Tutorial forward modelling 2D seismik menggunakan 2DSFpy

La Ode Marzujriban Masfara, 2020

Pendahuluan

Metode seismik merupakan salah satu metode populer dalam bidang geofisika, kebutuhan untuk mengolah data seismik pun menjadi hal utama dalam melakukan riset menggunakan metode ini namun tidak jarang riset akan terkendala dengan ketersediaan data. Data sintetik pun menjadi ujung tombak untuk mencoba keberhasilan "resep" baru metode seismik. Disisi lain perkembangkan pesat metode komputasi dan bahasa pemorgraman juga mendukung simulasi maupun pembuatan data sintetik. Python, merupakan salah satu bahasa pemograman populer dewasa ini bahkan merupakan salah satu yang memiliki pertumbuhan pengguna paling pesat. Selain fleksibilitas yang ditawarkan, sifatnya yang *open source* menjadikan python populer terutama dikalangan akademisi.

Menggambungkan python dan persamaan akustik, penulis menghadirkan 2DSFpy, module yang ditulis sepenuhnya menggunakan pyhon yang dapat digunakan untuk melakukan *forward modelling* untuk memperoleh data sintetik. Secara lengkap modul ini dapat di unduh melalui:

Sebuah file jupyter notebook pun disediakan agar pengguna bisa langsung menjalakan module dan kemudian dapat disesuaikan dengan kebutuhan pengguna. Melalui tulisan ini penulis ingin menjelaskan secara singkat penggunaan module utama yang digukan di file notebook yang disediakan.

Pastikan file jupyter notebook, data velocity dan source code, berada dalam satu folder yang sama

Sebelum menjalankan:

Pastikan anda telah memiliki python dan jupyter notebok terinstal. Jika belum anda dapat mengunduh Anaconda (https://www.anaconda.com/) yang akan menginstall python serta jupyter notebook secara otomatis.

1. Memuat data : Satu file data disediakan dalam format ".mat" yang memuat data P-velocity. Data ini kemudian di *unpack* menggunakan modul *scipy* dan diubah menjadi fila array 2 dimensi.

```
#load data
annots = loadmat('velocity.mat')
velocity = annots['vel']
velocity = velocity*1000
```

2. Buat project dengan memanggil module "forward". Contoh: project dibawah memiliki nama "for1" dibuat dengan memanggil module forward dengan input berupa 2D array "*velocity*, *density* dan *spacing*" untuk contoh kali ini density yang digunakan adalah *constant density* dan *spacing* 10 m "x" dan "y"

```
#define velocity density and spacing
dx = 10
density = 1500

#name the project for1
for1    = forward(velocity,density,dx)
```

3. Menentukan parameter simulasi: dalam menjalan kan skema *finite-difference* untuk pemodelan terdapat kriteria yang harus dipenuhi agar hasil pemodelan tidak mengalami dispersi numerik yang menjadikan simulasi tidak stabil [Juntunen, et.al 2000]. Untuk itu diperlukan perhitungan parameter berdasarkan *maximum velocity* dan *spacing* suatu model. Untuk memperoleh parameter tersebut ada dapat manggil sub-modul "FDpar()"

```
#call FDpar() to show FD parameter recomendation
for1.FDpar()
```

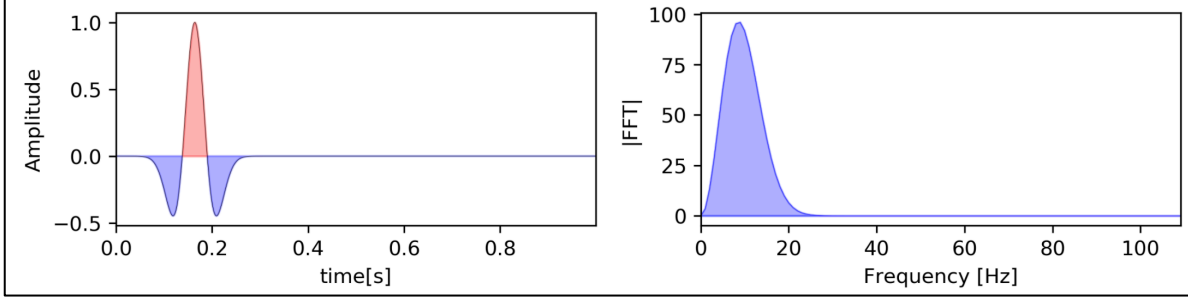
executed in 11ms, finished 12:45:34 2020-07-05

```
FD parameters:
|fc   | dominant frequency of ricker wavelet = 9.150000 Hz
|fmax| maximum frequency of our data        = 30.500000 Hz
|dt   | maximum sampling time                = 0.000829 s
```

