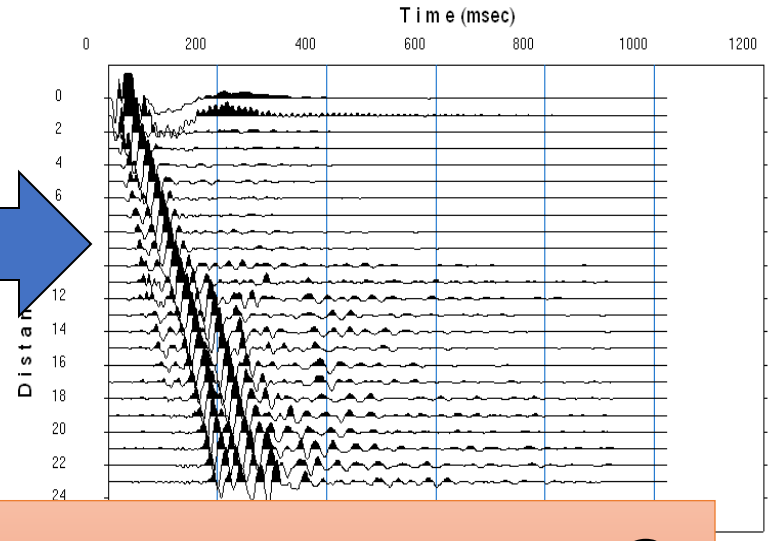
4. Inversion and uncertainty

- Inversion outline
- Least square methods
- Non-linear least square methods
- Uncertainty in surface wave investigations
- Appendices

Data acquisition & inversion of geophysical data

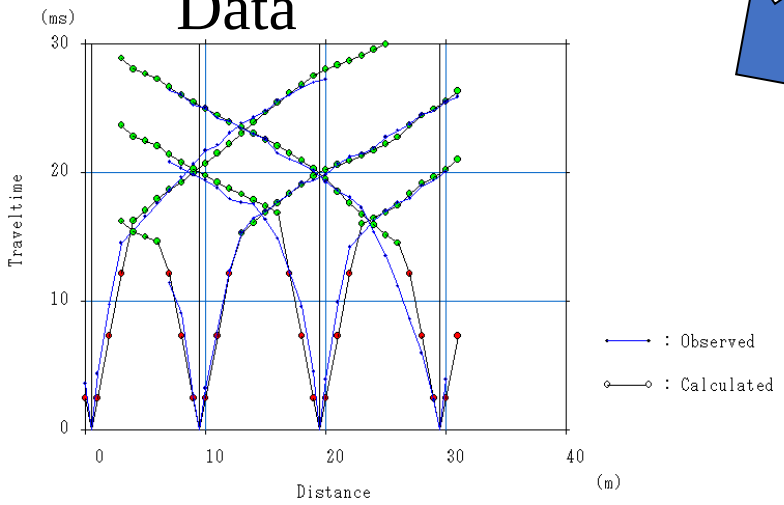
Data acquisition

Raw data

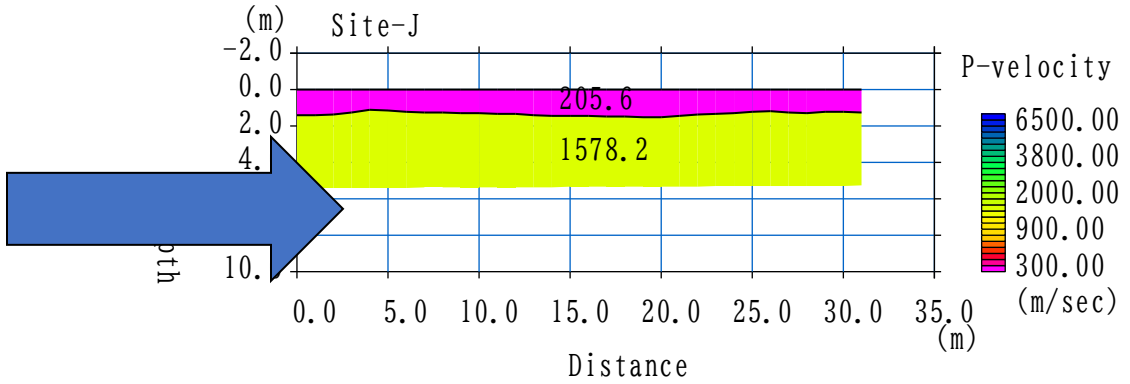


What is inversion ?

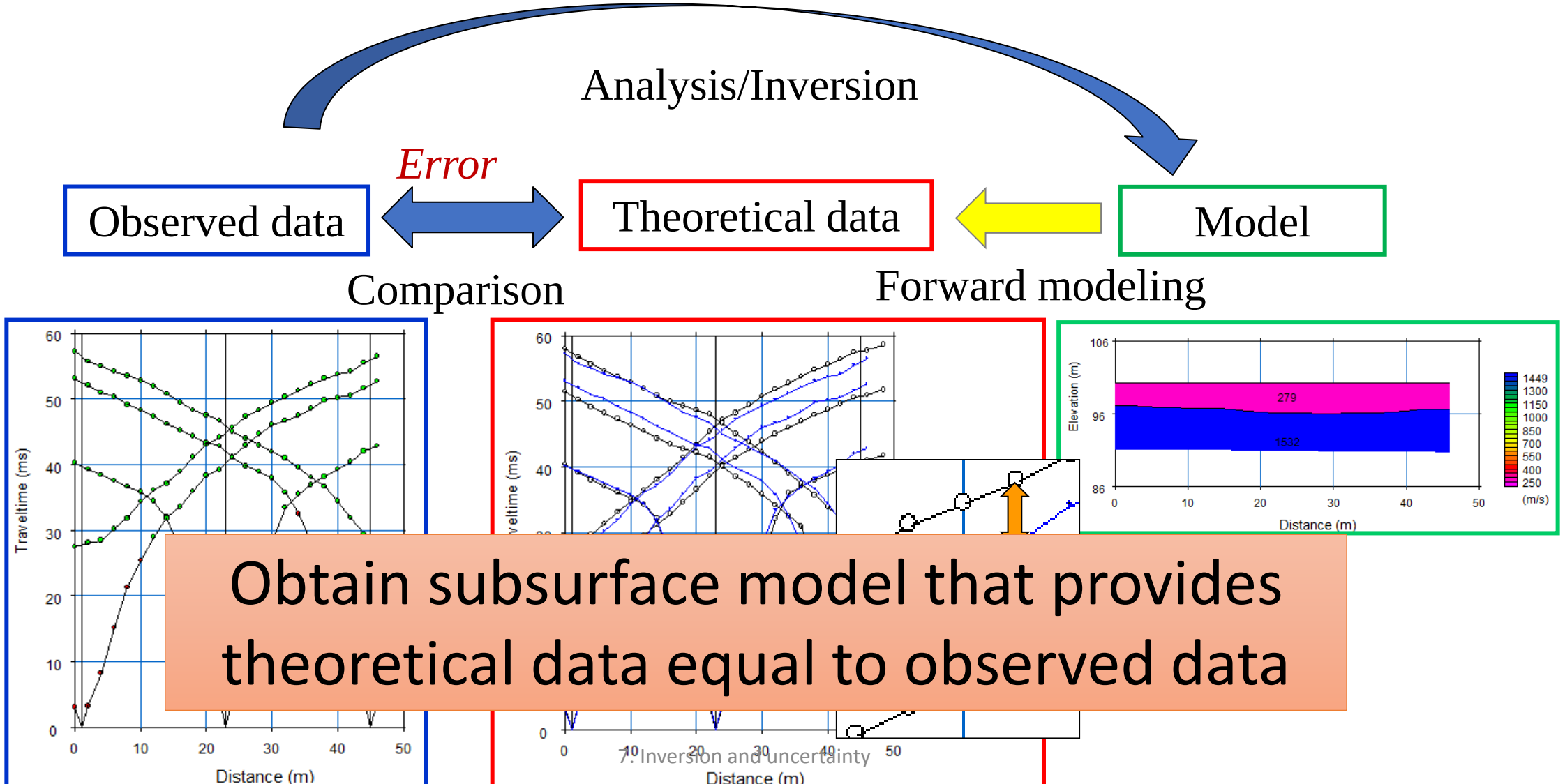
Data



Inverted result



What is inversion in geophysical method ?

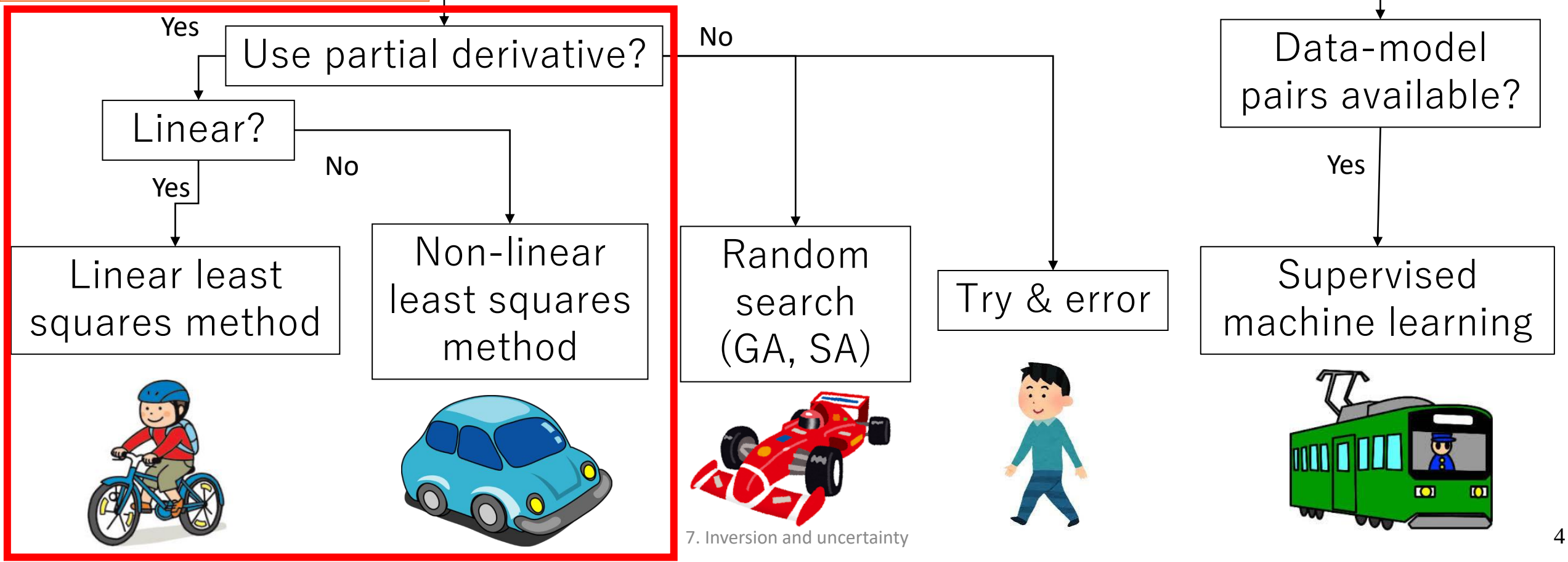


Obtain subsurface model that provides theoretical data equal to observed data

Inversion and Supervised Machine Learning

The important thing is to come to correct destination and how to come to there is not the issue

Sometimes a bicycle is better than an airplane or rocket

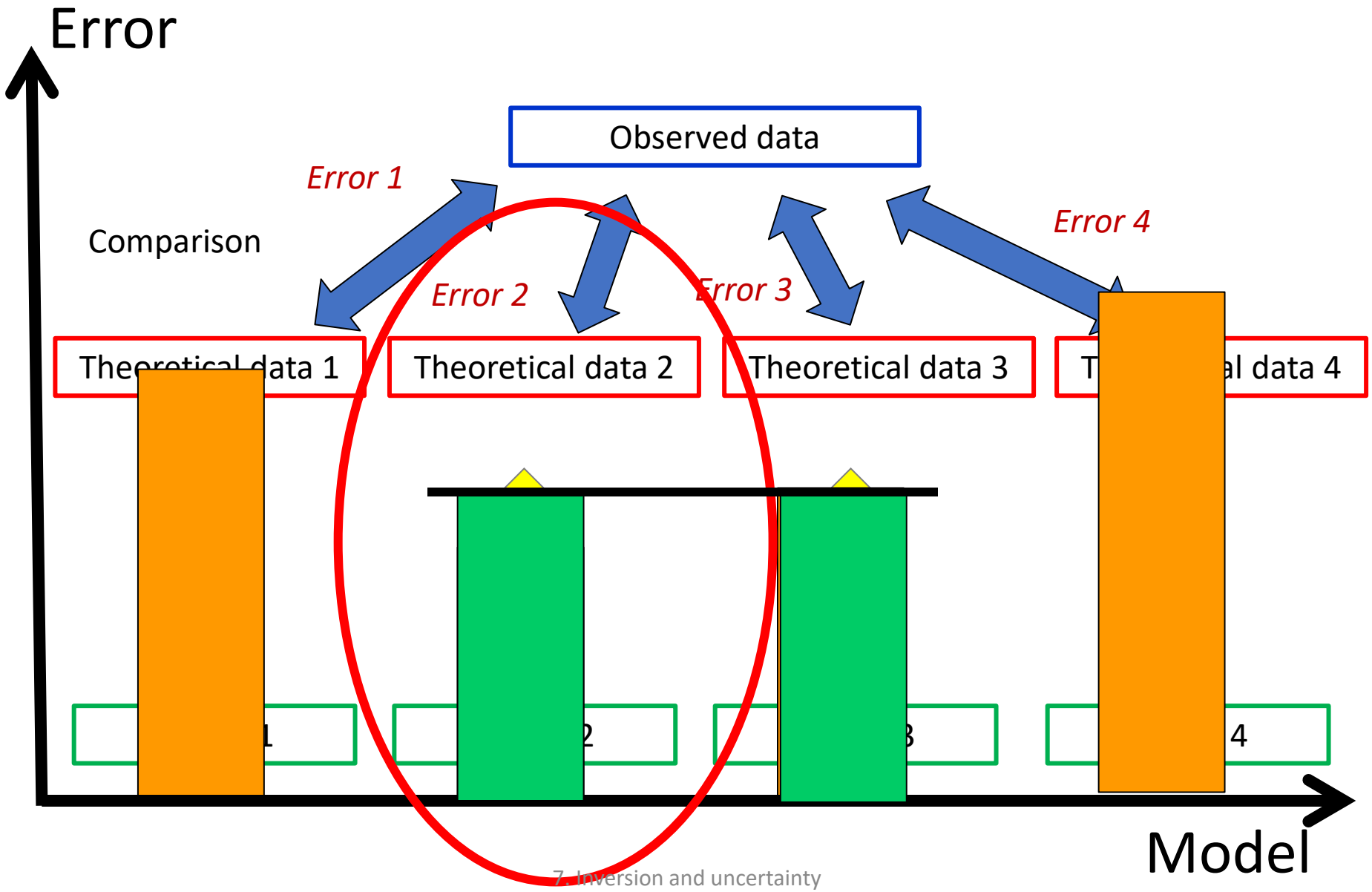


Cause of error or uncertainty

- Geophysical issues
  Non-uniqueness
  Model parameterization
  Dependence on initial conditions
  Constraints

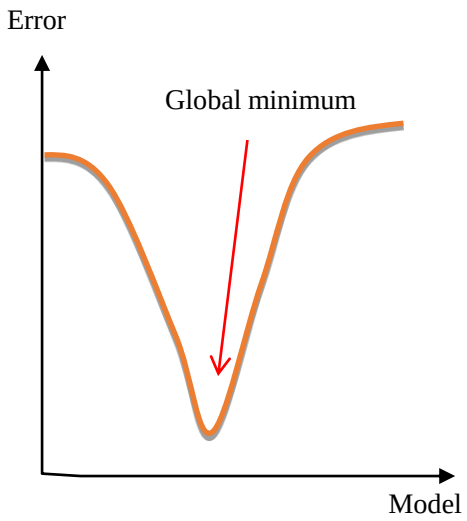
- Engineering issues
  Difference of resolution
  Difference between geophysical properties and engineering parameters

Analysis of geophysical data and its non-uniqueness

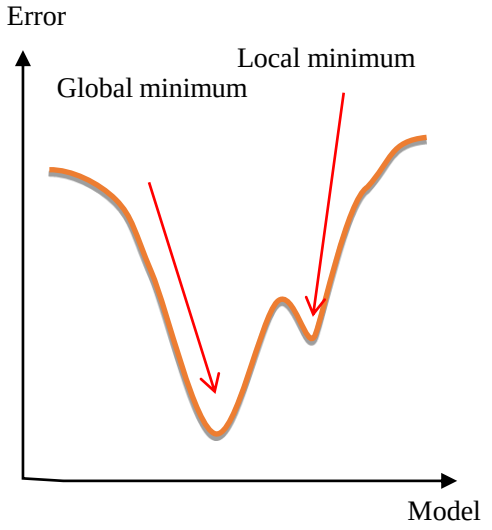


Non-uniqueness

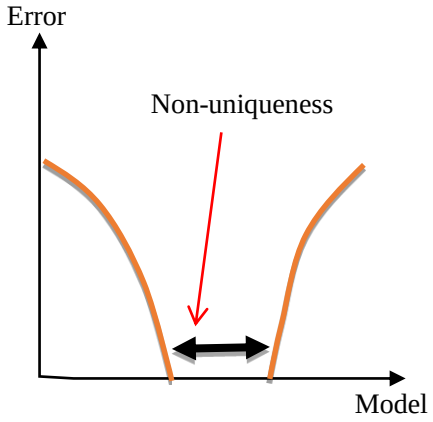
a) Global minimum



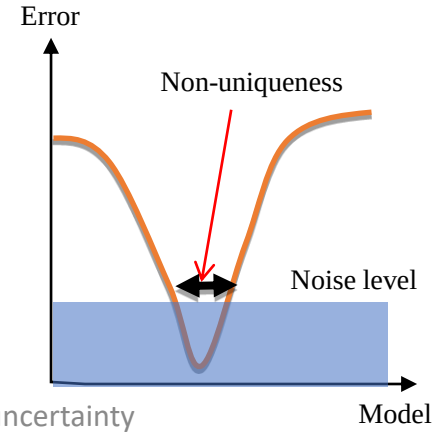
b) Local minimum



c) Non-uniqueness

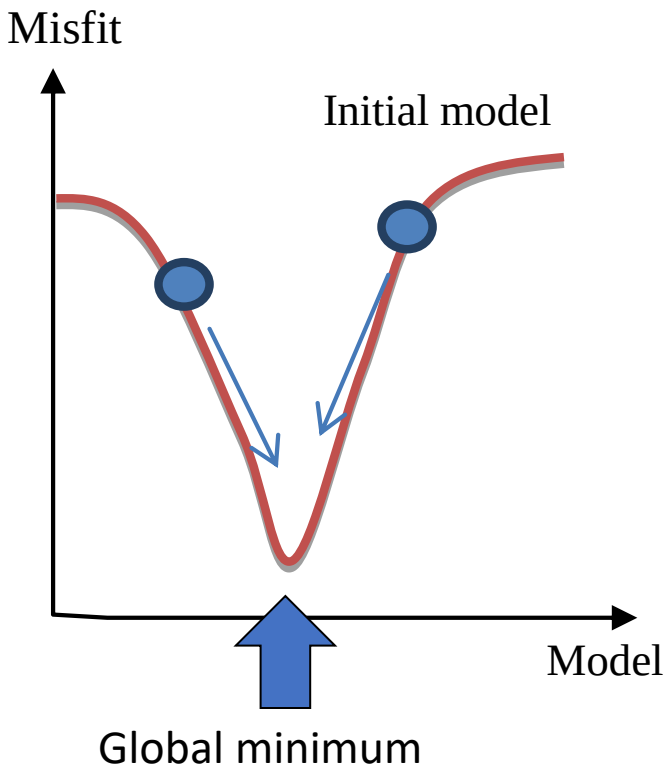


d) Non-uniqueness with noise

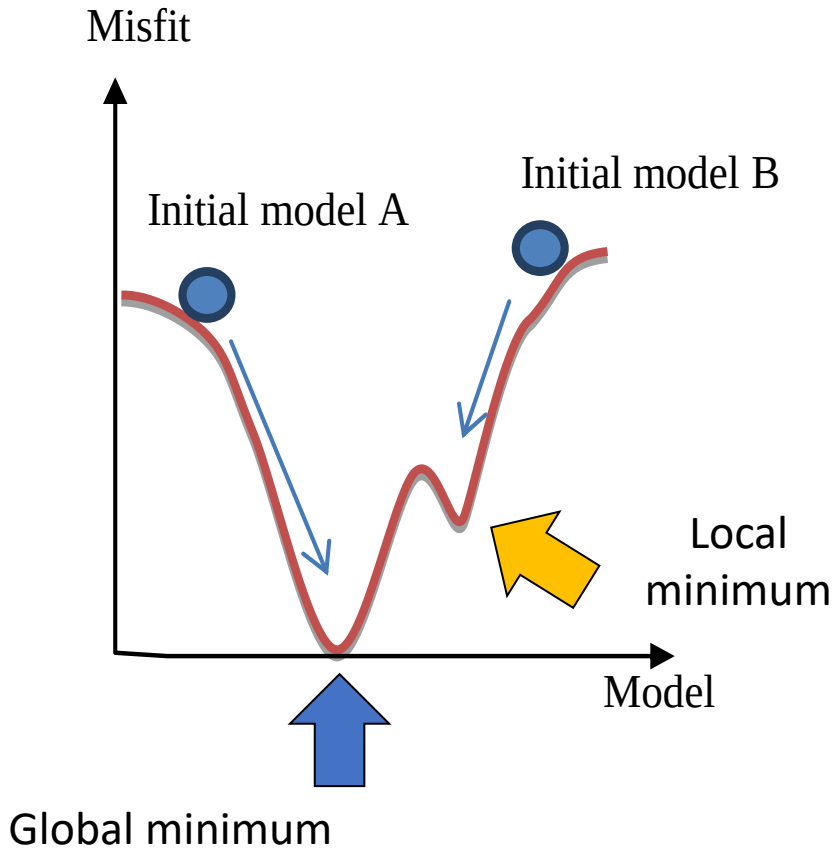


Dependence on initial conditions

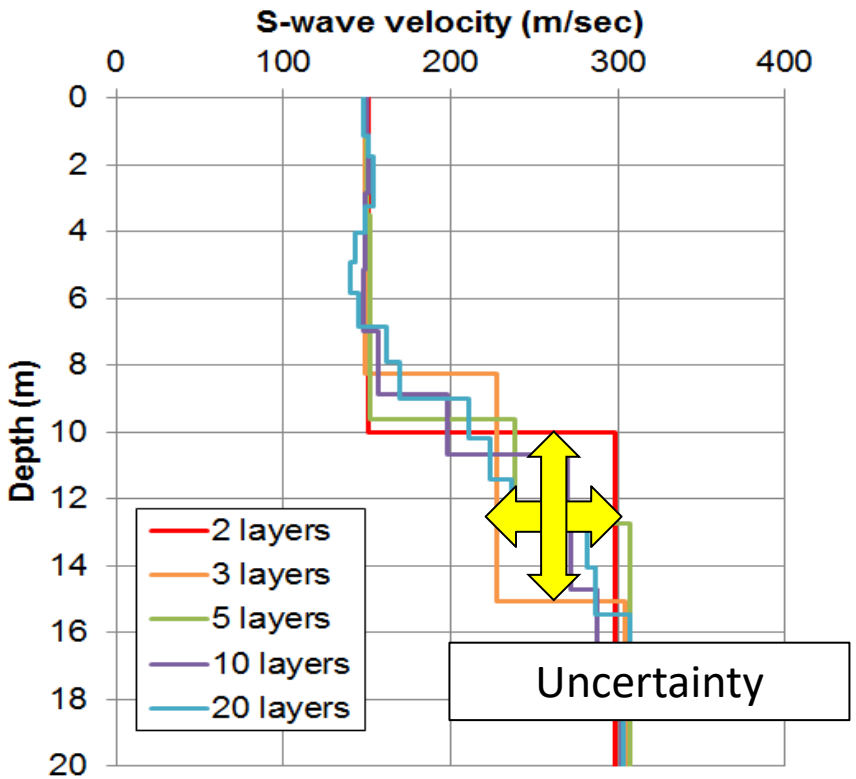
a) Global minimum

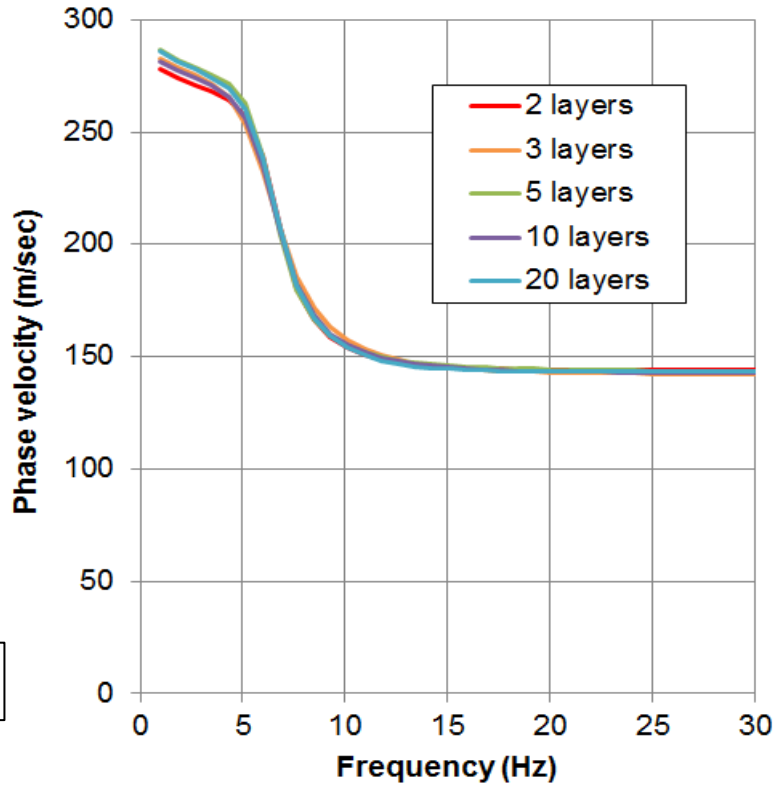


b) Local minimum

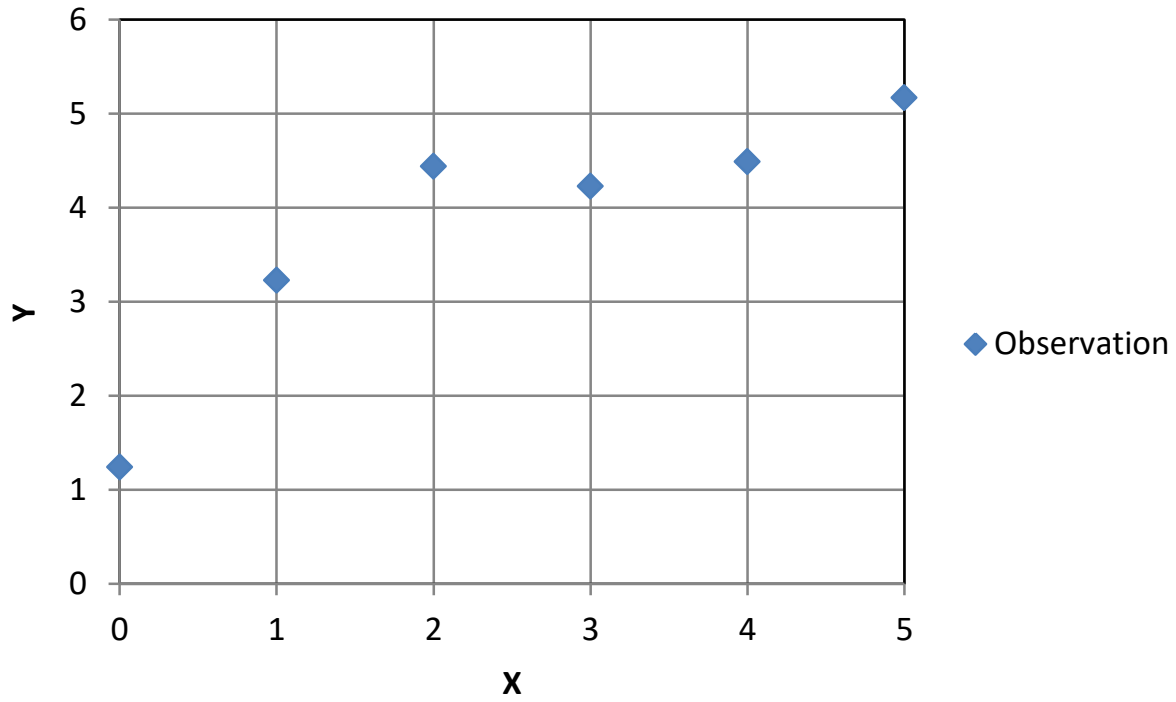


Non-uniqueness in surface wave data analysis





Model parameterization
Curve fitting by polynomial equation

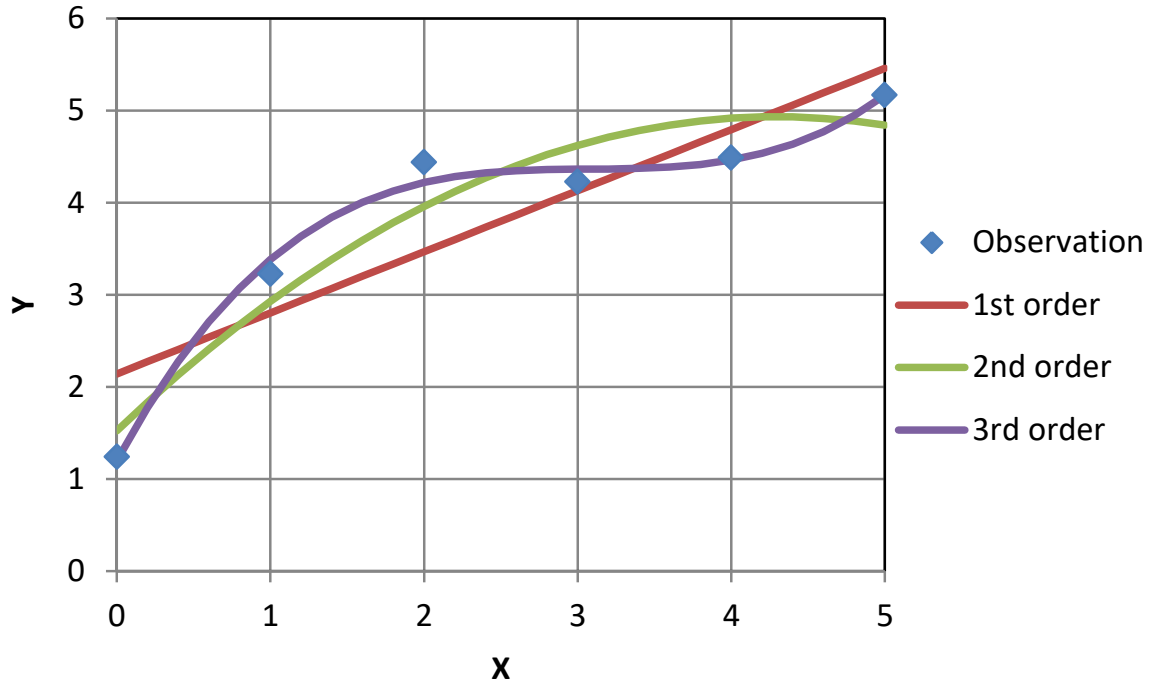


1st order : $y = ax + b$

2nd order : $y = ax^2 + bx + c$

3rd order : $y = ax^3 + bx^2 + cx + d$

Model parameterization
Curve fitting by polynomial equation



1st order : $y = ax + b$

2nd order: $y = ax^2 + bx + c$

3rd order : $y = ax^3 + bx^2 + cx + d$

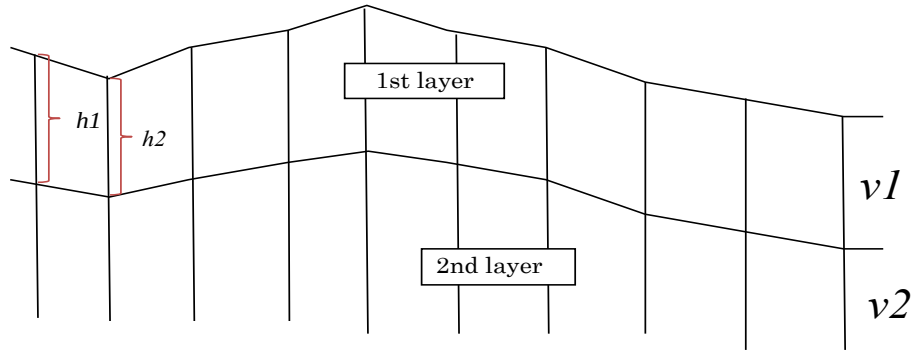
Model parameterization in seismic refraction

a) 2 layer model

Unknown parameters are;

$h_1, h_2 \ldots$

v_1, v_2



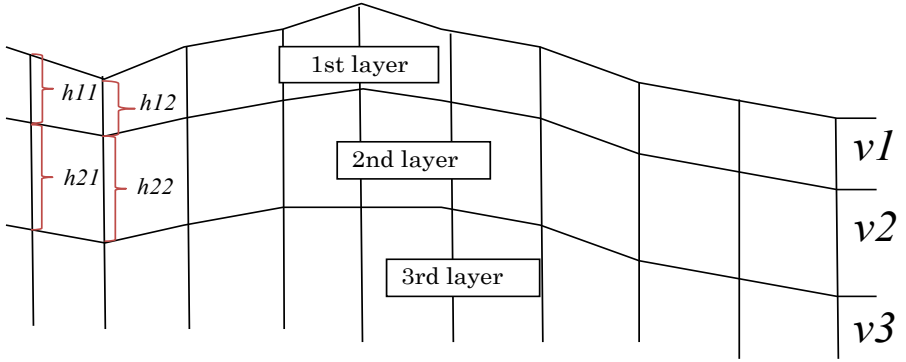
b) 3 layer model

Unknown parameters are;

$h_{11}, h_{12} \ldots$

$h_{21}, h_{22} \ldots$

v_1, v_2, v_3



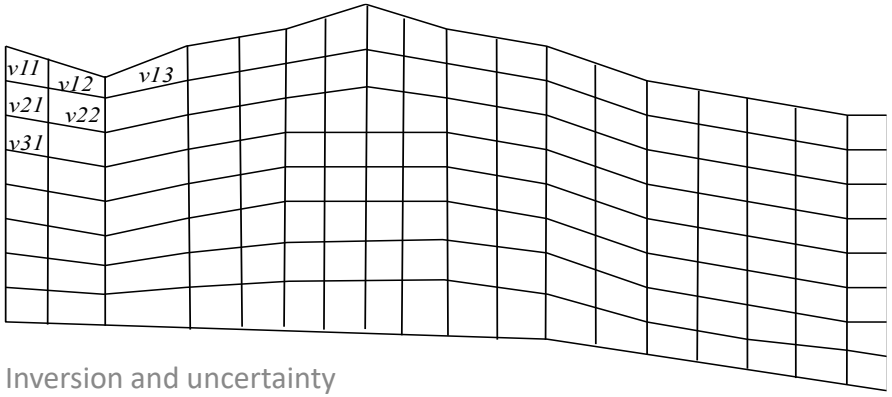
c) Tomographic model

Unknown parameters are;

$v_{11}, v_{12}, v_{13} \ldots$

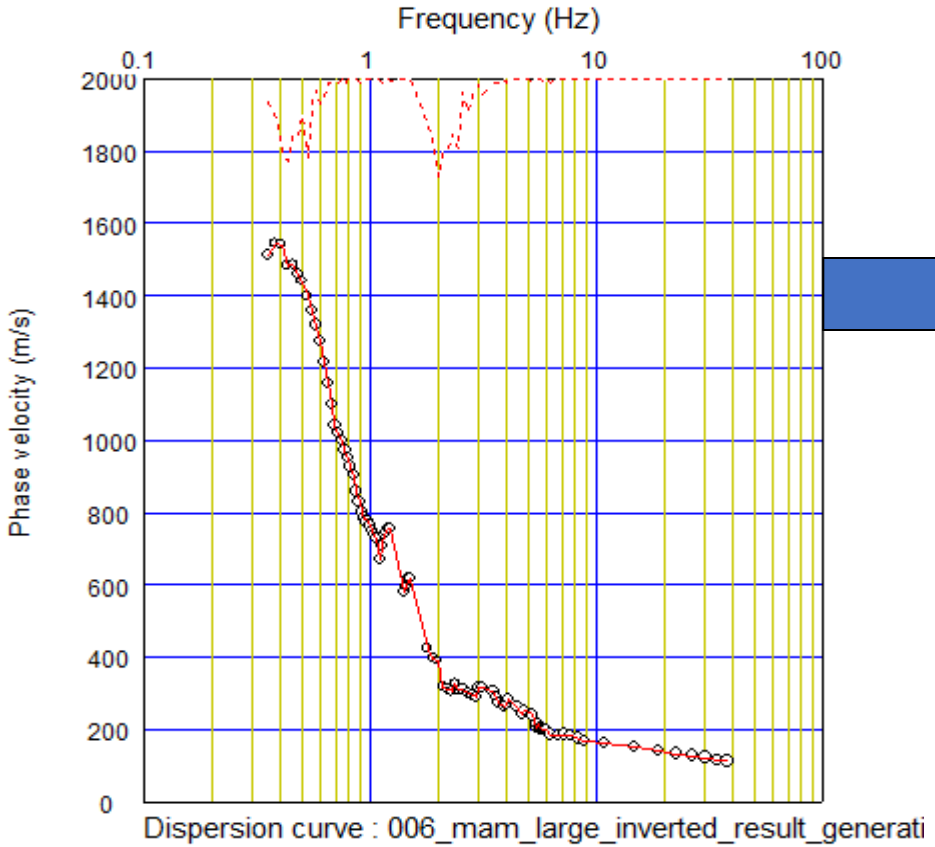
$v_{21}, v_{22} \ldots$

$v_{31} \ldots$

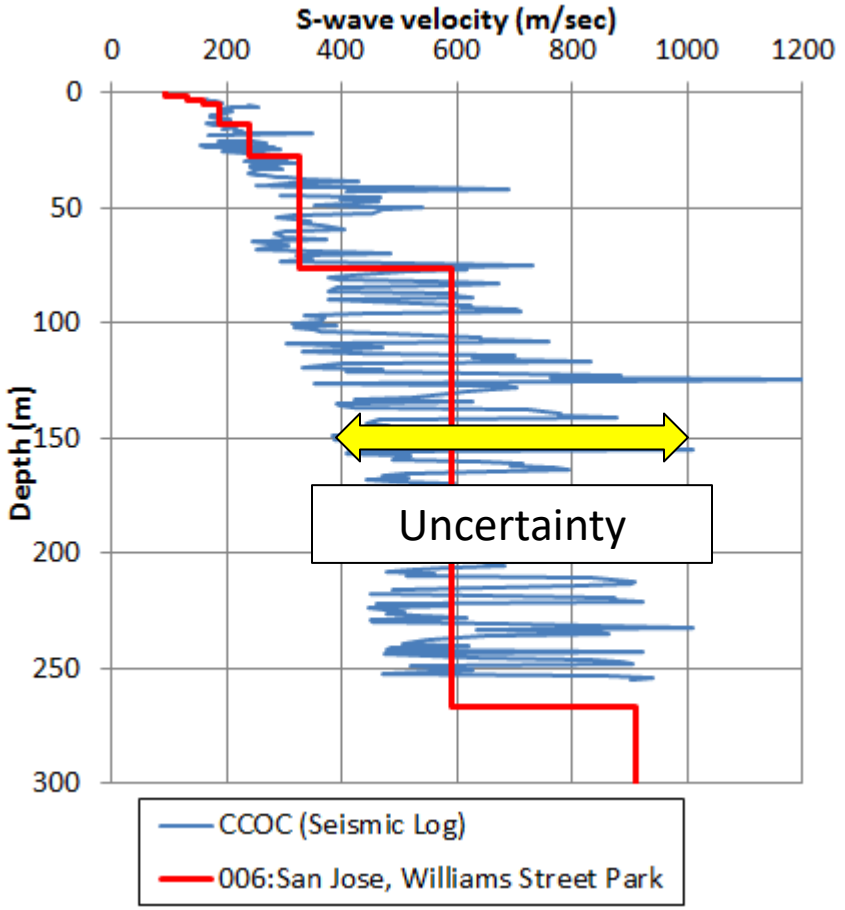


Uncertainty due to the difference in resolution

Observed data from
passive surface wave method



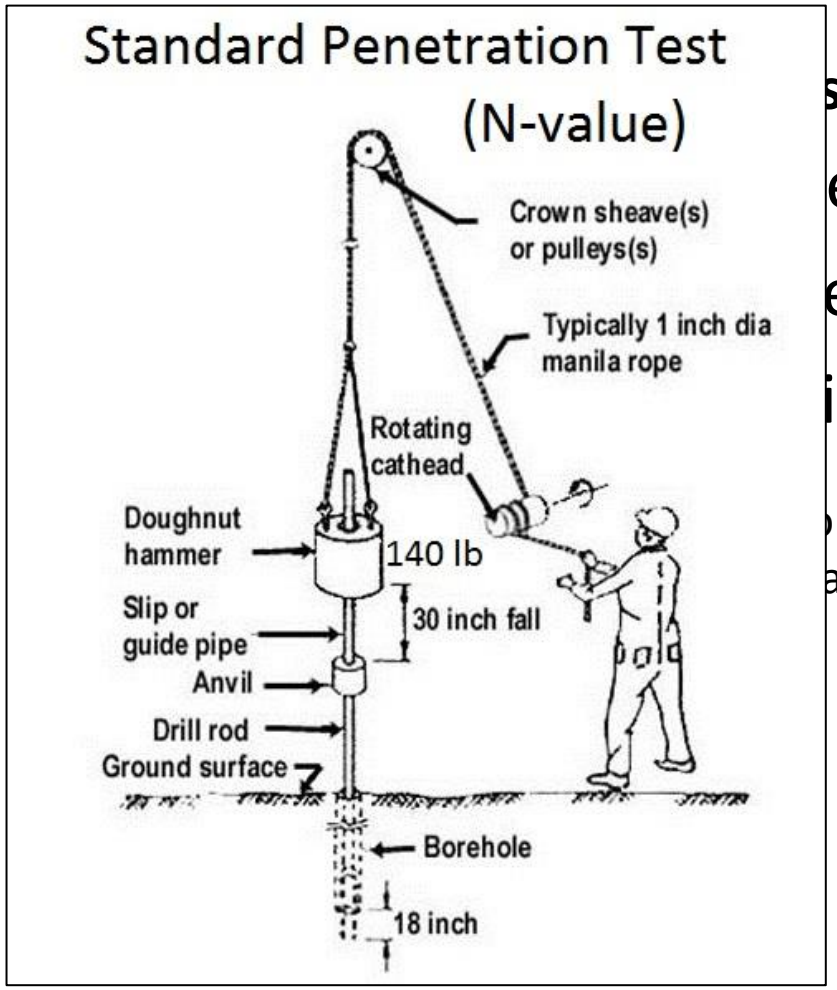
S-wave velocity model



Comparison with Seismic Log

Differences between geophysical properties and engineering parameters

Geophysical properties do not directly relate to engineering properties !



sical properties

e velocity

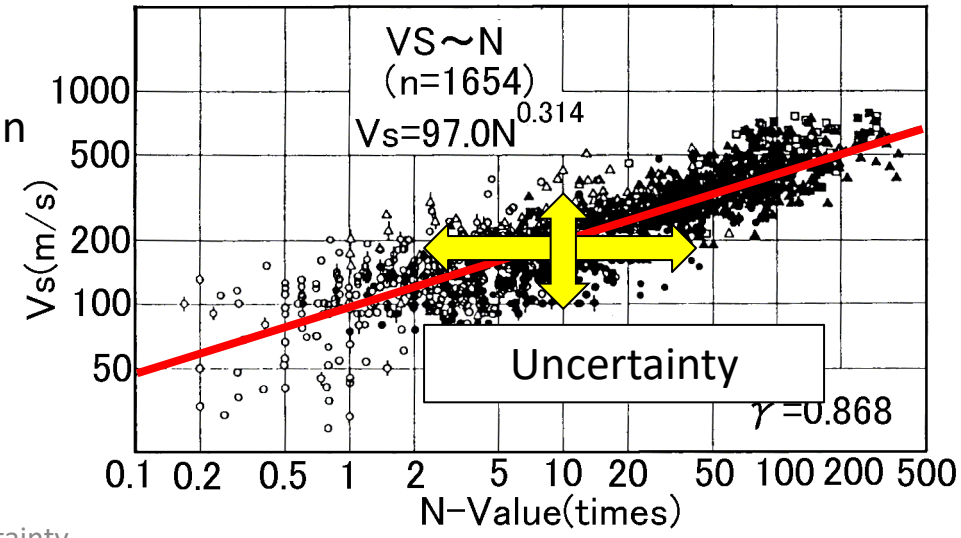
e velocity

ivity

on of S-wave velocity and N-
ained by Standard Penetration

Engineering parameters

- Cohesion and internal friction angle
- Grain size distribution
- Permeability



Example(Q1)

$$2x_1 + x_2 = 11$$
$$4x_1 + x_2 = 17$$
$$6x_1 + x_2 = 23$$

$$\downarrow$$

$$\left(A^T A\right)X = \begin{pmatrix} 2 & 4 & 6 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 4 & 1 \\ 6 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 & 4 & 6 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} 11 \\ 17 \\ 23 \end{pmatrix} = A^T Y$$

