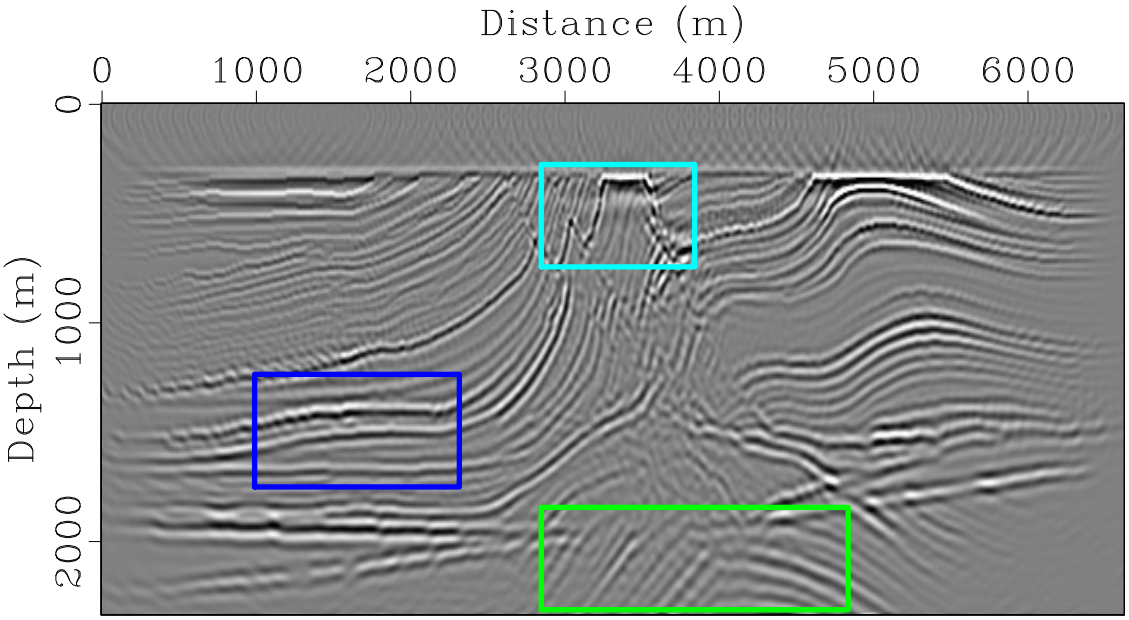

图 3: image from acoustic RTM

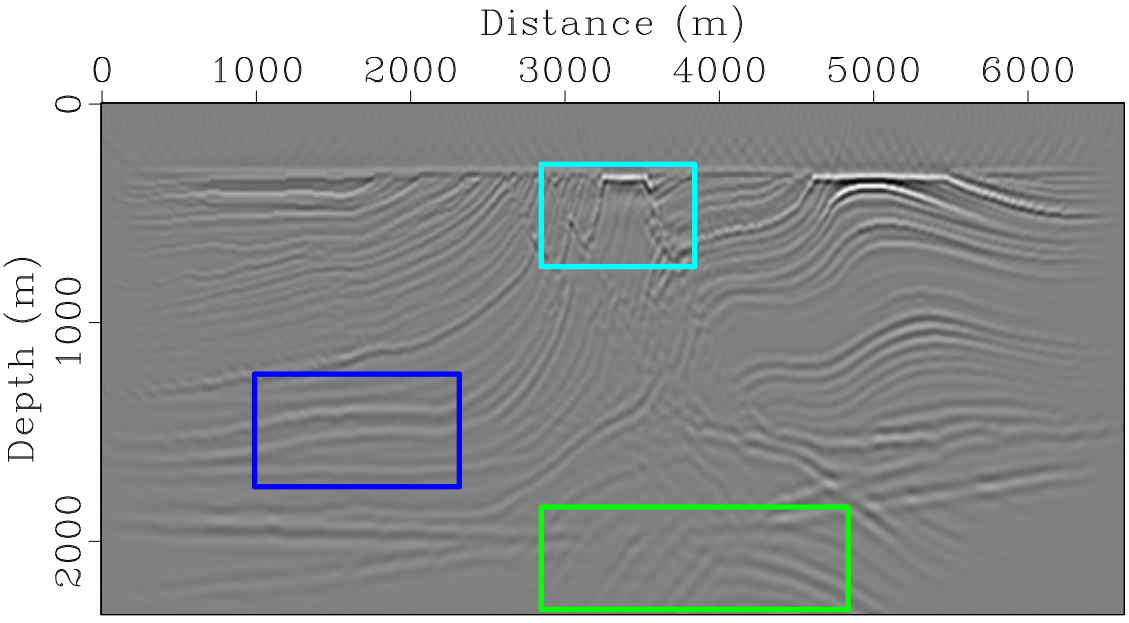

图 4: image from viscoacoustic RTM with compensation

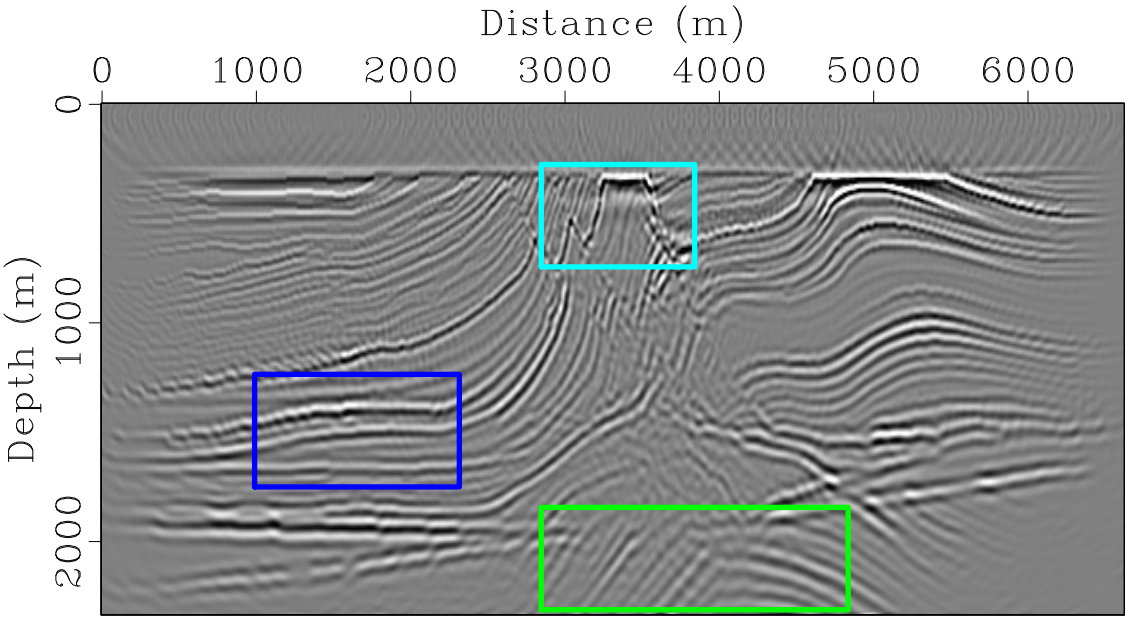

图 5: image from Q-RTM using low-pass filtering