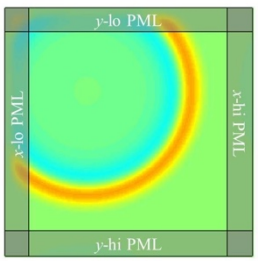
4. Membuat *source time function/wavelet*: Dalam melakukan simulasi, diperlukan *source* atau *wavelet* yang akan di-*inject* ke dalam *velocity model*. Untuk tahap ini 2DSFpy hanya mengakomodir *source* berupa *monopole accoustic pressure.* Pembuatan *source* dapat dilakukan dengan memanggil modul "source" serta memilih opsi "ricker" dengan input berdasarkan parameter yang diperoleh ditahap sebelumnya.

```
#define ricker source based on the above parameters
# t is the time axis ;
# s is the amplitude ;
# so you can plot plt.plot(t,s) or put plot = true in the source.ricker module
%matplotlib inline
dt = 0.0005
nt = 2000
fc = 9.15
t,s= sources(dt,nt, fc).ricker(a=4,plot=True)
```

executed in 766ms, finished 15:03:07 2020-06-30



5. Penambahan PML (*Perfect matched layer*): Setiap sisi dari model yang digunakan untuk melakukan simulasi akan memberikan pantulan dan akan direkam oleh *receiver*. Tentu saja hal ini bukanlah hal yang diinginkan sebab pantulan yang diharapkan hanyalah yang berasal dari lapisan model. Untuk mengantisipasi hal ini, diperlukan *layer* tambahan di keempat sisi (kasus 2D) dimana gelombang yang memasuki area tersebut akan mengalami absorbsi/attenuasi (*Absorbing boundary condition*) sehingga pantulan sisi model dapat diminimalisir [Berenger, 1994].



Contoh penggunaan PML dan Absorbing boundary condition (http://emlab.utep.edu/academics.htm)

Untuk menambahkan PML anda dapat memanggil modul "pml" dengan input jumlah layer yang ingin ditambakan kesetiap sisi (npml). Untuk mengaktifkan PML pada

model anda kemudian harus memanggil modul “ApplyPML” dengan input PML *factor* dan PML *exponent* yang menentukan attenuasi gelombang di area PML.

```
#add PML to the velocity model
npml    = 100
cpml    = for1.pml(npml)

#apply absorbing boundary condition to the PML
for1.ApplyPML(1,1)
```

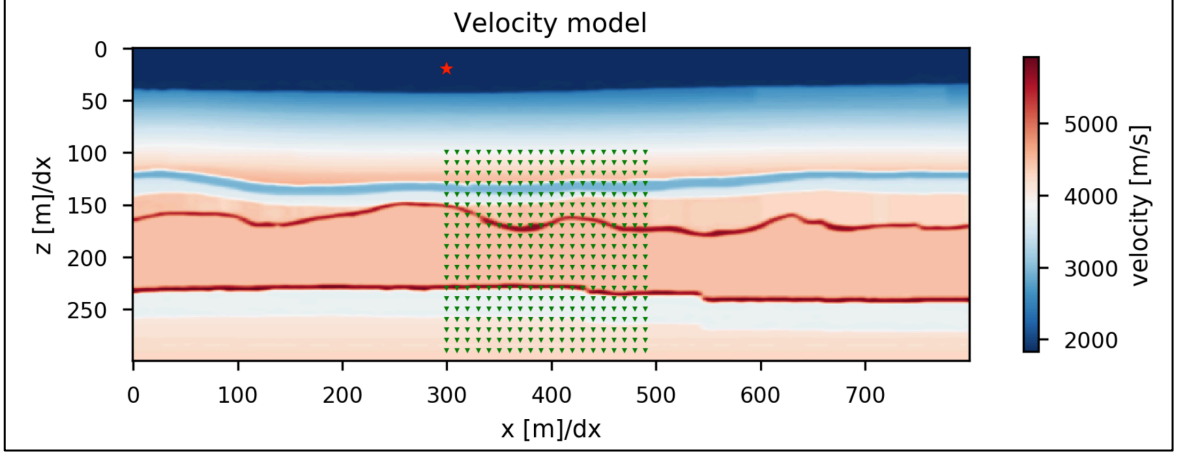
executed in 32ms, finished 15:03:07 2020-06-30

```
--- > 100 points are added to all sides
--- > absorbing boundaries applied to PML
--- > PML factor = 1.000   |    PML exponent = 1.000
```

6. *Acquisition geometry*: Tahap selanjutnya adalah menentukan lokasi receiver dan source pada model. Hal ini bebas anda lakukan dengan menggunakan modul bawaan python seperti *linspace* atau *arrange* melalui *numpy*. Akan tetapi secara default jumlah *source* yang diperbolehkan hanyalah satu *source* (dapat dikostumisasi). Setelah menentukan lokasi *receiver* dan *source* maka anda dapat memanggil modul “plotmodel” dengan input lokasi *source* dan *receiver*.

```
#plot the setup
%matplotlib inline
for1.plotmodel(sx,sy,recx,recz)
```