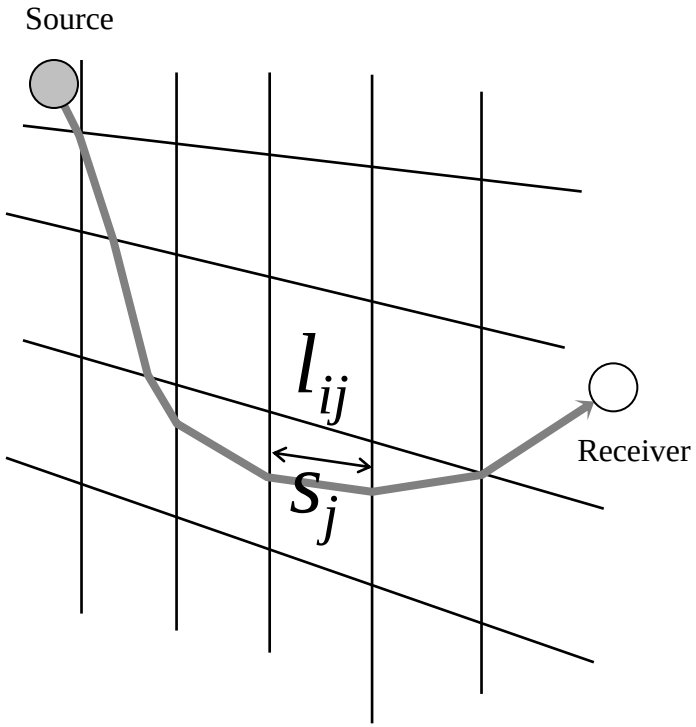
$$\downarrow$$

$$\left(A^T A\right)X = \begin{pmatrix} 56 & 12 \\ 12 & 3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 228 \\ 51 \end{pmatrix} = A^T Y$$ Normal equation

$$\downarrow$$

$$X = \left(A^T A\right)^{-1} A^T Y = \begin{pmatrix} 0.125 & -0.5 \\ -0.5 & 2.3333 \end{pmatrix}\begin{pmatrix} 228 \\ 51 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

Traveltime tomography
(Raypath and traveltime discretization)



$$t_i = \int_X \frac{dX}{\upsilon(X)} = \int_X s(X)dX$$

discretization

$$t_i = s_1 l_{i1} + s_2 l_{i2} + s_3 l_{i3} + s_4 l_{i4} + \cdots + s_N l_{iN}$$

$$t_i = \sum_{j=1}^{N} s_j l_{ij}$$

$$s = \frac{1}{\upsilon}$$

υ:velocity
s:slowness

Traveltime tomography

M simultaneous equations(M traveltime, N unknown)

$$
\begin{aligned}
t_1 &= l_{11}s_1 + l_{12}s_2 + \cdots + l_{1N}s_N \\
t_2 &= l_{21}s_1 + l_{22}s_2 + \cdots + l_{2N}s_N \\
t_3 &= l_{31}s_1 + l_{32}s_2 + \cdots + l_{3N}s_N \\
&\vdots \\
t_M &= l_{M1}s_1 + l_{M2}s_2 + \cdots + l_{MN}s_N
\end{aligned}
$$

Matrix notation

$$
LS = \begin{pmatrix} l_{11} & l_{12} & \cdot & l_{1N} \\ l_{21} & l_{22} & \cdot & l_{2N} \\ l_{31} & l_{32} & \cdot & l_{3N} \\ \cdot & \cdot & \cdot & \cdot \\ l_{M1} & l_{M2} & \cdot & l_{MN} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \cdot \\ s_N \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ t_M \end{pmatrix} = T
$$

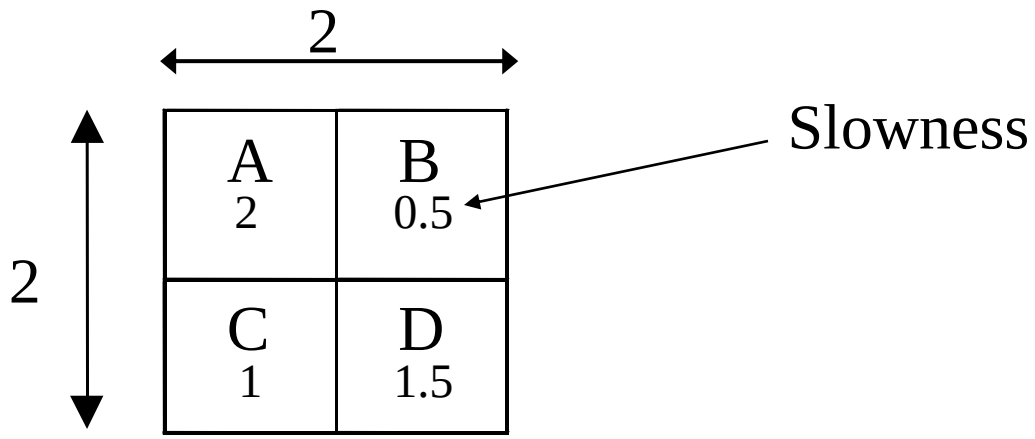
Raypaths model Traveltime

→ Least Square Method

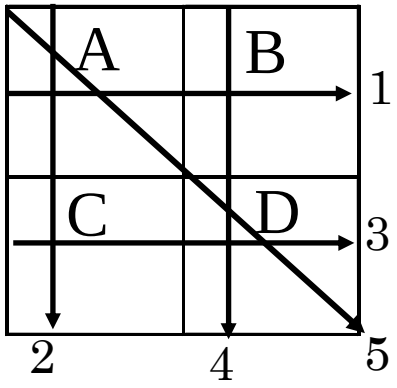
Generally M>N

Traveltime tomography example(Q2)

4 cells (unknown)



5 ray-path (data)



Observed traveltime

$$T = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{pmatrix} = \begin{pmatrix} 2+0.5 \\ 2+1 \\ 1+1.5 \\ 0.5+1.5 \\ 2\sqrt{2}+1.5\sqrt{2} \end{pmatrix} = \begin{pmatrix} 2.5 \\ 3 \\ 2.5 \\ 2 \\ 4.949747 \end{pmatrix}$$

Jacobian matrix A (Ray length passing through each cell)

$$L = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix}$$

$$t_i = s_1 l_{i1} + s_2 l_{i2} + s_3 l_{i3} + s_4 l_{i4} + \cdots + s_N l_{iN}$$

$$\frac{\partial t_i}{\partial s_j} = l_{ij}$$

Equation to be solved

$$LS = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} = \begin{pmatrix} 2.5 \\ 3 \\ 2.5 \\ 2 \\ 4.949747 \end{pmatrix} = T$$

Normal equation

$$L^T LS = \begin{pmatrix} 4 & 1 & 1 & 2 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 2 & 1 & 1 & 4 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} = \begin{pmatrix} 12.5 \\ 4.5 \\ 5.5 \\ 11.5 \end{pmatrix} = L^T T$$

Solve it !

$$S^T = \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \end{pmatrix} = \begin{pmatrix} 2 & 0.5 & 1 & 1.5 \end{pmatrix}$$

Solve traveltime tomography (example 2) by python

Tomography.ipynb

```
import numpy as np

A=np.array([[1,1,0,0],[1,0,1,0],[0,0,1,1],[0,1,0,1],[np.sqrt(2),0,0,np.sqrt(2)]])
B=np.array([[2.5],[3],[2.5],[2],[3.5*np.sqrt(2)]])

print(A)
print(B)

AT=np.transpose(A)
ATA=np.matmul(AT,A)
ATB=np.matmul(AT,B)
ATAI=np.linalg.inv(ATA)
X=np.matmul(ATAI,ATB)

print(X)
```

Results

```
[[1. 1. 0. 0. ]
 [1. 0. 1. 0. ]
 [0. 0. 1. 1. ]
 [0. 1. 0. 1. ]
 [1.41421356 0. 0. 1.41421356]]

[[2.5 ]
 [3. ]
 [2.5 ]
 [2. ]
 [4.94974747]]

[[2. ]
 [0.5]
 [1. ]
 [1.5]]
```

Normal equation

$$(A^T A)X = A^T B$$

Non-linear least square method

Jacobian matrix requires ray-path

↓

Ray-path can not be calculated with out a velocity model !

↓

Can not solve at once

↓

Non-linear least square method

Non-linear least square method

If the Jacobian matrix is not a constant,

$$y(Z) = x_1 Z - x_2 e^{-Z x_3}$$ Estimate x_1, x_2, x_3.

$$A = \begin{pmatrix} \frac{\partial y(z_1)}{\partial x_1} & \frac{\partial y(z_1)}{\partial x_2} & \frac{\partial y(z_1)}{\partial x_3} \\ \frac{\partial y(z_2)}{\partial x_1} & \frac{\partial y(z_2)}{\partial x_2} & \frac{\partial y(z_2)}{\partial x_3} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \frac{\partial y(z_m)}{\partial x_1} & \frac{\partial y(z_m)}{\partial x_2} & \frac{\partial y(z_m)}{\partial x_3} \end{pmatrix} = \begin{pmatrix} z_1 & -e^{-Z_1 x_3} & -x_2 Z_1 e^{-Z_1 x_3} \\ z_2 & -e^{-Z_2 x_3} & -x_2 Z_2 e^{-Z_2 x_3} \\ & & \\ & & \\ z_m & -e^{-Z_m x_3} & -x_2 Z_m e^{-Z_m x_3} \end{pmatrix}$$

Unknown parameter x_2 and x_3 is in the Jacobian matrix A!

Iterative solution of non-linear least square method

1 : Calculate theoretical value Y_0 for initial value X_0.

$$Y_0(Z) = Y(Z, X_0)$$

2 : Calculate residuals (ΔY) between theoretical value Y_0 and observed value Y.

$$\Delta Y = Y - Y_0$$

3 : Calculate correction value for X (ΔY) by the least square method.

$$\left(A^T A\right)\Delta X = A^T \Delta Y$$

4 : Calculate new estimate X_1.

$$X_1 = X_0 + \Delta X$$

5 : Return to step 1.

6 : Stop if the residuals are enough small.

Example (Q3)

Model $$y(Z) = x_1 Z - x_2 e^{-Z x_3}$$

True solution(answer) $$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Eleven observed data

z	y(z)
0	-2
1	0.264241
2	1.729329
3	2.900426
4	3.963369
5	4.986524
6	5.995042
7	6.998176
8	7.999329
9	8.999753
10	9.999909

Partial differential $\frac{\partial y}{\partial x_1} = Z \qquad \frac{\partial y}{\partial x_2} = -e^{-Zx_3} \qquad \frac{\partial y}{\partial x_3} = x_2 Z e^{-Zx_3}$

Initial model $X_0 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix}$

Jacobian matrix A

$$A_0 = \begin{pmatrix} \frac{\partial y(z_1)}{\partial x_1} & \frac{\partial y(z_1)}{\partial x_2} & \frac{\partial y(z_1)}{\partial x_3} \\ \frac{\partial y(z_2)}{\partial x_1} & \frac{\partial y(z_2)}{\partial x_2} & \frac{\partial y(z_2)}{\partial x_3} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \frac{\partial y(z_{11})}{\partial x_1} & \frac{\partial y(z_{11})}{\partial x_2} & \frac{\partial y(z_{11})}{\partial x_3} \end{pmatrix} = \begin{pmatrix} z_1 & -e^{-Z_1 x_3} & -x_2 Z_1 e^{-Z_1 x_3} \\ z_2 & -e^{-Z_2 x_3} & -x_2 Z_2 e^{-Z_2 x_3} \\ & & \\ & & \\ z_{11} & -e^{-Z_{11} x_3} & -x_2 Z_{11} e^{-Z_{11} x_3} \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & -0.1353352832 & 0.4060058497 \\ 2 & -0.0183156389 & 0.1098938333 \\ 3 & -0.0024787522 & 0.0223087696 \\ 4 & -0.0003354626 & 0.0040255515 \\ 5 & -0.0000453999 & 0.0006809989 \\ 6 & -0.0000061442 & 0.0001105958 \\ 7 & -0.0000008315 & 0.0000174621 \\ 8 & -0.0000001125 & 0.0000027008 \\ 9 & -0.0000000152 & 0.0000004112 \\ 10 & -0.0000000021 & 0.0000000618 \end{pmatrix}$$

Observed data

$$Y^T = \begin{pmatrix} -2.0000 & 0.264241 & 1.729329 & 2.900426 & 3.963369 & 4.986524 & 5.995042 & 6.998176 & 7.999329 & 8.999753 & 9.999909 \end{pmatrix}$$

Theoretical data for initial the model

$$Y_0^T = \begin{pmatrix} -3.0000 & 1.5940 & 3.9451 & 5.9926 & 7.9990 & 9.9999 & 12.0000 & 14.0000 & 16.0000 & 18.0000 & 20.0000 \end{pmatrix}$$

Residual vector $\Delta Y = Y_0 - Y$

$$\Delta Y_0^T = \begin{pmatrix} -1.0000 & 1.3298 & 2.2157 & 3.0921 & 4.0356 & 5.0133 & 6.0049 & 7.0018 & 8.0007 & 9.0002 & 10.0001 \end{pmatrix}$$

RMSE(Root Mean Square Error)

$$RMSE_0 = \sqrt{\frac{\Delta Y_0^T \Delta Y_0}{11}} = 5.9449$$

$$A_0^T A_0 = \begin{pmatrix} 385 & -0.181 & 0.71304 \\ -0.181 & 1.0187 & -0.057 \\ 0.71304 & -0.057 & 0.17743 \end{pmatrix} \qquad A_0^T \Delta Y_0 = \begin{pmatrix} 386.3 \\ 0.7702 \\ 0.8728 \end{pmatrix}$$

Solve $\left(A_0^T A_0\right)\Delta X_0 = A_0^T \Delta Y_0$ get $\Delta X_0 = \begin{pmatrix} 1.0016 \\ 1.0021 \\ 1.2162 \end{pmatrix}$

New estimated value for X (X_1) $X_1 = X_0 - \Delta X$

$$X_1 = \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix} - \begin{pmatrix} 1.0016 \\ 1.0021 \\ 1.2162 \end{pmatrix} = \begin{pmatrix} 0.9984 \\ 1.9979 \\ 0.7838 \end{pmatrix}$$

Calculate residuals (RMSE) from new estimation of X (X_1)

$$RMSE_1 = \sqrt{\frac{\Delta Y_1^T \Delta Y_1}{11}} = 0.0793$$

In the 2nd calculation

$$A_1^T A_1 = \begin{pmatrix} 385 & -1.543 & 8.19332 \\ -1.543 & 1.2635 & -0.6652 \\ 8.19332 & -0.6652 & 2.02955 \end{pmatrix} \quad A_1^T \Delta Y_1 = \begin{pmatrix} -1.854 \\ 0.123 \\ -0.372 \end{pmatrix}$$

Correction is

$$\Delta X_1 = \begin{pmatrix} -0.001 \\ 0.002 \\ -0.179 \end{pmatrix}$$

Corrected model is

$$X_2 = \begin{pmatrix} 0.9984 \\ 1.9979 \\ 0.7838 \end{pmatrix} - \begin{pmatrix} -0.001 \\ 0.002 \\ -0.179 \end{pmatrix} = \begin{pmatrix} 0.9994 \\ 1.9959 \\ 0.9625 \end{pmatrix} \cong \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Residuals are

$$RMSE_2 = \sqrt{\frac{\Delta Y_2^T \Delta Y_2}{11}} = 0.0122 \cong 0$$

Linear Vs Non-linear

Linear $(A^T A)X = A^T Y$

Non-linear $\left({A_0}^T A_0\right)\Delta X = {A_0}^T \Delta Y$

Inversion in surface wave methods

Unknown parameters (S-wave velocity model)

$$x^T = \left(Vs_1, Vs_2, \cdots, Vs_M\right)$$

Observed data (phase velocities)

$$f_i^{obs}$$

Observed data (phase velocities)

$$f_i^{cal}(x)$$

Objective function

$$\sum_i^N \left(f_i^{obs} - f_i^{cal}\left(Vs_1, Vs_2, \cdots Vs_N\right)\right)^2 = \sum_i^N \left(f_i^{obs} - f_i^{cal}(x)\right)^2 \rightarrow \text{Minimize}$$

Inversion in surface wave methods

Now, $f_i = f_i^{cal}(x)$ $(i = 1 \sim N : N \text{ is the number of observed data})$

Jacobian matrix (a) goes to

$$a = \begin{pmatrix} \frac{\partial f_1}{\partial Vs_1} & \frac{\partial f_1}{\partial Vs_2} & \cdot & \frac{\partial f_1}{\partial Vs_N} \\ \frac{\partial f_2}{\partial Vs_1} & \frac{\partial f_2}{\partial Vs_2} & \cdot & \frac{\partial f_2}{\partial Vs_N} \\ \frac{\partial f_3}{\partial Vs_1} & \frac{\partial f_3}{\partial Vs_2} & \cdot & \frac{\partial f_3}{\partial Vs_N} \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial f_n}{\partial Vs_1} & \frac{\partial f_n}{\partial Vs_2} & \cdot & \frac{\partial f_n}{\partial Vs_N} \end{pmatrix}$$

Unknown parameter x is in the partial differentials !

→ Non-linear problem

↓

Iteration

Inversion in surface wave methods

Residual vector y is

$$y = \begin{pmatrix} f_1^{obs} - f_i^{cal}(x) \\ f_2^{obs} - f_2^{cal}(x) \\ f_3^{obs} - f_3^{cal}(x) \\ \cdot \\ f_N^{obs} - f_N^{cal}(x) \end{pmatrix}$$

Observed

Calculated

Correction vector Δx is,

$$\left(a^T a + \varepsilon I\right)\Delta x = a^T y$$

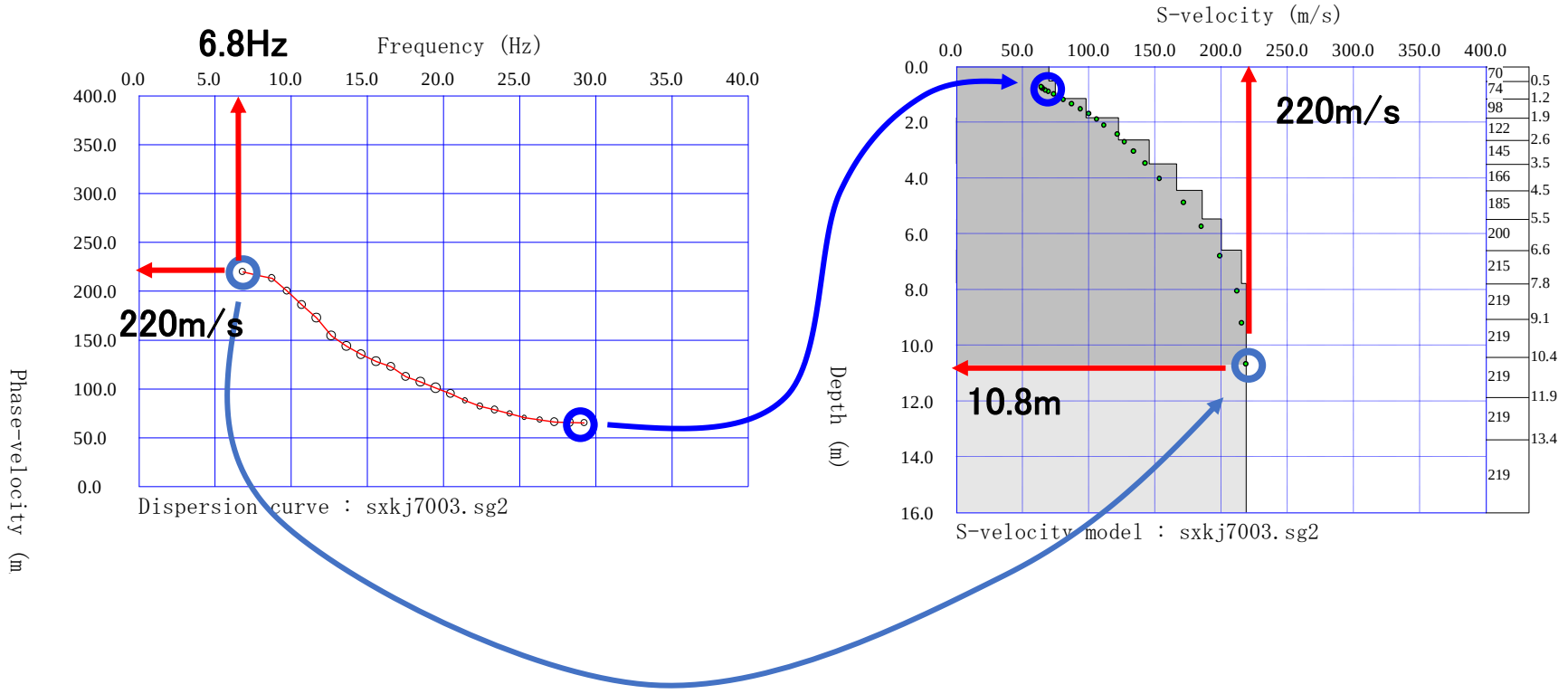
Damping parameter

New model estimation x^{l+1} in l_{th} iteration is,

$$x^{l+1} = x^l + \gamma \Delta x$$

Estimating an initial velocity model in terms of 1/3 wave-length theory

Depth=Phase-velocity/Frequency/3



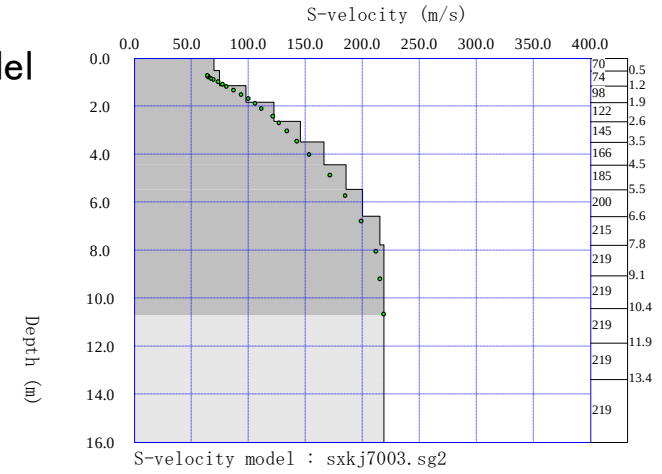
Depth=10.8=220/6.8/3=Phase-velocity/Frequency/3

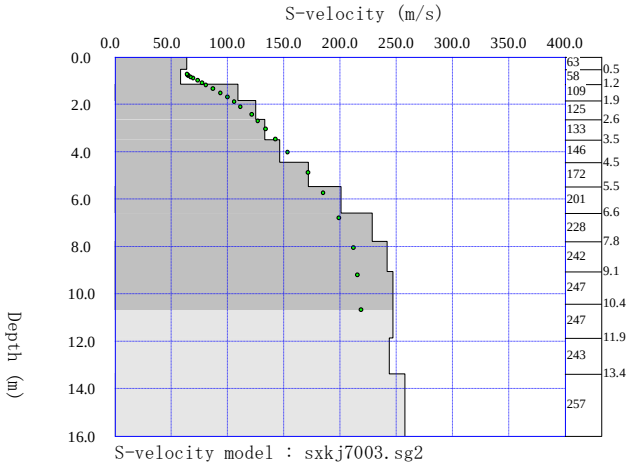
Estimating an initial velocity model in terms of 1/3 wave-length theory

Initial velocity model

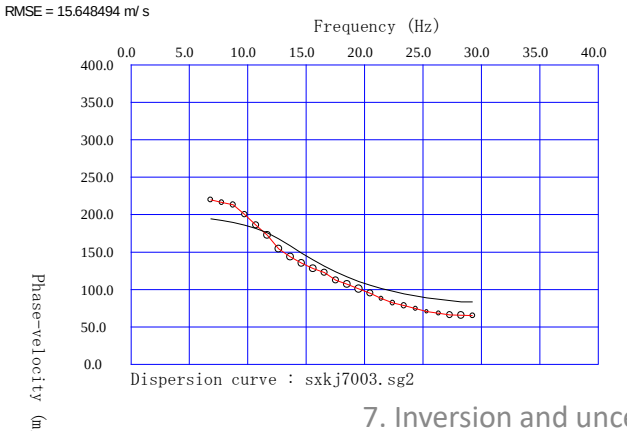
Inverted result

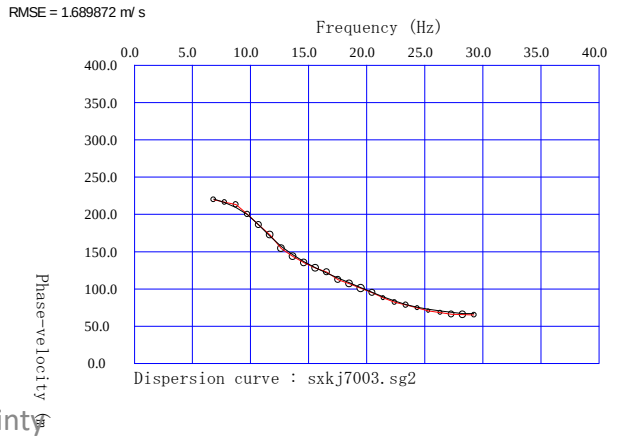
S-wave velocity model





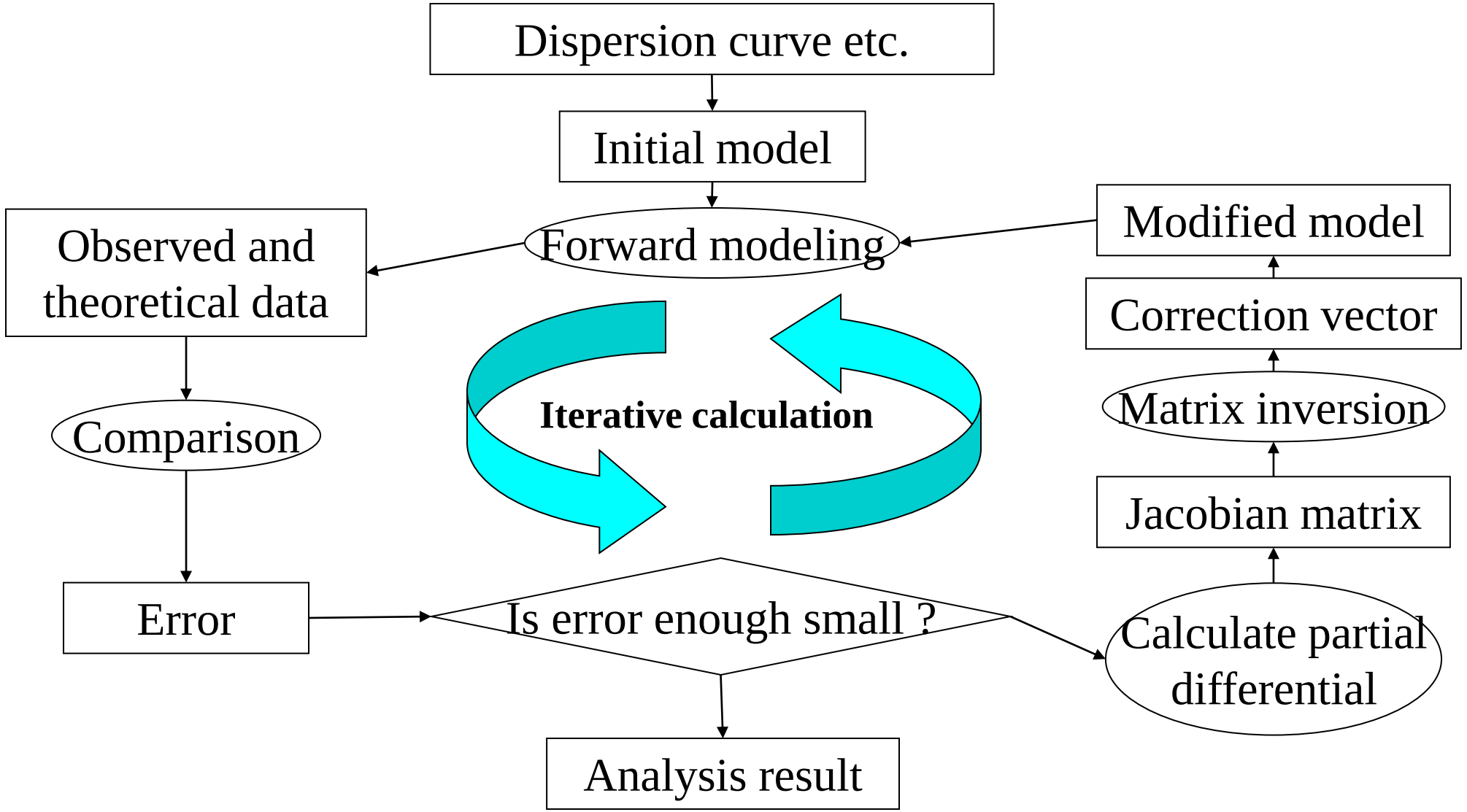
Comparison of theoretical and observed dispersion curves





Inversion in terms of Non-linear Least squares Method

Calculation flow



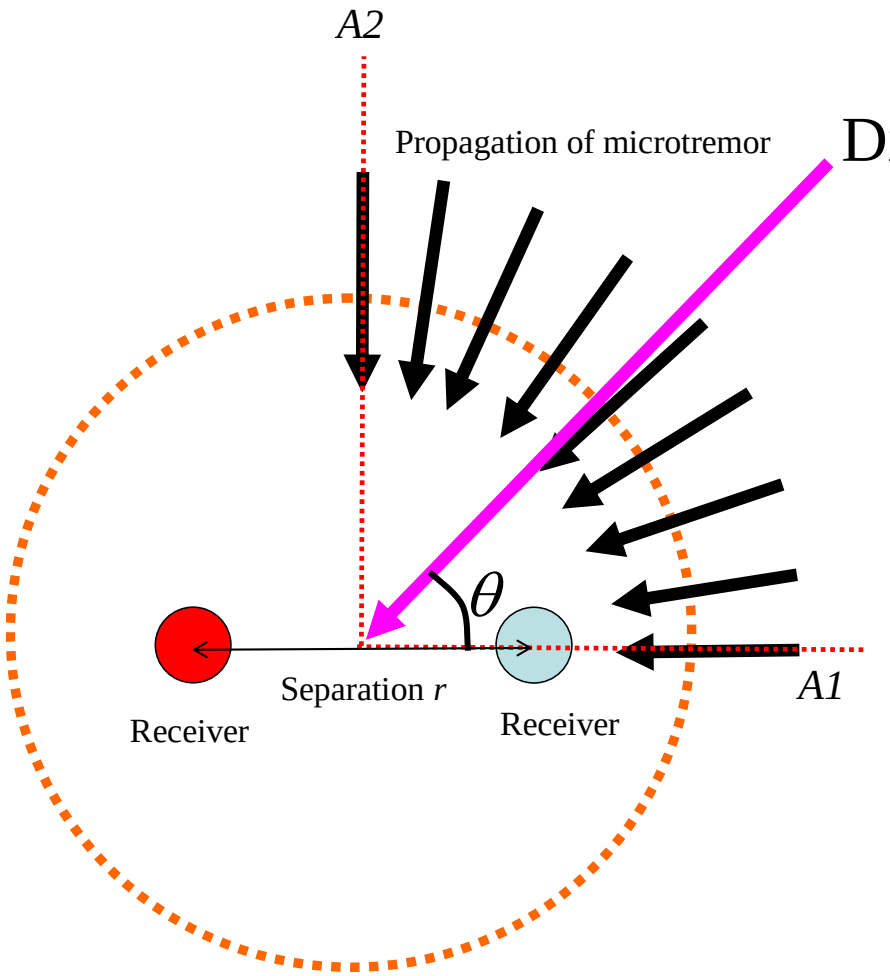
Uncertainties in surface wave investigations

- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

Uncertainties in surface wave investigations

- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

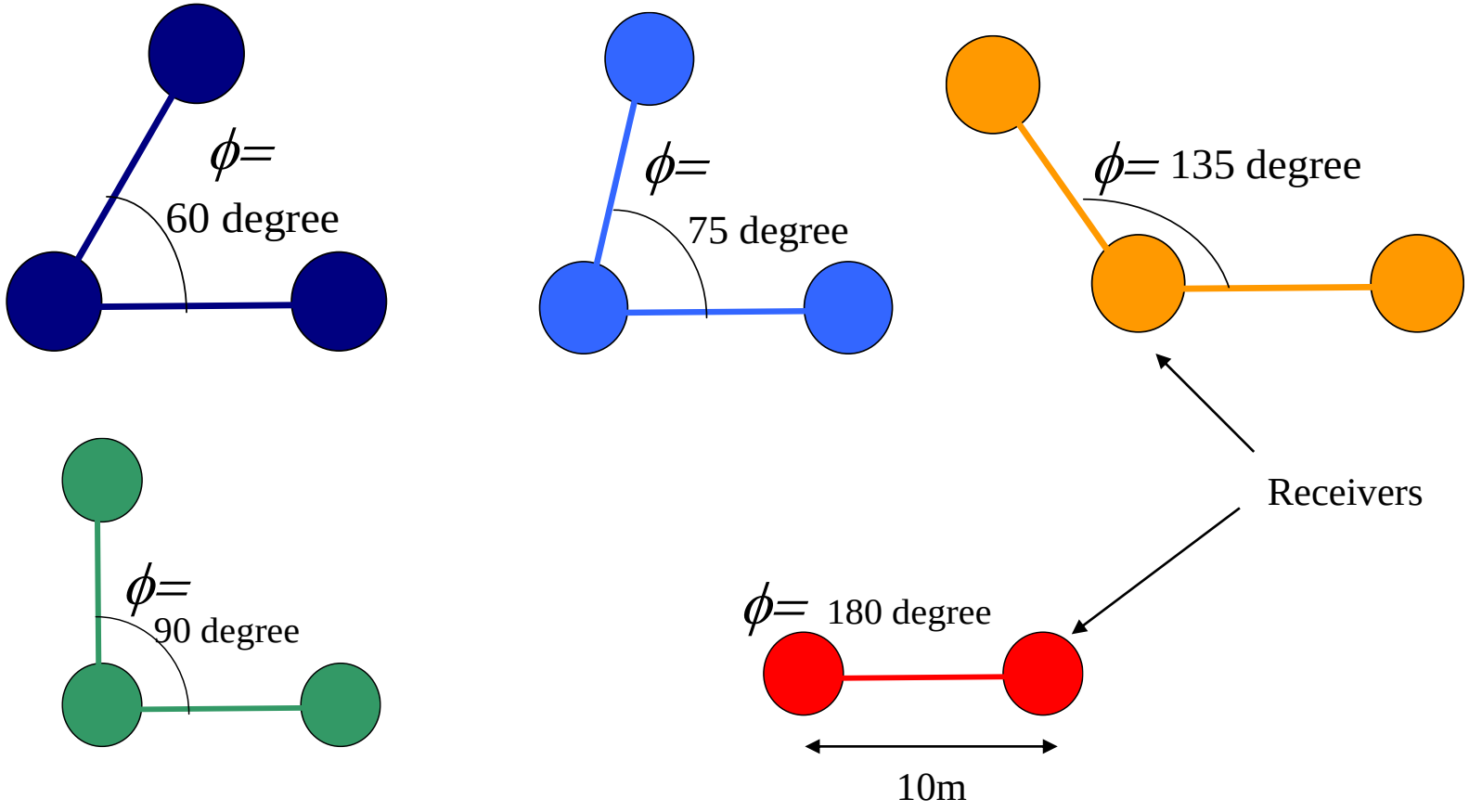
Directional Average for Limited Propagation Direction



$$DA(kr) = \frac{1}{A2 - A1} \int_{\varphi=A1}^{\varphi=A2} e^{ikr\cos\varphi} d\varphi$$

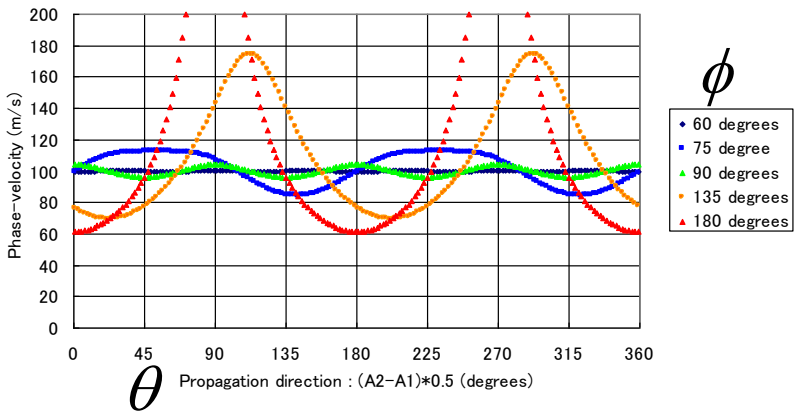
$$c(\omega) = \frac{\omega \cdot r}{J_0^{-1}(DA(kr))}$$

Irregular Arrays for Directional Average

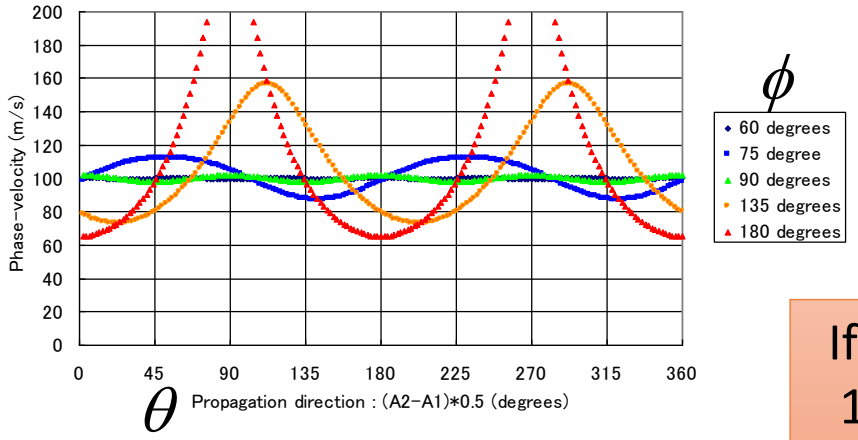


Error Associated with Limited Propagation Direction of Microtremor

A2−A1=30 degrees



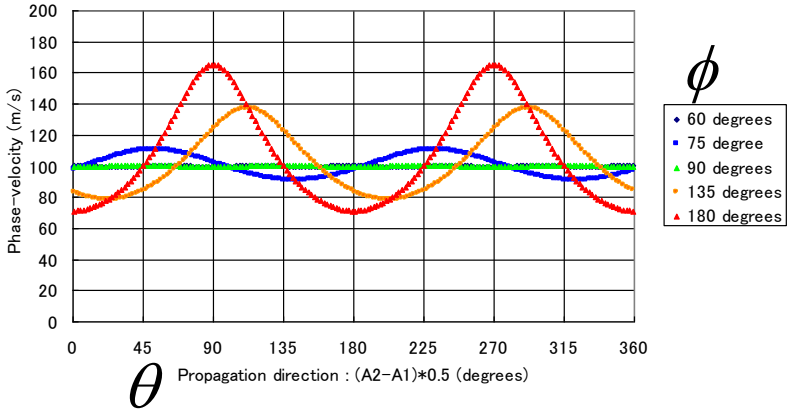
A2−A1=60 degrees



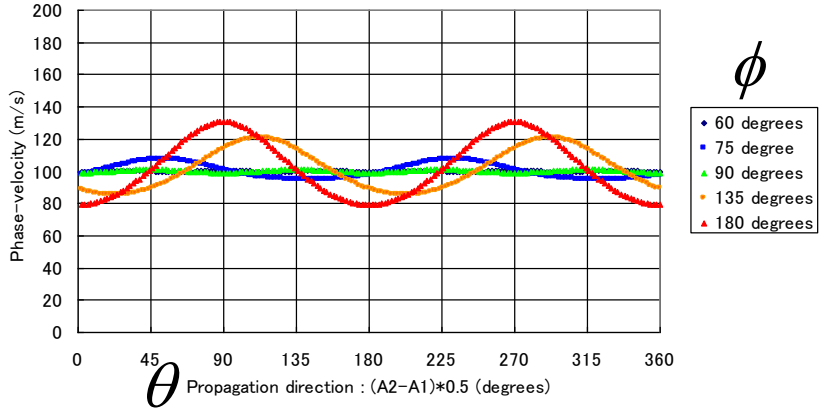
True velocity is 100 m/sec

If ambient noise comes from 120 degrees, error of phase velocity is less than 20 %

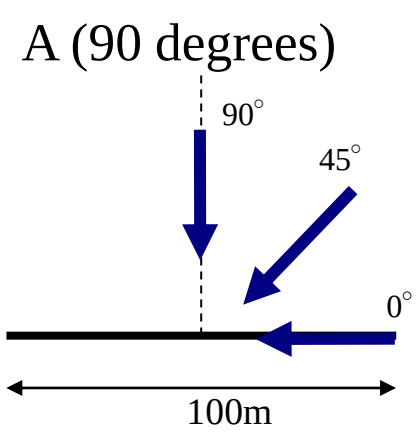
A2−A1=90 degrees



A2−A1=120 degrees

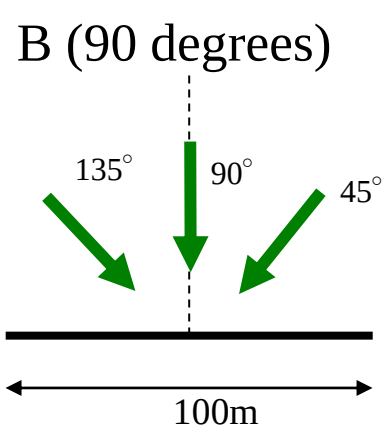


Simulation of Microtremor Measurement Using Linear Array



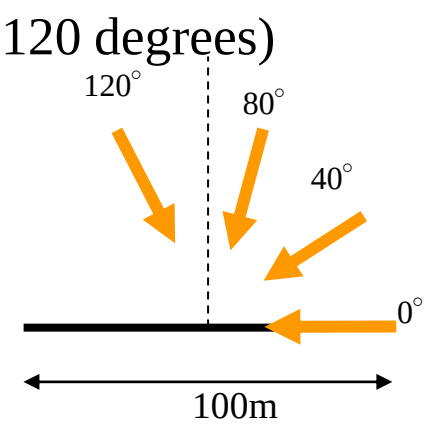
Linear array

3 directions
90 degrees (1)



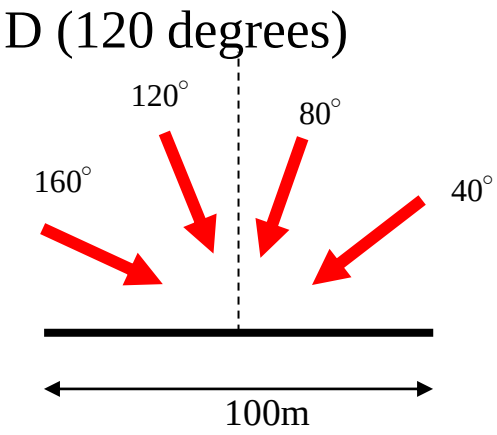
Linear array

3 directions
90 degrees (2)



Linear array

4 directions
120 degrees (1)

