EOSC 454 / 556B Assignment 2: Linear Inverse Problem

DUE:

Name: Parth Pokar
Student #: 43949999

Note: The code used to generate all figures is in the Jupyter notebook attached along with this assignment and also at this Github repo: `https://github.com/pokarparth/EOSC_556_Applied_Geophysics.git`. Snippets of the code from this notebook are included here for some questions that ask for code.

Q1. In this question, we will set up the forward simulation using kernels that are decaying exponential functions. The kernel functions take the form

$$g_j(x) = e^{jpx} \cos(2\pi jqx) \tag{1}$$

and each datum is

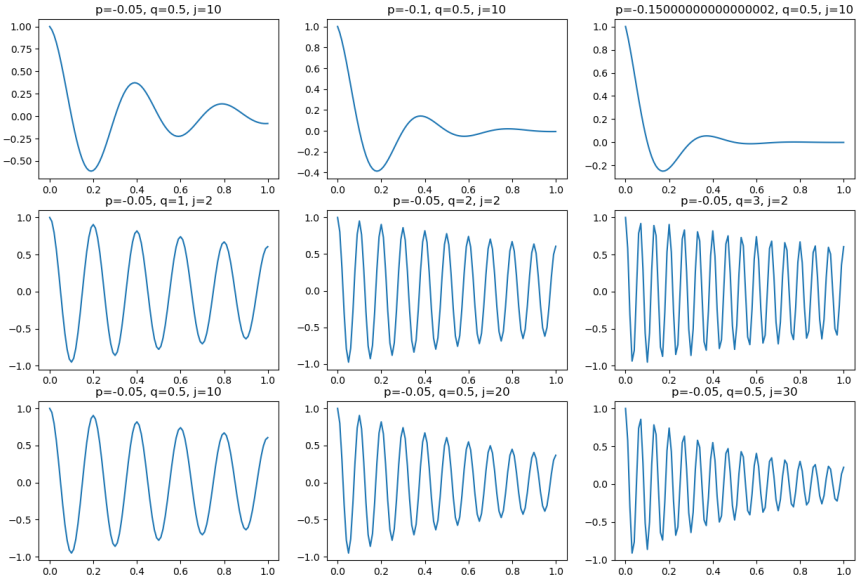
$$d_j = \int_v g_j(x) m(x) dx \tag{2}$$

As we discussed in class, this can be written in discrete form as

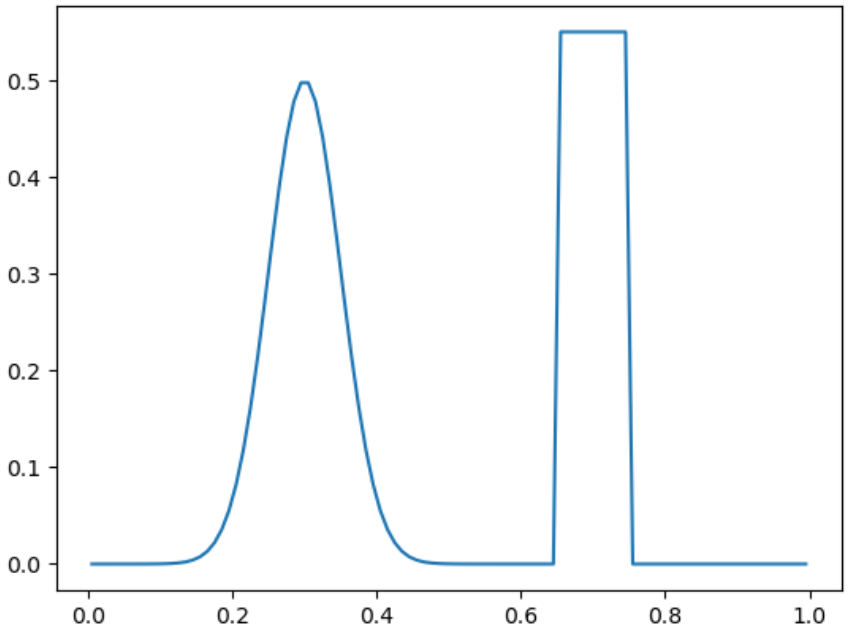
$$\mathbf{d} = \mathbf{Gm} \tag{3}$$

a. Create a function that generates the kernel functions $g(x)$ given scalar values for j, p, q and a vector of locations x. Define a mesh with 100 cells with the domain $0 \leq x \leq 1$. Describe the roles of p, q and j in the nature of the kernel function (plots will be helpful).