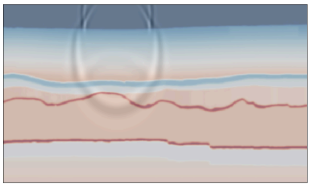
executed in 705ms, finished 15:03:10 2020-06-30



7. *Modelling*: Setelah menentukan semua parameter, anda dapat memanggil modul “solve” untuk melakukan *forward modelling* dengan input lokasi *source* dan *receiver* serta *time axis* dan *source*. Untuk menampilkan proses simulasi anda dapat mengatifkan opsi plot (True).

```
#run the setup
#to be able to show the movie add '%matplotlib notebook' before the solve module and set plotmov to 'True'
#to make plotted images appear inline add '%matplotlib inline'
%matplotlib notebook
res = for1.solve(recx,recz,sx,sy,t,s,plotmov=True)
%matplotlib inline
```

execution queued 13:53:40 2020-07-05



Download plot

67%| | 1340/1999 [01:37<00:41, 15.82it/s]

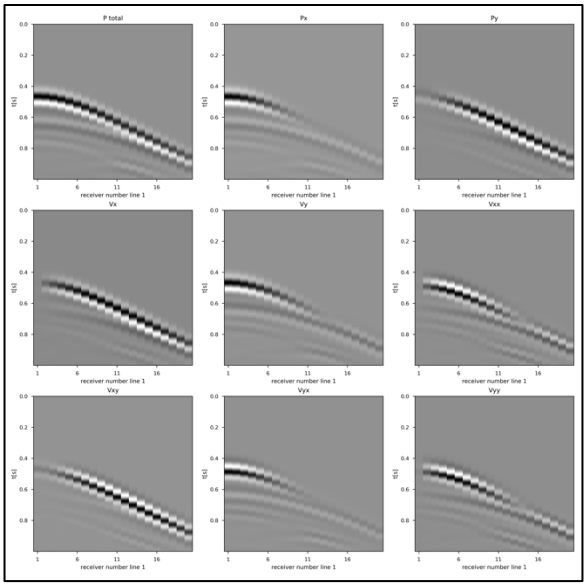
Setelah simulasi dilakukan anda dapat melihal hasil simulasi dengan me-list hasil simulasi:

```
#list the results of the forward simmulation
#Vx is horizontal particel velocity and Vxx is the derivative of Vx in x direction
list(res)
```

executed in 8ms, finished 14:21:04 2020-07-05

```
['Px', 'Py', 'Ptot', 'Vx', 'Vy', 'Vxx', 'Vxy', 'Vyy', 'Vyx']
```

Hasil simulasi berupa *pressure* (P) dan *particle velocity* (V), *subscript* pertama merupakan orientasi – contoh Vx = *particle velocity* pada arah "x" sedang *subscript* kedua merupakan *spatial derivative* – contoh Vxx = $\frac{dVx}{dx}$ sedangkan Ptot adalah *total pressure*.



8. Selain *forward modelling*, juga terdapat modul untuk melakukan dekonvolusi dengan metode *water level deconvolution* [Ligorria, et.al 1999]. Dengan input yakni *trace* yang anda pilih (pada contoh adalah salah satu trace dari Vxx), wavelet yang digunakan serta property taper/filter yang akan digunakan untuk mem-filter *band-limited Green's function* anda dapat memanggil modul "data" dan submodule "deconv" untuk