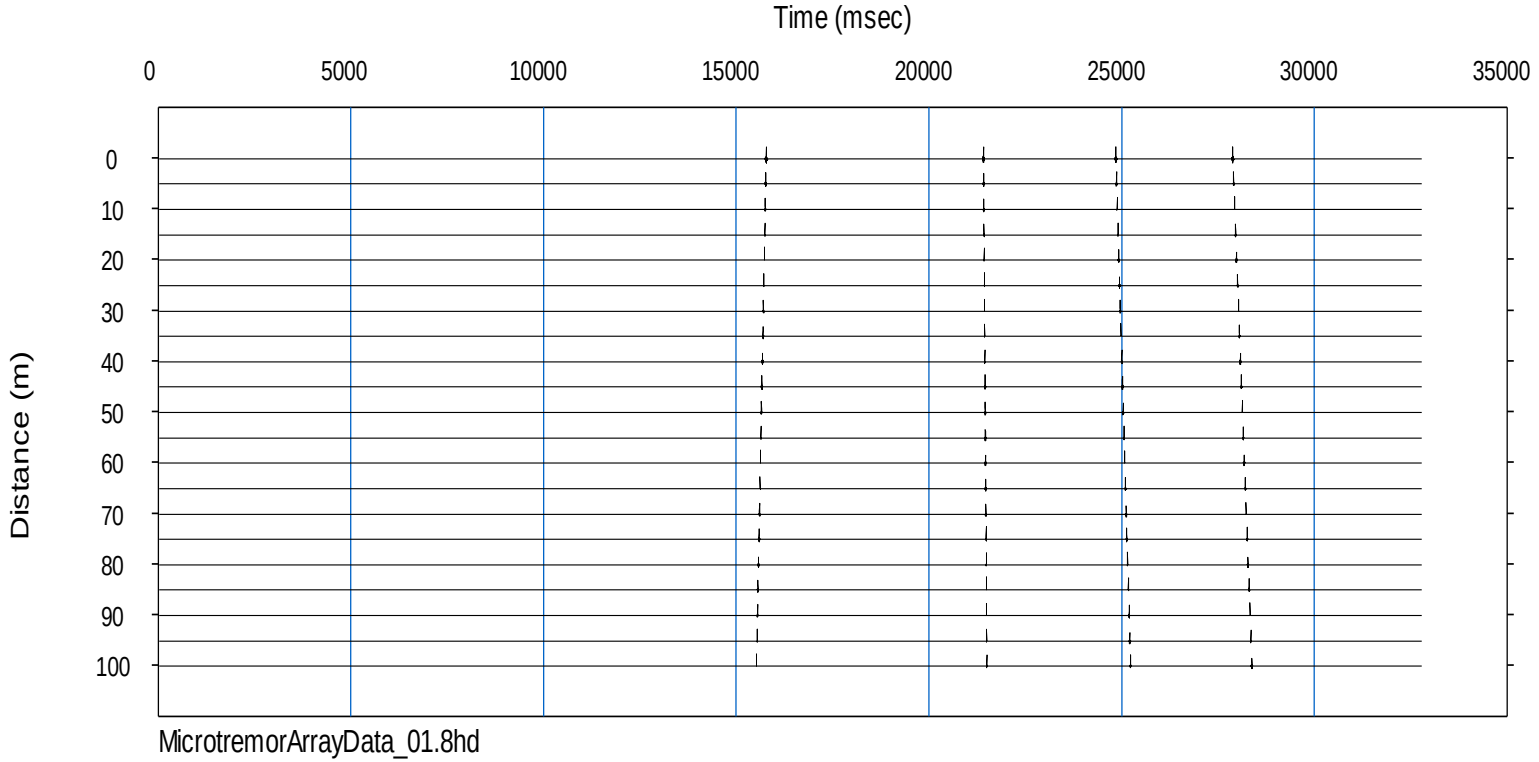


Linear array

4 directions
120 degrees (2)

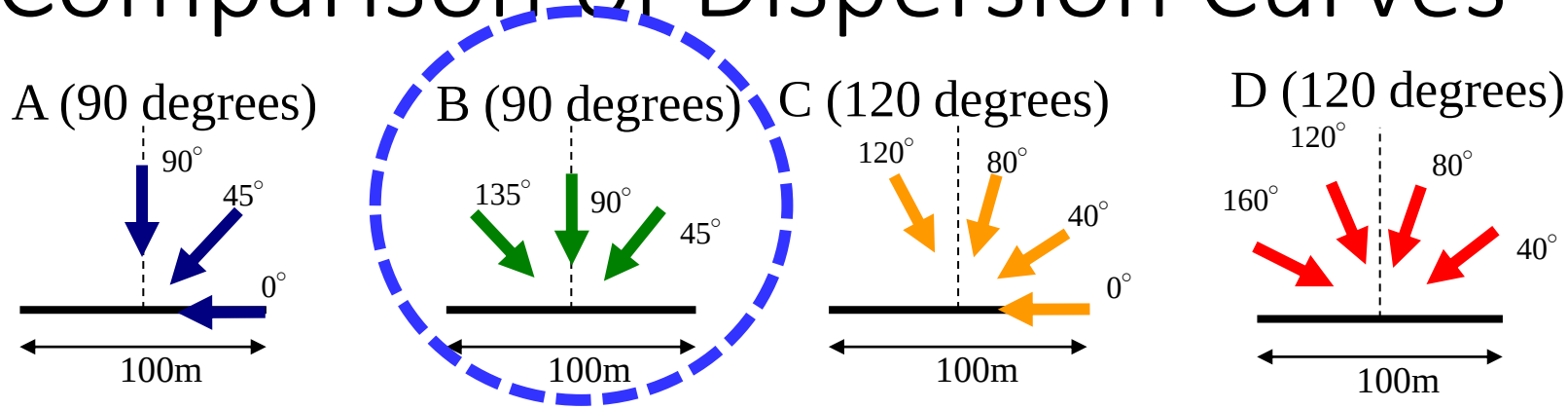
Simulation of Microtremor Measurement Using Linear Array

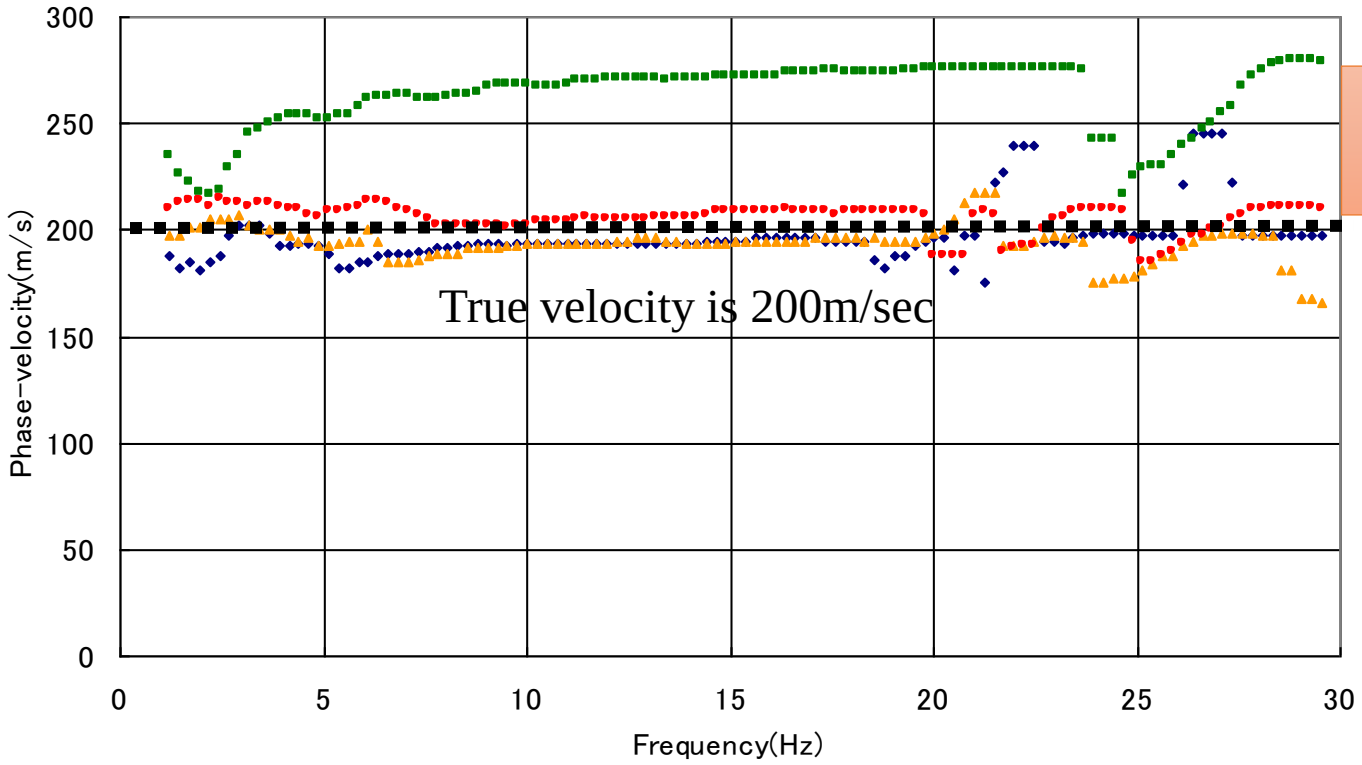
Example of synthetic waveform data



20 files are averaged in coherency

Comparison of Dispersion Curves

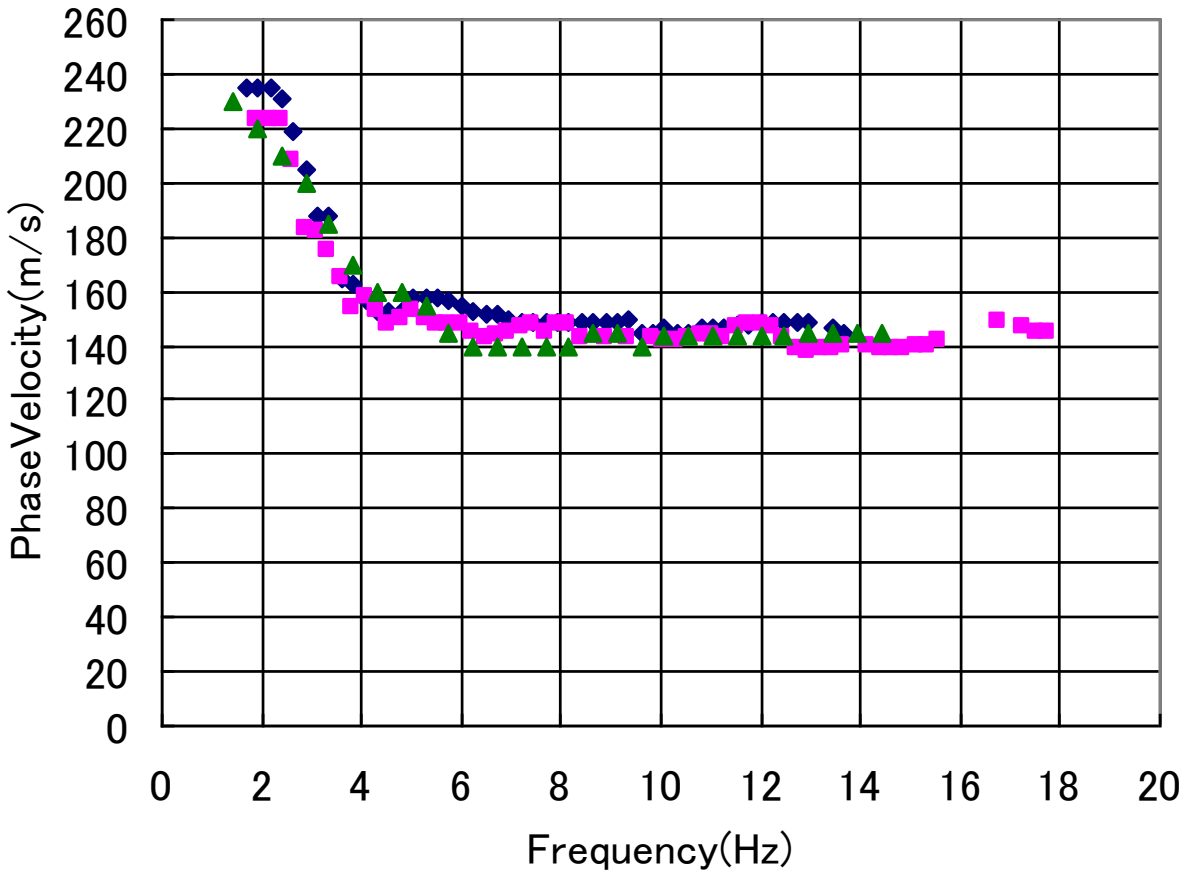


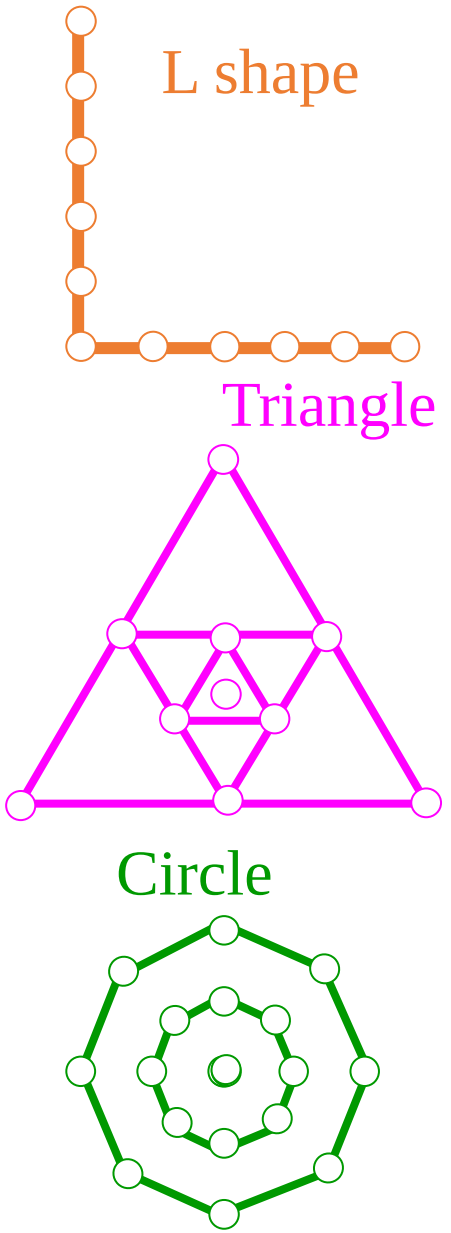


Except Case B, almost correct phase velocity can be calculated

If ambient noise comes from 120 degrees, error of phase velocity is less than 20 %

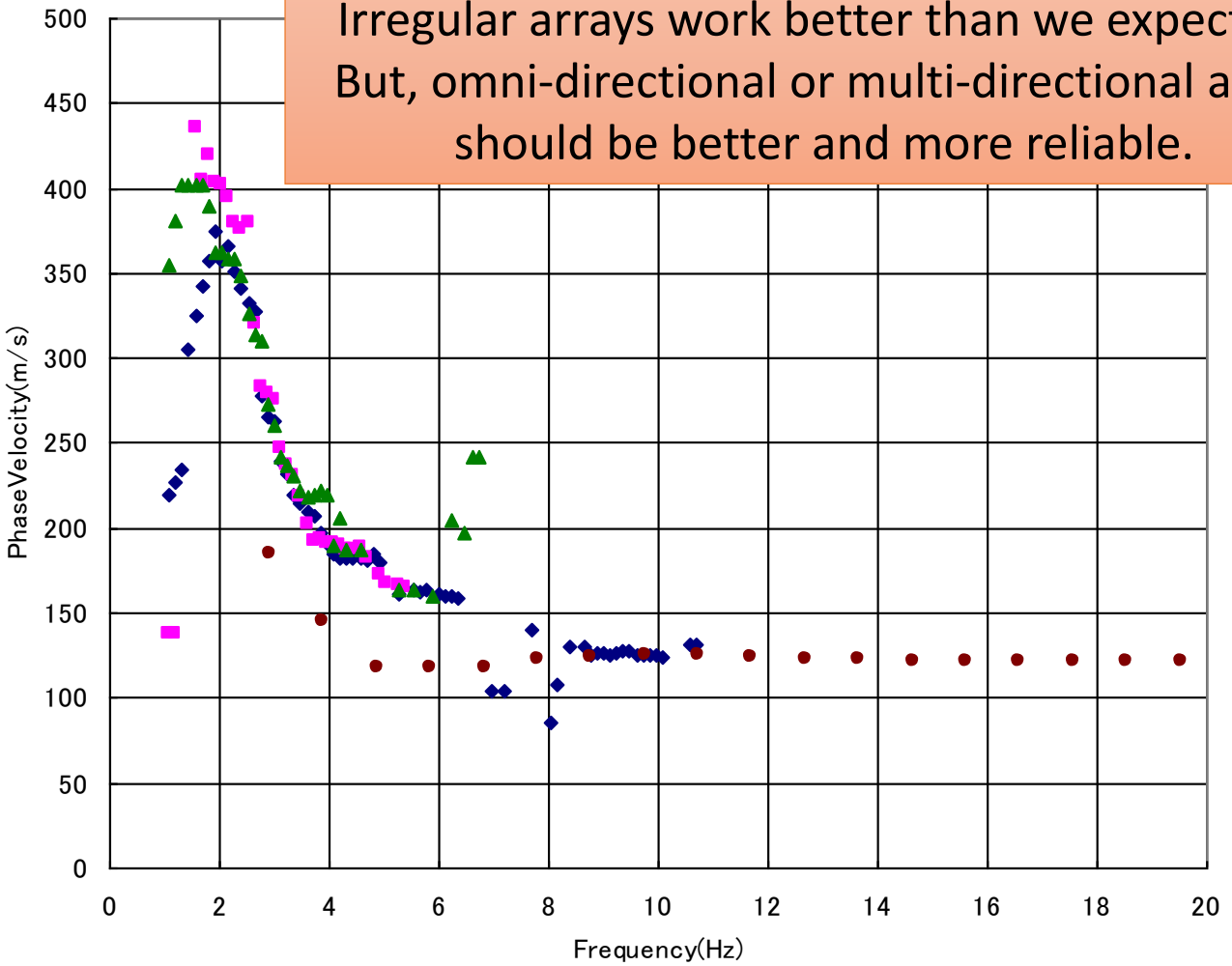
Effect of array shape on dispersion curves Pisa, Italy, in 2003

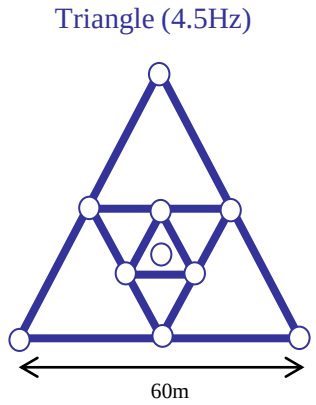


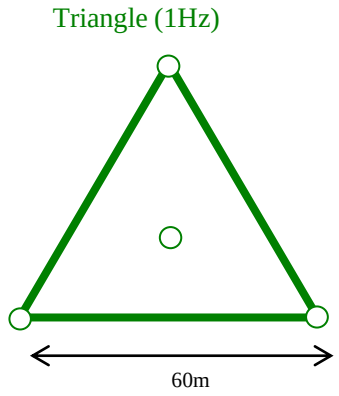


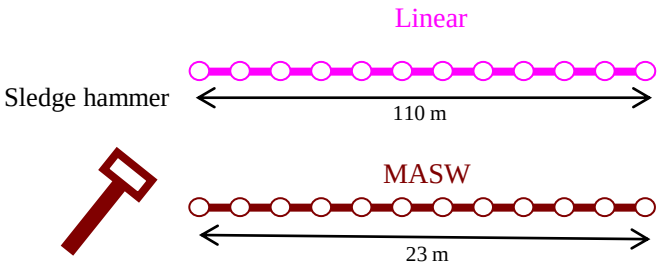
Comparison of Dispersion Curves Los Angeles, U.S., in 2004

Irregular arrays work better than we expected.
But, omni-directional or multi-directional array should be better and more reliable.





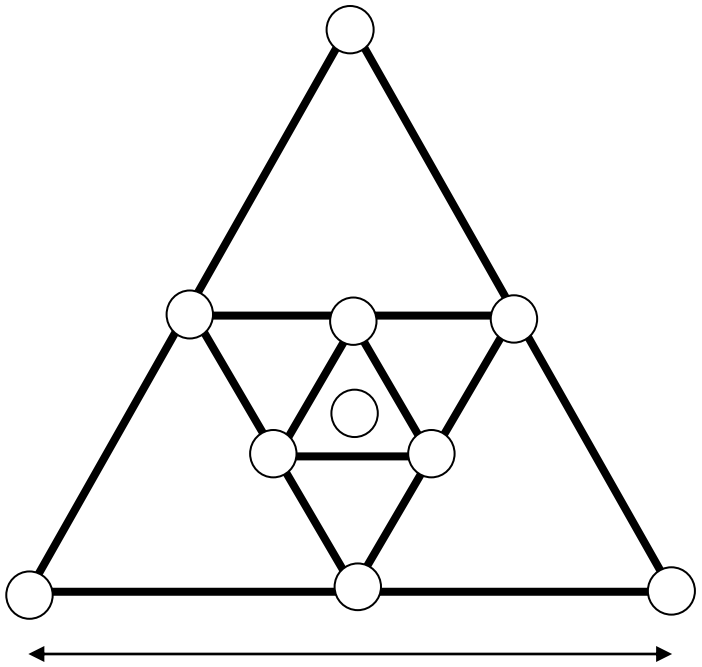




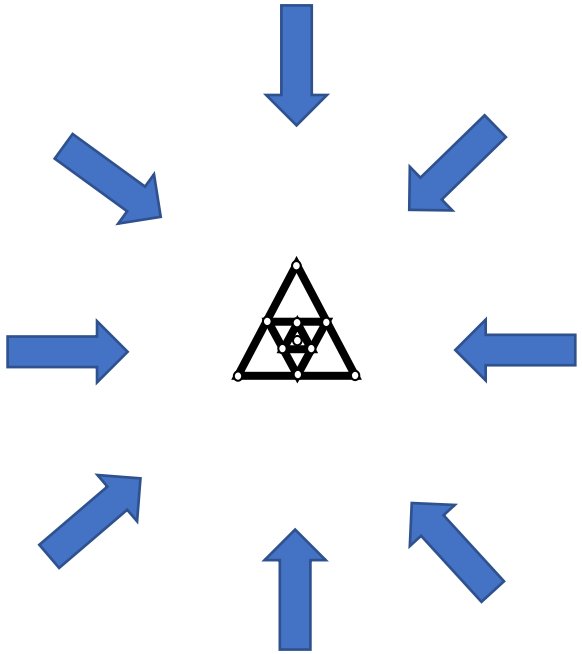
Uncertainties in surface wave investigations

- Review Spatial Auto-correlation method
- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

Numerical simulation

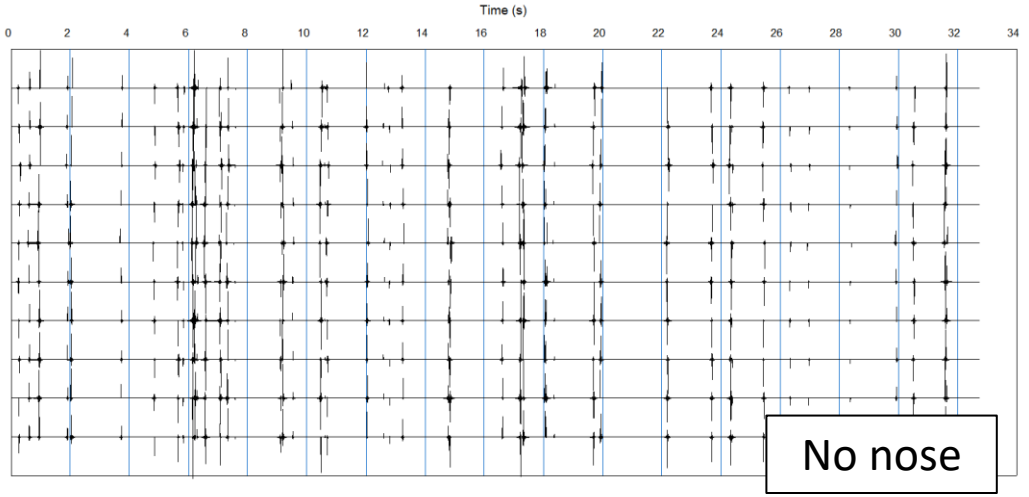


15, 50, 150, 500 m

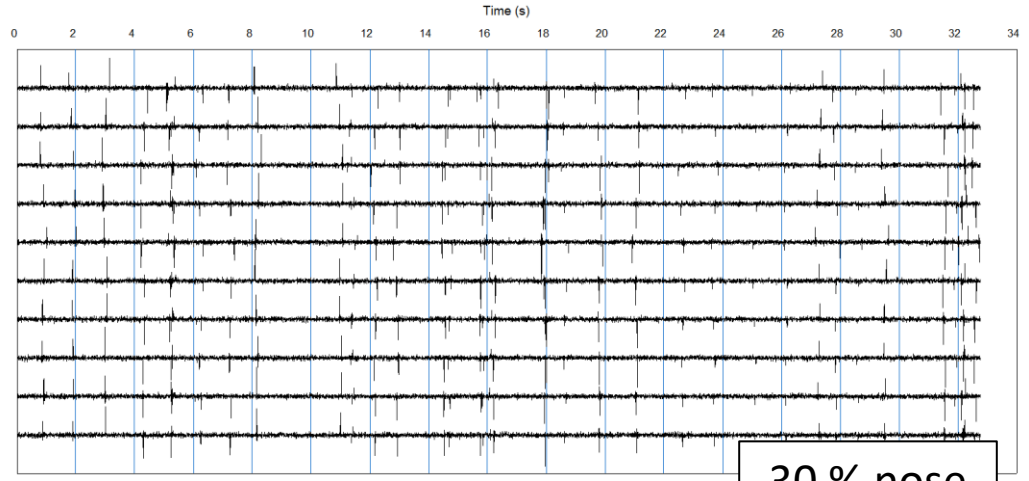


50 directions

Averaged 100 files in coherency



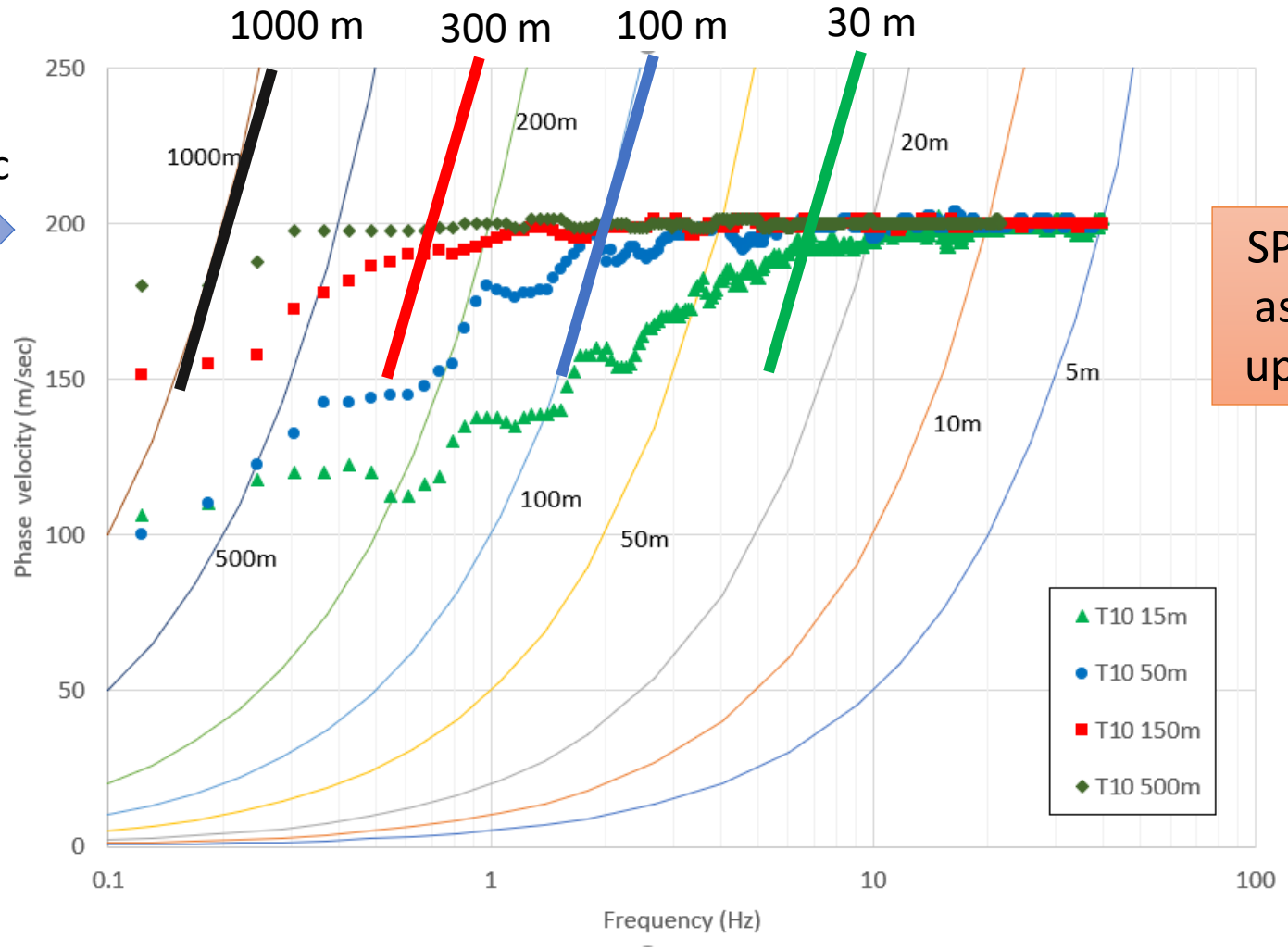
No nose



30 % nose

Effect of array size (maximum receiver separation)

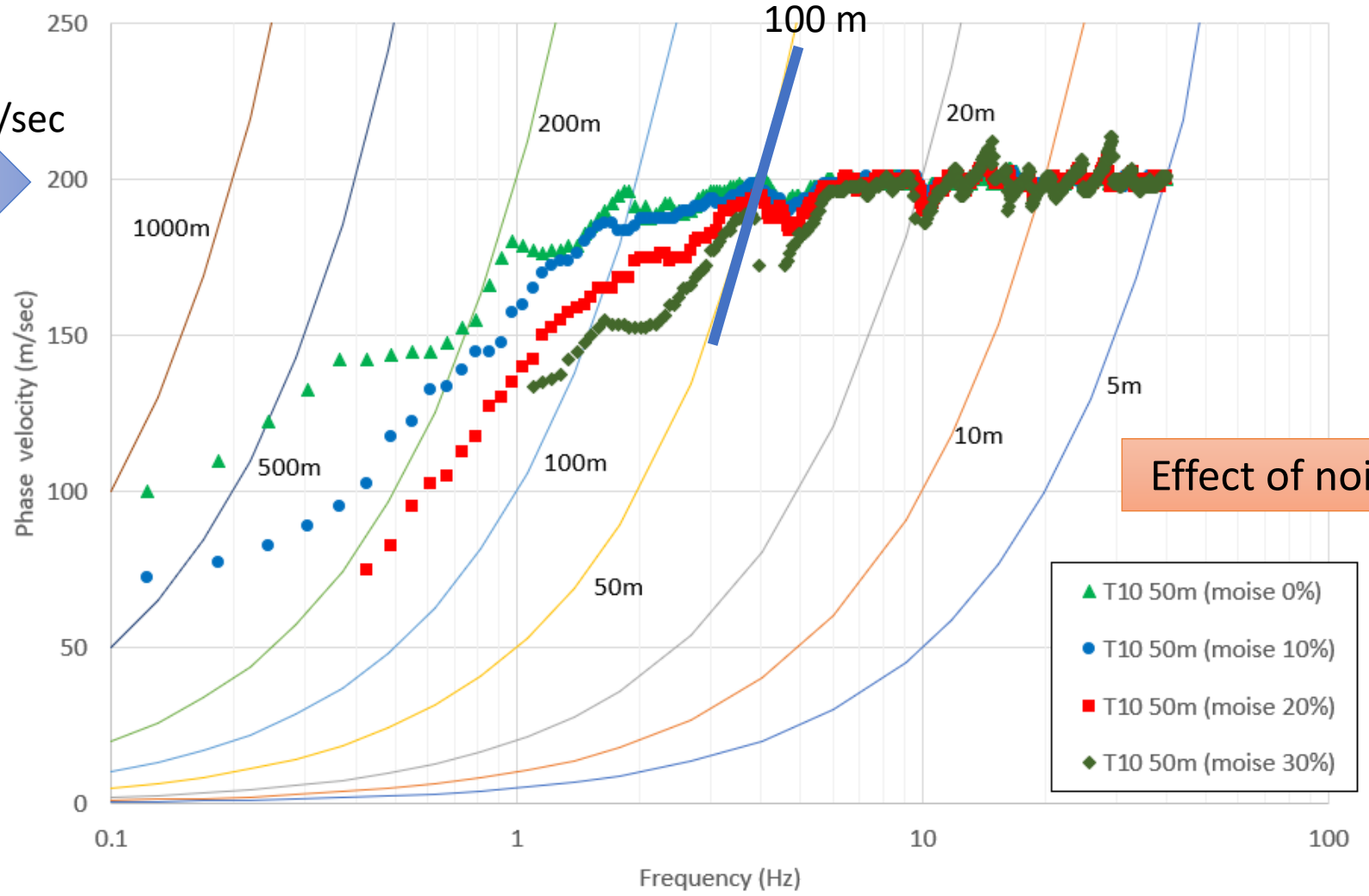
True velocity is 200 m/sec



SPAC provides phase velocity associated with wave length up to the double of array size

Effect of noise (50 m)

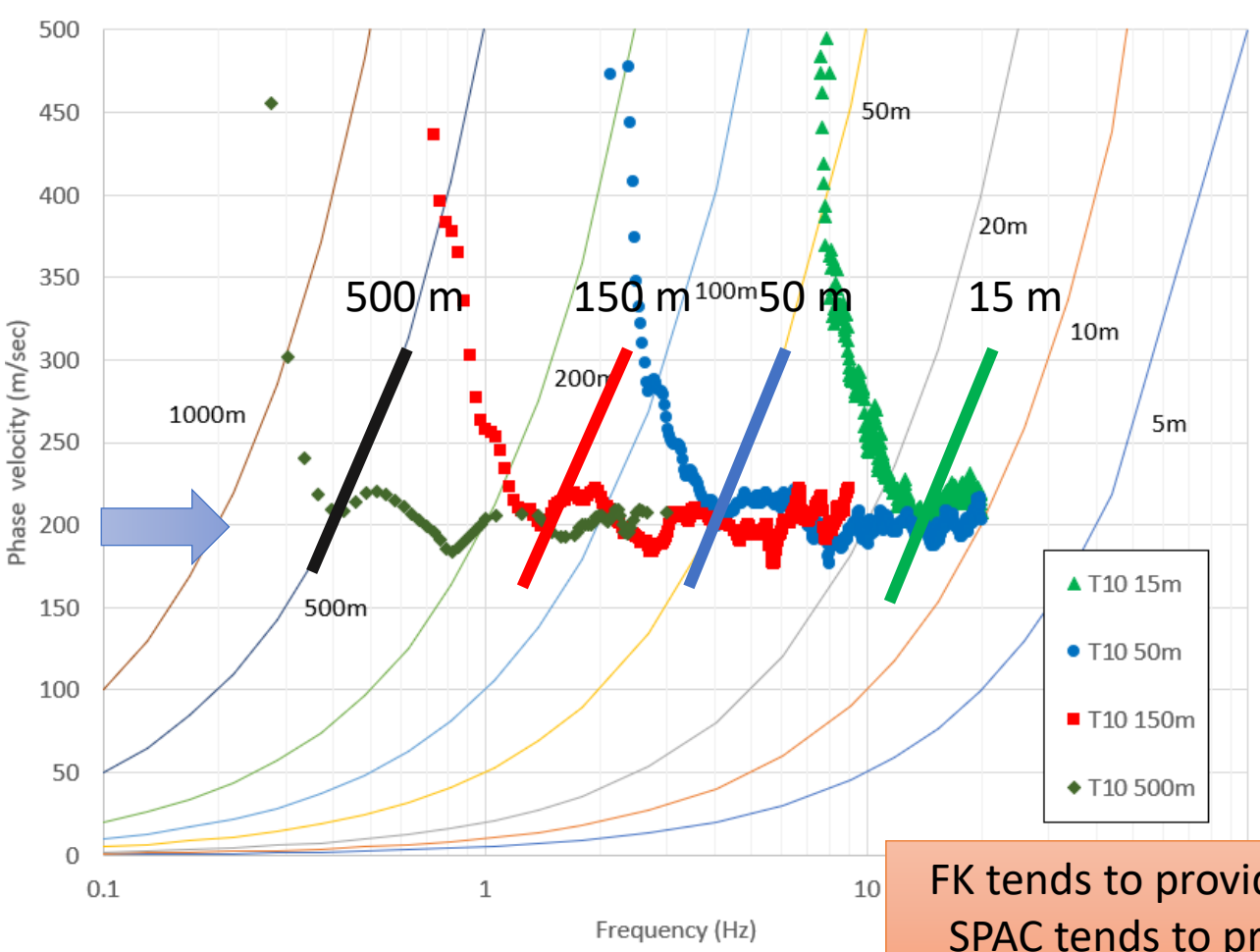
True velocity is 200 m/sec

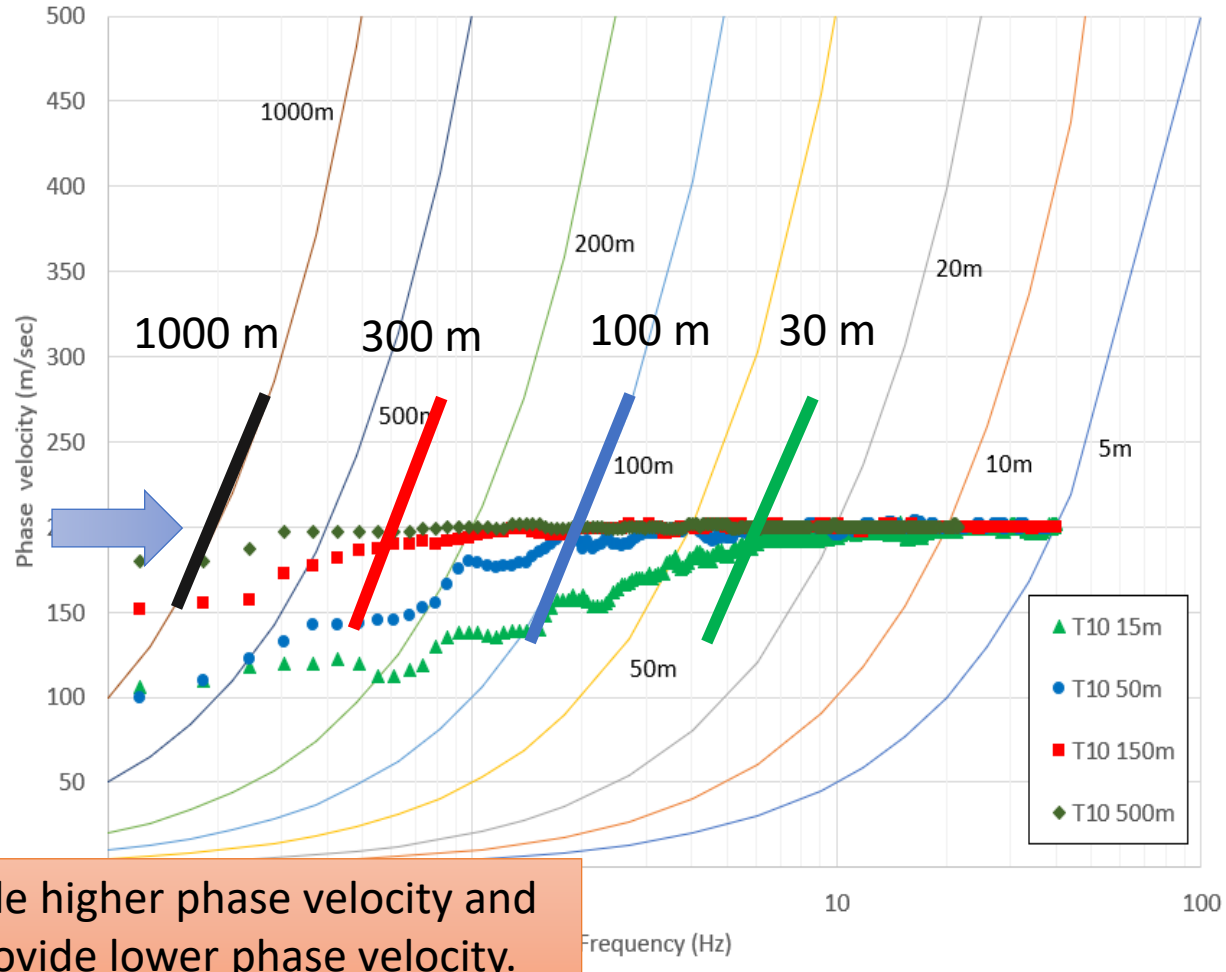


Effect of noise is relatively small

Comparison with FK

FK provides phase velocity associated with wave length up to the same as array size.
SPAC provides phase velocity associated with wave length up to the double of array size.





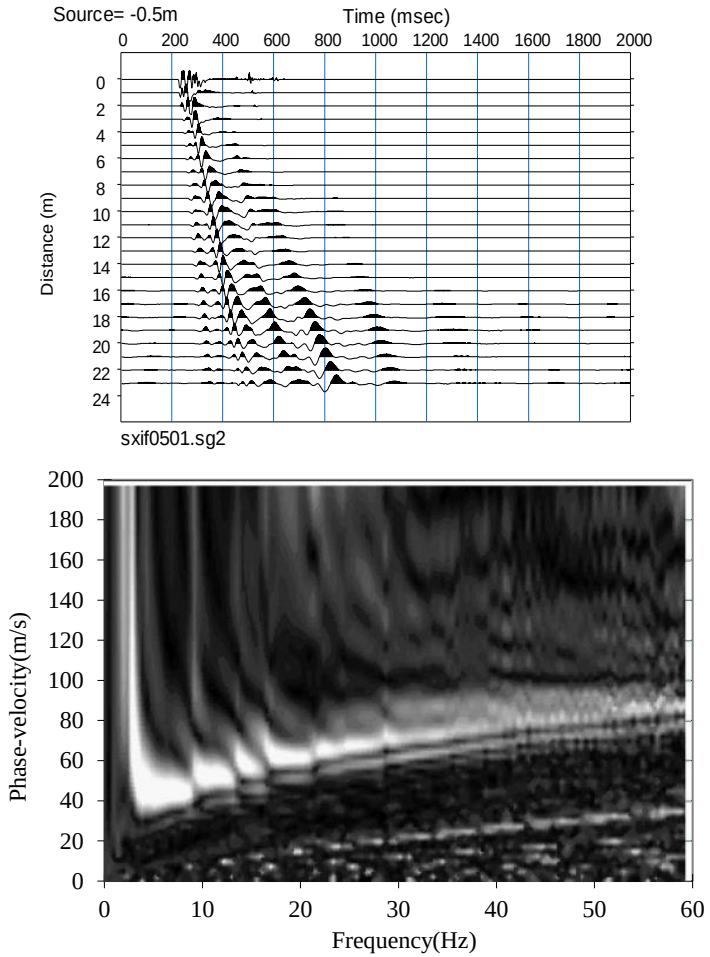
FK tends to provide higher phase velocity and SPAC tends to provide lower phase velocity.