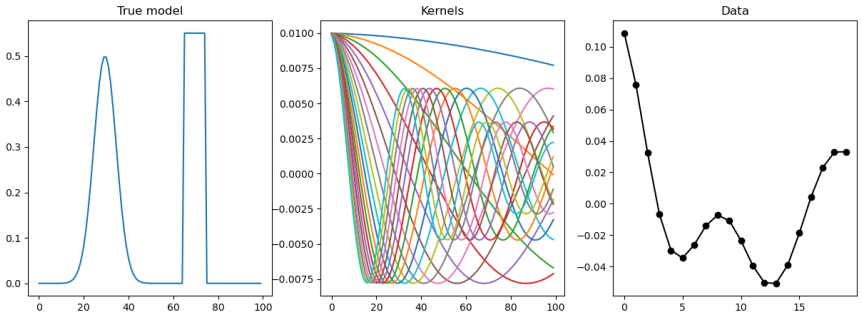
- Parameter p controls the decay rate of the kernel function. As p increases, the decay rate of the kernel function increases. Row 1 in the plot shows the effect of p.
- Parameter q controls the periodicity/frequency of the kernel function. As q increases, the number of oscillations in the kernel function increases. Row 2 in the plot shows the effect of q.
- Parameter j is a scaling factor for both periodicity and decay rate of the kernel function. As j increases, the number of oscillations and decay rate of the kernel function both increase. Row 3 in the plot shows the effect of j.



b. Design a model that consists of a Gaussian and a boxcar and use different parameters than we used in class (e.g. change the location of the boxcar / Gaussian, change the amplitude, make one have a negative amplitude, etc). Plot your model.



c. Next create a function that builds the forward simulation matrix $\mathbf{G}$ and simulate data using 20 kernels with $p = -0.05$, $q = 0.1$, $0 \leq j \leq 30$, with the values for j being evenly spaced. Create a plot with 3 subplots that show the model, kernels and data.



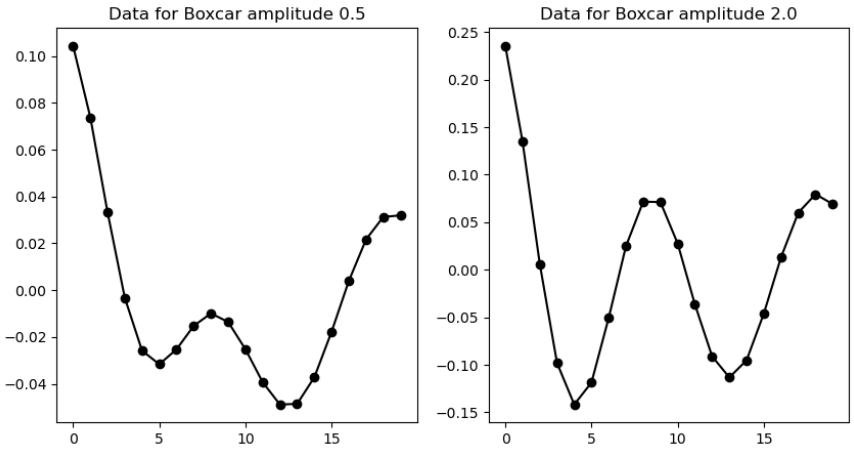
d. In this part, we will examine the concept of sensitivity: how a change in the model impacts the data. For these examples, we can quantify the sensitivity using

$$\frac{\|\mathbf{d}(m_1) = \mathbf{d}(m_2)\|}{||\Delta m||} \tag{4}$$

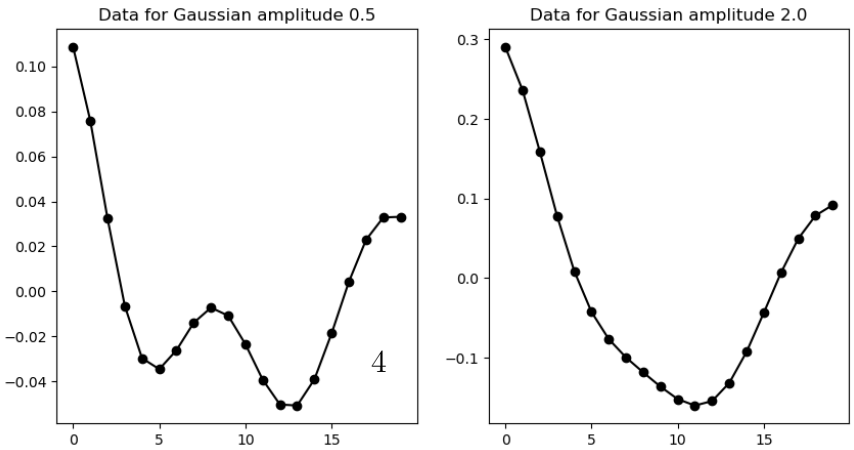
Where m_1 is your original model, m_2 is the new model, and Δm is the change in the amplitude (of either the boxcar or Gaussian, independently). Change the amplitude of the boxcar, compute the sensitivity using equation 4, plot the data, and comment on how a chance in the amplitude of the boxcar changes the data. Next repeat these steps changing the amplitude of the Gaussian.