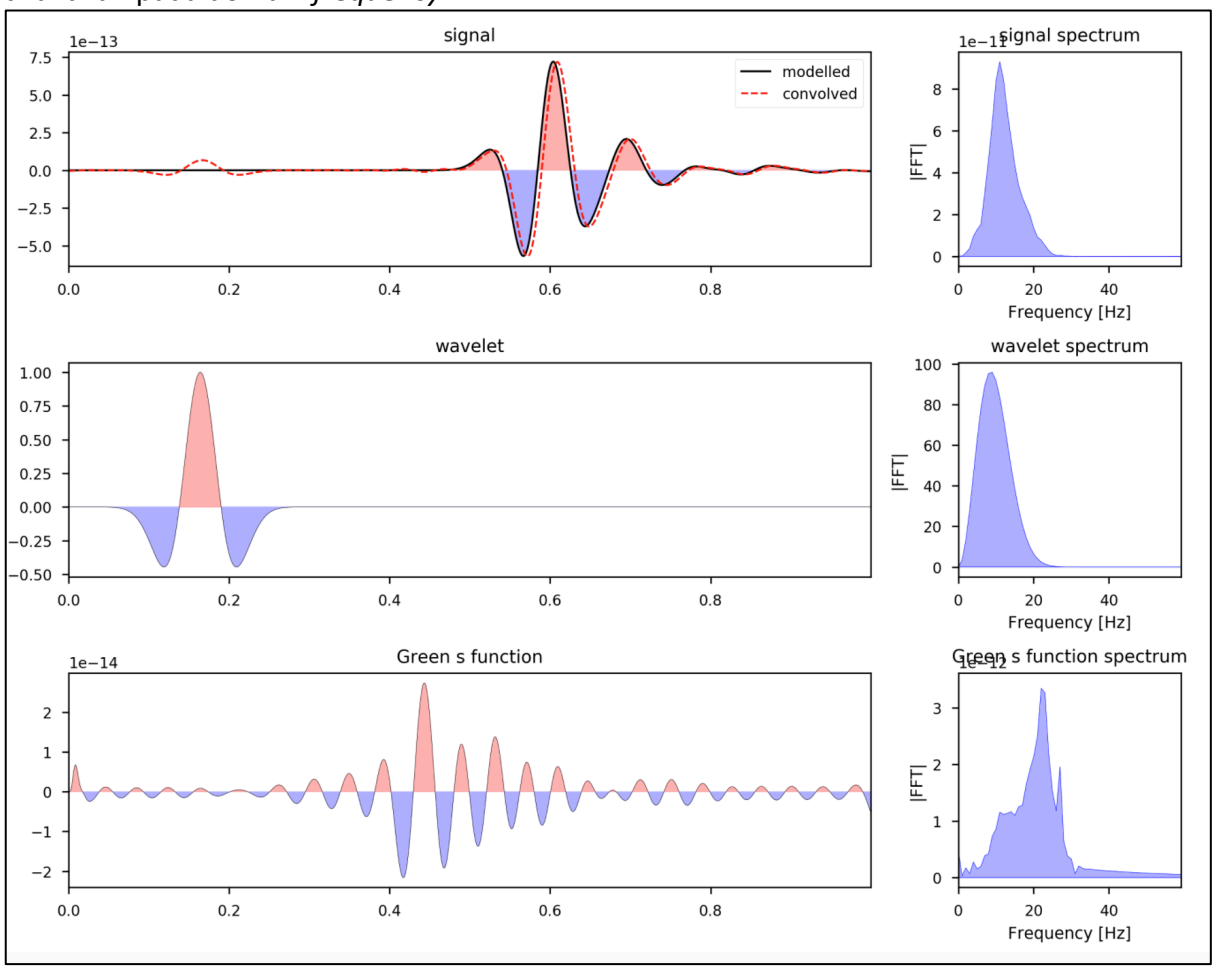
melakukan deconvolusi serta memilih opsi “plotdeconv=True” untuk menampikan hasilnya.

```
#deconvolve data call module 'data'
wavelet     = s
timesample  = t
freqsamp    = 1/dt
datatodecon = vxx
cutofftaper = 100
ordertaper  = 5
waveletfc   = 9.15

dataprep = data(freqsamp,wavelet,timesample)
decon = dataprep.deconv(datatodecon,cutofftaper,ordertaper,waveletfc,plotdeconv=True)
```

executed in 2.30s, finished 14:45:57 2020-07-05

Pada panel pertama, menunjukkan sinyal yang dipilih (*trace* hitam) sedangkan trace merah merupakan hasil konvolusi wavelet pada panel kedua dengan Green’s function di panel ketiga. Green’s function dipanel ketiga merupakan band limited Green’s function hasil dekonvolusi *trace* hitam panel pertama dengan *wavelet*. Proses dekonvolusi dan konvolusi dilakukan pada domain *frequency*.



Demikianlah tutorial dan penjelasan singkat modul ini. Modul ini dibuat untuk pengenalan metode seismik dan juga dapat digunakan untuk memperoleh data sintetik untuk kebutuhan riset seperti skripsi maupun penulisan karya ilmiah lainnya (dengan men-sitasi dokumen ini atau URL Github penulis). Pertanyaan dan saran dapat dikirim melalui redaksi atau penulis melaui email: l.o.m.masfara@tudelft.nl selamat mencoba.

[1] Juntunen, Jaakko S., and Theodoros D. Tsiboukis. "Reduction of numerical dispersion in FDTD method through artificial anisotropy." *IEEE Transactions on Microwave Theory and Techniques* 48.4 (2000): 582-588.

[2] Berenger, Jean-Pierre. "A perfectly matched layer for the absorption of electromagnetic waves." *Journal of computational physics* 114.2 (1994): 185-200.

[3]Ligorria, Juan Pablo, and Charles J. Ammon. "Iterative deconvolution and receiver-function estimation." *Bulletin of the seismological Society of America* 89.5 (1999): 1395-1400.