Uncertainties in surface wave investigations

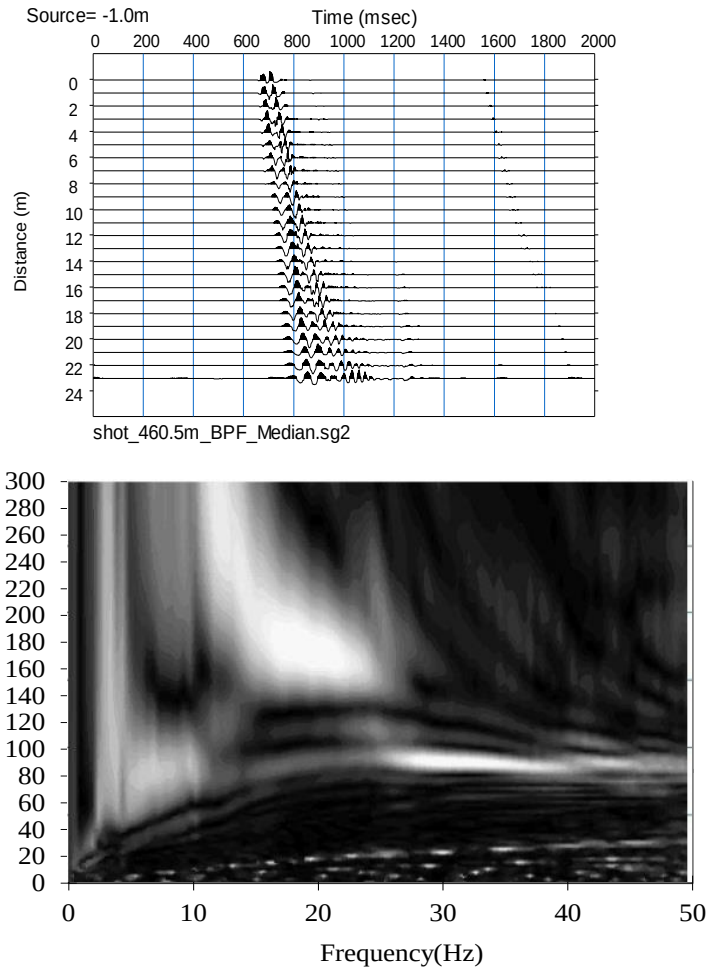
- Review Spatial Auto-correlation method
- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

Example of observed waveform data including higher modes

Case-1



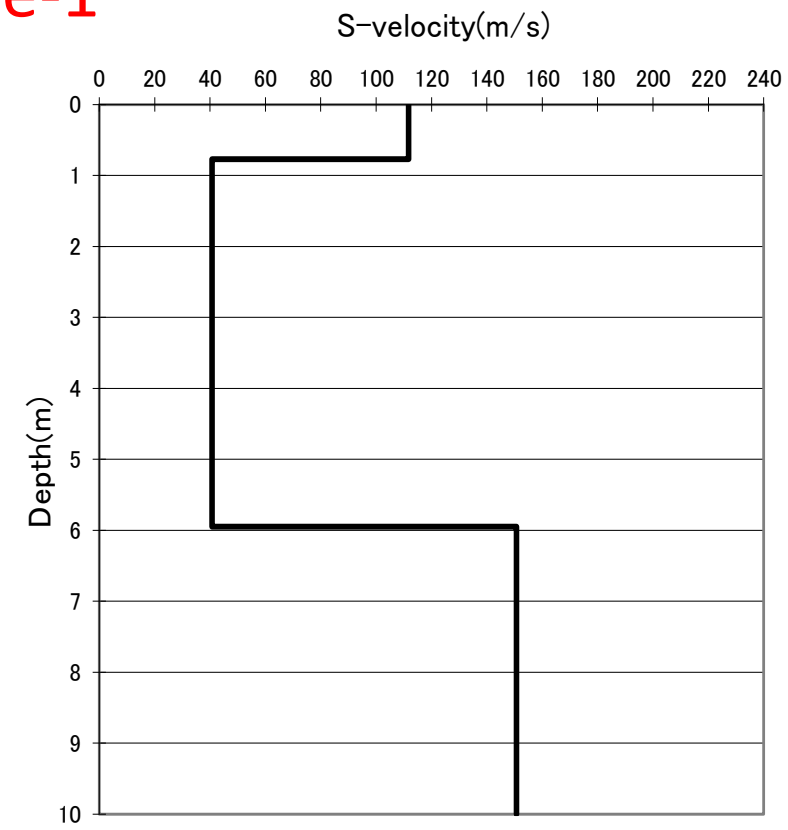
Case-2



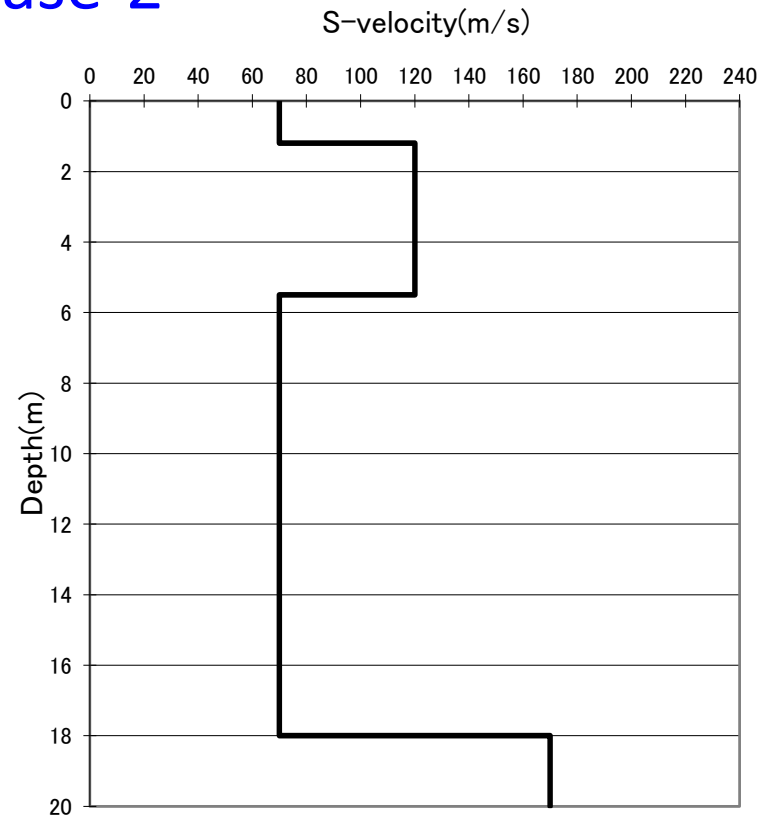
b)

Typical velocity model generating higher modes

Case-1

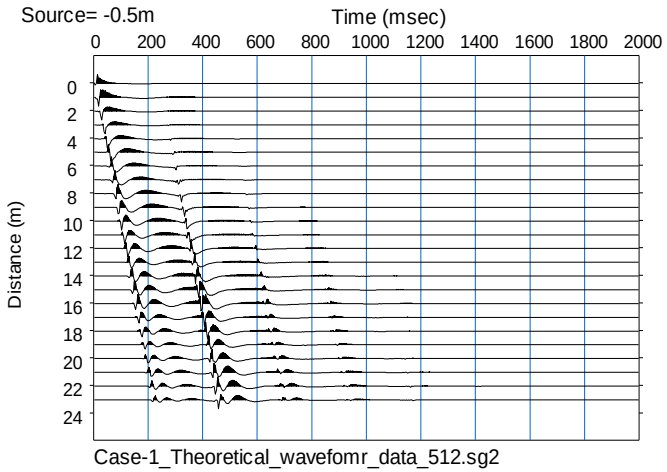


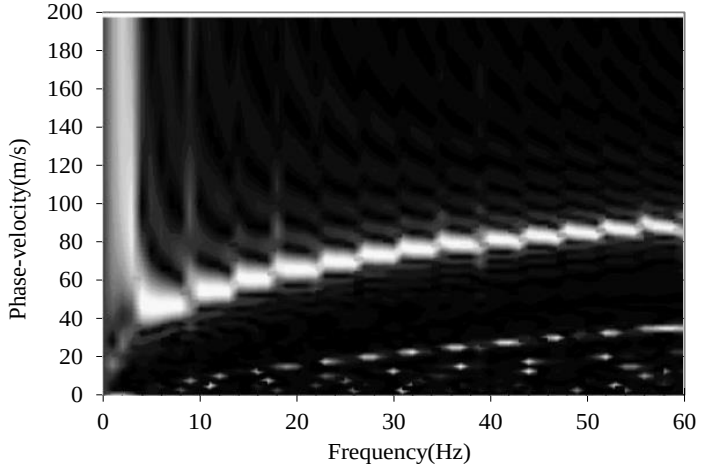
Case-2



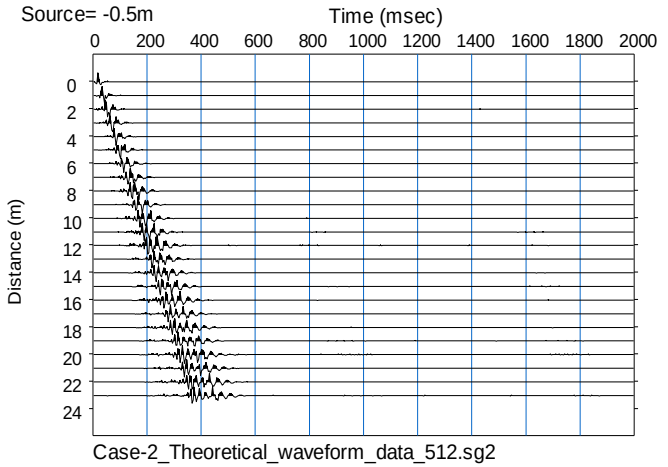
Synthetic waveform data calculated by discrete wave-number method (DWM)

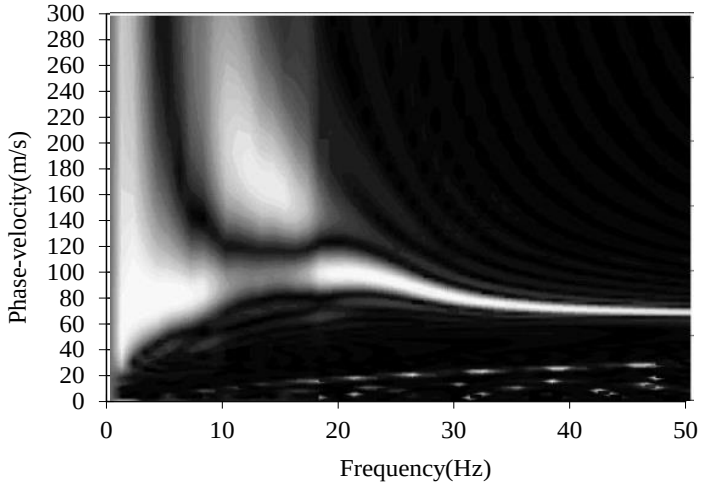
Case-1



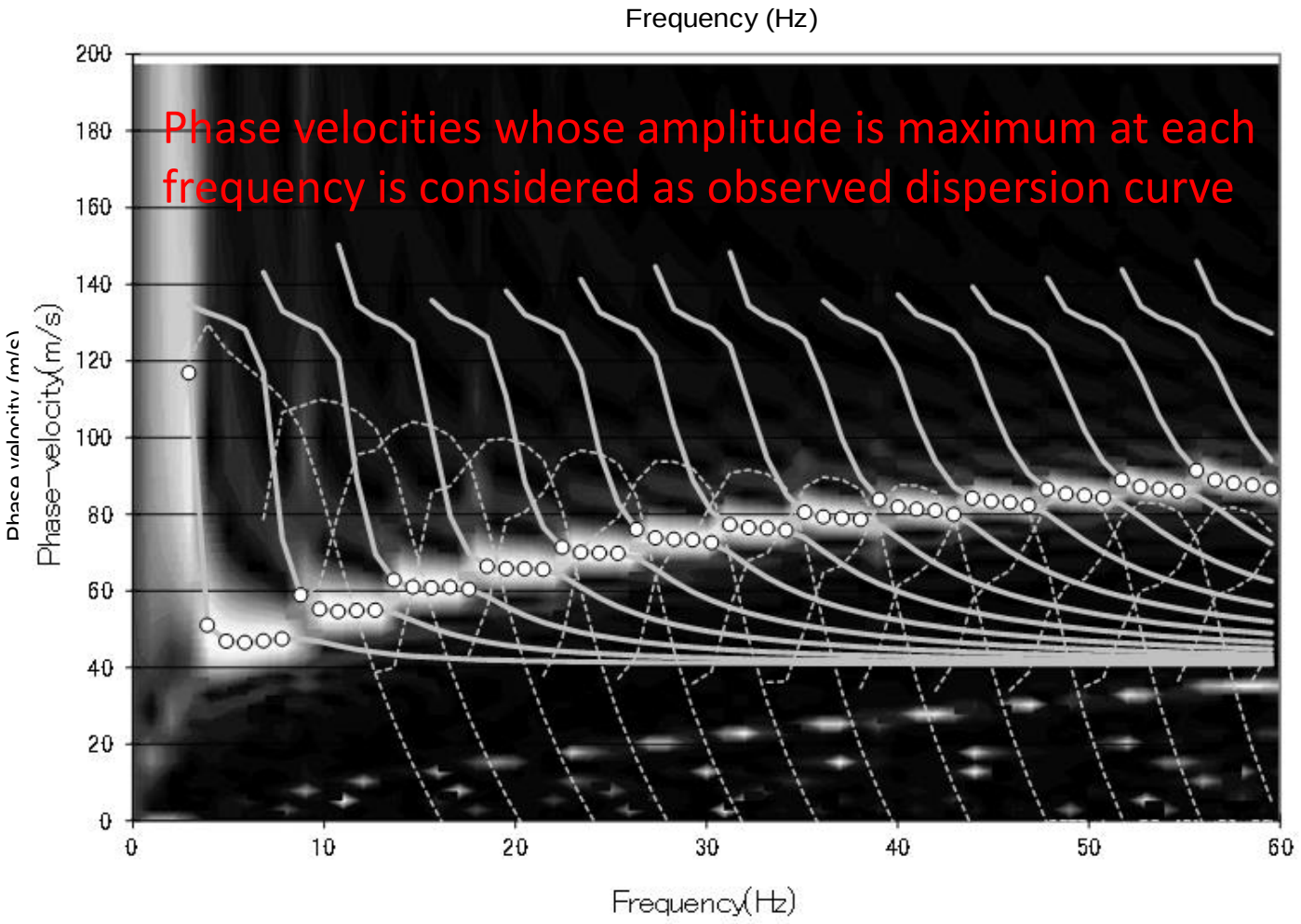


Case-2

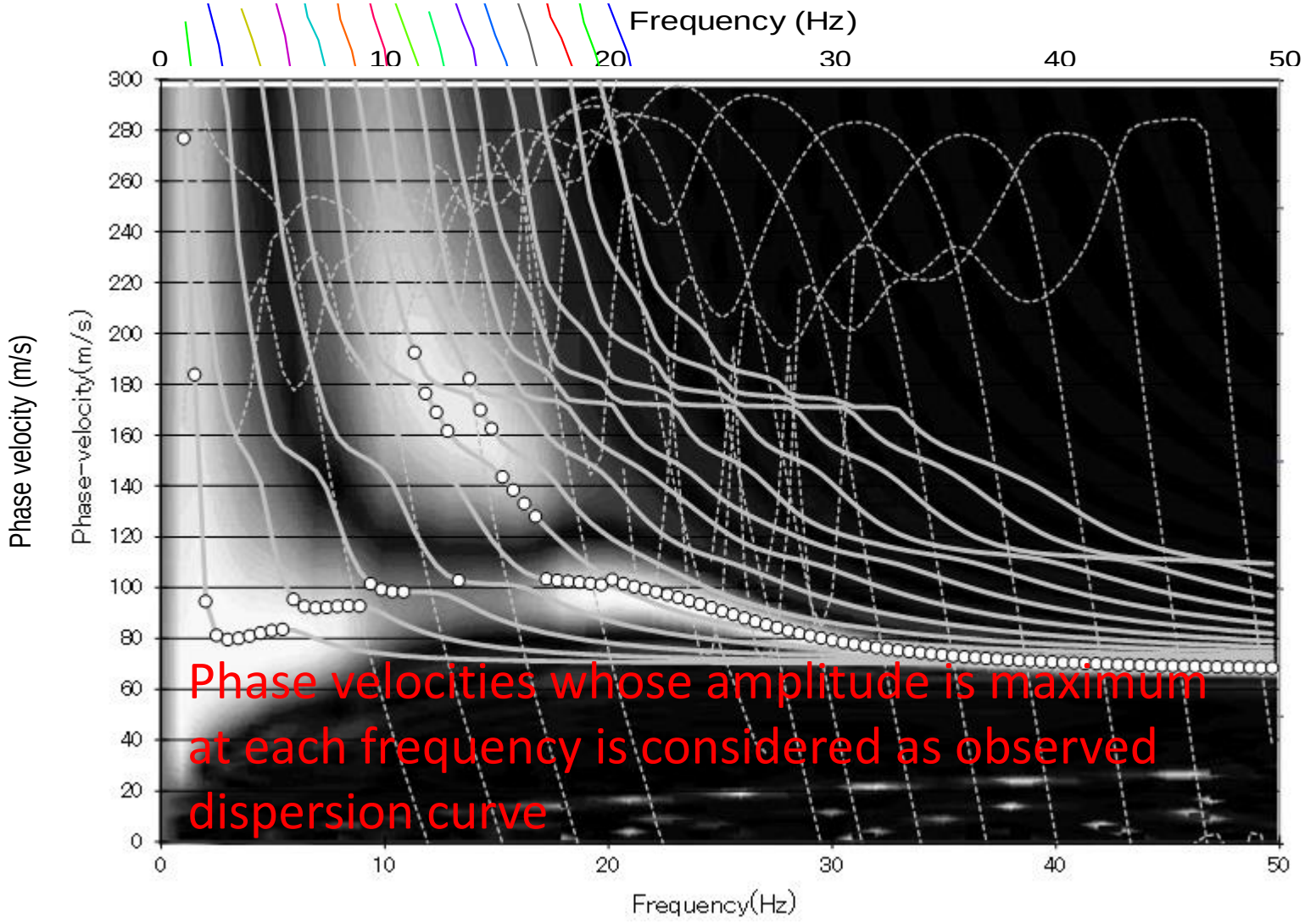




Dispersion curves and their relative amplitude calculated by NMS for Case-1

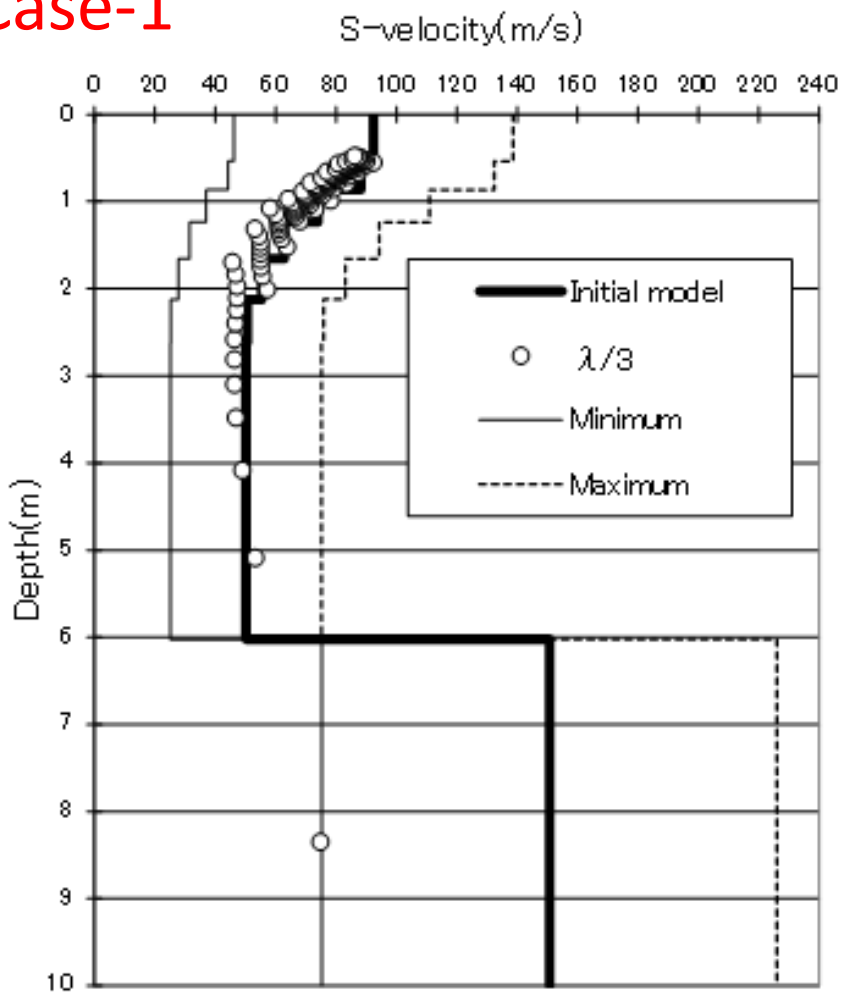


Dispersion curves and their relative amplitude calculated by NMS for Case-2

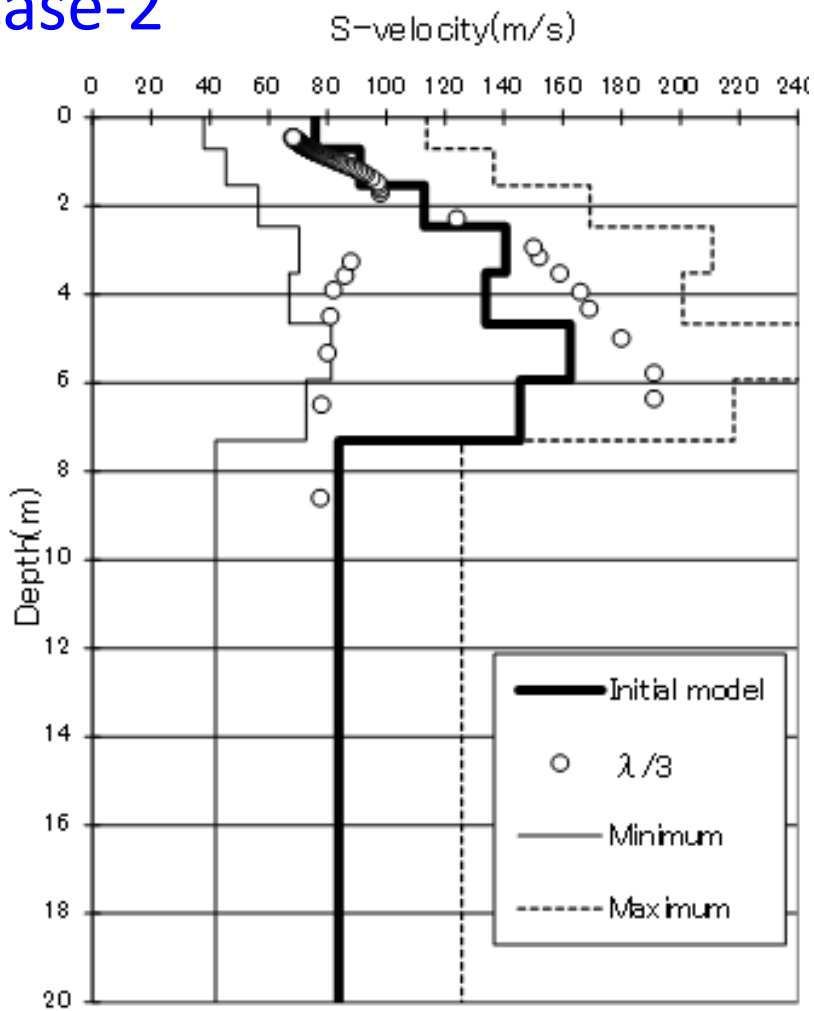


Initial model and search area for GA

Case-1

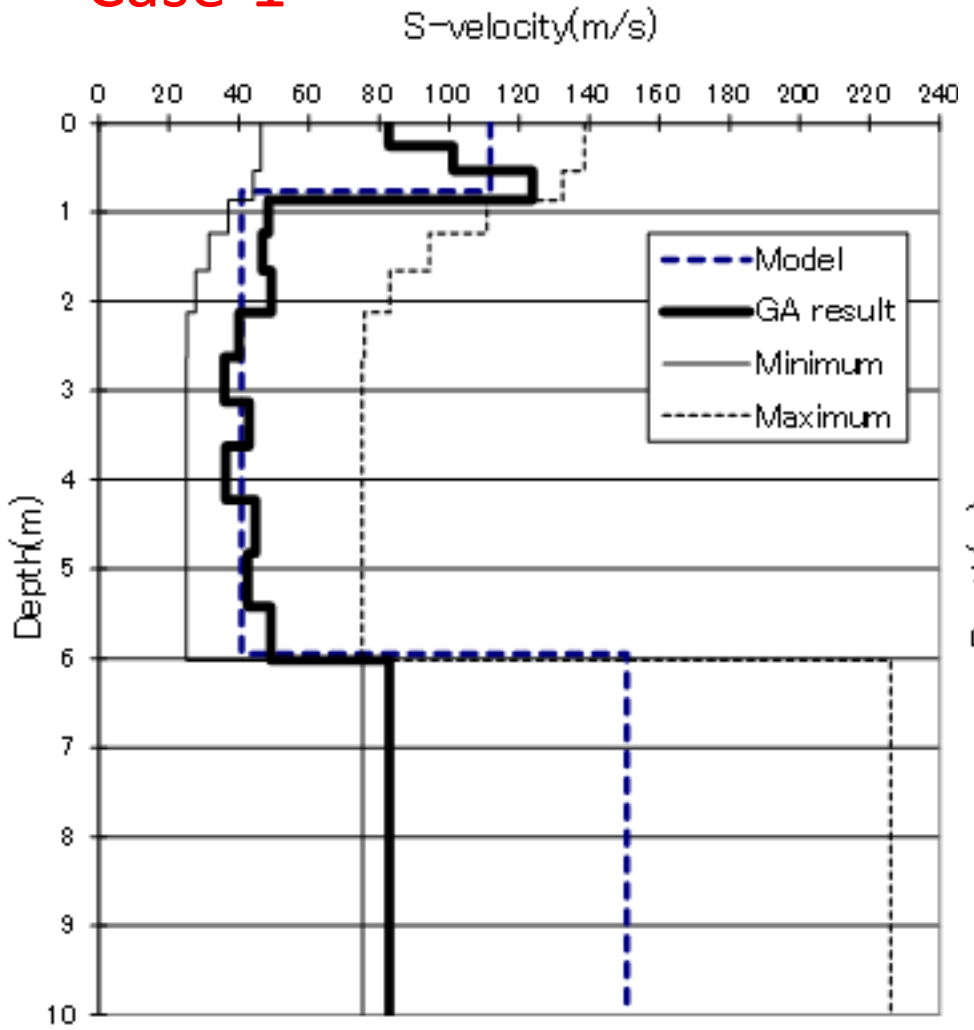


Case-2

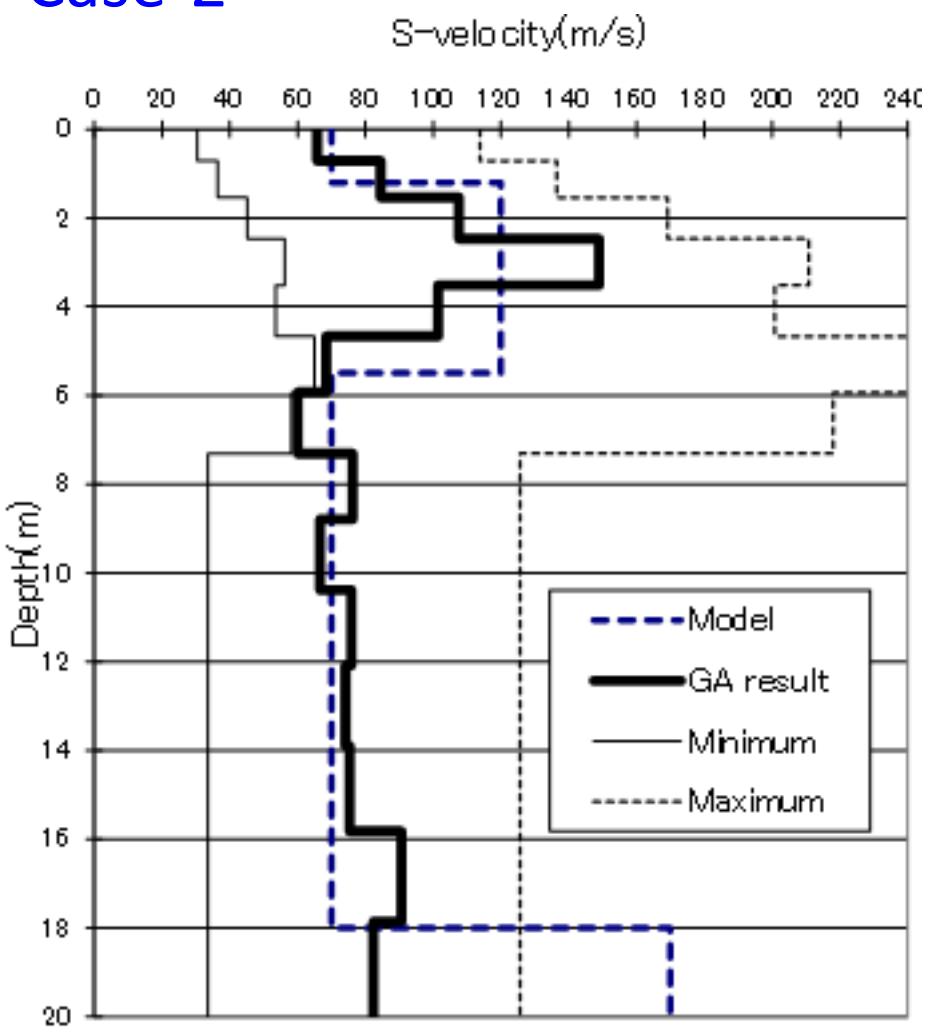


Inverted S-wave Velocity Model

Case-1

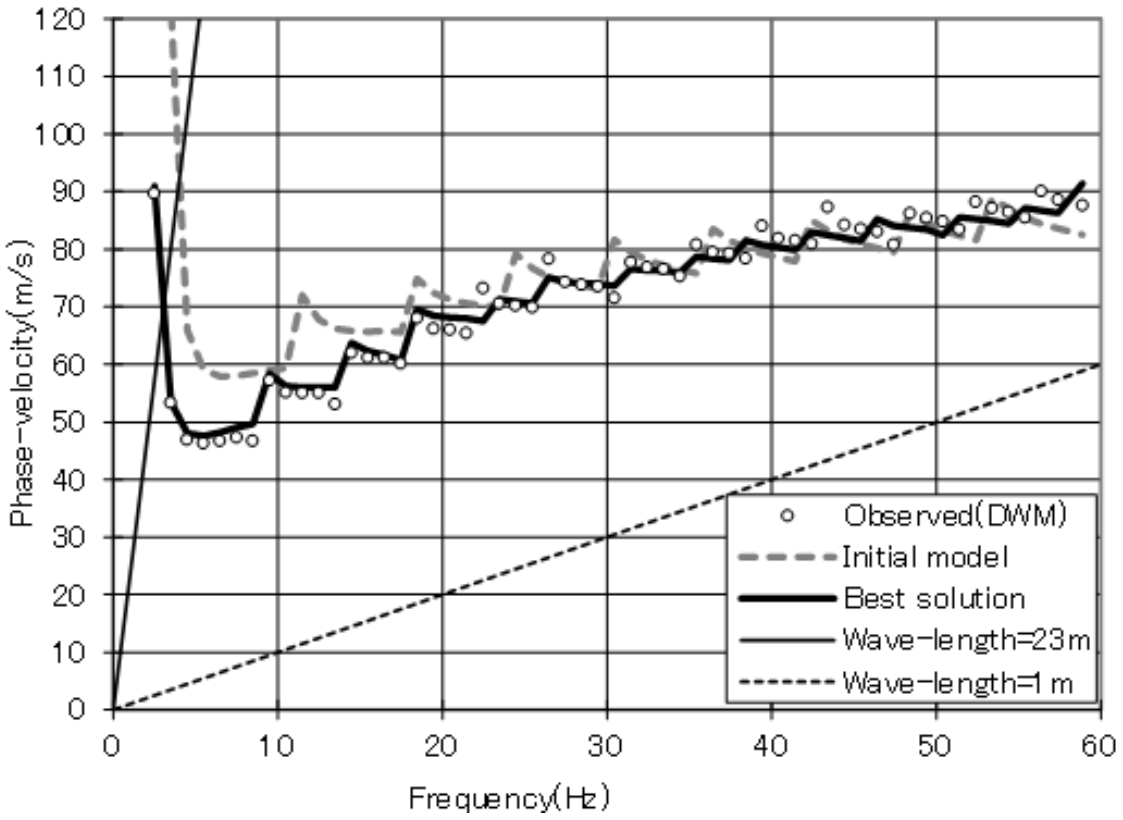


Case-2

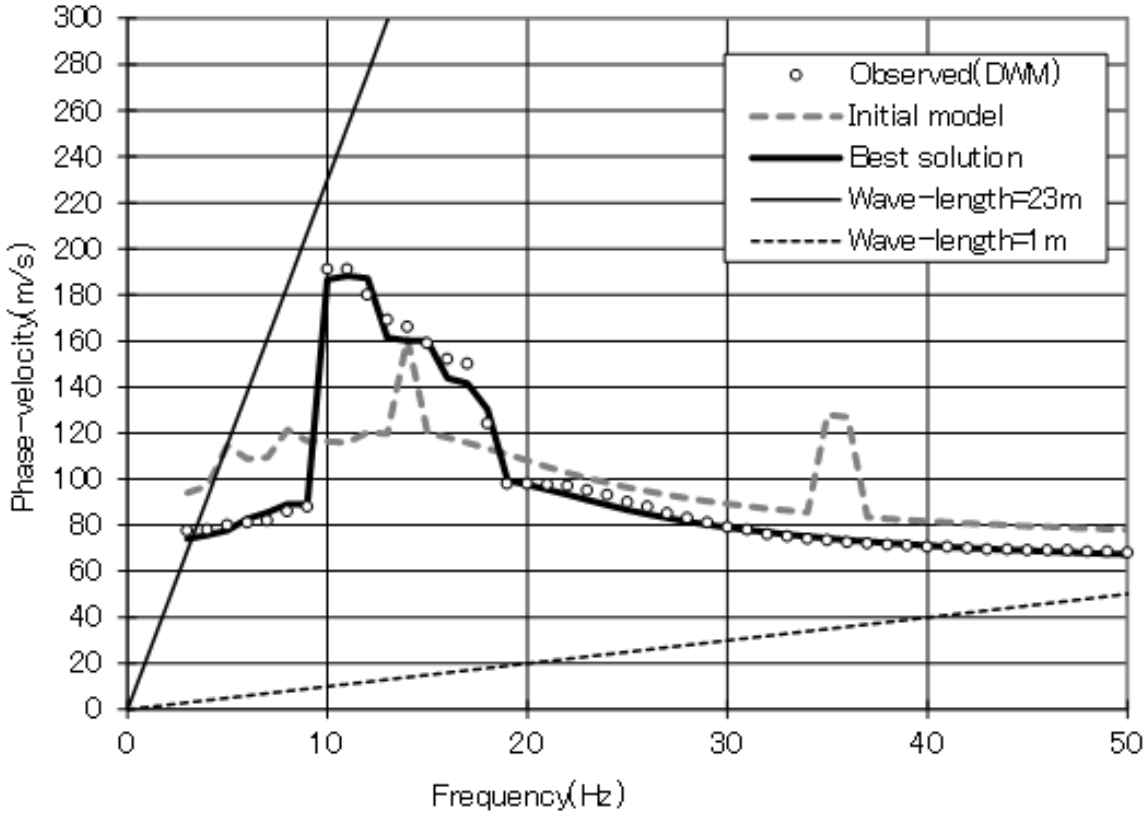


Comparison of observed (DWM) and theoretical (NMS) dispersion curves

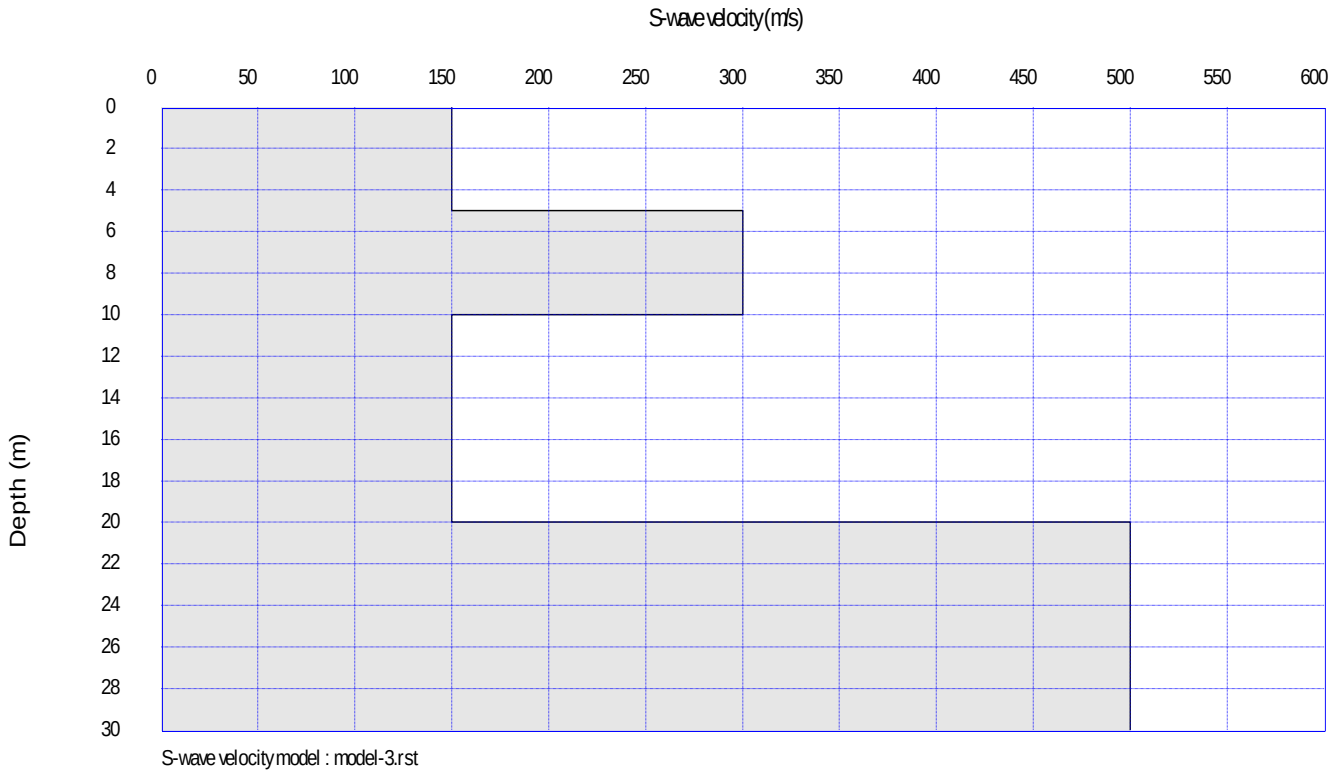
Case-1

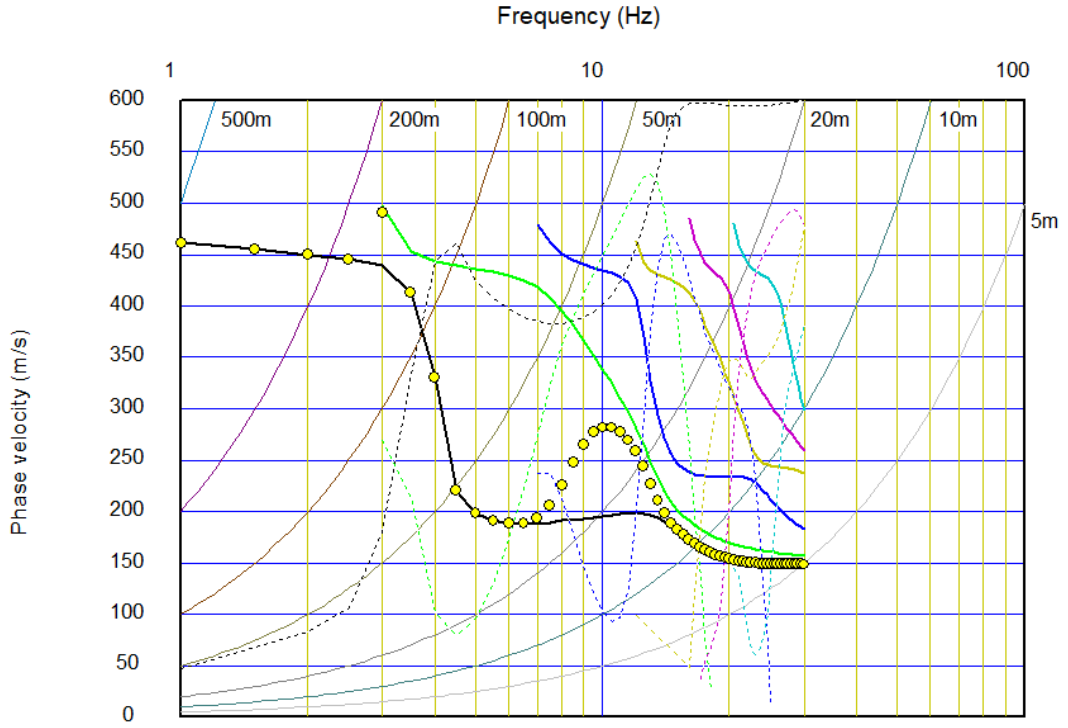


Case-2

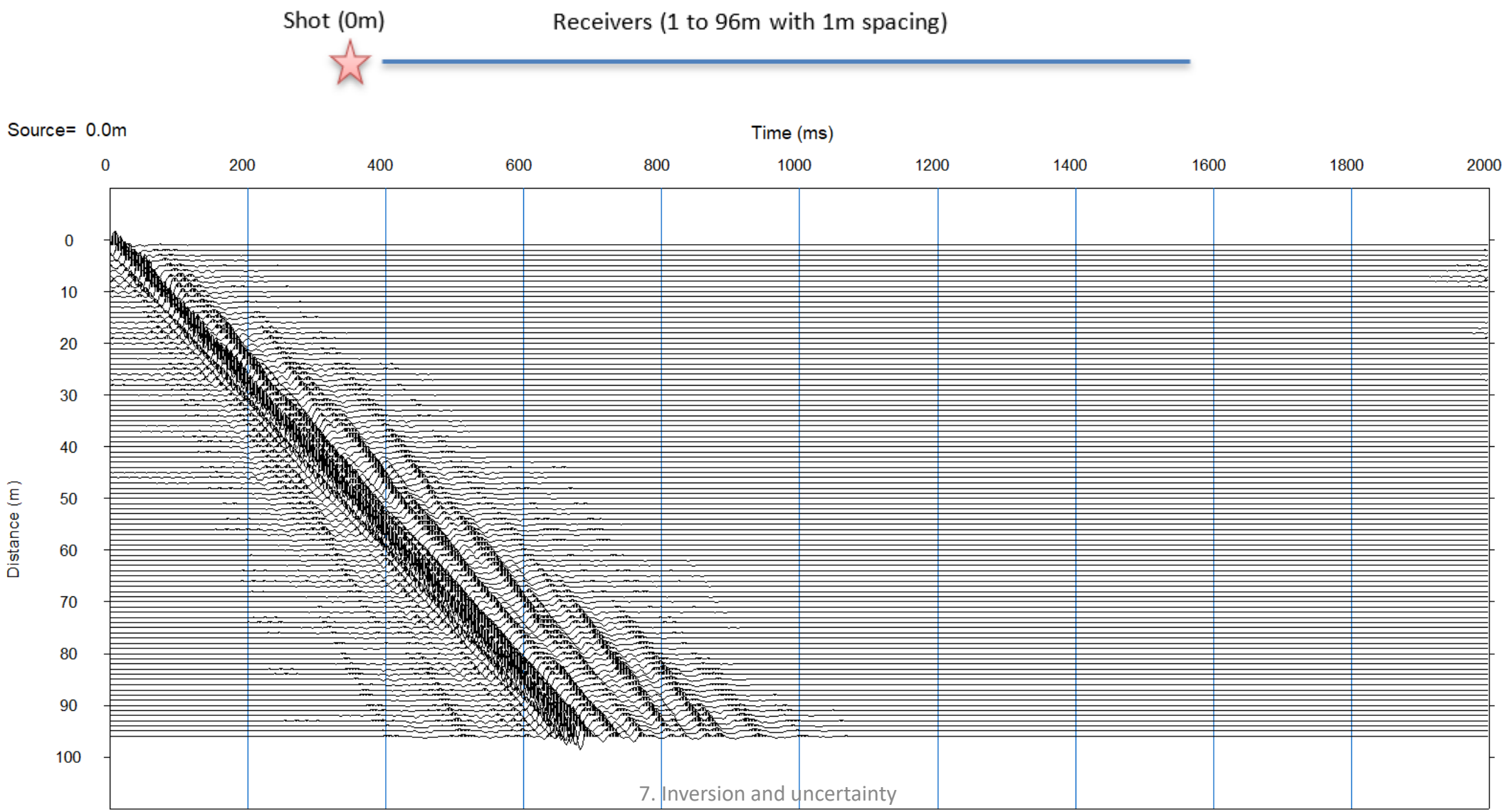


Velocity model generates significant higher modes

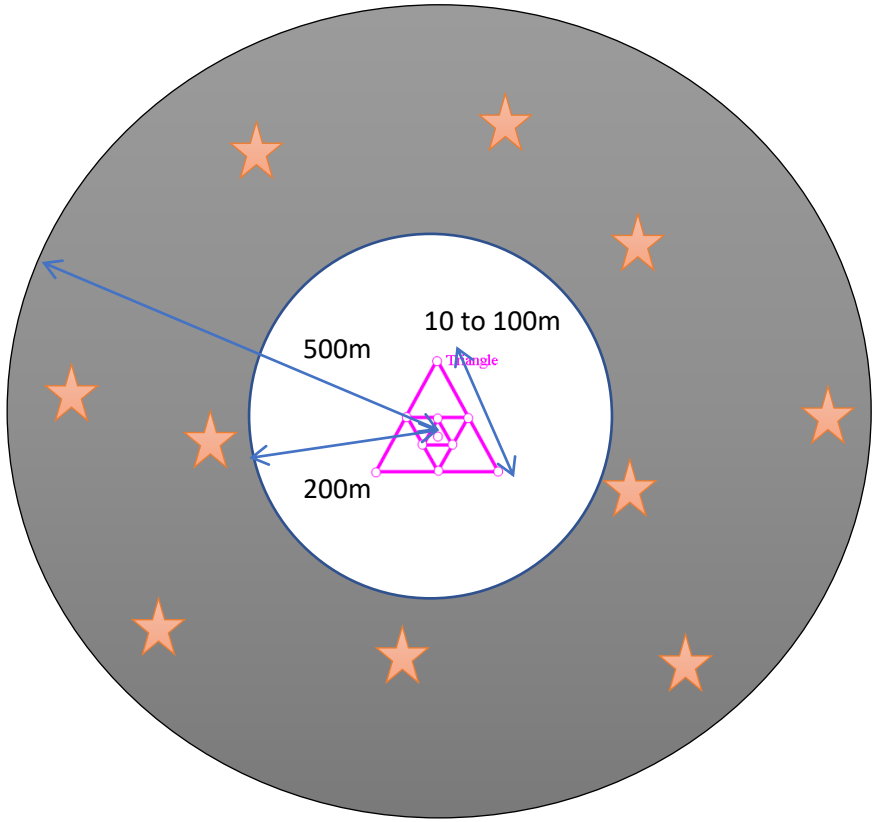




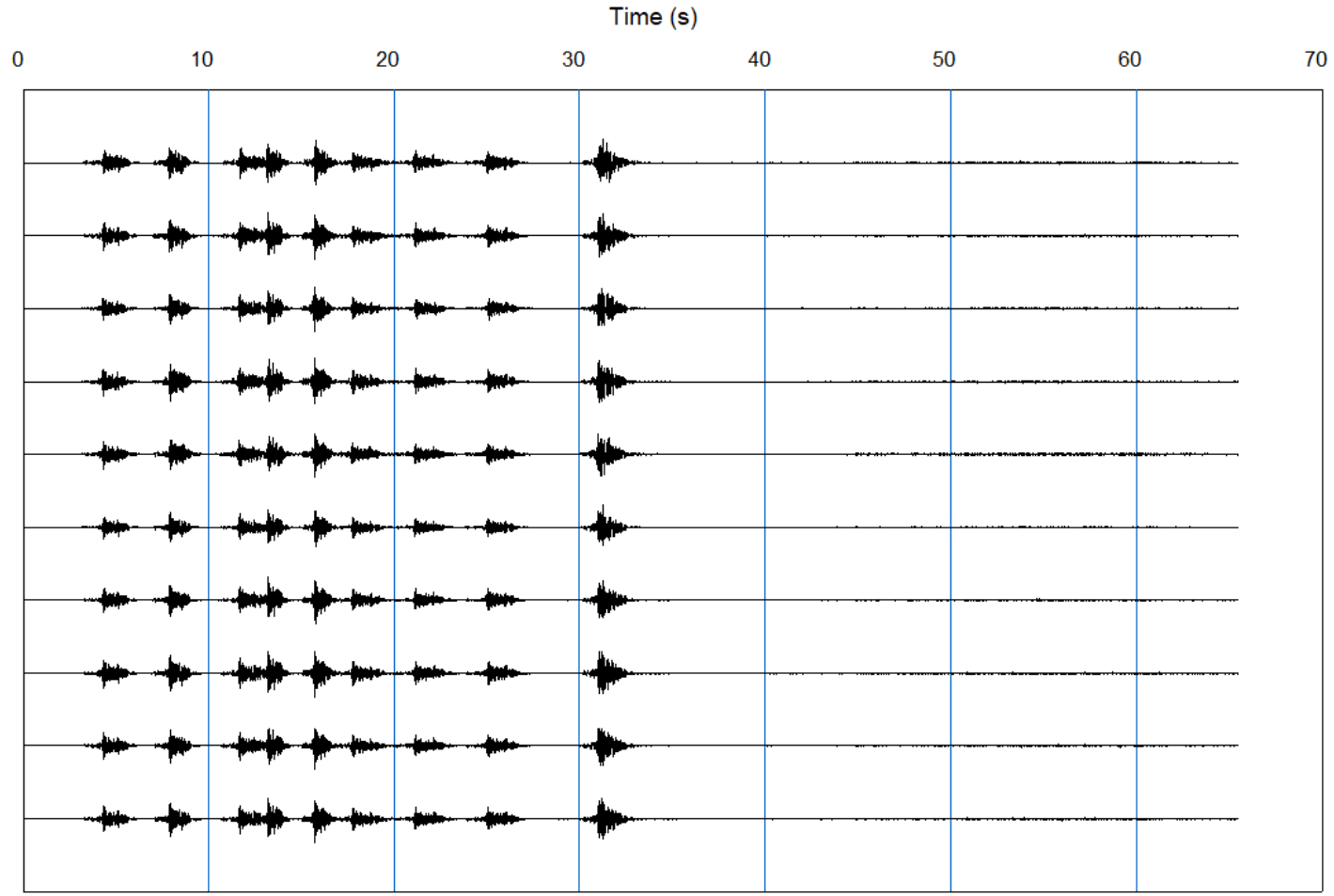
Numerical simulation using DWM (Active)