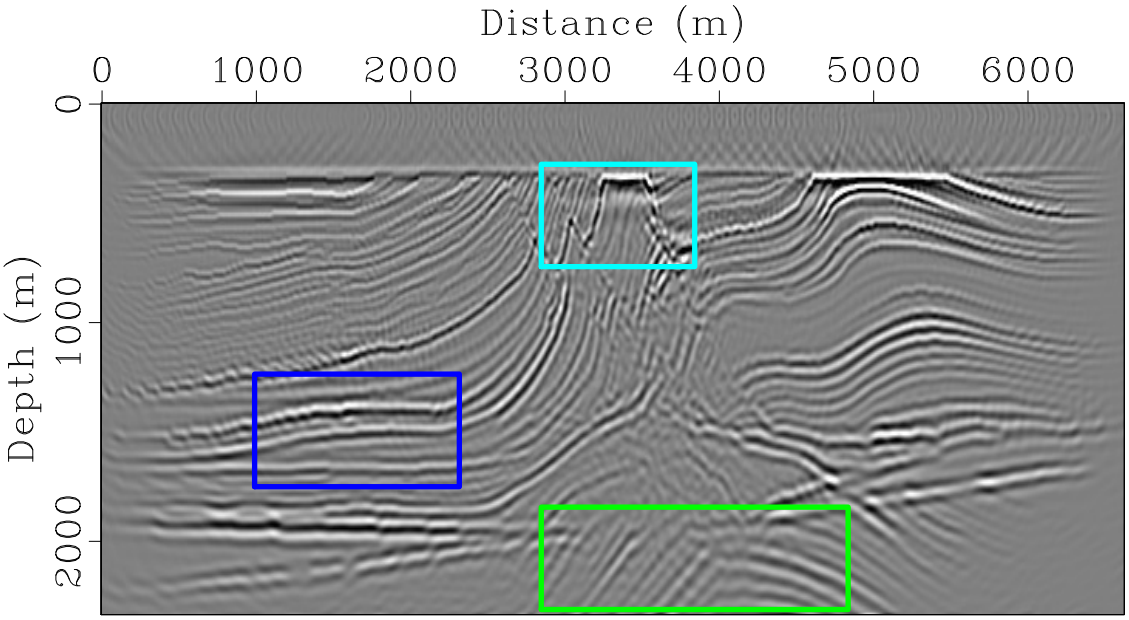

图 6: image from Q-RTM using adaptive stabilization

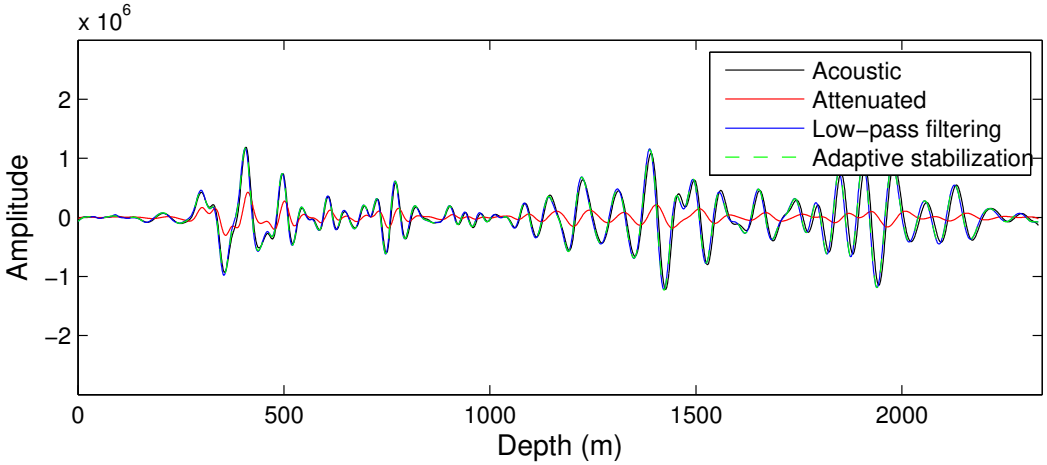

图 7: trace at X=1500m

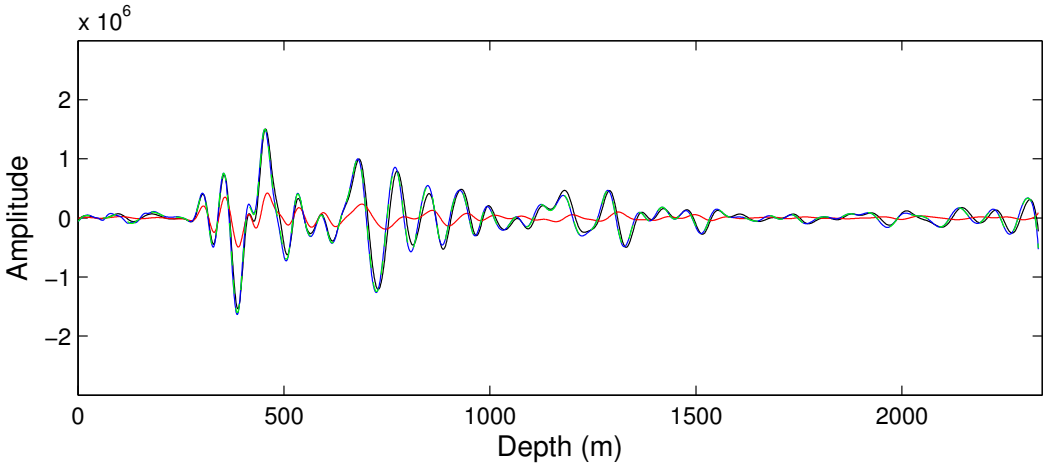

图 8: trace at X=3600m

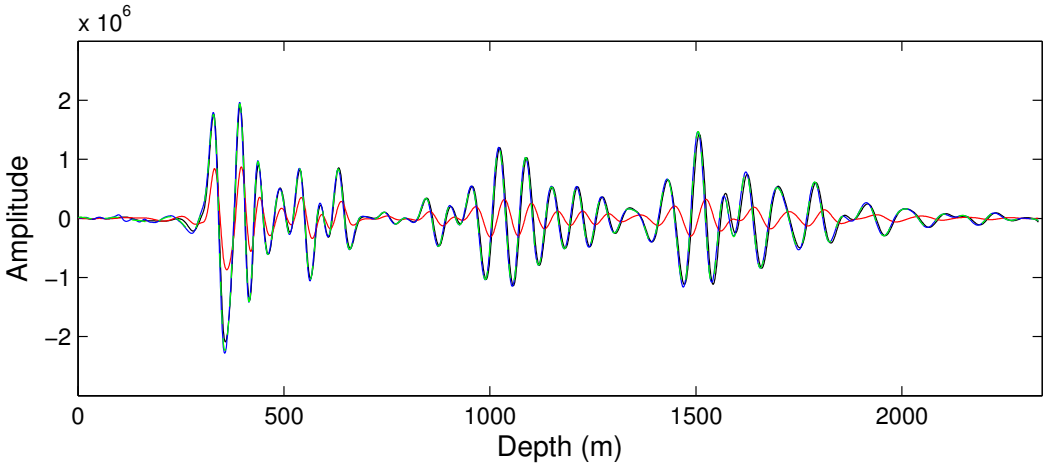

图 9: trace at X=5200m

5 Contact me

I am Yufeng Wang, a PhD candidate from China University of Petroleum, Beijing. If you have any question about this coda package, please feel free to contact me by Email:hellowangyf@163.com.

6 Copyright