Numerical simulation using DWM

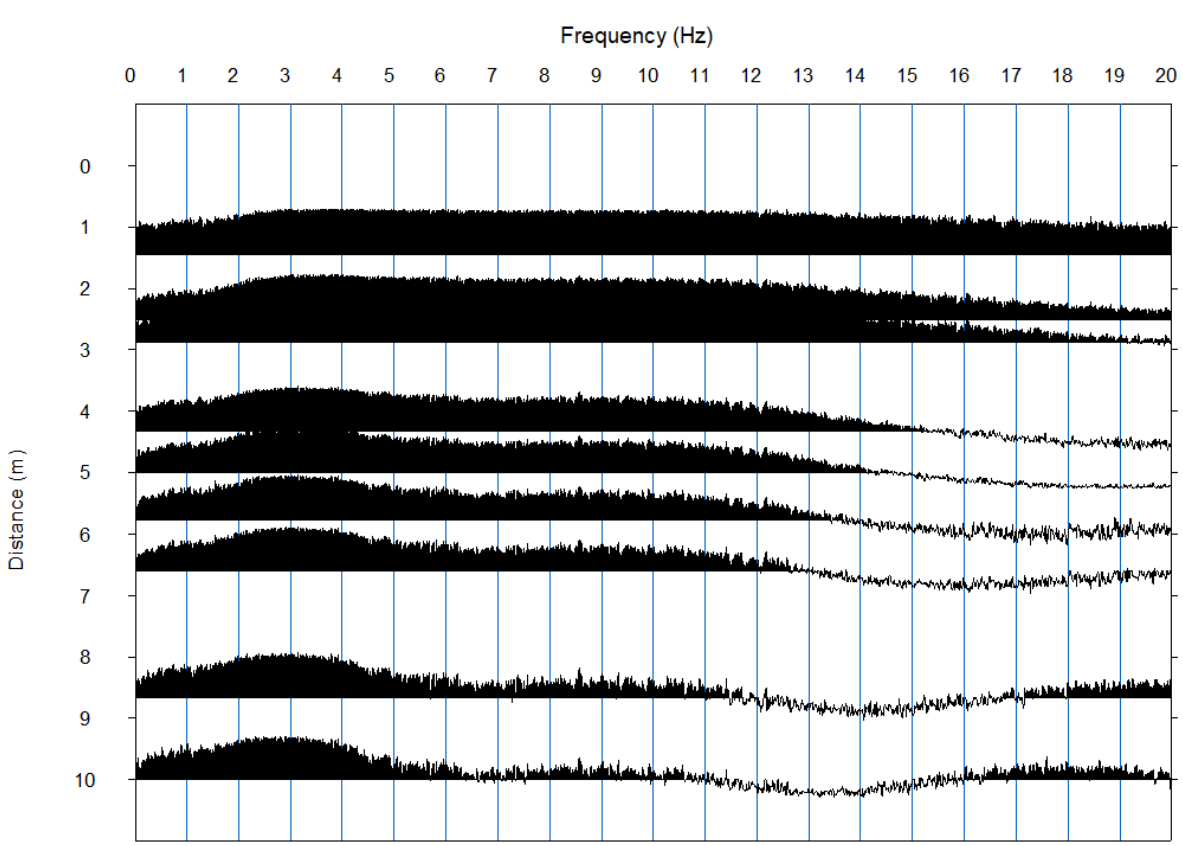


Averaged 20 files in frequency domain

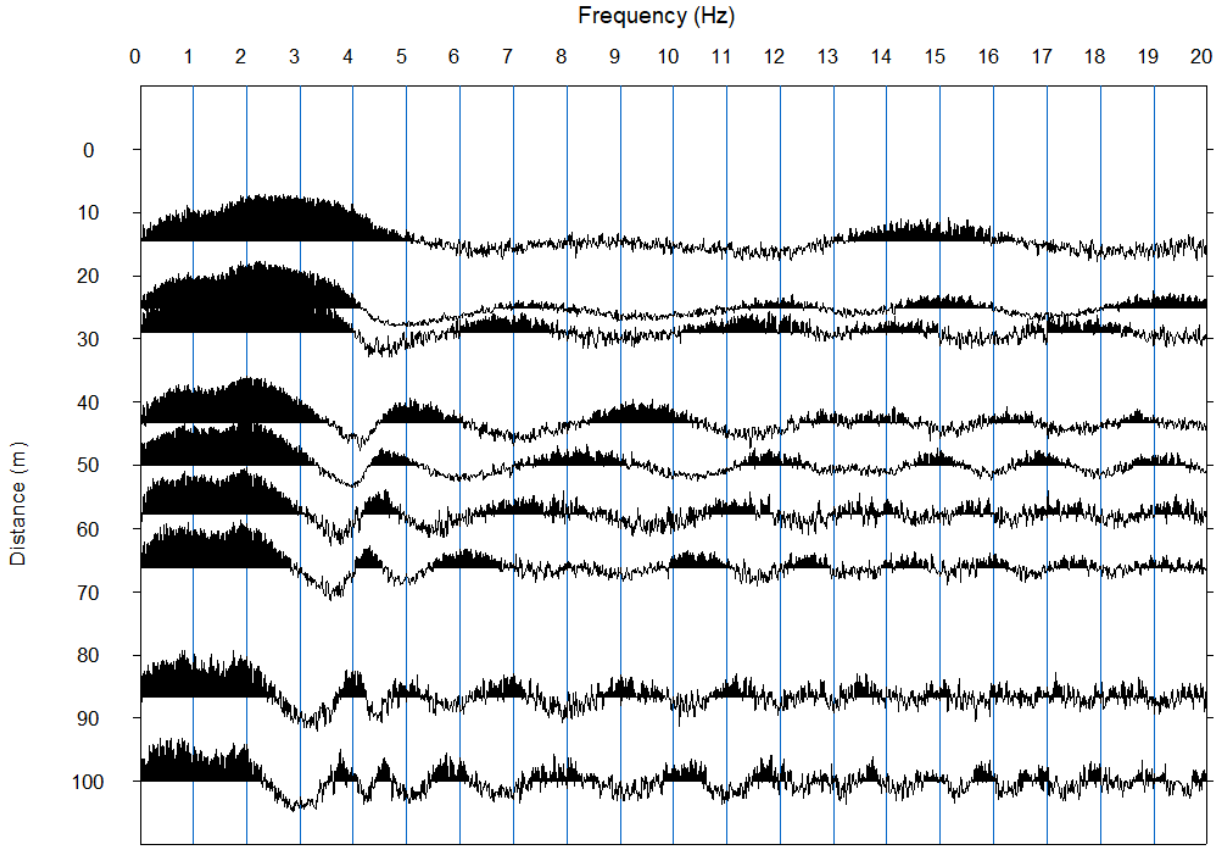


Examples of coherencies

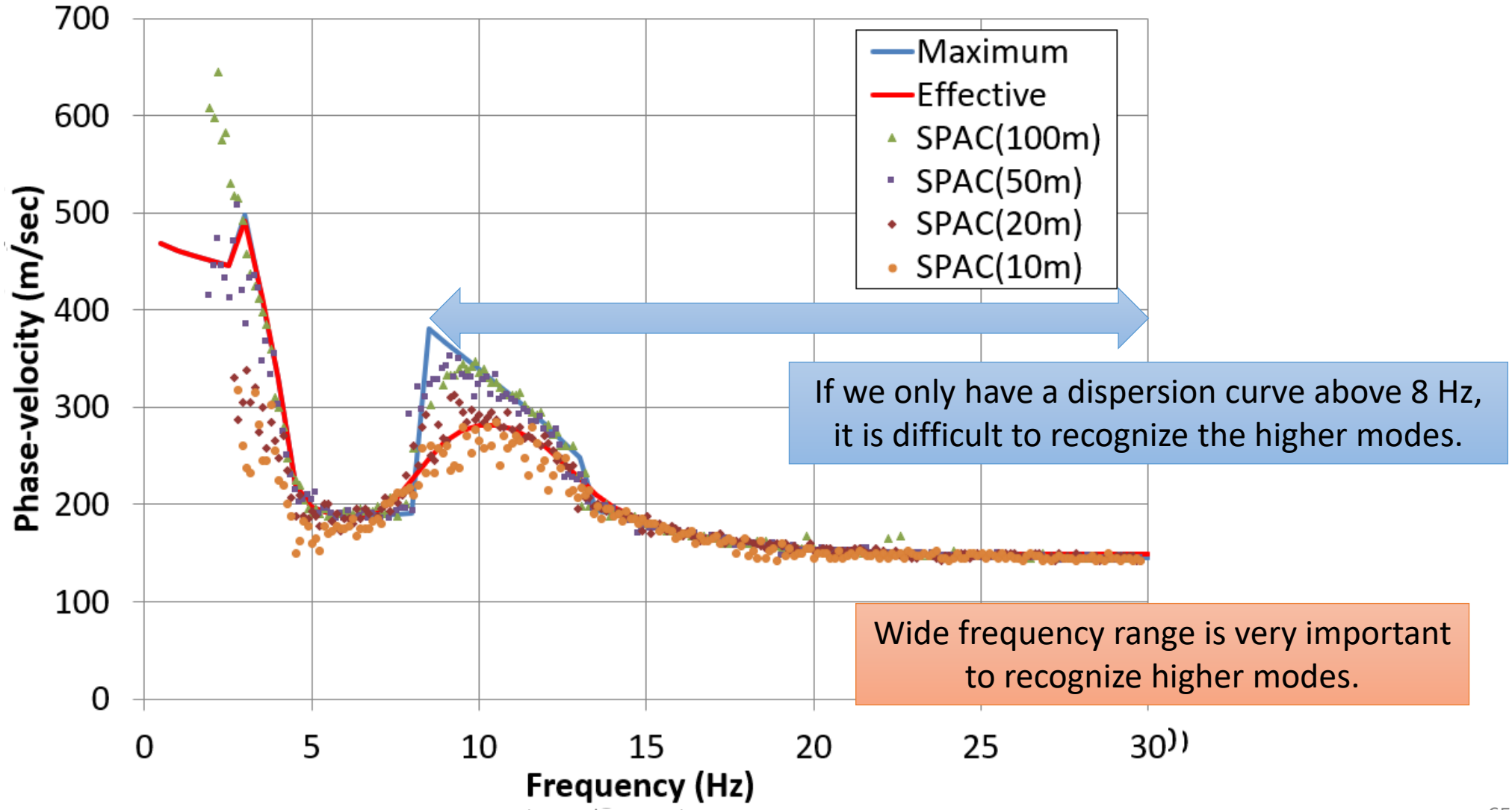
10 m array



100 m array



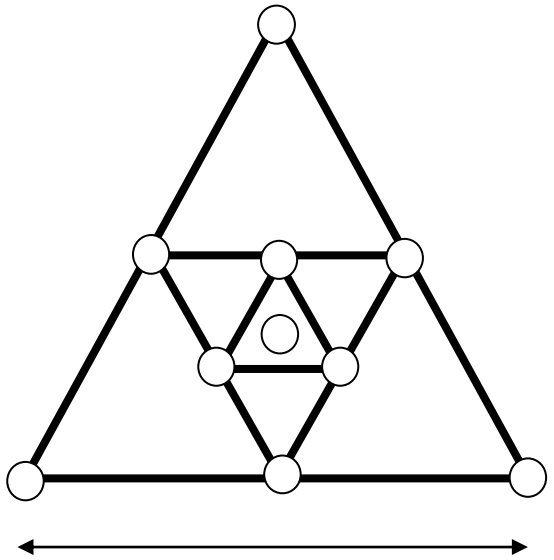
Comparison of dispersion curves with different array size



Uncertainties in surface wave investigations

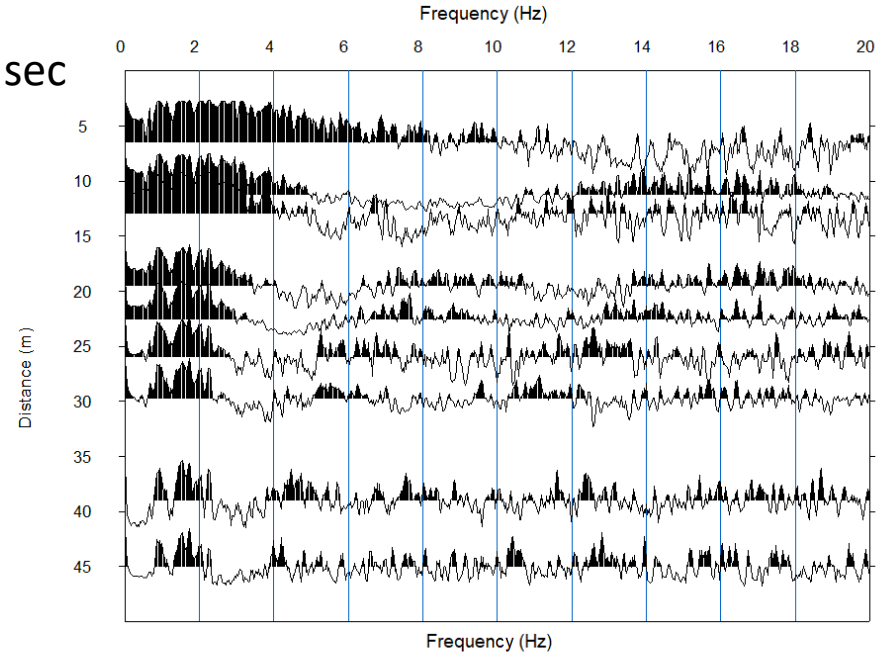
- Review Spatial Auto-correlation method
- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

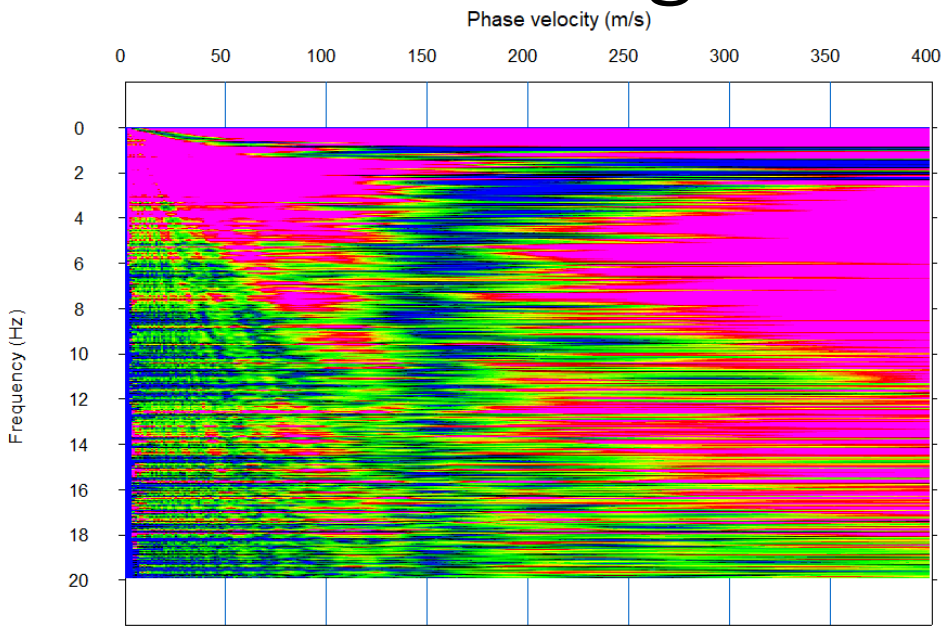
Change of dispersion curve associated with data length



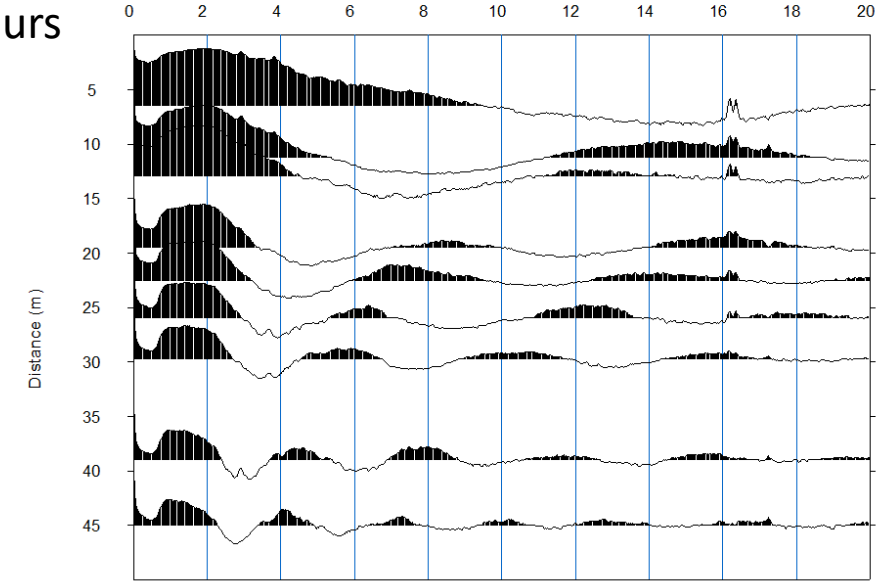
45 m

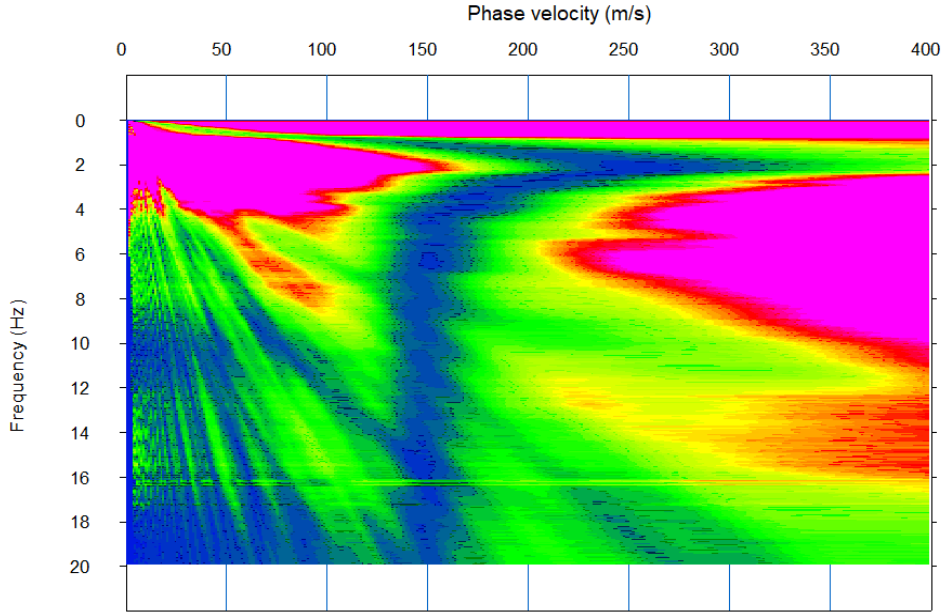
32 sec



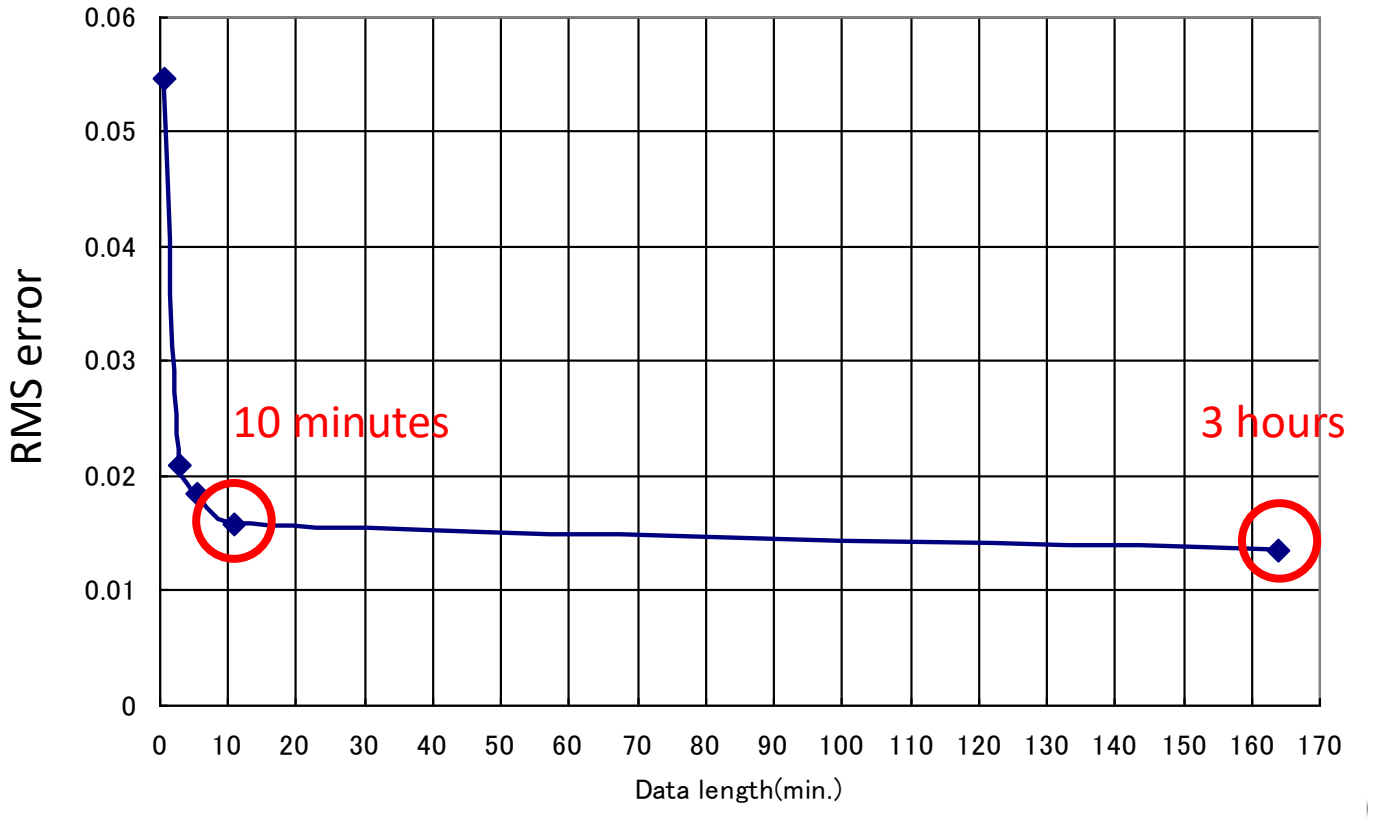


3 hours





RMS error between observed coherency (SPAC) and Bessel function



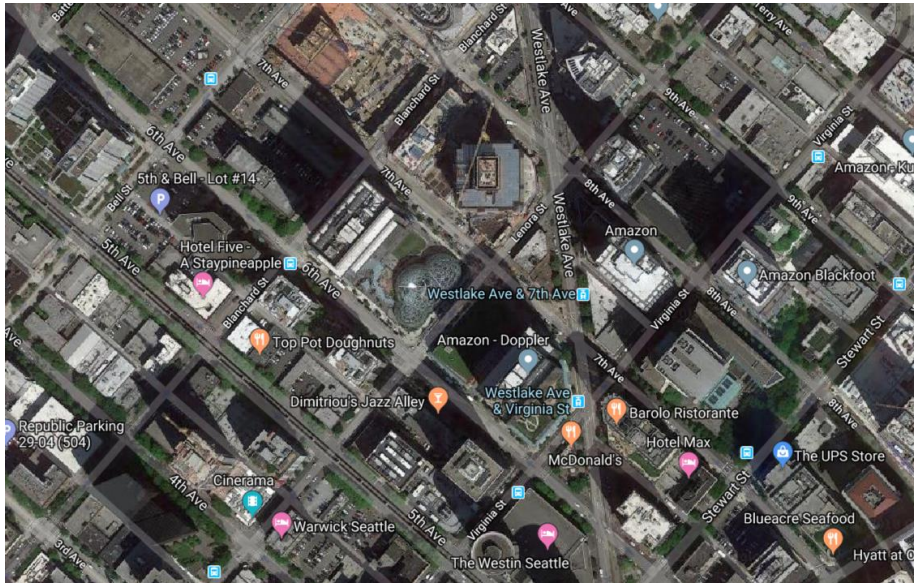
Array size (m)	Data length (min.)
< 50	15
50 ~ 500	30
< 500	60

We know necessary data length empirically

Uncertainties in surface wave investigations

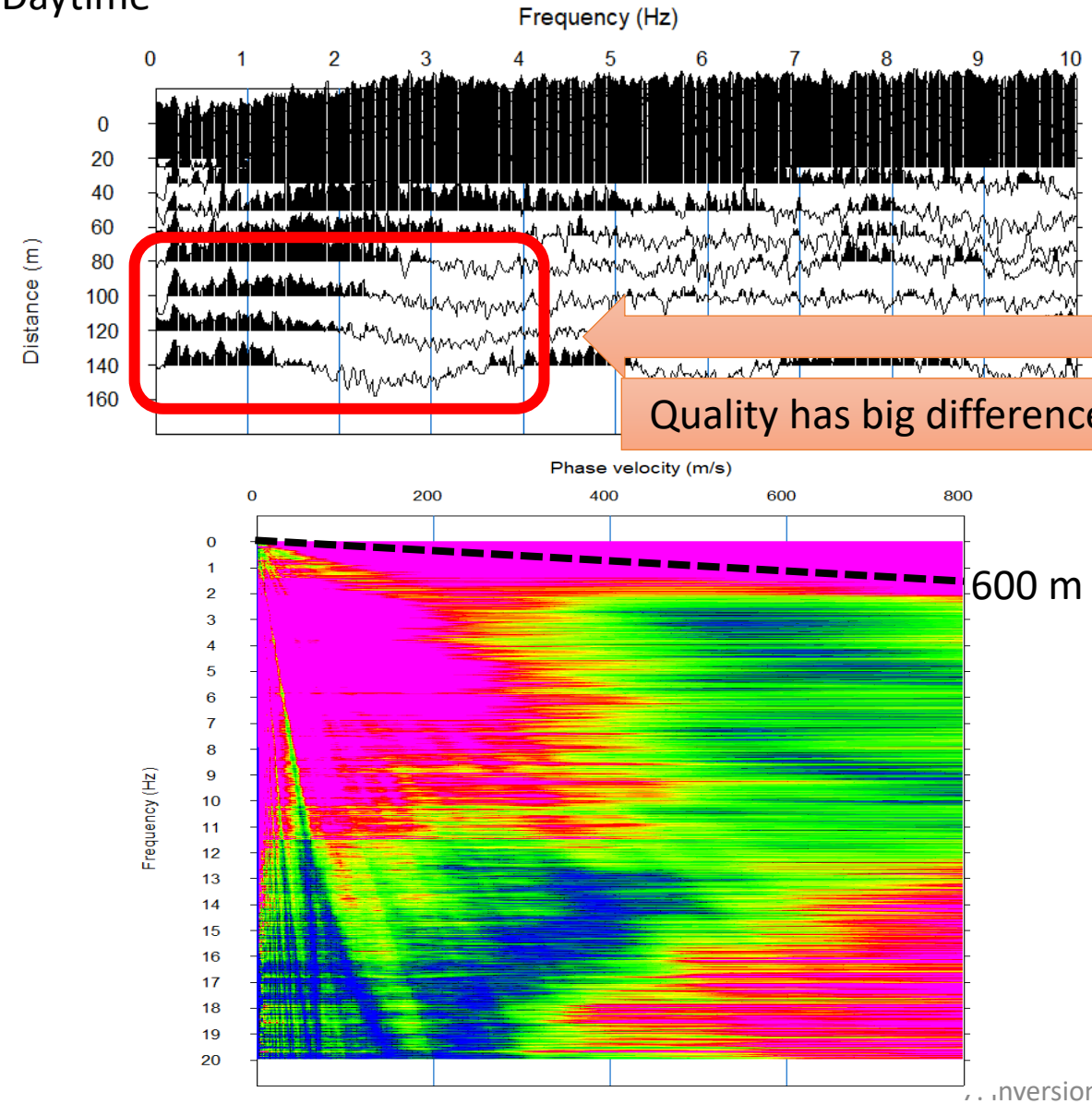
- Review Spatial Auto-correlation method
- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

Seattle downtown



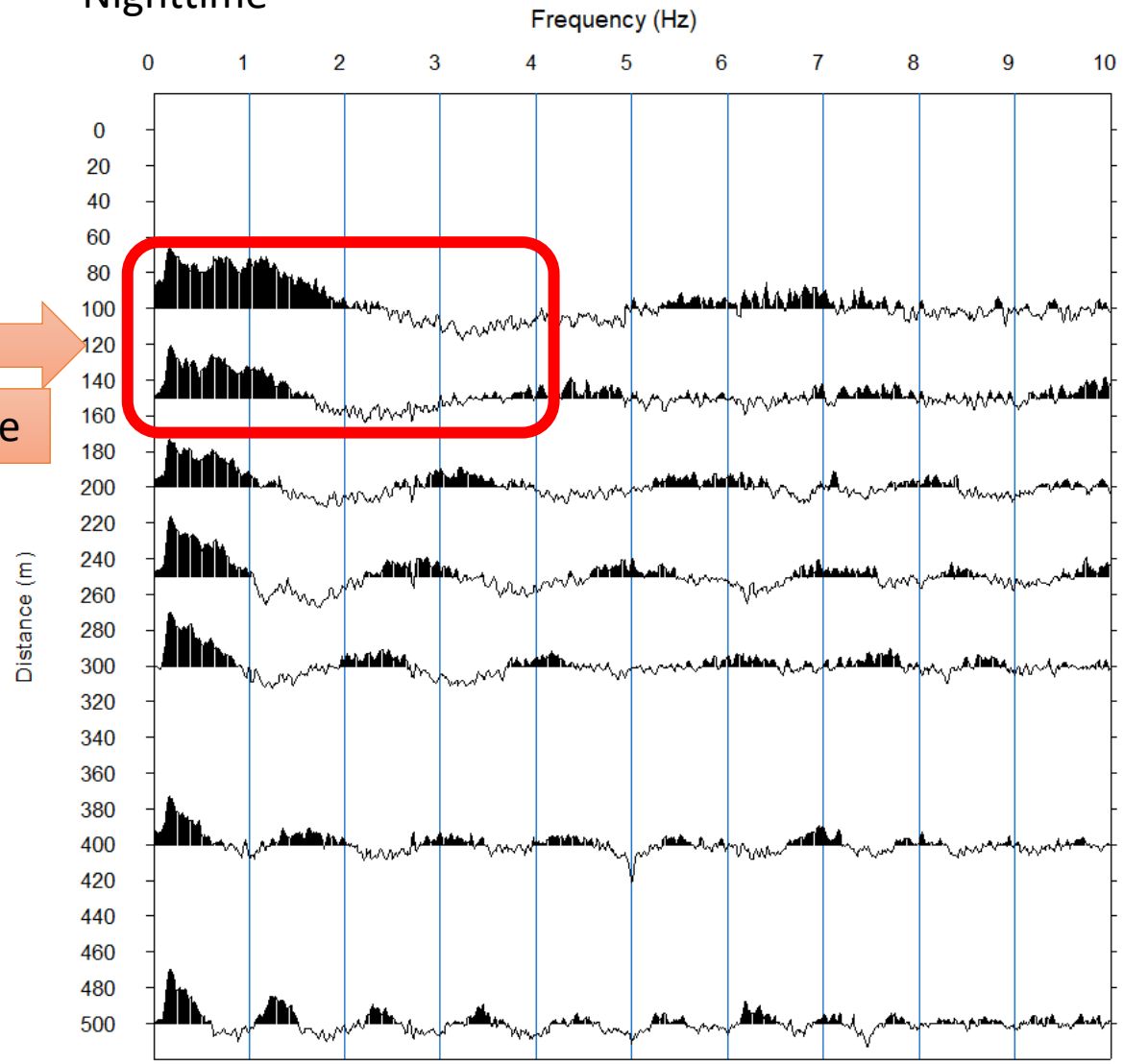
Coherencies obtained at day time and night time

Daytime



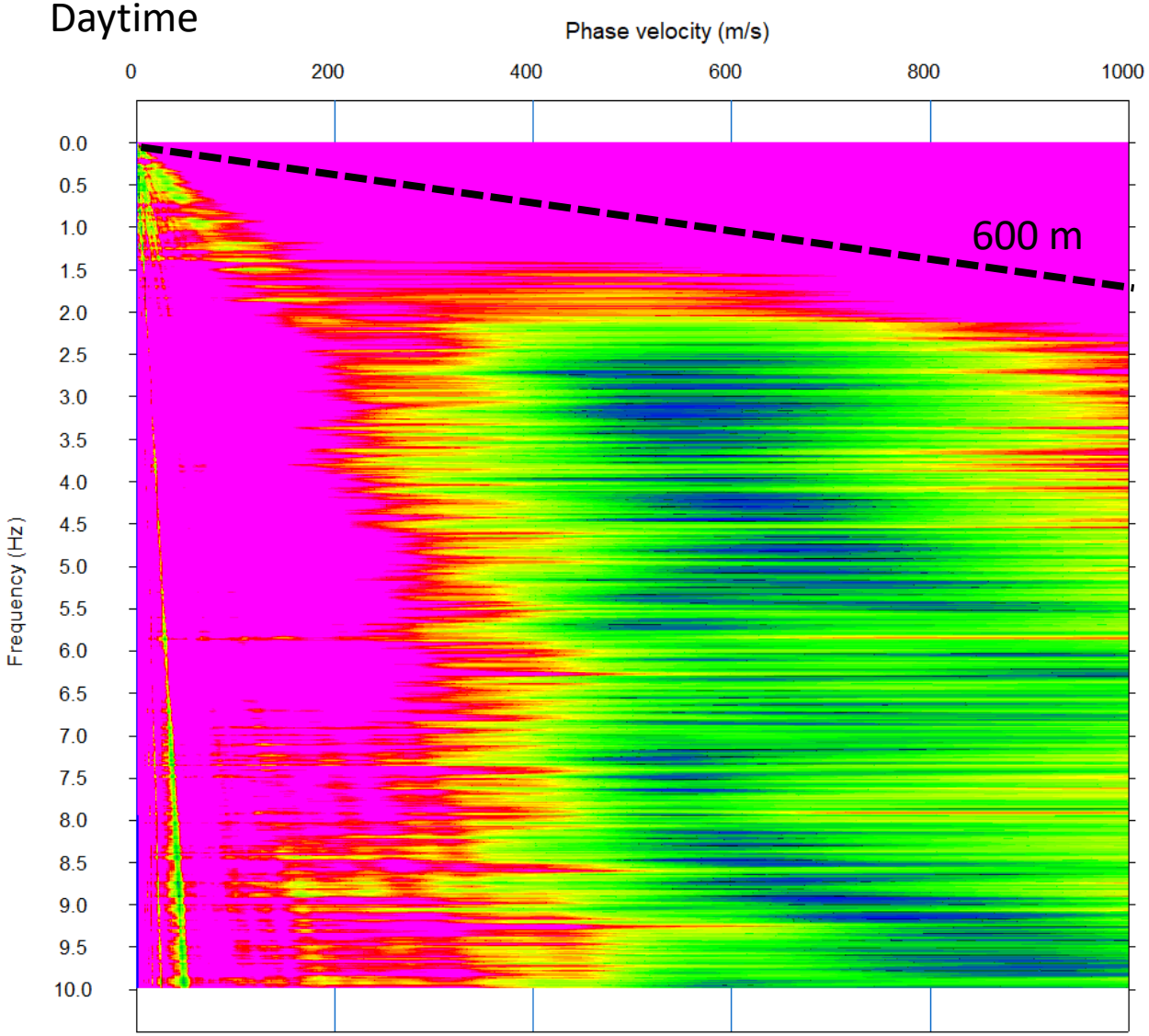
Quality has big difference

Nighttime

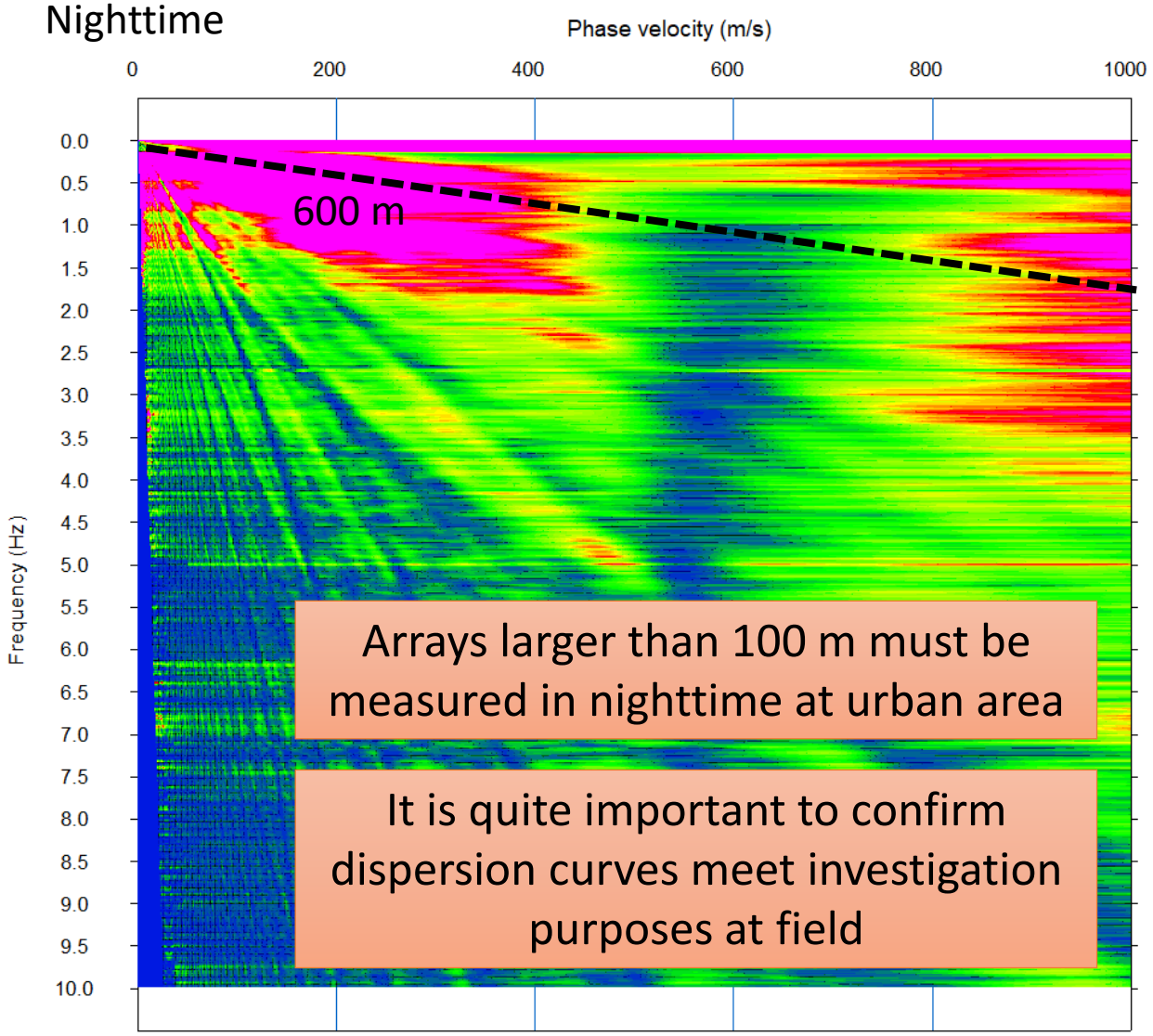


Coherencies obtained at day time and night time

Daytime



Nighttime



Arrays larger than 100 m must be measured in nighttime at urban area

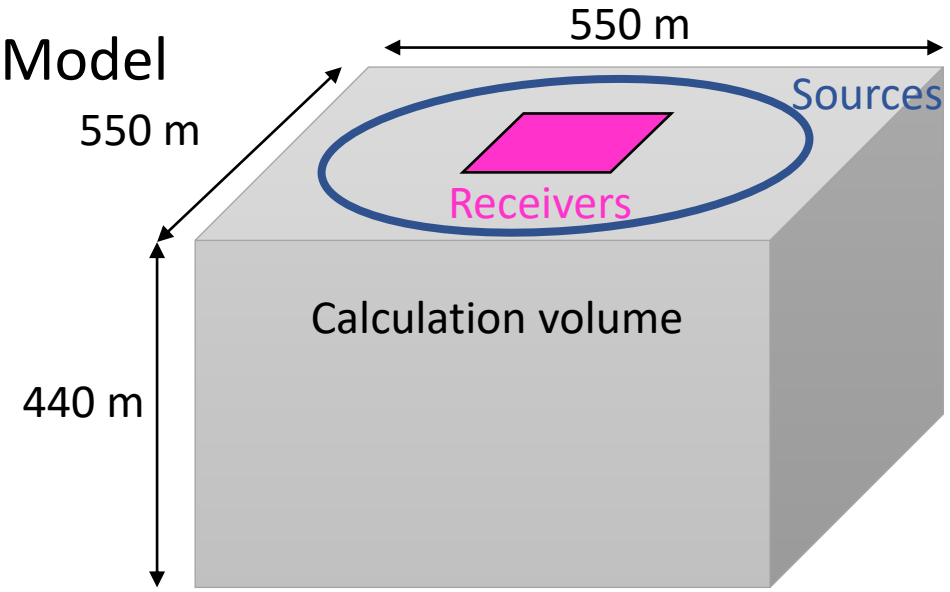
It is quite important to confirm dispersion curves meet investigation purposes at field

Uncertainties in surface wave investigations

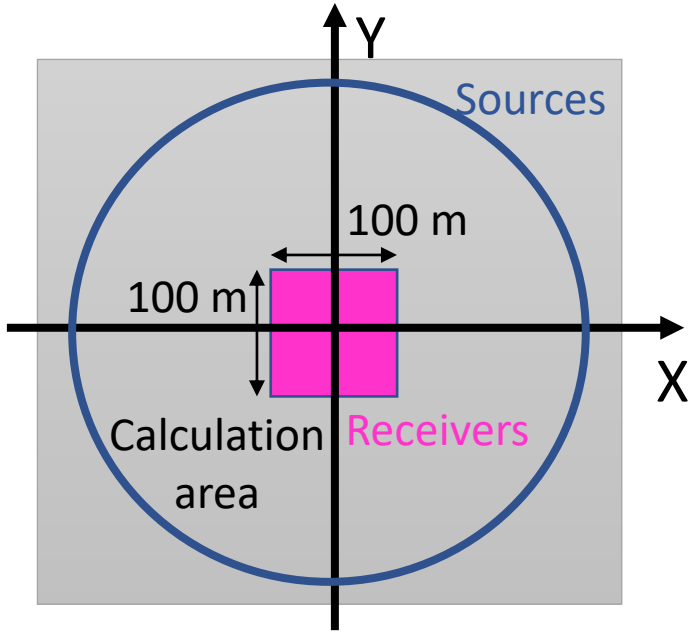
- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

3D Finite-difference simulation

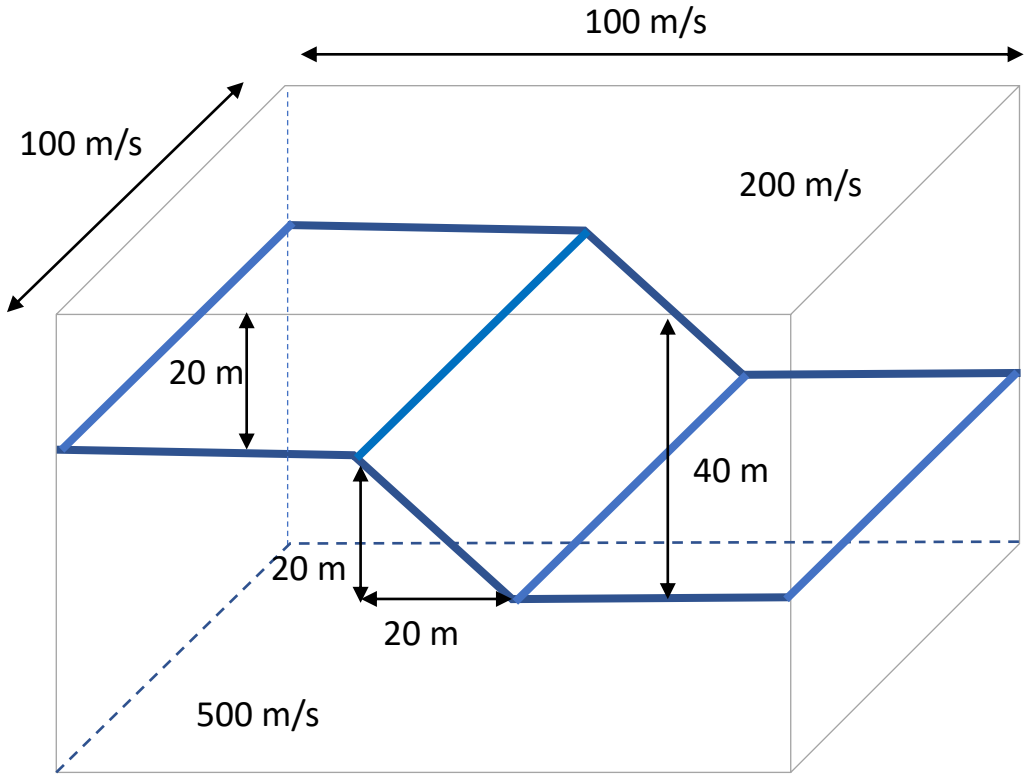
3D Model



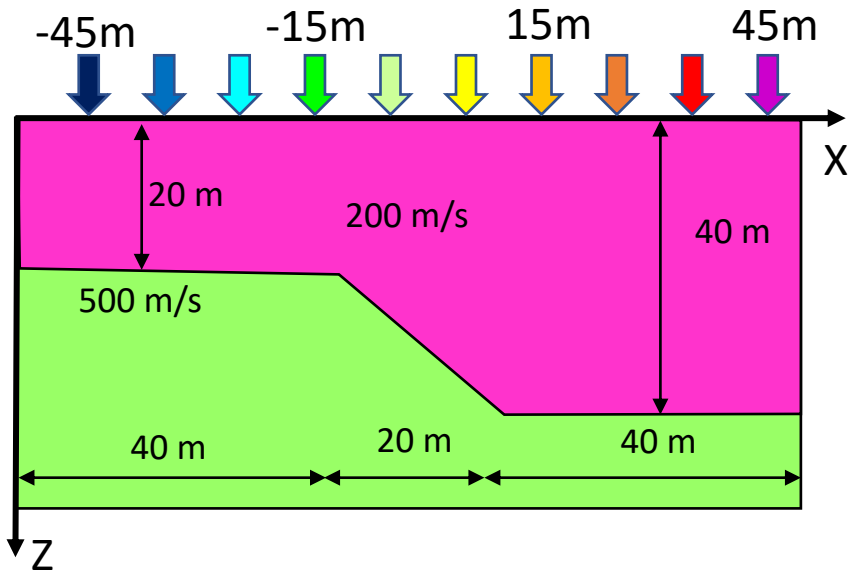
Plan view

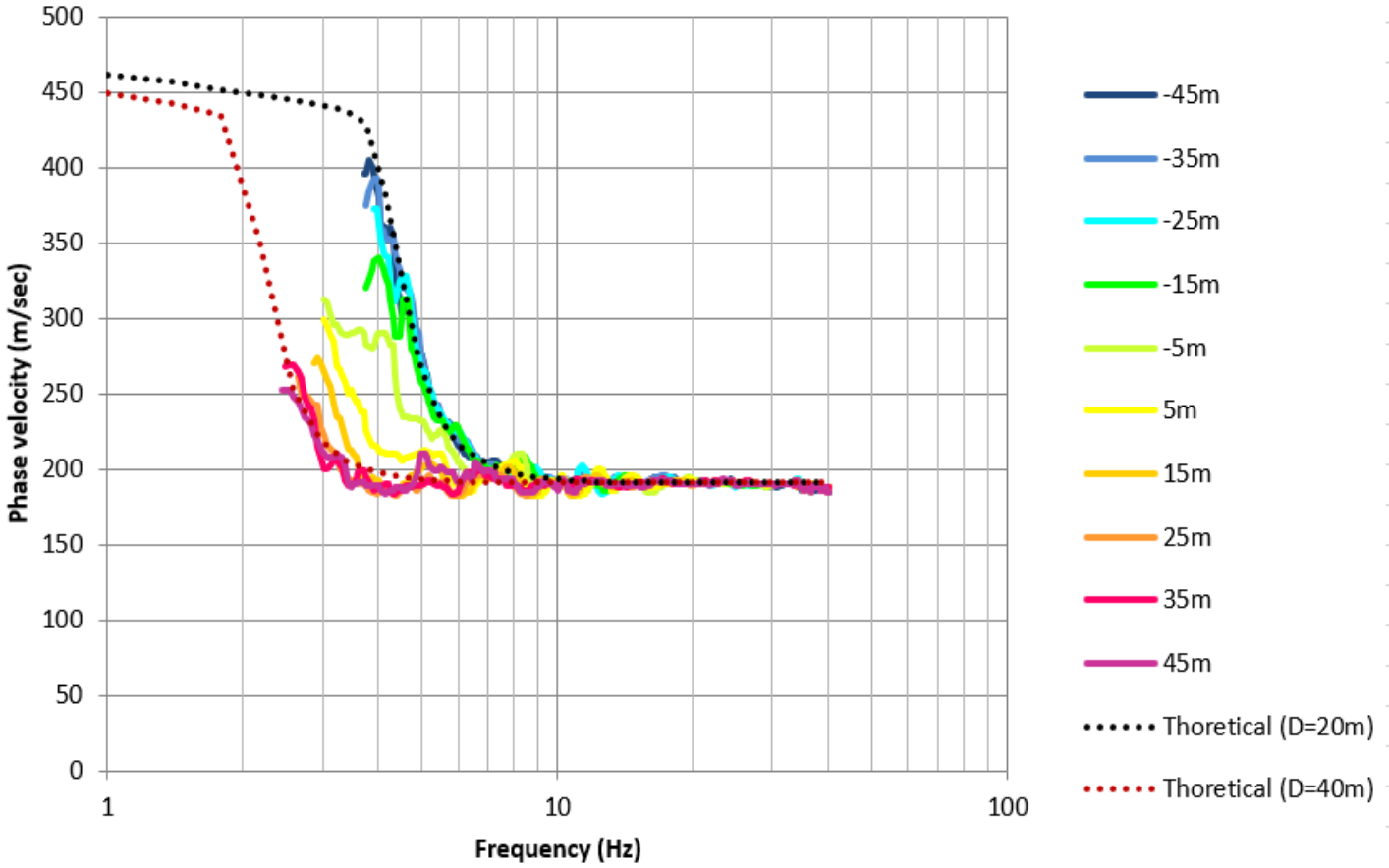


2 layer velocity model with a step



Comparison of dispersion curves cross the step





Dispersion curves obtained from ambient noise mainly reflect S-wave velocity beneath receiver array

Uncertainties in surface wave investigations

- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

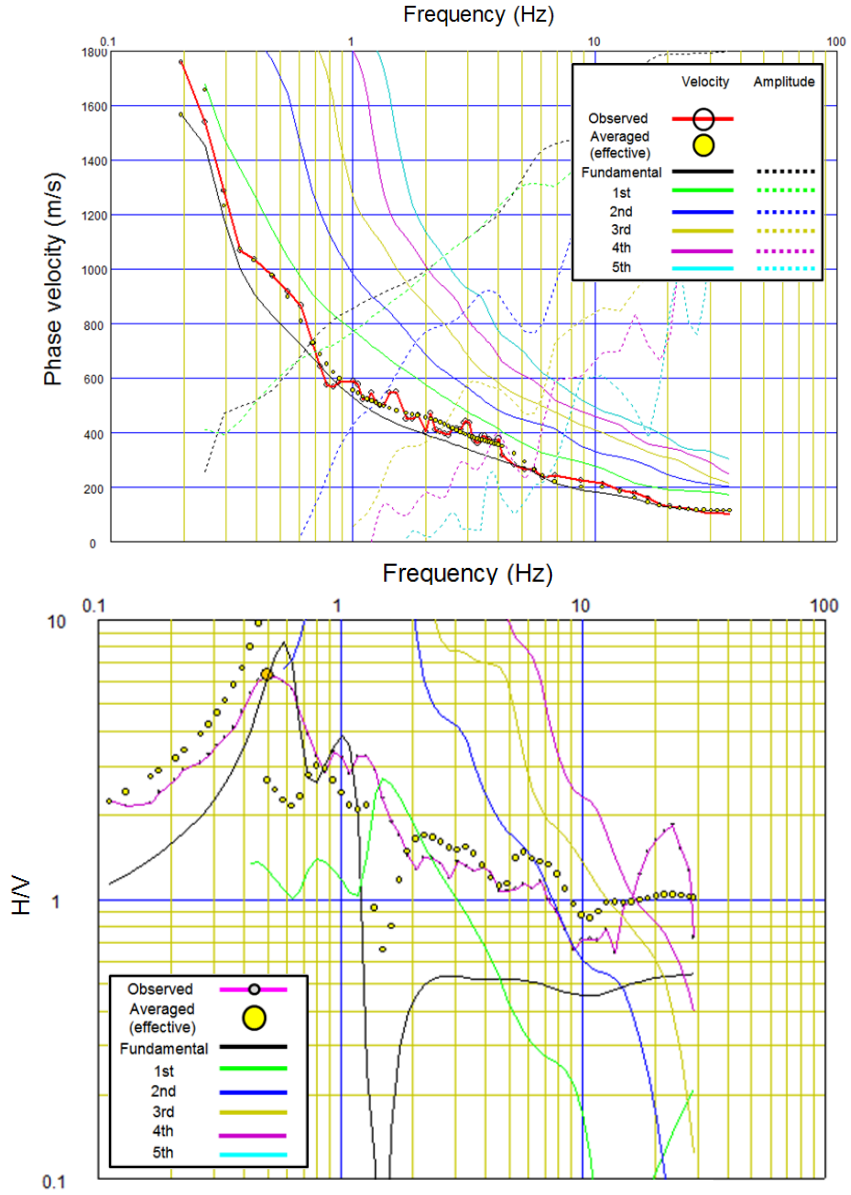
Conclusions

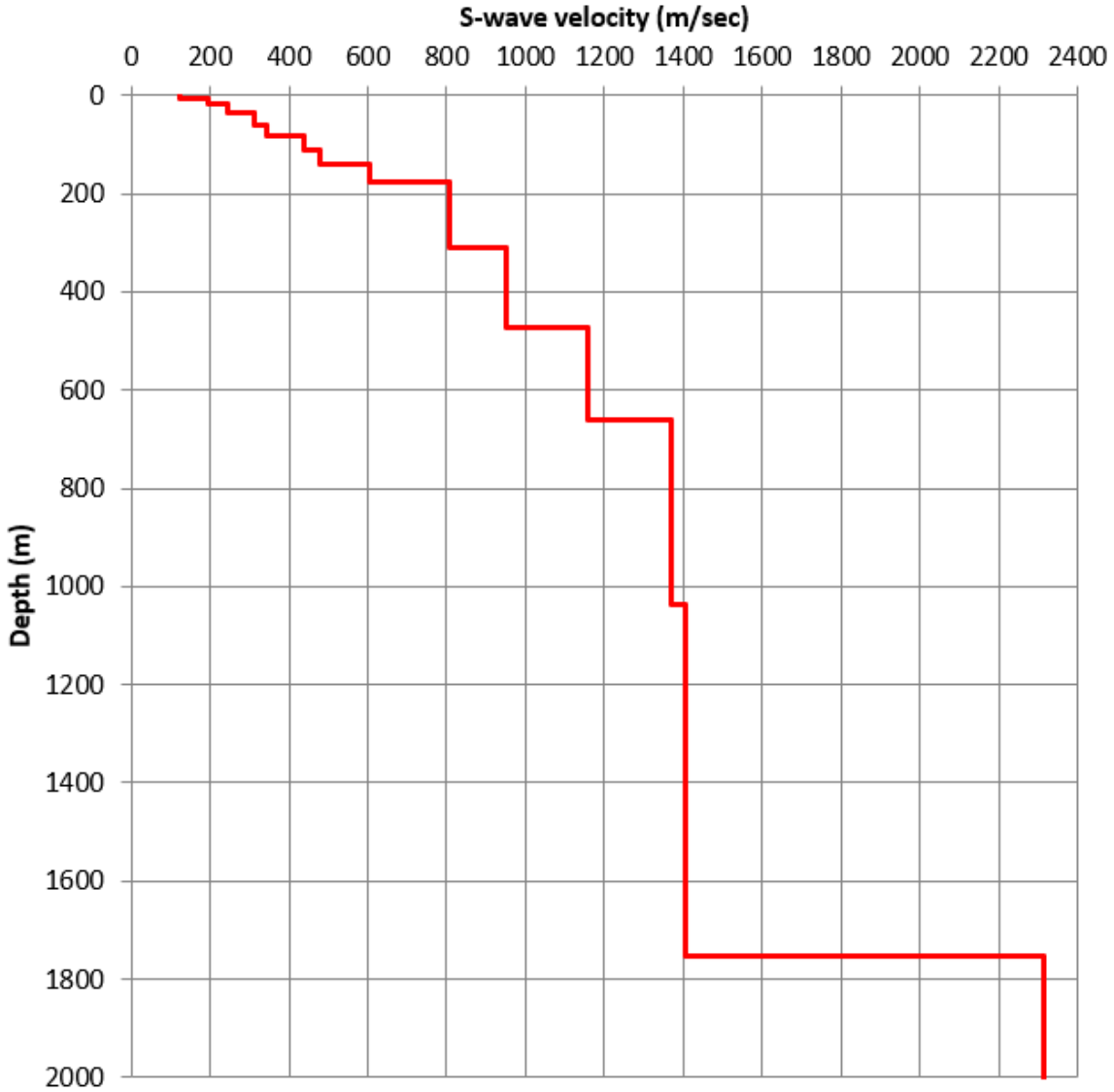
- Array shape and propagation direction of ambient noise
  Irregular arrays work better than we expected.
  But, omni-directional or multi-directional array should be better and more reliable.
- Array size and investigation depth
  FK provides phase velocity associated with wave length up to the same as array size.
  SPAC provides phase velocity associated with wave length up to the double of array size.
  FK tends to provide higher phase velocity and SPAC tends to provide lower phase velocity.
- Higher modes
  Wide frequency range is very important to recognize higher modes.
- Data length
  We know necessary data length empirically.
- Daytime or nighttime
  Arrays larger than 100 m must be measured in nighttime at urban area.
- Horizontal velocity change
  Dispersion curves obtained from ambient noise mainly reflect S-wave velocity beneath receiver array.

Uncertainties in surface wave investigations

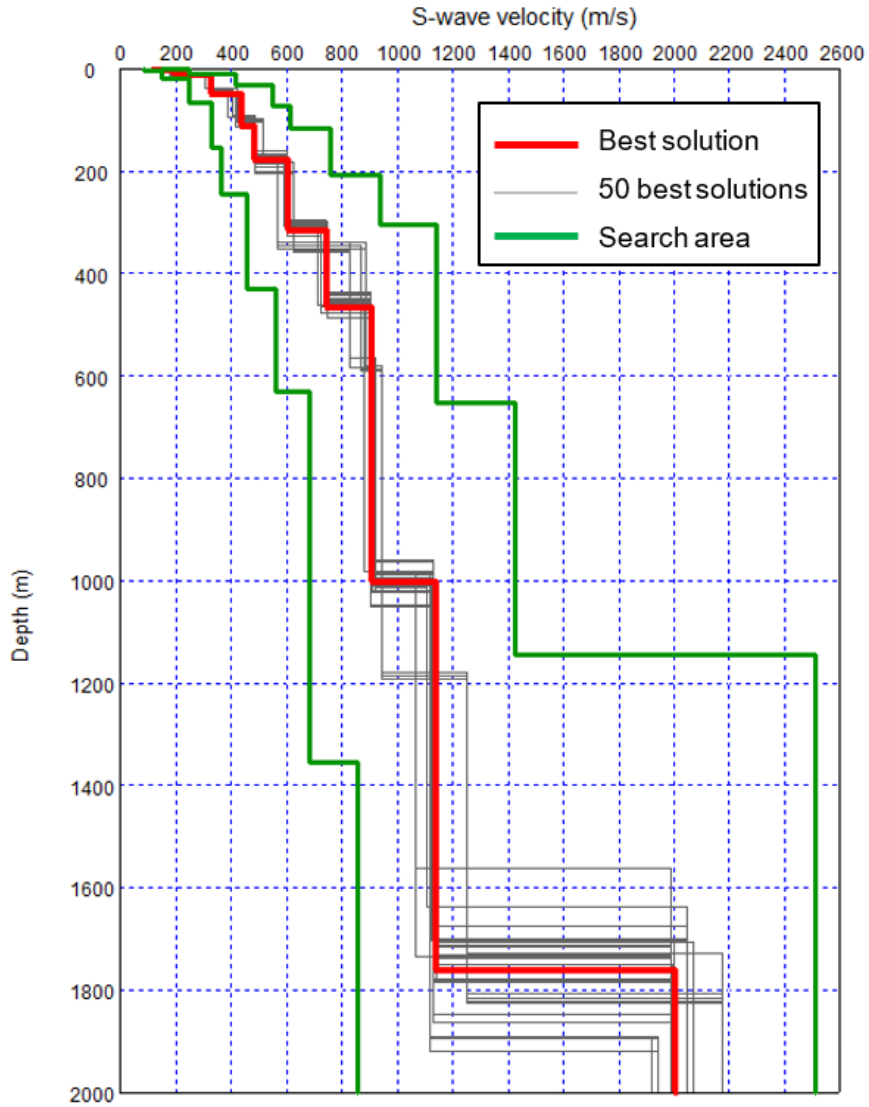
- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

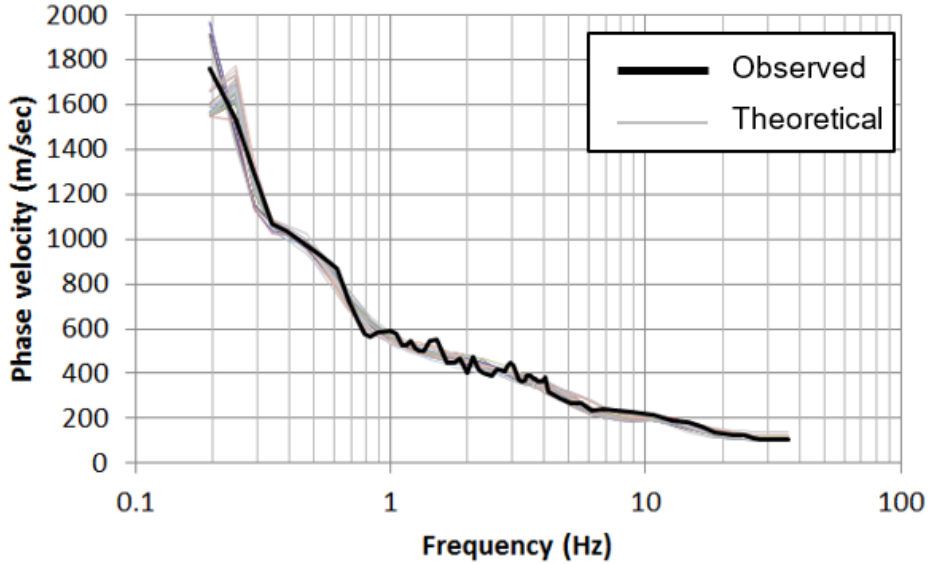
GA and uncertainty evaluation

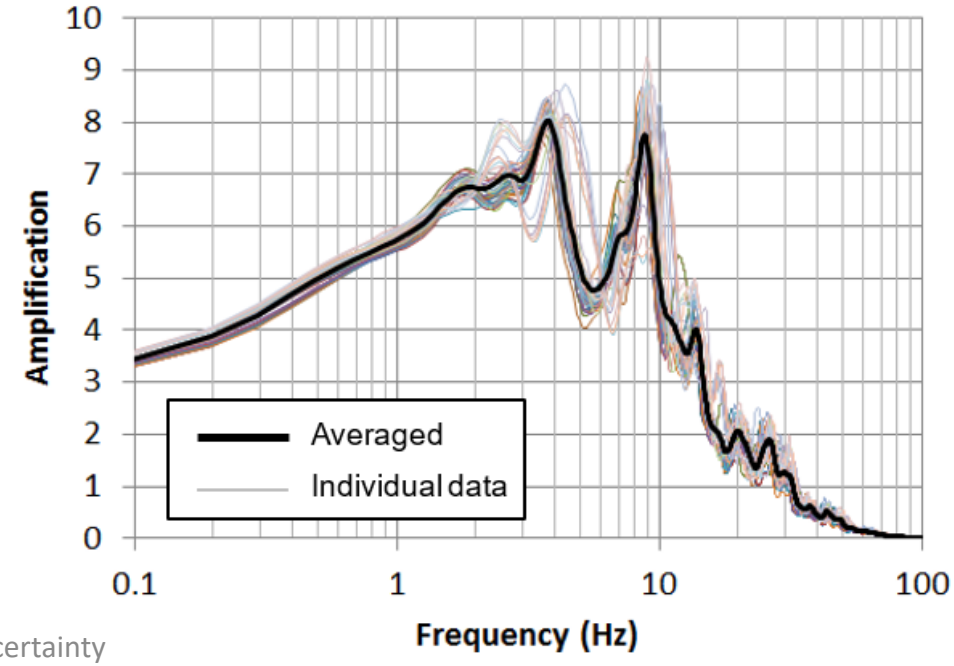




GA and uncertainty evaluation

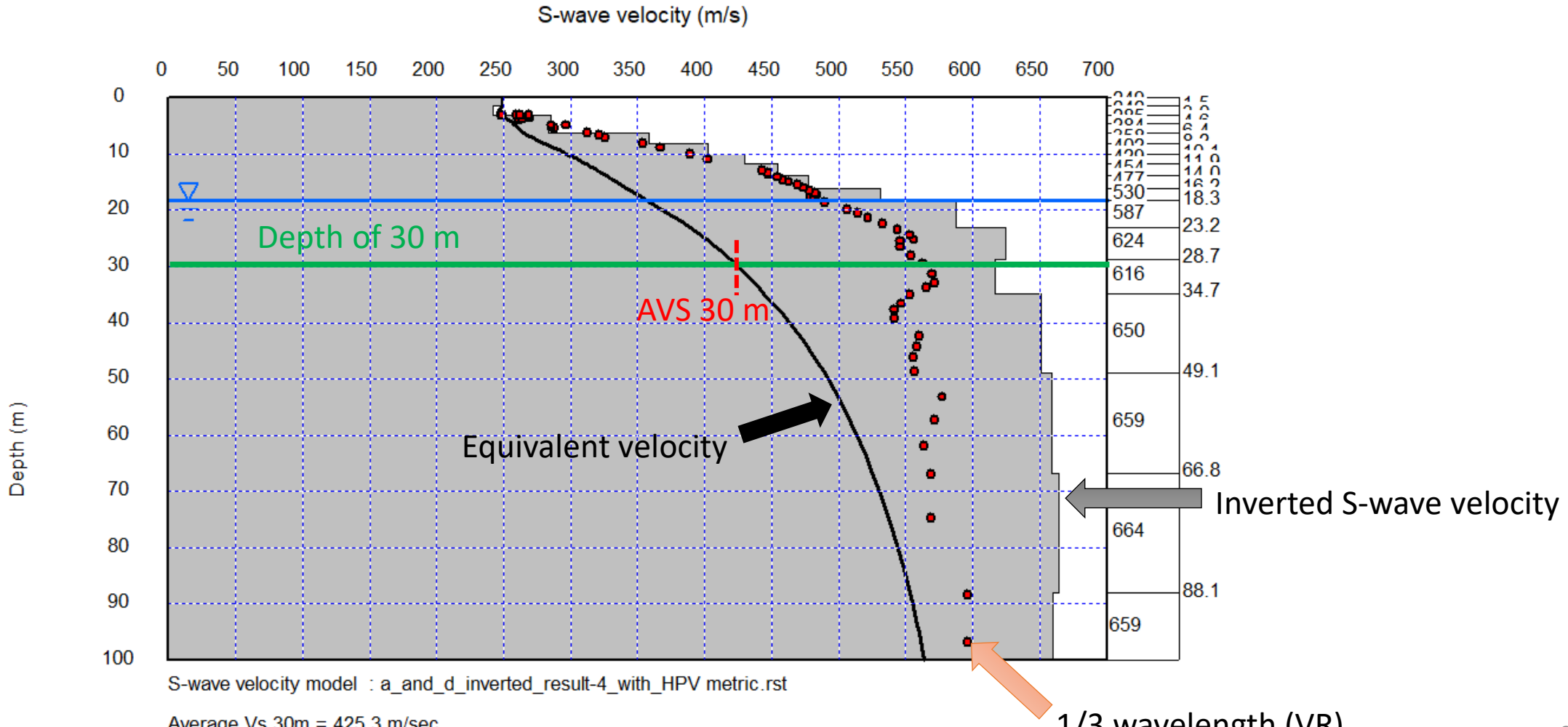






Equivalent velocity

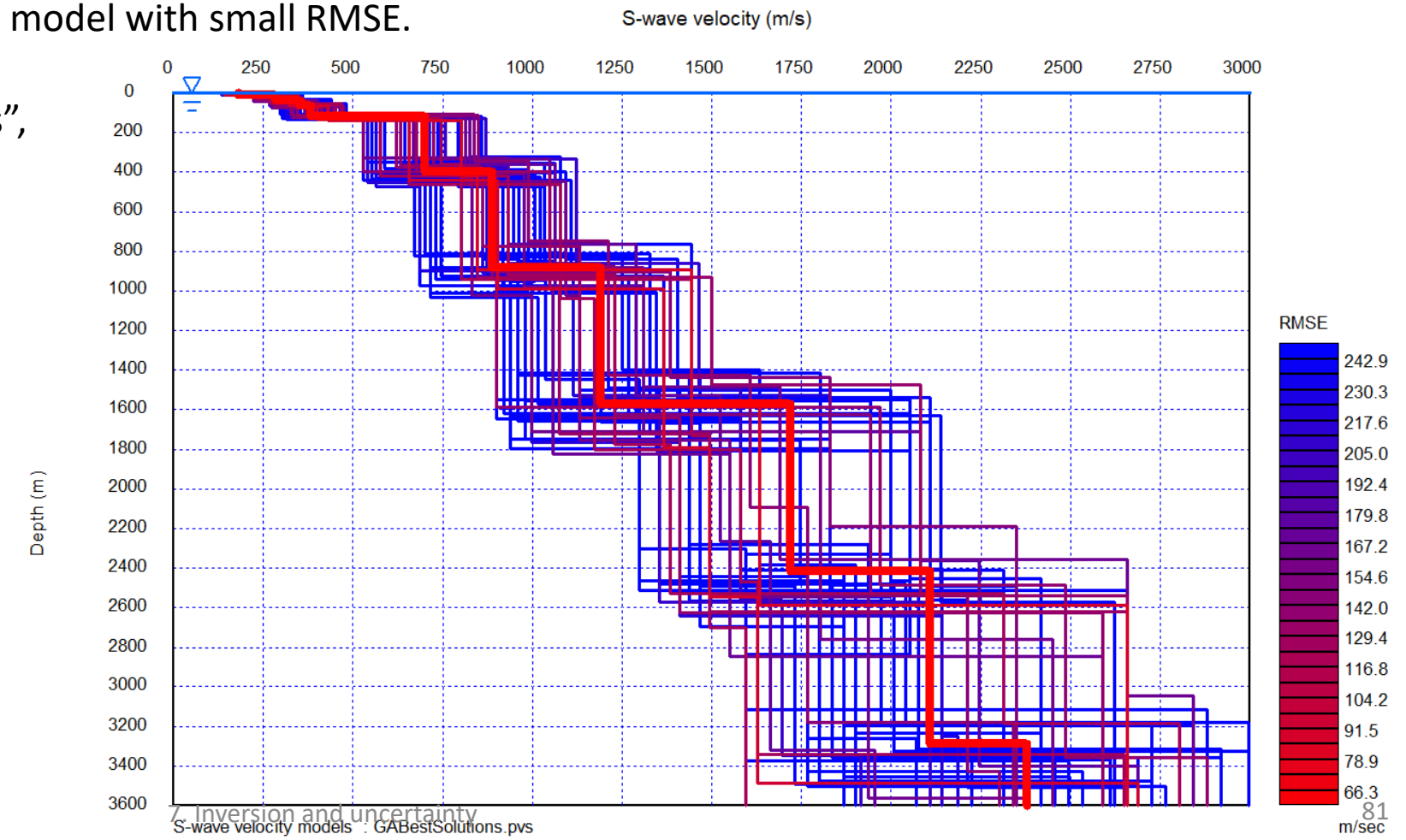
Equivalent velocity *Ev(d)* is a time averaged velocity to a depth of *d*. For example, *Ev(30 m)* corresponds to AVS 30 m. The equivalent velocity is quite important to evaluate site amplification of earthquakes. Check “Velocity model”, “Show equivalent velocity” to on to show the equivalent velocity. Note that it only works for 1D data.



Solution members obtained from GA shown together with their RMSE

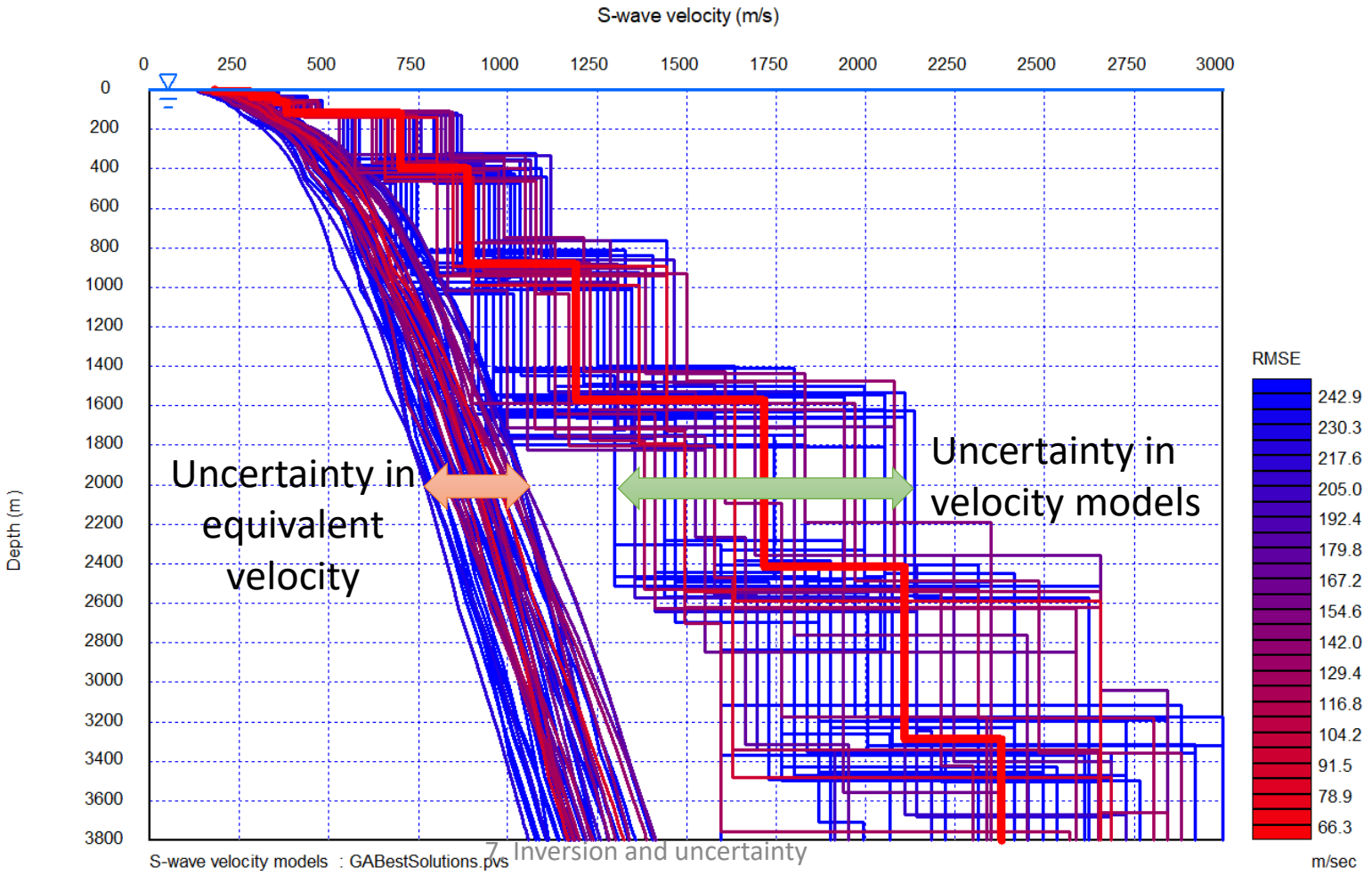
Genetic Algorithm (GA) yields candidates of solution (members) together with the best solution. The members are automatically saved in “GABestSolutions.pvs” after the calculation of GA completed. The members can be used to evaluate uncertainty associated with the inversion. WaveEq shows the solution members with their RMSE as shown below. Velocity models shown as red color correspond to the model with small RMSE.

Check “Velocity model”, “Advanced options”, “Show GA RMSE” to on to show RMSE.



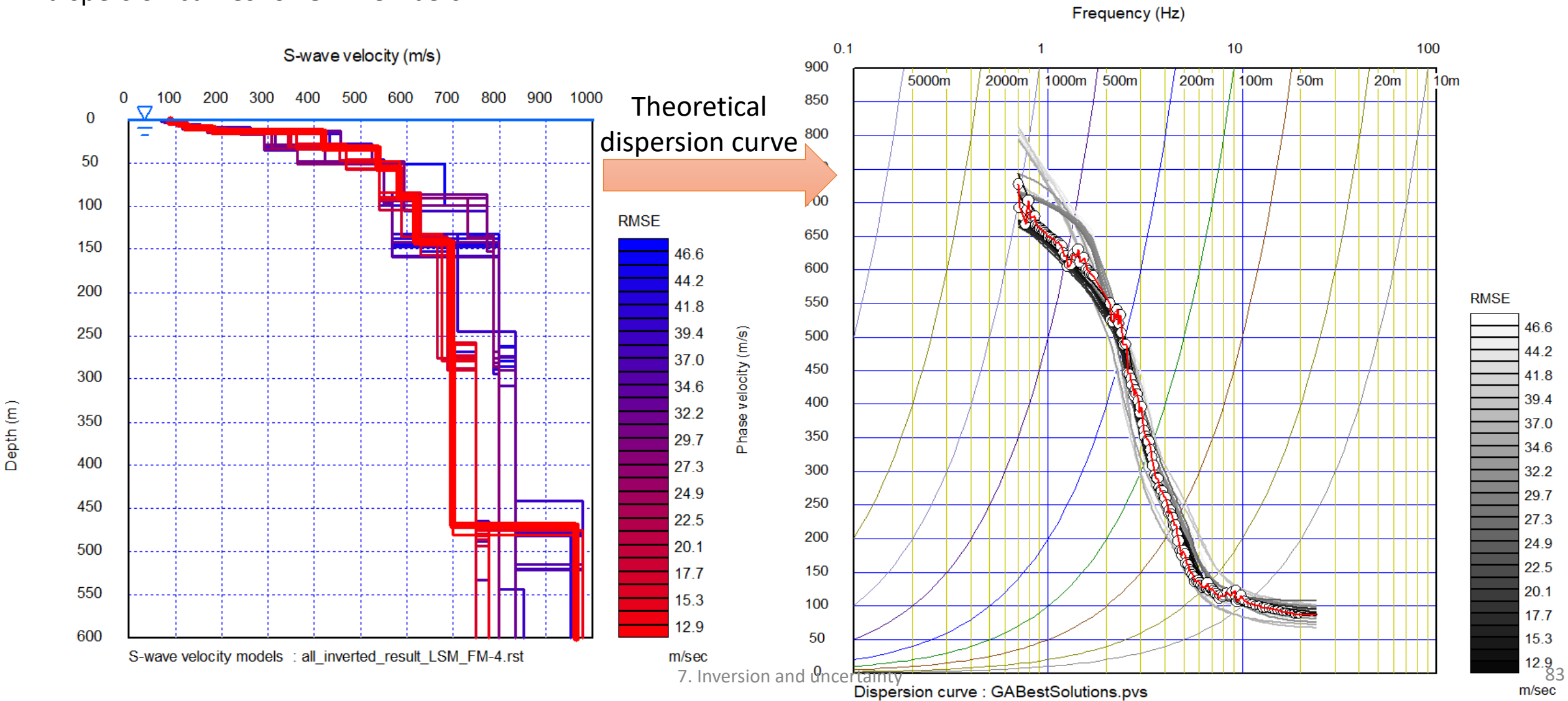
Equivalent velocity for solution members

Equivalent velocity can be shown for solution members obtained by GA. We can recognize that the uncertainty of the equivalent velocities is much smaller than one of velocity models.



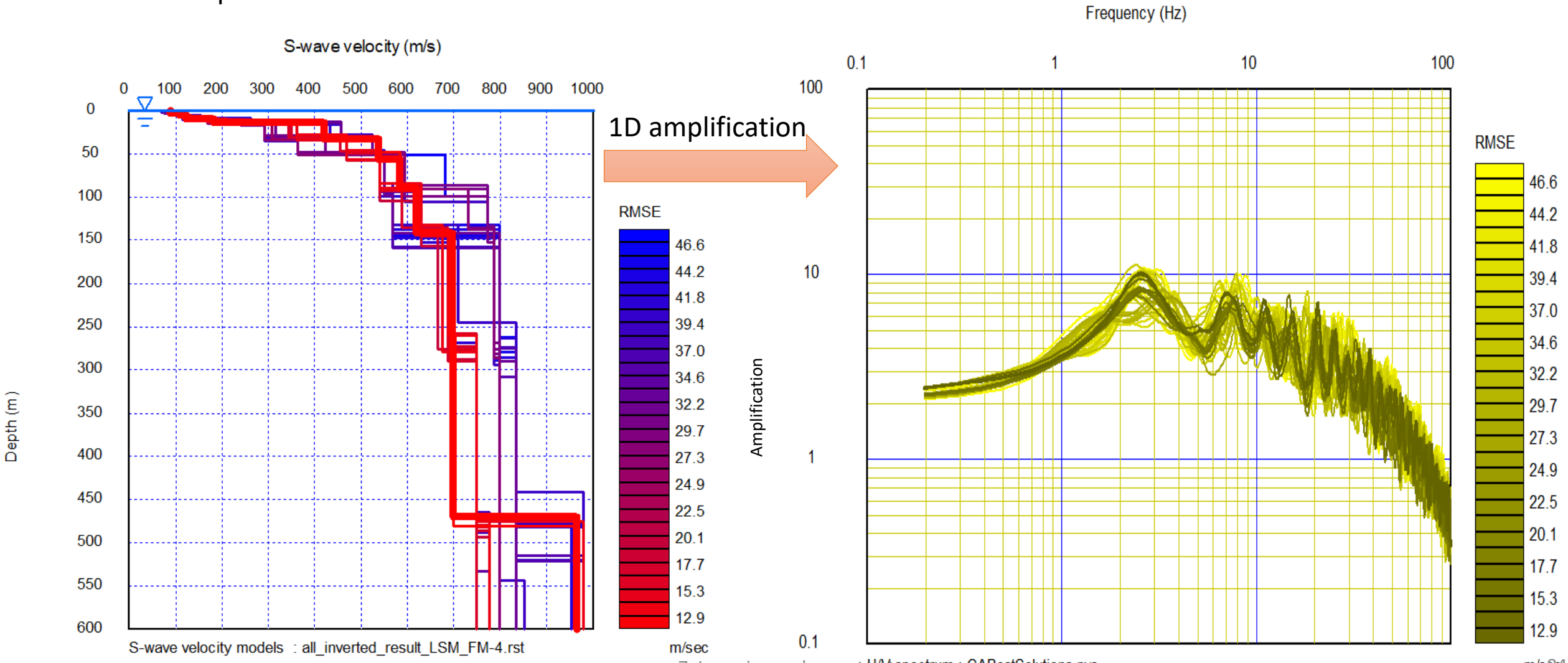
Theoretical dispersion curves for solution members

Check “MASW/MAM (1D)”, “Advanced inversion”, “Show all theoretical data for GA members ” to on to show all theoretical dispersion curves for GA members



Theoretical 1D amplification for solution members

Check “MASW/MAM (1D)”, “Advanced inversion”, “Calculate 1D amplification for GA members ” to on to show all theoretical dispersion curves for GA members



Uncertainties in surface wave investigations

- Array shape and propagation direction of ambient noise
- Array size and investigation depth
- Higher modes
- Data length
- Daytime or nighttime
- Horizontal velocity change
- Conclusions
- Uncertainty evaluation
- What can we do for reducing uncertainty ?

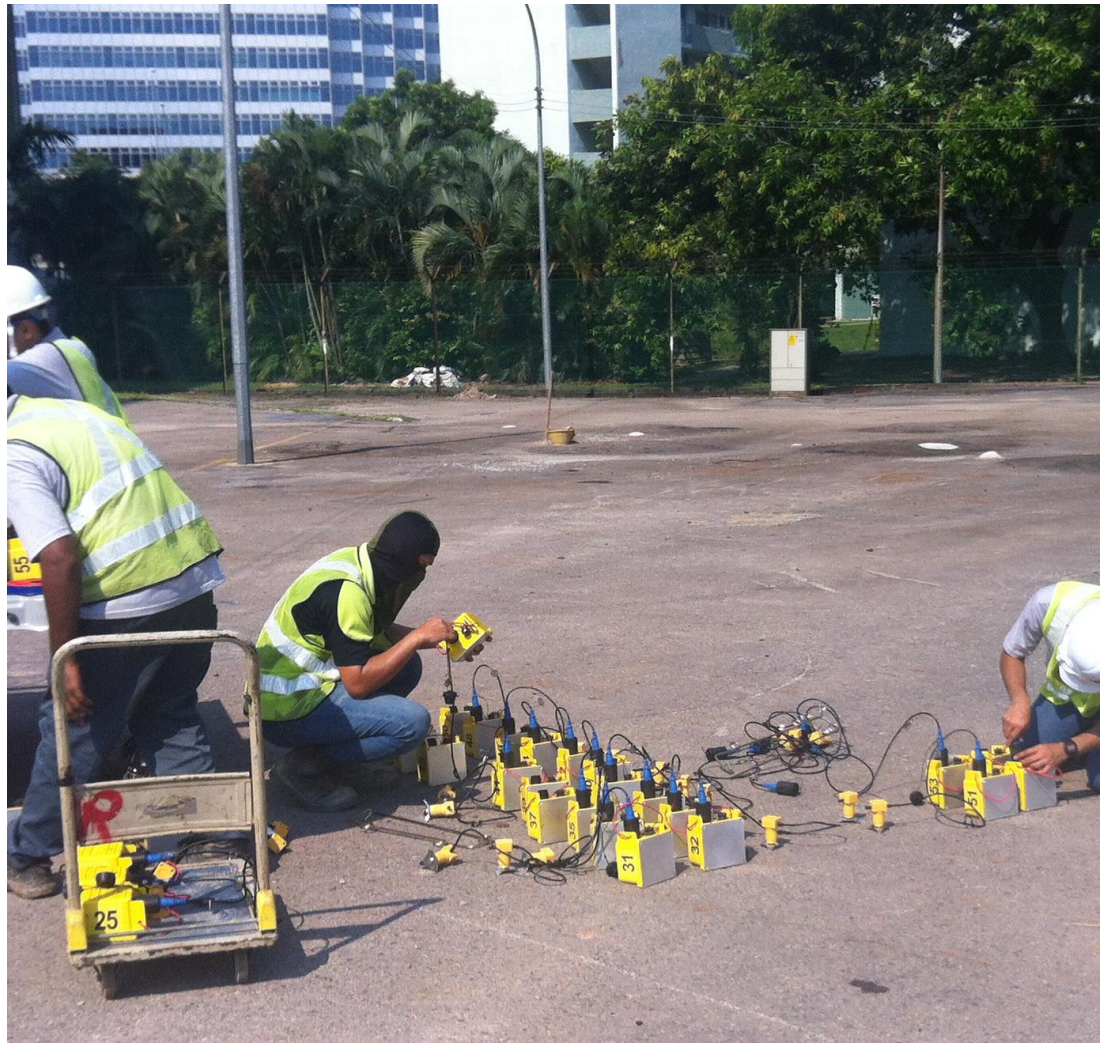
What can we do for reducing uncertainty ?

- Improve data quality
  - Use omni-directional or multi-directional arrays rather than linear array
  - Enlarge frequency range
  - Increase maximum sensor spacing
  - Decrease minimum sensor spacing
  - Increase number of sensors

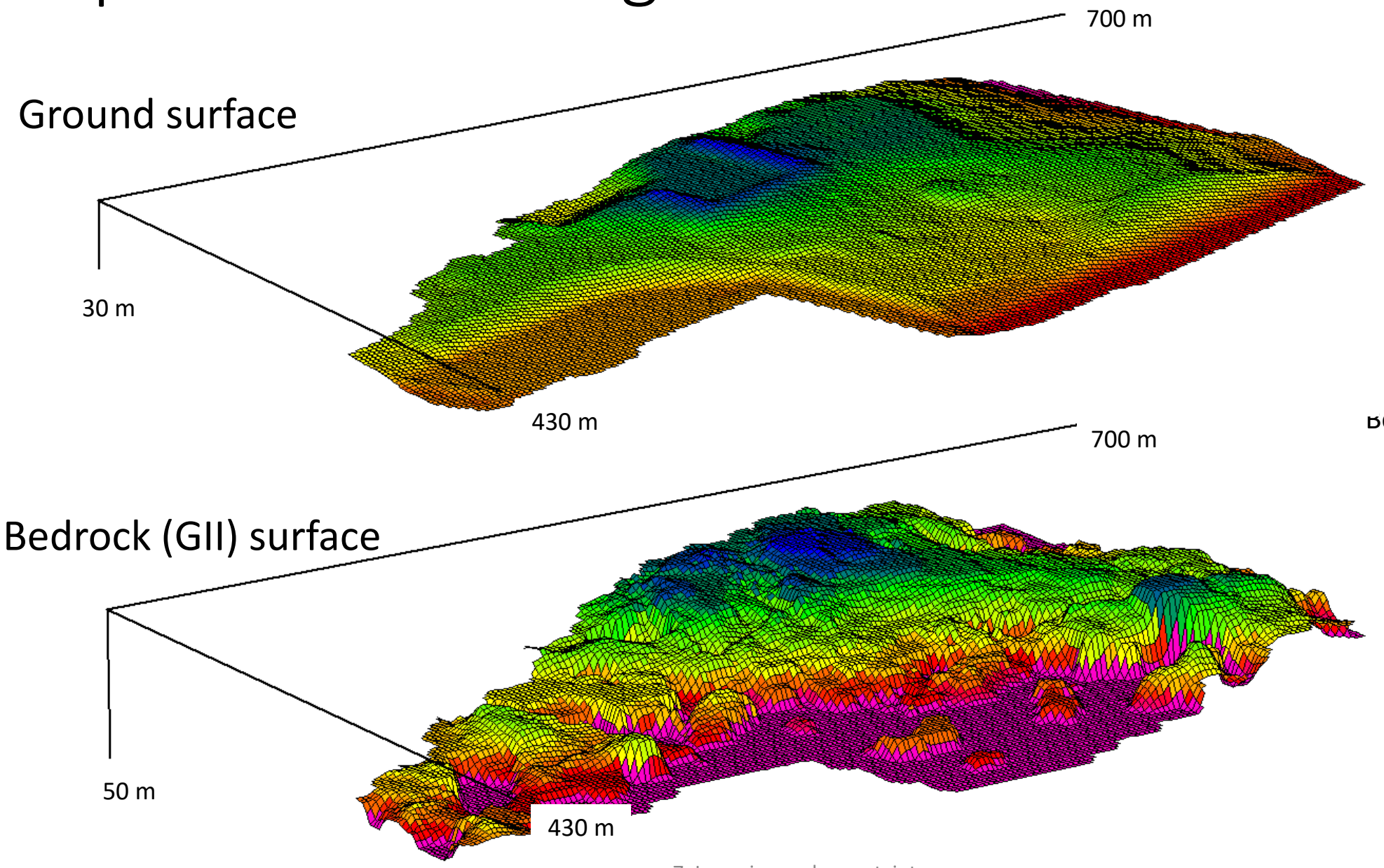
  -> Develop small, cableless, easy-to-use, and inexpensive seismographs and sensors
- Decrease uncertainty in inversion
  - Use a priori information
  - Use appropriate constraint

  -> Develop and use database

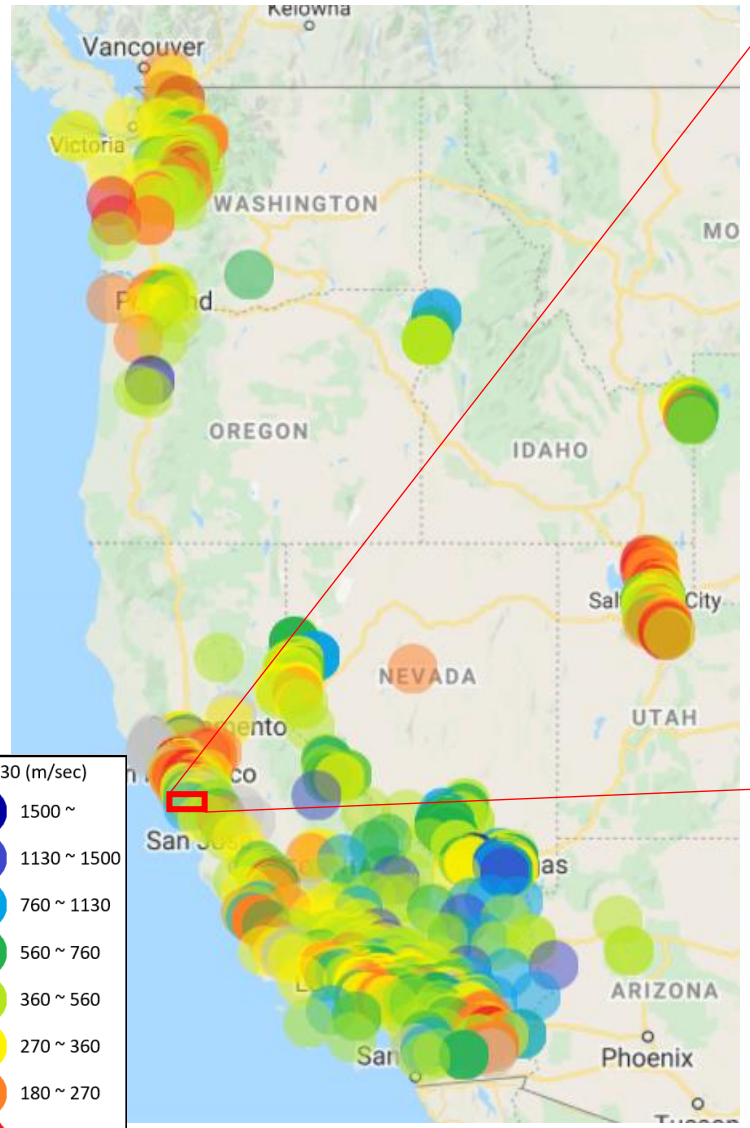
Develop small, cable-less, easy-to-use, and inexpensive seismographs and sensors

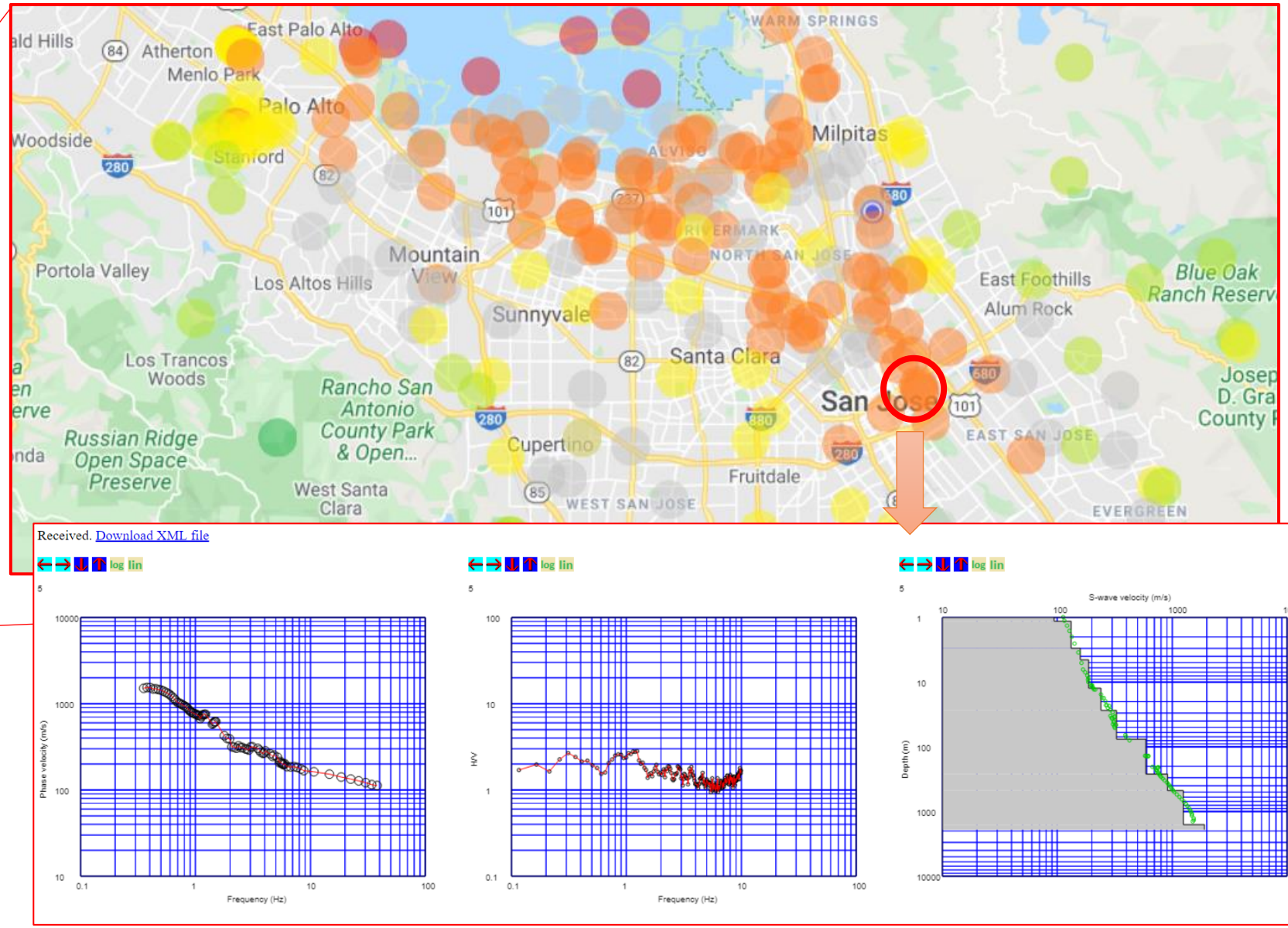


Example of 3D investigation

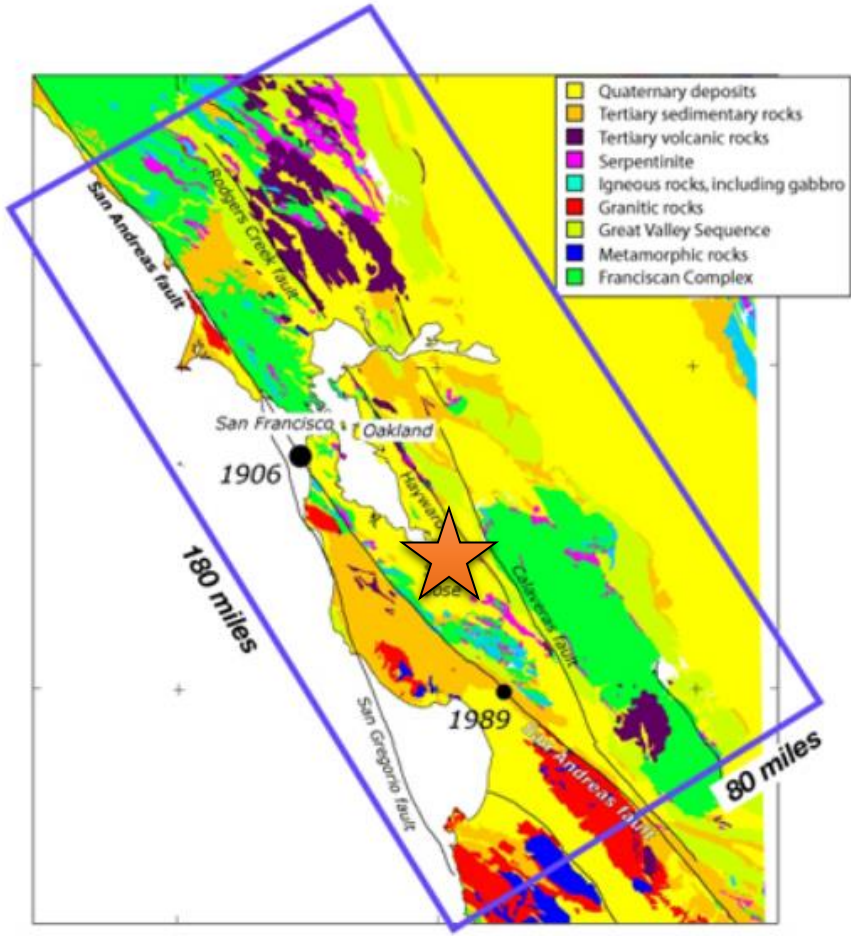


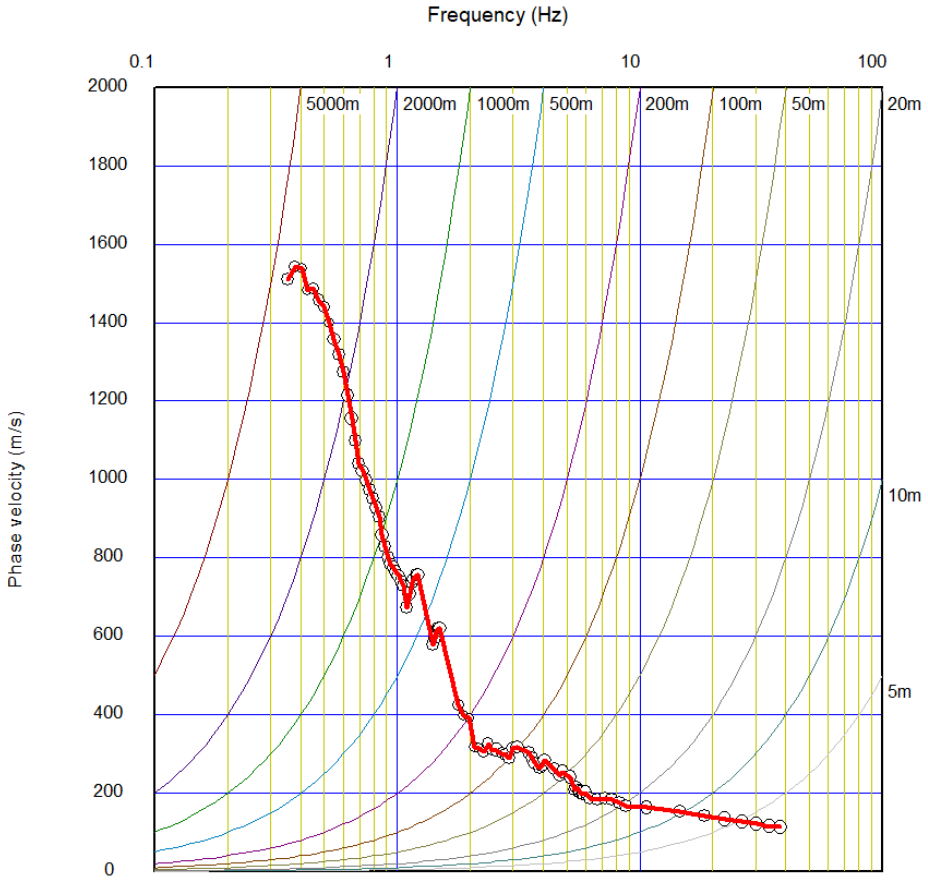
Database of investigation results (SeisImager.com)

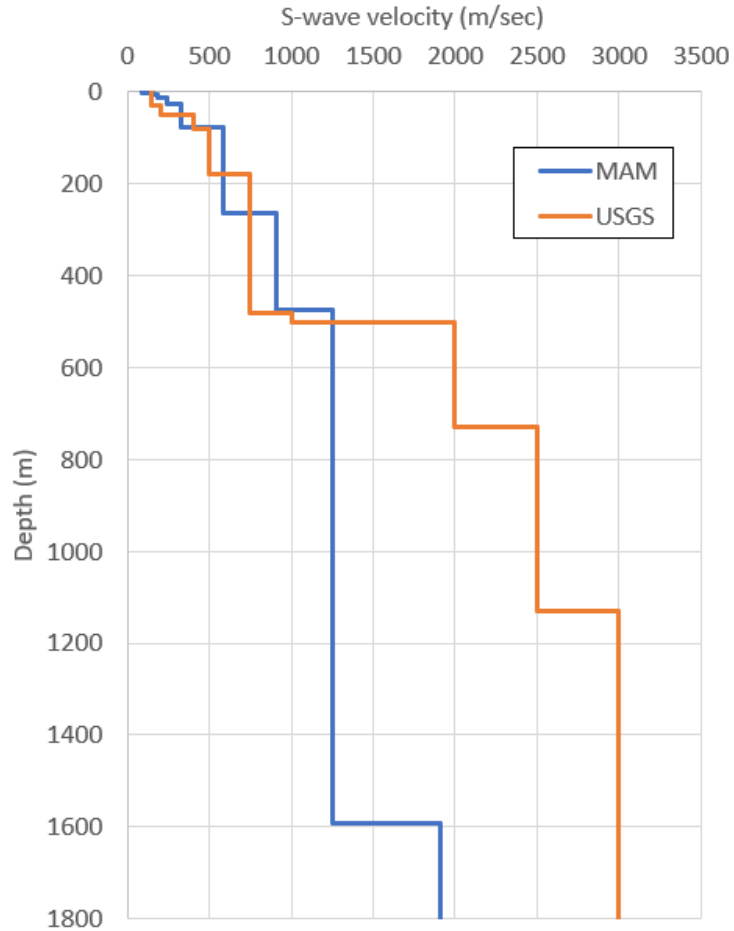




Database of 3D geological/geophysical model (SeisImager.com)







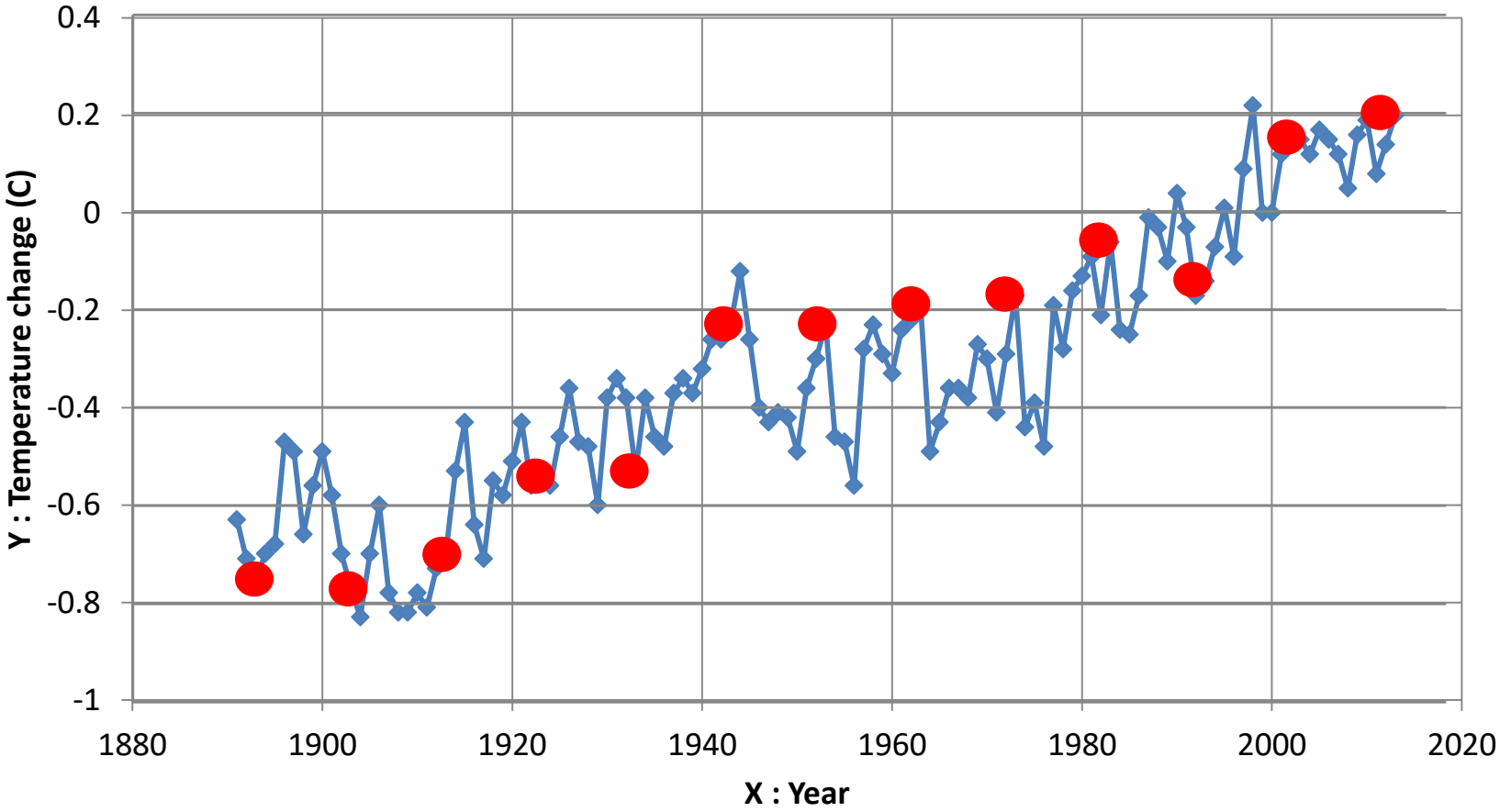
Appendices

- Non-linear least square method example
- Neural network

Appendix 1:
Non-linear least square method example

- Global warming

Example 4 : Global warming



Try to represent by a straight line and a trigonometric function.

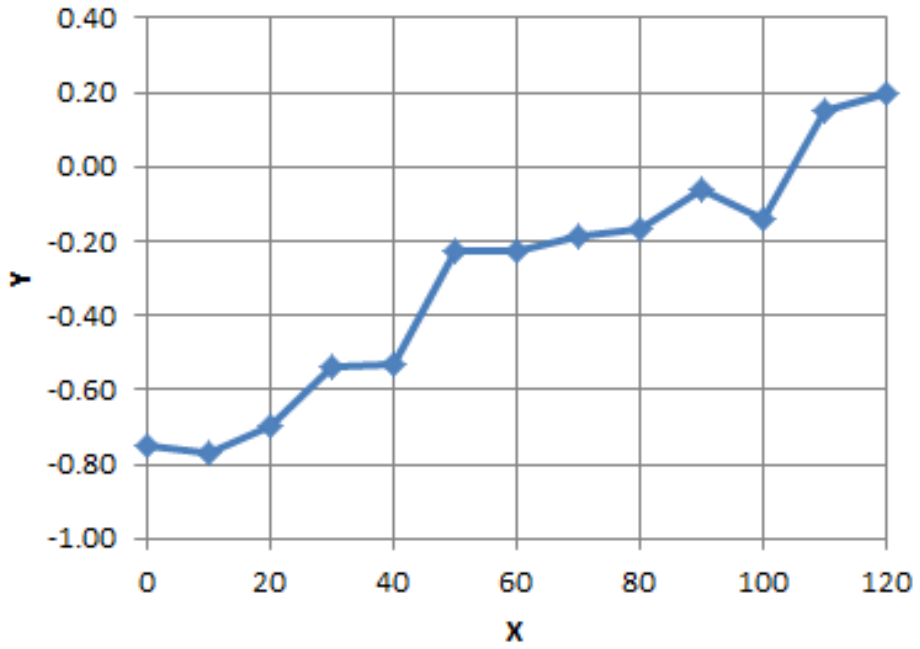
$$Y=aX+b+e*\cos(cX+d)$$

Example 4 : Global warming

Model

$$Y=aX+b+e*\cos(cX+d)$$

13 observed data



X	Y
0	-0.75
10	-0.77
20	-0.70
30	-0.54
40	-0.53
50	-0.23
60	-0.23
70	-0.19
80	-0.17
90	-0.06
100	-0.14
110	0.15
120	0.20

Model

$$Y=aX+b+e*\cos(cX+d)$$

Partial derivatives

$$\frac{\partial Y}{\partial a} = X$$

$$\frac{\partial Y}{\partial b} = 1$$

$$\frac{\partial Y}{\partial c} = -eX\sin(cX+d)$$

$$\frac{\partial Y}{\partial d} = -e\sin(cX+d)$$

$$\frac{\partial Y}{\partial e} = \cos(cX+d)$$

Initial model

$$M_0 = \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} 0.01 \\ -1 \\ 0.1 \\ 1 \\ 0.1 \end{pmatrix}$$

Jacobian matrix A

Non-linear Least Squares Method.xlsx

$$A_0 = \begin{pmatrix} \frac{\partial Y_1}{\partial a} & \frac{\partial Y_1}{\partial b} & \frac{\partial Y_1}{\partial c} & \frac{\partial Y_1}{\partial d} & \frac{\partial Y_1}{\partial e} \\ \frac{\partial Y_2}{\partial a} & \frac{\partial Y_2}{\partial b} & \frac{\partial Y_2}{\partial c} & \frac{\partial Y_2}{\partial d} & \frac{\partial Y_2}{\partial e} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{\partial Y_{13}}{\partial a} & \frac{\partial Y_{13}}{\partial b} & \frac{\partial Y_{13}}{\partial c} & \frac{\partial Y_{13}}{\partial d} & \frac{\partial Y_{13}}{\partial e} \end{pmatrix} = \begin{pmatrix} X_1 & 1 & -eX_1\sin(cX_1+d) & -e\sin(cX_1+d) & \cos(cX_1+d) \\ X_2 & 1 & -eX_2\sin(cX_2+d) & -e\sin(cX_2+d) & \cos(cX_2+d) \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ X_{13} & 1 & -eX_{13}\sin(cX_{13}+d) & -e\sin(cX_{13}+d) & \cos(cX_{13}+d) \end{pmatrix}$$

=

0	1	0	-0.08415	0.540302
10	1	-0.9093	-0.09093	-0.41615
20	1	-0.28224	-0.01411	-0.98999
30	1	2.270407	0.07568	-0.65364
40	1	3.835697	0.095892	0.283662
50	1	1.397077	0.027942	0.96017
60	1	-3.94192	-0.0657	0.753902
70	1	-6.92551	-0.09894	-0.1455
80	1	-3.29695	-0.04121	-0.91113
90	1	4.89619	0.054402	-0.83907
100	1	9.999902	0.099999	0.004426
110	1	5.902302	0.053657	0.843854
120	1	-5.042	-0.04202	0.907447

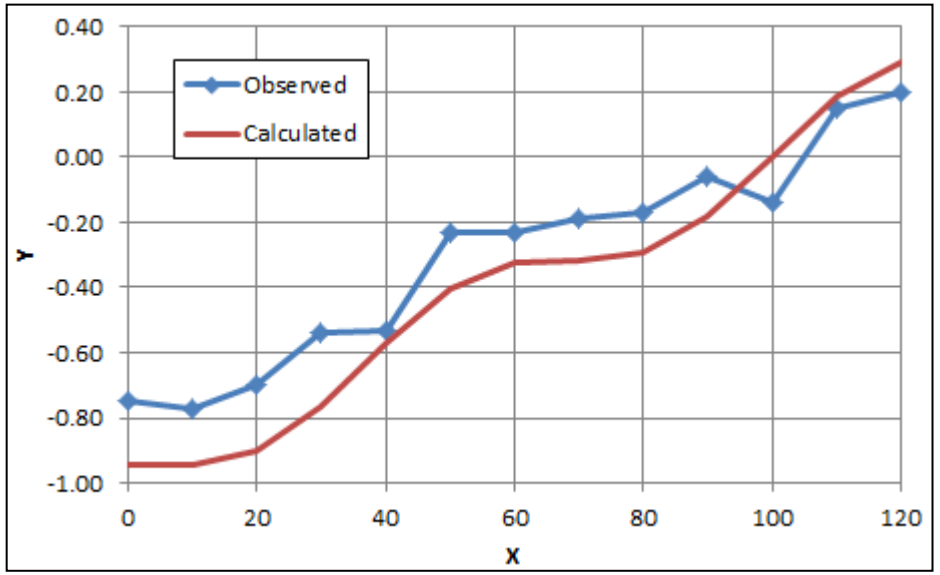
Observation (Y^{obs})

Theoretical value from an initial model (Y_0^{cal})

Error vector(ΔY_0)

$$\Delta Y = Y_{obs} - Y_0$$

X	Yobs	Ycal	Error
0	-0.75	-0.94597	0.19596977
10	-0.77	-0.941615	0.17161468
20	-0.70	-0.898999	0.19899925
30	-0.54	-0.765364	0.22536436
40	-0.53	-0.571634	0.04163378
50	-0.23	-0.403983	0.17398297
60	-0.23	-0.32461	0.09460977
70	-0.19	-0.31455	0.12455
80	-0.17	-0.291113	0.12111303
90	-0.06	-0.183907	0.12390715
100	-0.14	0.0004426	-0.1404426
110	0.15	0.1843854	-0.0343854
120	0.20	0.2907447	-0.0907447



RMSE（Root Mean Square Error）

$$RMSE_0 = \sqrt{\frac{\Delta Y_0^T \Delta Y_0}{13}} = 0.145146$$

$$A_0^T A_0 = \begin{bmatrix} 65000 & 780 & 776.4597 & 7.903659 & 104.5868 \\ 780 & 13 & 7.903659 & -0.02948 & 0.338279 \\ 776.459666 & 7.903659 & 281.3266 & 3.540267 & -1.01562 \\ 7.90365915 & -0.02948 & 3.540267 & 0.064659 & -0.02473 \\ 104.586768 & 0.338279 & -1.01562 & -0.02473 & 6.534138 \end{bmatrix} \qquad A_0^T \Delta Y_0 = \begin{bmatrix} 29.34132318 \\ 1.206172131 \\ -1.47575951 \\ -0.03785912 \\ -0.40408251 \end{bmatrix}$$

Solve $\left(A_0^T A_0\right)\Delta M_0 = A_0^T \Delta Y_0$ $\qquad \Delta M_0 = \begin{bmatrix} -0.00218086 \\ 0.231168514 \\ -0.00960364 \\ 0.297261754 \\ -0.03926993 \end{bmatrix}$

Updated estimation X_1 for X, $M_1 = M_0 + \Delta M$ yields

$$M_1 = \begin{bmatrix} 0.01 \\ -1 \\ 0.1 \\ 1 \\ 0.1 \end{bmatrix} + \begin{bmatrix} -0.00218086 \\ 0.231168514 \\ -0.00960364 \\ 0.297261754 \\ -0.03926993 \end{bmatrix} = \begin{bmatrix} 0.007819 \\ -0.76883 \\ 0.090396 \\ 1.297262 \\ 0.06073 \end{bmatrix}$$

Calculate new error (RMSE) using new estimation X_1 for X

$$RMSE_1 = \sqrt{\frac{\Delta Y_1^T \Delta Y_1}{13}} = 0.056088$$

Second calculation

$$A_1^T A_1 = \begin{bmatrix} 65000 & 780 & 7.903659 & 79.03659 & 104.5868 \\ 780 & 13 & -0.02948 & -0.29479 & 0.338279 \\ 7.90365915 & -0.02948 & 0.064659 & 0.646586 & -0.02473 \\ 79.0365915 & -0.29479 & 0.646586 & 6.465862 & -0.24732 \\ 104.586768 & 0.338279 & -0.02473 & -0.24732 & 6.534138 \end{bmatrix}$$

$$A_1^T \Delta Y_1 = \begin{bmatrix} 0.4790406 \\ -0.01422331 \\ -0.00674581 \\ -0.06745808 \\ 0.046622923 \end{bmatrix}$$

Modification

Updated model

$$\Delta M_1 = \begin{bmatrix} 0.000141417 \\ -0.00277015 \\ -0.00836384 \\ 0.228335161 \\ 0.018218837 \end{bmatrix} \quad M_2 = \begin{bmatrix} 0.007819 \\ -0.76883 \\ 0.090396 \\ 1.297262 \\ 0.06073 \end{bmatrix} + \begin{bmatrix} 0.000141417 \\ -0.00277015 \\ -0.00836384 \\ 0.228335161 \\ 0.018218837 \end{bmatrix} = \begin{bmatrix} 0.007961 \\ -0.7716 \\ 0.082033 \\ 1.525597 \\ 0.078949 \end{bmatrix}$$

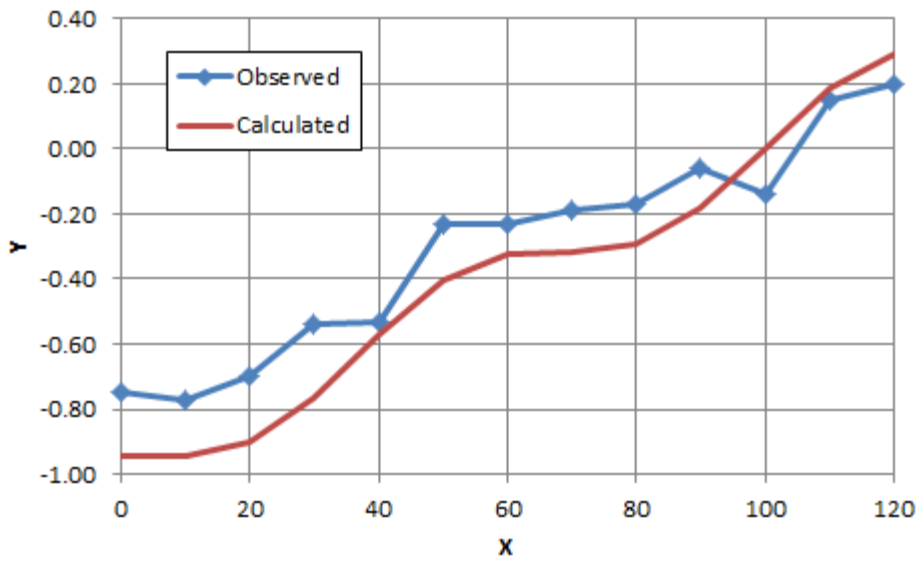
New error is

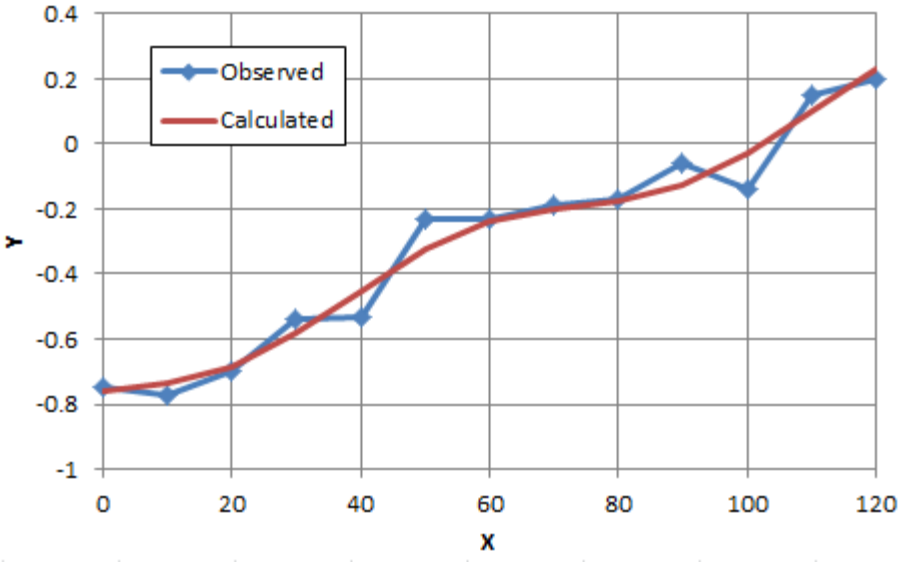
$$RMSE_2 = \sqrt{\frac{\Delta Y_2^T \Delta Y_2}{13}} = 0.052662$$

Example 3 : Global warming

$$Y=aX+b+e*\cos(cX+d)$$

$$Y=0.0079X-0.77+0.079*\cos(0.082X+1.5)$$





Solve global warming(example 4) by python

Non-linear Least Squares Method.ipynb

```
mport numpy as np
import math

XT=np.array([0,10,20,30,40,50,60,70,80,90,100,110,120]) # Year
YoT=np.array([-0.75,-0.77,-0.70,-0.54,-0.53,-0.23,-0.23,-0.19,-0.17,-0.06,-0.14,0.15,0.20]) #
Observation (temperature)

X=np.transpose(XT)
Yo=np.transpose(YoT)

nData=13

A=np.ones((nData,5))

MT=np.array([0.01,-1,0.1,1,0.1]) # Initial model
M=np.transpose(MT)  # initial model

Yc=np.ones((nData))

nIteration=3
```

Solve global warming(example 3) by python

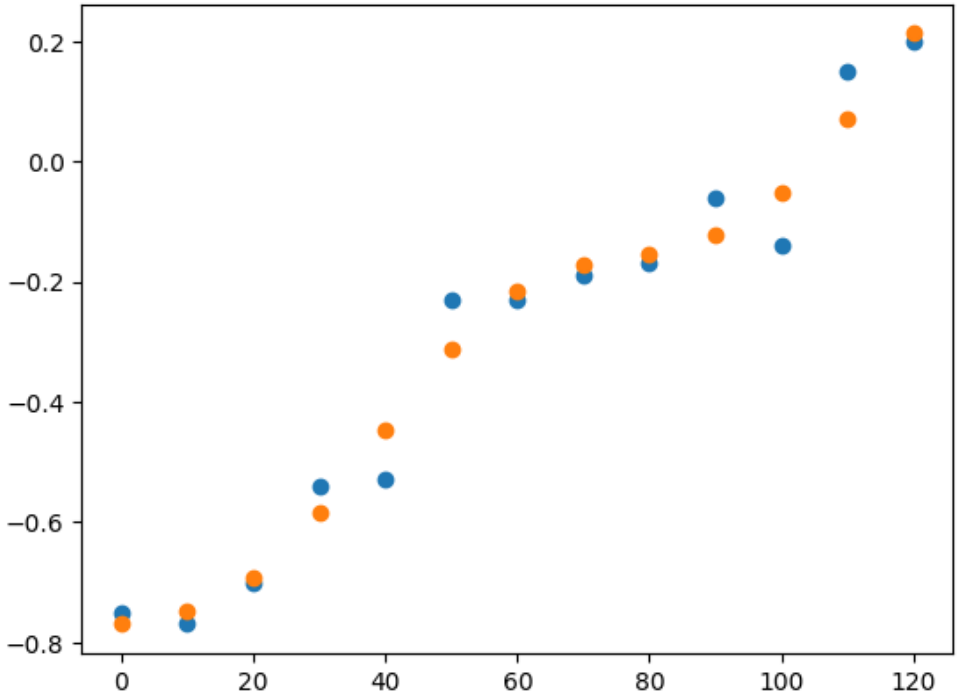
```
for i in range (nIteration):
  for i in range (nData):
    A[i,0]=X[i]
    A[i,2]=-M[4]*X[i]*math.sin(M[2]*X[i]+M[3])
    A[i,3]=-M[4]*math.sin(M[2]*X[i]+M[3])
    A[i,4]=math.cos(M[2]*X[i]+M[3])

  for i in range (nData):
    Yc[i]=M[0]*X[i]+M[1]+M[4]*math.cos(M[2]*X[i]+M[3])

  dY=Yo-Yc

  dYT=np.transpose(dY)
  ms=np.matmul(dYT,dY)/nData
  rms=math.sqrt(ms)
  print (rms)

  AT=np.transpose(A)
  ATA=np.matmul(AT,A)
  ATdY=np.matmul(AT,dY)
  ATAI=np.linalg.inv(ATA)
  dM=np.matmul(ATAI,ATdY)

  M+=dM

  print(M)
```

Solve global warming(example 3) by python

Results

```
year=[]
obs=[]
cal=[]

for i in range (nData):
  year.append(X[i])
  obs.append(Yo[i])
  cal.append(Yc[i])

import matplotlib.pyplot as plt

# 散布図を描画
plt.scatter(year, obs)
plt.scatter(year, cal)
```

```
0.1451458473978012
 [ 0.00781914 -0.76883149 0.09039636 1.29726175 0.06073007]
0.05608817320861406
[ 0.00796056 -0.77160164 0.08203252 1.52559691 0.07894891]
0.05266210438680151
 [ 0.0079748 -0.7717075 0.08392778 1.46748093 0.08294966]
```



Appendix 2:
Neural network

- How neural network works?
- MNIST
- Traveltime tomography

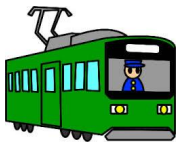
How neural network works?

M is model and D is data.
Conventional linear or non-linear inversion using Jacobian matrix G.

$$GM=D \qquad G^{-1}D=M$$

Random search (when unable to calculate G^{-1}).

$$GM=D$$



Neural network (no Jacobian matrix).

Hidden layer

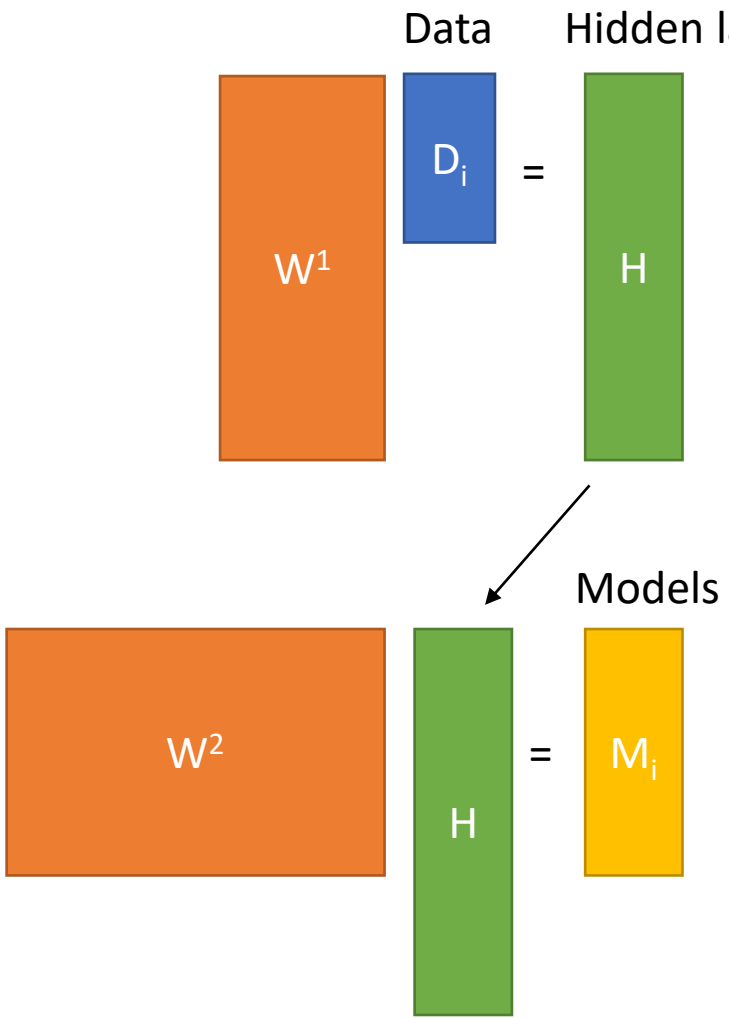
$$WD=M$$

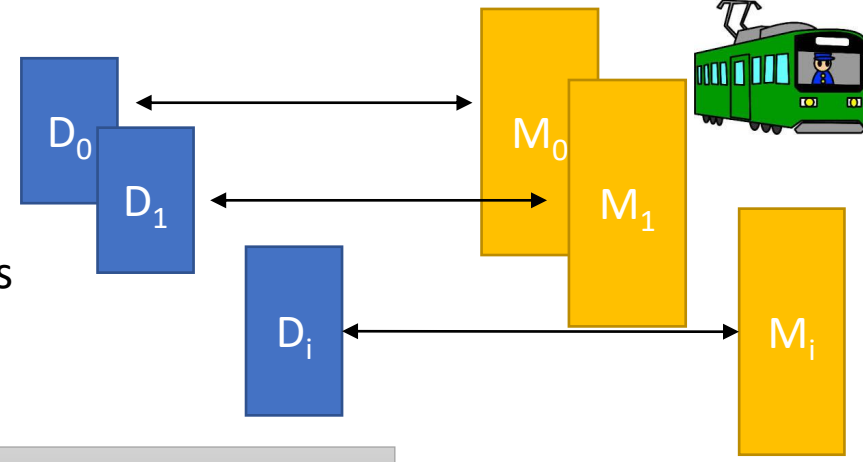
$$W^1D=H$$

$$W^2H=M$$

1. Training :
   Calculate W^1 and W^2 using existed data pairs of D and M.
2. Prediction :
   Predict M from D using the trained wight W^1 and W^2.

Neural network : training





Data- model pairs

Randomly generate W^1 and W^2 for initial guess.

Learning

Calculate M_{ci} and compare with true value M_{oi}.

Modify W^1 and W^2 for next guess.

Difference between M_{ci} and M_{oi} is enough small ?

No

Yes

Training completed.

Non-linear least square methods and neural network

D : Data
G : System (Jacobian matrix)
M : Model

$$D_{obs} = GM \qquad M = G^{-1}D_{obs}$$

Conventional non-linear least square method
(try to estimate model)

$$D_{obs} \neq D_{cal0} = GM_0$$

Initial model

$$D_{obs} \neq D_{cal1} = GM_1$$

$$D_{obs} \neq D_{cal2} = GM_2$$

.
.
.
.
.

Change model M until
D_{cal} becomes equal to D_{obs}

Inversion

$$D_{obs} = D_{caln} = GM_n$$

Need to know system G

Neural network (try to estimate system)

$$M_{true\ a} \neq M_{cal\ a} = G_0^{-1} D_{obs\ a}$$

Initial system

$$M_{true\ b} \neq M_{cal\ b} = G_1^{-1} D_{obs\ b}$$

$$M_{true\ c} \neq M_{cal\ c} = G_2^{-1} D_{obs\ c}$$

.
.
.
.
.

Change System G until
M_{cal} becomes equal to M_{true}

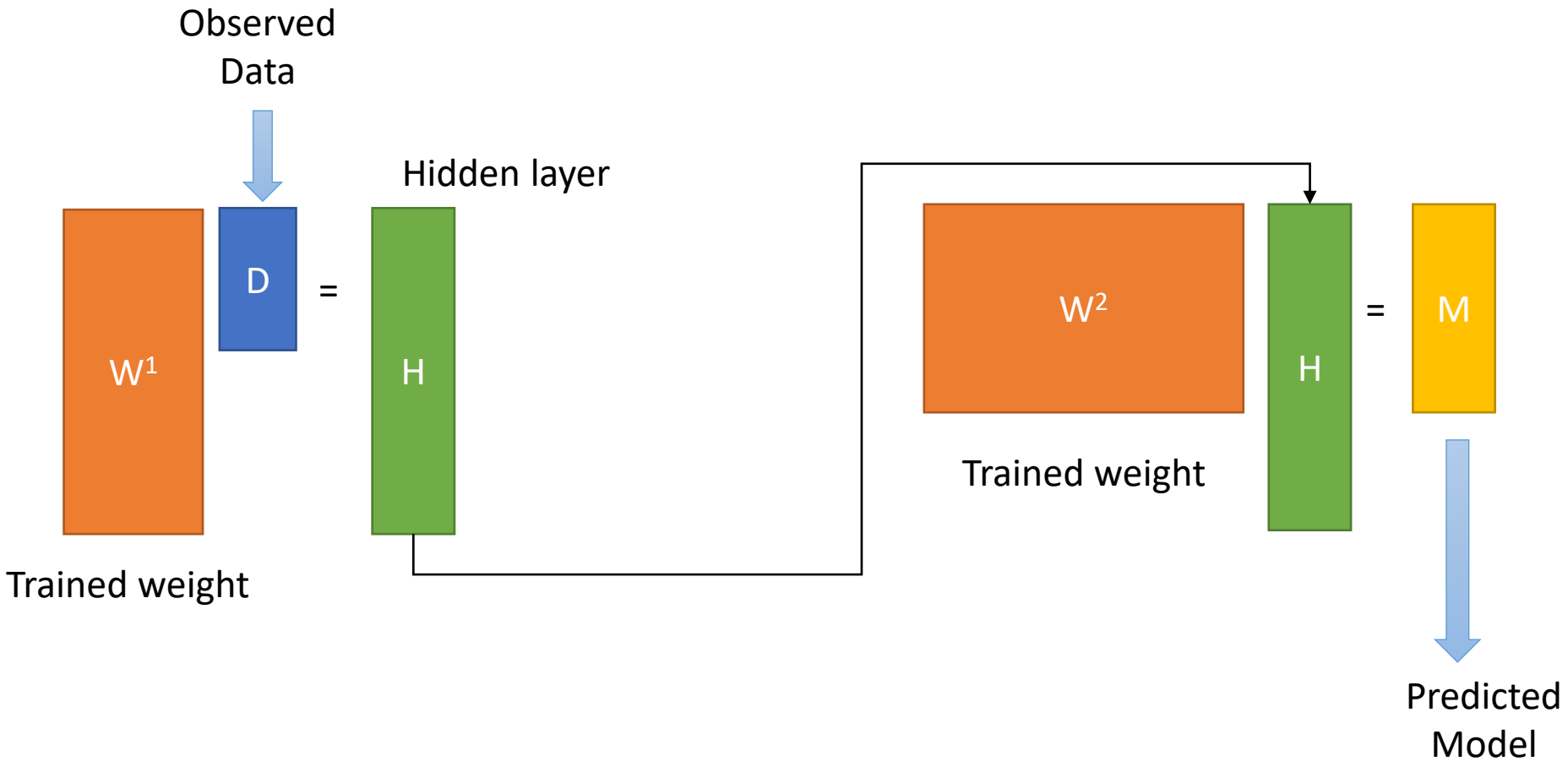
Training

$$M_{true\ n} = M_{cal\ n} = G_n^{-1} D_{obs\ n}$$

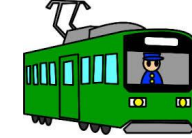
$$M_{guess} = G_n^{-1} D_{obs}$$

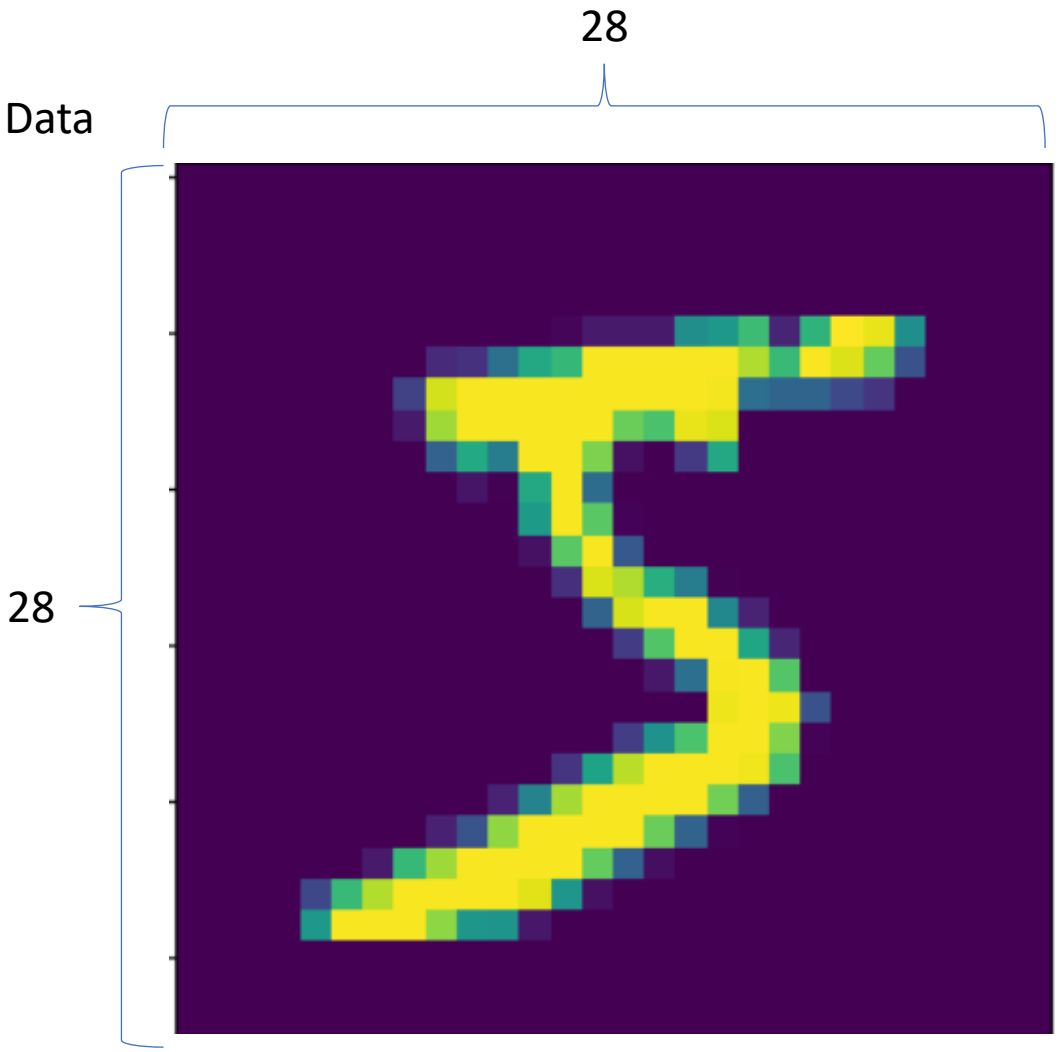
Need to prepare a lot of M_{true}-D_{obs} pairs

Neural network : prediction

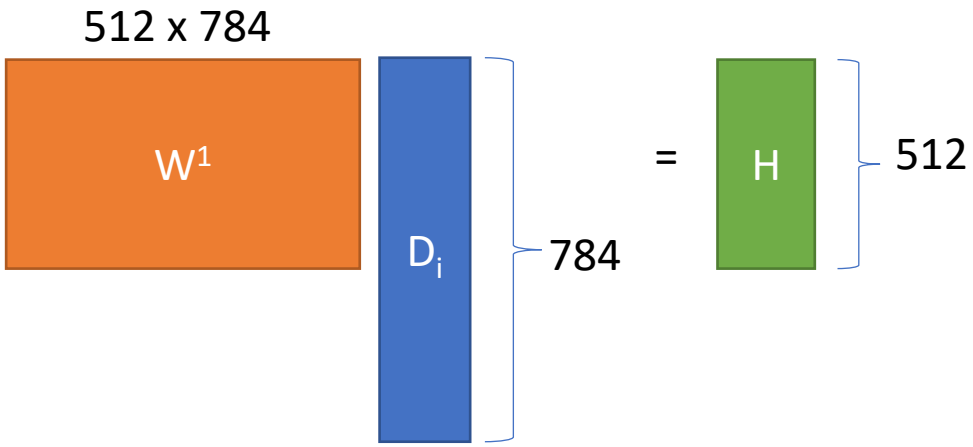


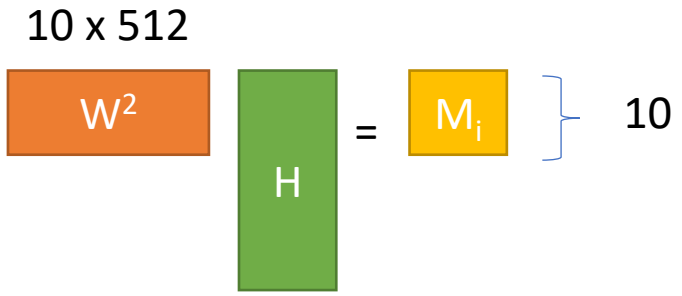
Example 4 : MNIST





28x28=784





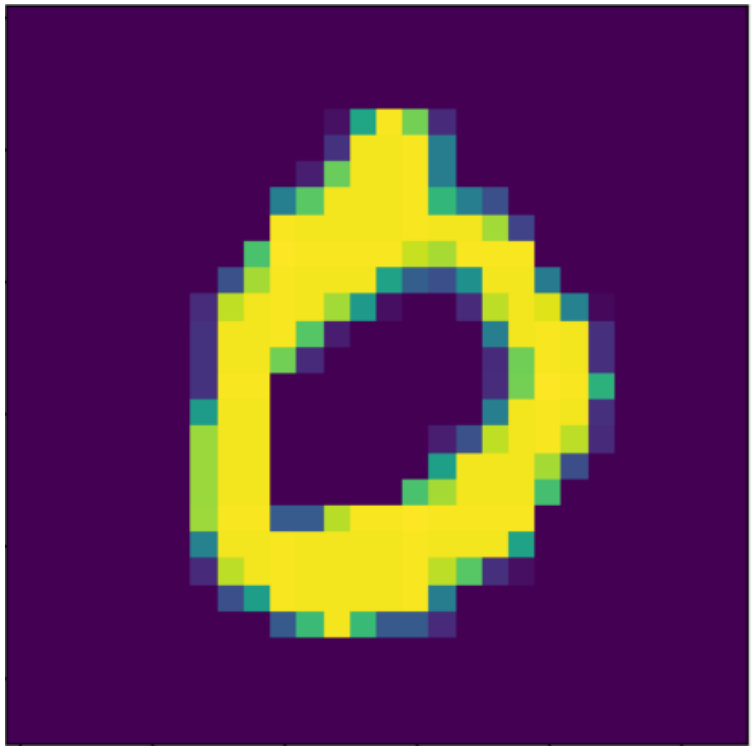
Model

M_i

0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0

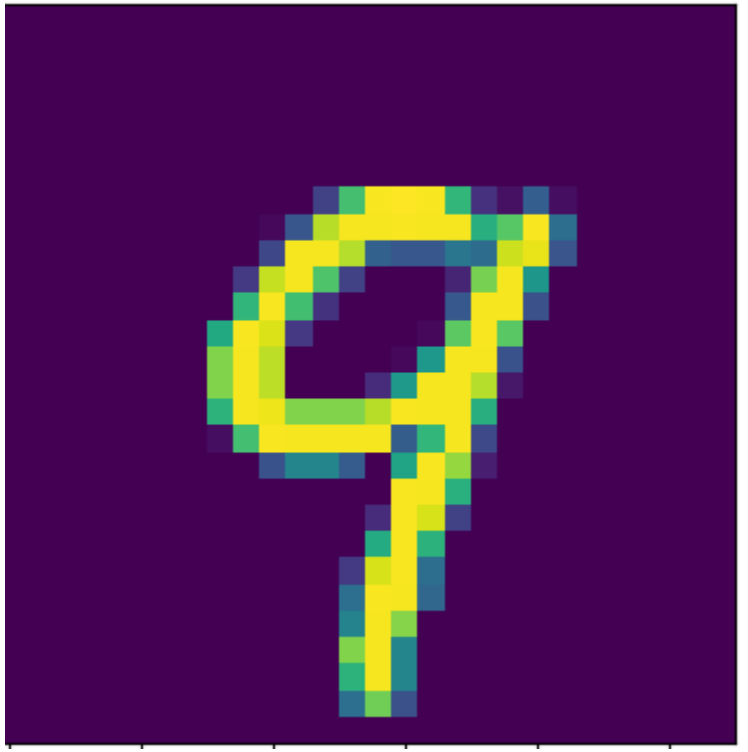
Example 1 : MNIST

0



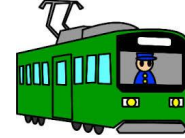
0	0.99999994
1	0.0
2	5.04571e-35
3	0.0
4	0.0
5	0.0
6	0.0
7	0.0
8	8.43885e-38
9	0.0

9



0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
5	0.0
6	0.0
7	0.0
8	3.90426e-37
9	0.99999994

Solve MNIST (example 4) by python



mnist keras.ipynb

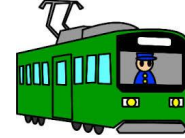
Import training and test pairs

```
from tensorflow.keras.datasets import mnist
(train_images,train_labels),(test_images,test_labels)=mnist.load_data()
train_images.shape
```

➡ Results

(60000, 28, 28)

Solve MNIST (example 4) by python

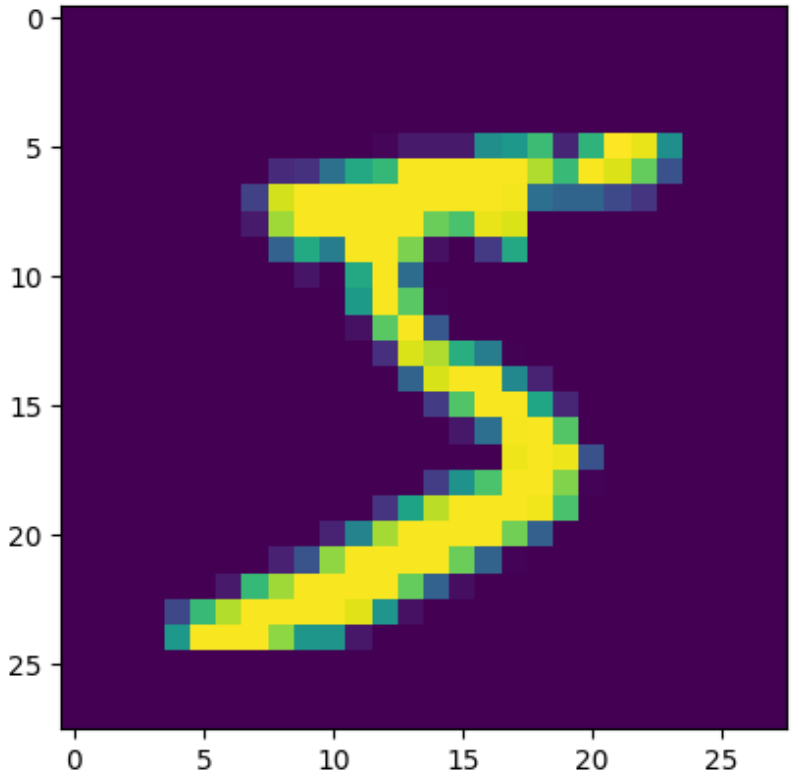


Reshape training and test pairs

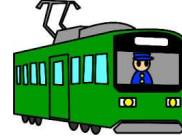
```
import numpy as np
import matplotlib.pyplot as plt

train_images_vector=train_images.reshape((60000,28*28))
train_images=train_images.astype("float32")/255
test_images_vector=test_images.reshape((10000,28*28))
test_images=test_images.astype("float32")/255

print(train_labels[0])

plt.imshow(train_images[0])
```

Results

5



Solve MNIST (example 4) by python

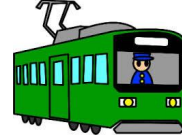


```
from tensorflow import keras
from tensorflow.keras import layers

model=keras.Sequential([
    layers.Dense(512,activation="relu"),
    layers.Dense(10,activation="softmax")])

model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

model.fit(train_images,train_labels,epochs=5,batch_size=128)
```

Results

```
Epoch 1/5
469/469 [==============================] - 11s 21ms/step - loss:
0.2634 - accuracy: 0.9245
Epoch 2/5
469/469 [==============================] - 6s 13ms/step - loss:
0.1064 - accuracy: 0.9685
Epoch 3/5
469/469 [==============================] - 8s 17ms/step - loss:
0.0706 - accuracy: 0.9790
Epoch 4/5
469/469 [==============================] - 9s 20ms/step - loss:
0.0512 - accuracy: 0.9844
Epoch 5/5
469/469 [==============================] - 5s 11ms/step - loss:
0.0380 - accuracy: 0.9887
```

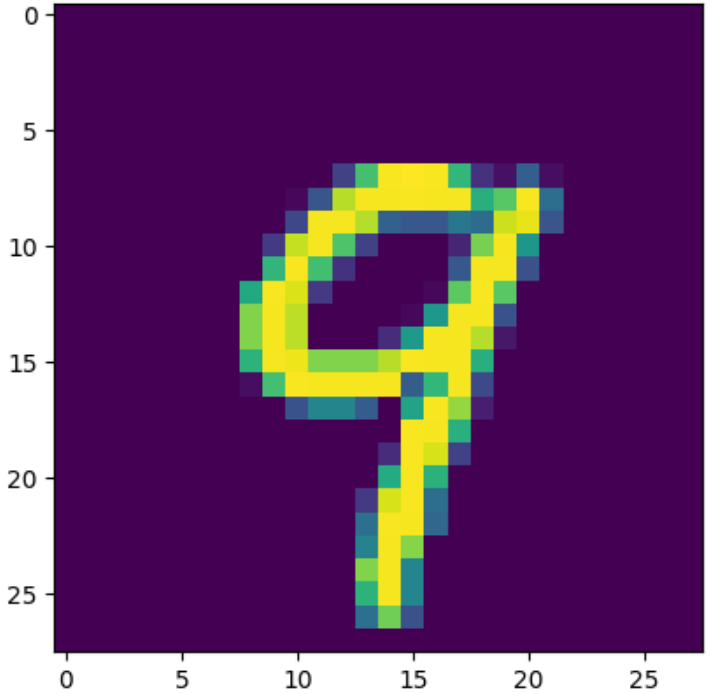
Solve MNIST (example 4) by python



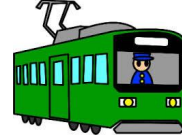
```
import numpy as np
import matplotlib.pyplot as plt

index=12

test_digits=test_images_vector[0:100]
predictions=model.predict(test_digits)
print(test_labels[index])
print(predictions[index])

plt.imshow(test_images[index])
```

Results

```
9
[0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
3.9042601e-37 9.9999994e-01]
```

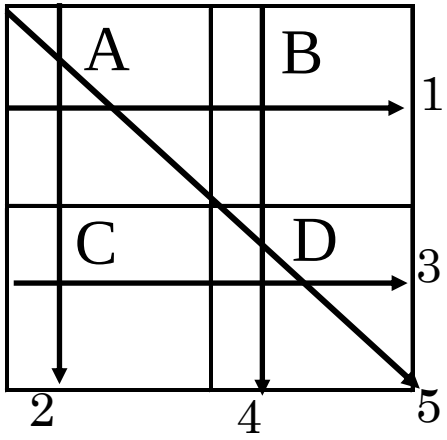


Example 2 : Traveltime tomography



Observed traveltime

5 ray-path (data)

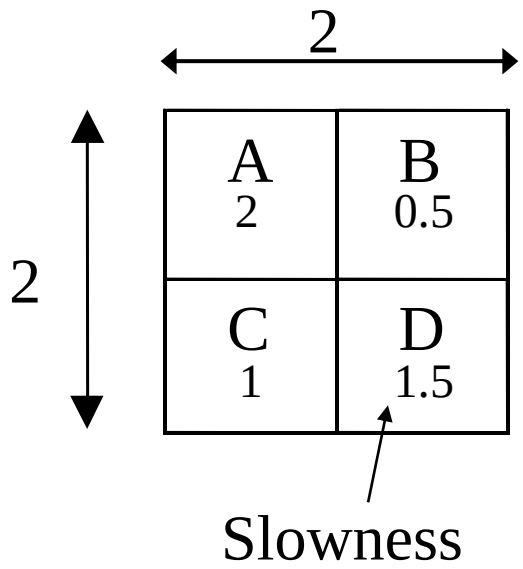


Jacobian matrix A

(Ray length passing through each cell)

$$L = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix}$$

4 cells (targets)

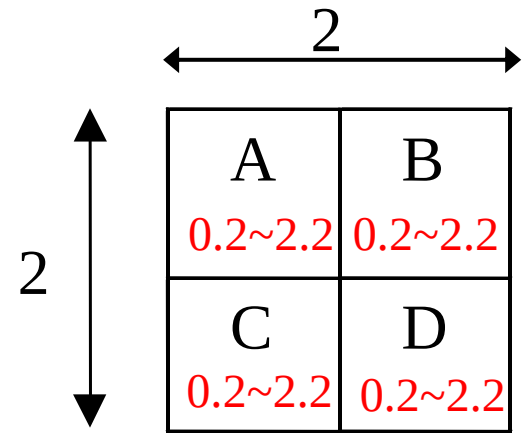


$$T = \begin{pmatrix} 2.5 \\ 3 \\ 2.5 \\ 2 \\ 4.949747 \end{pmatrix}$$

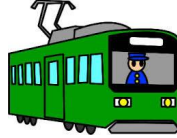
Target (slowness)

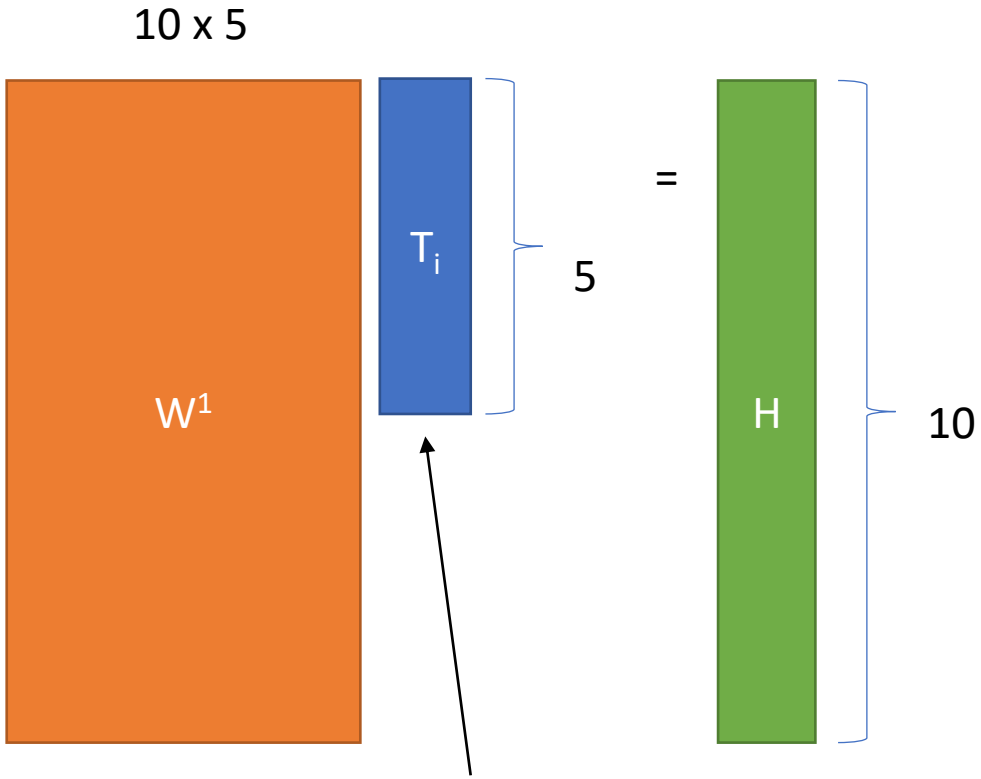
$$S = \begin{pmatrix} 2 \\ 0.5 \\ 1 \\ 1.5 \end{pmatrix}$$

Randomly change slowness and prepare traveltime-slowness pairs for training



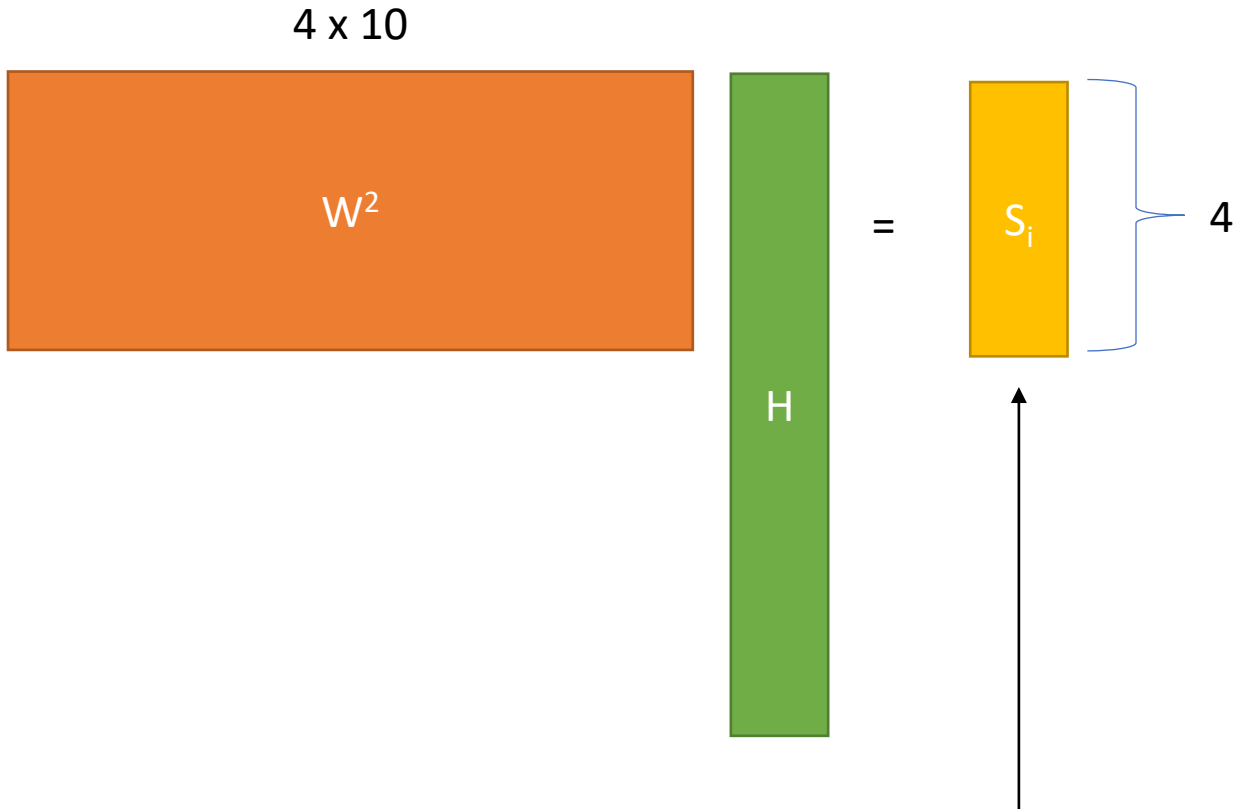
Example 2 : Traveltime tomography





$$T=\begin{pmatrix}2.5\\3\\2.5\\2\\4.949747\end{pmatrix}$$

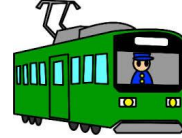
2.5
3
2.5
2
4.949747



$$S=\begin{pmatrix}2\\0.5\\1\\1.5\end{pmatrix}$$

2
0.5
1
1.5

Example 2 : Traveltime tomography



Generate training pairs

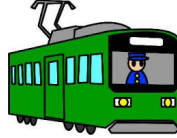
```
import numpy as np

slowness=np.random.rand(4,100)
slowness*=2
slowness+=0.2 # between 0.2 and 2.2

raypath=np.array([[1,1,0,0],[1,0,1,0],[0,0,1,1],[0,1,0,1],[np.sqrt(2),0,0,np.sqrt(2)]])

traveltime=[[]]

traveltime=np.matmul(raypath,slowness)

train_data=np.transpose(traveltime)
train_targets=np.transpose(slowness)

print(train_data.shape)
print(train_targets.shape)

print(train_data)
print(train_targets)
```

Results

Tomography keras.ipynb

```
(100, 5)
(100, 4)
[[2.91388276 2.18675355 1.5615076  2.28863681 3.65152196]
 [3.2968924  2.86442028 2.93388528 3.36635741 5.28811038]
 [2.23434069 2.31363787 1.11020268 1.03090551 3.26564963]
 [2.70686542 2.57213827 2.48419541 2.61892257 6.03137111]
 .
 .
 [2.4655604  3.48459768 2.92579835 1.90676108 4.41433625]
 [3.15218277 2.61675686 3.13720562 3.67263153 4.94473896]
 [2.56232353 2.94505932 2.9115759  2.52884012 2.72956877]
 [1.99732166 1.54923849 1.39849323 1.84657639 2.26682469]]
[[1.60363094 1.31025181 0.58312261 0.97838499]
 [1.83489685 1.46199555 1.02952343 1.90436185]
 [1.75629909 0.4780416  0.55733877 0.55286391]
 [2.17638314 0.53048229 0.39575513 2.08844028]
 [1.93545362 2.09593115 1.8177563  1.76106714]
 [1.06271321 0.83994294 1.61987338 0.63068731]
 .
 .
 [0.21176484 1.87613569 1.26569859 1.97418329]
 [0.75673625 0.96122831 1.65799706 0.39291112]
 [0.53243576 1.9057105  1.84557216 0.2680977 ]
 [1.05724176 0.64528736 0.2966809  0.20785675]
 [1.84010321 0.62545719 1.64449447 1.28130388]
 [1.48800484 1.66417793 1.12875202 2.00845361]
 [0.98179    1.58053353 1.96326932 0.94830658]
 [0.87681619 1.12050547 0.6724223  0.72607092]]
```

Example 2 : Traveltime tomography



Calculate mean and SD

```
data_mean=train_data.mean(axis=0)
train_data-=data_mean
data_std=train_data.std(axis=0)
train_data/=data_std

print(data_mean,data_std)

targets_mean=train_targets.mean(axis=0)
train_targets-=targets_mean
targets_std=train_targets.std(axis=0)
train_targets/=targets_std

print(targets_mean,targets_std)
```

Results

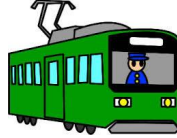
Data

Mean [2.36243515 2.44565284 2.28746888 2.20425118 3.31138003]
SD [0.8622478 0.78062335 0.84245784 0.77192784 1.21938181]

Model (target)

Mean [1.24984162 1.11259353 1.19581122 1.09165765]
SD [0.5847847 0.57108855 0.5513168 0.59928691]

Example 2 : Traveltime tomography



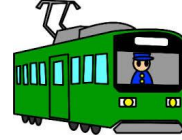
Define neural network model and calculate weights W^1 and W^2

```
from tensorflow import keras
from tensorflow.keras import layers

def build_model():
  model=keras.Sequential([layers.Dense(10,activation="relu"),
                          layers.Dense(4)])

  model.compile(optimizer="rmsprop",loss="mse",metrics="mae")

  return model

model=build_model()

num_epochs=1000

history=model.fit(train_data,train_targets,epochs=num_epochs,verbose=0)

mse,mae=model.evaluate(train_data,train_targets,verbose=0)

print(mse,mae)
```

Results

```
3.289747473900206 4e-05
0.0038549371529370546
```

Example 2 : Traveltime tomography



Predict velocity model from observed traveltime using trained W^1 and W^2

```
import matplotlib.pyplot as plt

test_data=np.array([[2.5,3,2.5,2,4.949747]])
test_data-=data_mean
test_data/=data_std

predictions=model.predict(test_data)
predictions*=targets_std
predictions+=targets_mean

print(predictions)

loss=history.history["loss"]
mae=history.history["mae"]

plt.plot(range(0,len(mae)),mae)
plt.xlabel("epochs")
plt.ylabel("Validation MAE")
plt.show()
```

Observed traveltime

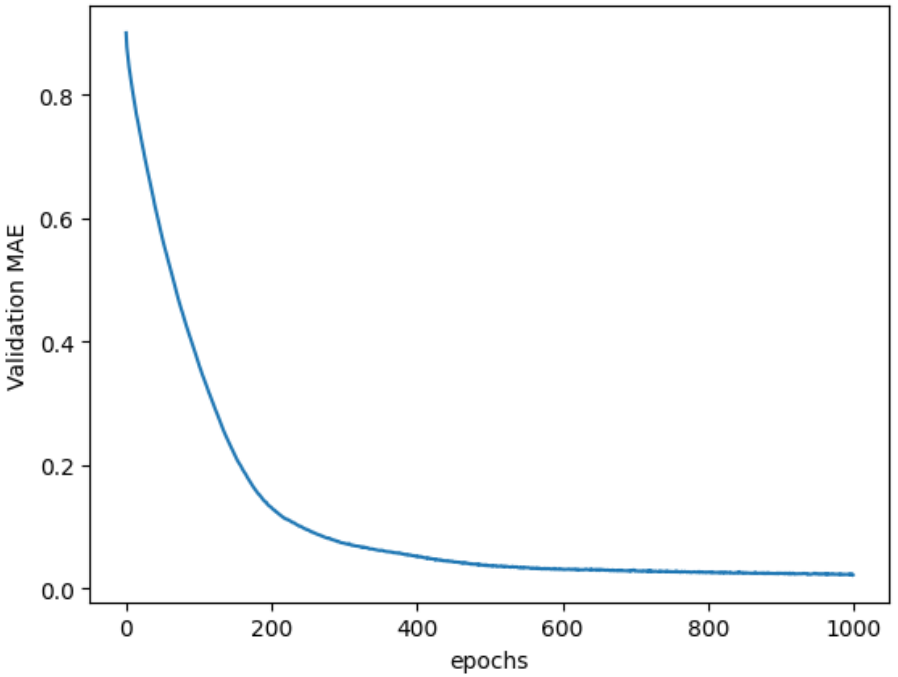
Results

Prediction

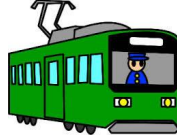
[1.9966546 0.48463768 0.9899436 1.5137079]

Target (slowness)

[2.0 0.5 1.0 1.5]



Example 2 : Traveltime tomography



Retrieve trained weights

```
model.summary()
```

Results

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                60

 dense_1 (Dense)             (None, 4)                 44

=================================================================
Total params: 104 (416.00 Byte)
Trainable params: 104 (416.00 Byte)
Non-trainable params: 0 (0.00 Byte)
```

Number of elements in weight matrices

(5+1)*10=60

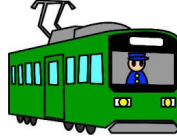
(10+1)*4=44

```
l0 = model.layers[0]
print(l0.get_weights())

l1 = model.layers[1]
print(l1.get_weights())
```

Results

```
_________________________________________________________________
[array([[-0.3829258 , -0.18755202,  0.22799419,  0.4709892 , -0.27799228,
         0.03346295,  0.7313882 , -0.66752845,  0.23819262, -0.10034406],
        [ 0.71042573,  0.09020863, -0.0879085 , -0.6082602 ,  0.19179393,
         -0.07183067,  0.6346195 , -0.63333905, -0.60649025,  0.45061472],
        [-0.2497883 , -0.5445761 ,  0.36182728, -0.04827076, -0.06770287,
         0.05016776, -0.29021642,  0.30371803,  0.10881497, -0.3357046 ],
        [-0.22593565,  0.0277824 ,  0.23321283, -0.38766268, -0.33902004,
         0.09804326,  0.14106004, -0.16256832,  0.41806746,  0.32522824],
        [-0.61730266,  0.05580708, -0.56693864,  0.08645135,  0.61277306,
         -0.1649657 , -0.19356838,  0.18711595,  0.66081595, -0.04122322]],
       dtype=float32), array([-0.2666976 ,  0.9270561 ,  1.0015459 ,  0.9200742 ,  1.0945895 ,
         0.17699714,  0.09031737, -0.10001677,  0.285381  ,  0.16973662],
       dtype=float32)]
[array([[-0.14626415, -0.28690726,  0.47345722, -0.8281264 ],
        [-0.01698786, -0.27996433, -0.31821382, -0.4092804 ],
        [-0.6655223 ,  0.5989425 ,  0.8401682 ,  0.08153339],
        [ 0.1035963 ,  0.38328812, -1.0497649 , -0.3896314 ],
        [ 0.58353484, -0.67936105,  0.2793591 ,  0.3634506 ],
        [-0.17626432, -0.01337478, -0.11364847, -0.1242265 ],
        [ 0.5877383 ,  0.46626562,  0.10157645, -0.41417316],
        [-0.679952  , -0.51919657, -0.15134722,  0.40519232],
        [ 0.13679464,  0.26696086, -0.4404578 ,  0.7690209 ],
        [ 0.31158996,  0.15141848,  0.22909169,  0.13553093]],
       dtype=float32), array([-0.15913346, -0.09428249,  0.21936116,  0.07833888], dtype=float32)]
```

Example 2 : Traveltime tomography



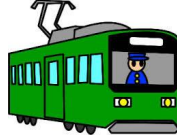
Predict velocity model from traveltime using trained weights by Excel

Keras weight.xlsx

Traveltime	Mean	SD	T
2.5	2.362435	0.862248	0.159542
3	2.445653	0.780623	0.710134
2.5	2.287469	0.842458	0.252275
2	2.204251	0.771928	-0.2646
4.949747	3.31138	1.219382	1.343605

(Traveltime-Mean)/SD=T

Example 2 : Traveltime tomography



Predict velocity model from traveltime using trained weights by Excel

W^1

0.3289117	0.3100661	0.0518733	-0.08238	-0.108733	0.0665823
-0.056284	-0.585552	-0.140331	-0.119161	0.3223119	0.395904
0.2187911	-0.25161	-0.421834	0.0679366	0.0653371	0.5763623
-0.135704	0.561525	-0.402515	0.7623872	-0.483435	0.7636405
0.7555688	-0.759614	0.5183554	0.4231131	0.0521938	-0.160831
-0.015172	0.0928923	0.2038522	-1.092998	-0.050584	0.1636207
0.1699794	0.3239634	0.7140482	-0.245875	-0.337596	0.3338379
-0.337227	-0.18118	0.361335	0.270821	-0.815223	-0.056668
-0.729801	0.237316	-0.250041	0.5297543	0.4563882	0.8693018
0.2539147	0.3432414	-0.568281	-0.191523	0.9769327	0.0694447

T

0.159542
0.710134
0.252275
-0.2646
1.343605
1

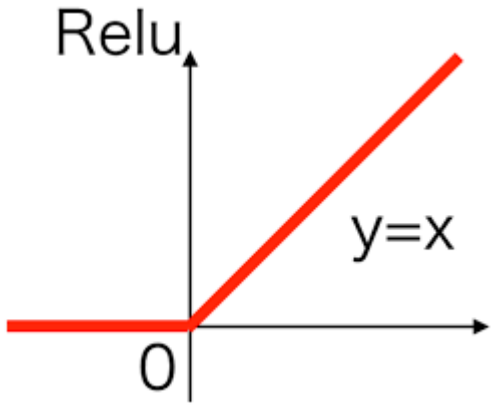
=

H

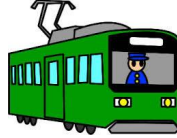
0.2280362
0.4002917
0.395985
0.1879318
-0.490773
0.4998346
0.3826129
-1.314973
1.3313479
1.573627
1

H(Relu)

0.2280362
0.4002917
0.395985
0.1879318
0
0.4998346
0.3826129
0
1.3313479
1.573627
1



Example 2 : Traveltime tomography



Predict velocity model from traveltime using trained weights by Excel

W^2

0.0572287	-0.09517	-0.06794	0.0214456	-0.060815	0.0558894	0.0529566	-1.054866	-0.133641	0.8733705	0.080358
0.1054724	-0.079642	0.086948	0.8782455	0.6016411	-0.590669	-0.019119	0.0187649	-0.701471	-0.039473	0.007341
0.2897521	-0.74344	-0.725053	0.3636039	-0.155156	0.1278185	0.5302092	0.2270992	0.040184	-0.289159	0.2116617
-0.048361	-0.091536	-0.293267	-0.780442	0.6084266	-0.620079	0.149375	-0.00384	0.8495807	0.0026193	0.132298

H(Relu)

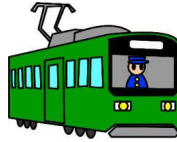
0.2280362
0.4002917
0.395985
0.1879318
0
0.4998346
0.3826129
0
1.3313479
1.573627
1

=

S

1.277073
-1.09958
-0.37341
0.704254

Example 2 : Traveltime tomography



Predict velocity model from traveltime using trained weights by Excel

S	SD	Mean	Slowness
1.277073	0.584785	0.584785	1.996655
-1.09958	0.571089	0.571089	0.484638
-0.37341	0.551317	0.551317	0.989944
0.704254	0.599287	0.599287	1.513708

S*SD+Mean=Slowness

Target (slowness)

$$S=\begin{pmatrix}2\\0.5\\1\\1.5\end{pmatrix}$$