The data is sensitive to both boxcar and Gaussian function in the model. The computed sensitivity to changing the boxcar is 0.0617 and the sensitivity to changing the Gaussian is higher at 0.095.

Increasing the amplitude of the boxcar makes the boxcar kink in the more pronounced in the data.



On the other hand, increasing the amplitude of the Gaussian has the effect of smoothing out the data and the kink in the data is not present.That the kink in the data completely disappears on increasing the amplitude of the Gaussian reflects the higher sensitivity of the Gaussian feature in the model.

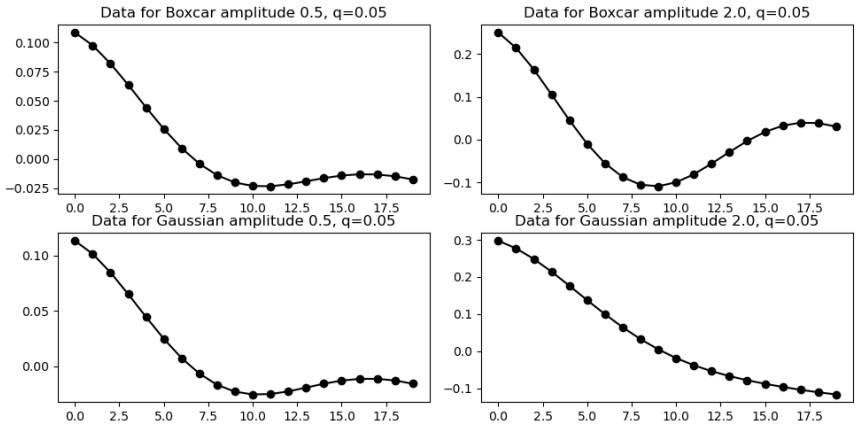


e. Now, we will look at how the kernels impact the sensitivity to features in the model. Make the kernel functions less oscillatory by decreasing q; use $q = 0.05$. Now change the amplitude of the boxcar and Gaussian independently. Describe how this impacts the data and quantify the sensitivity using equation 4. What happens if the kernels are more oscillatory ($q = 0.2$)?

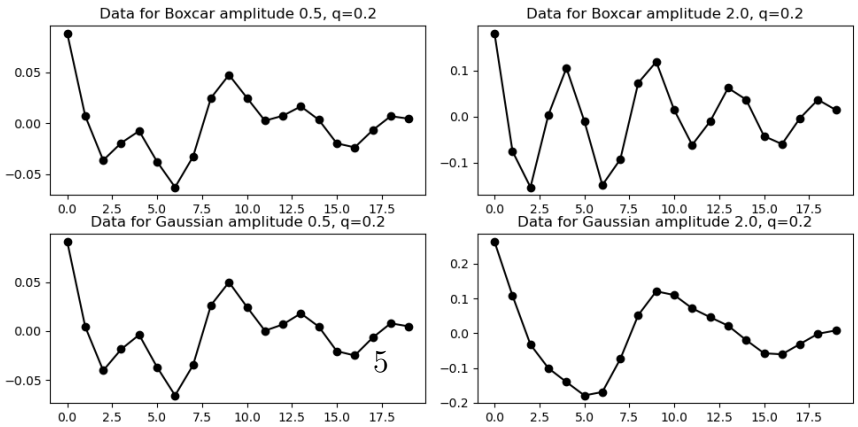
With lower oscillatory kernels, the boxcar has lower sensitivity. So when the amplitude of boxcar is increased, the data is slightly flatter but mostly unchanged. However, when the amplitude of the Gaussian is increased the data changes and is lot flatter reflecting the greater sensitivity of the low oscillation kernels to the Gaussian. With higher oscillatory kernels, the sensitivity of boxcar is greater and the data have more structure when boxcar has greater amplitude.

In general, the data is more oscillatory when the kernels are more oscillatory, in particular when the amplitude of the boxcar is increased. The data is flatter when the amplitude of the Gaussian is increased.

For lower oscillatory kernels with $q = 0.05$:
Boxcar's Sensitivity : 0.0566
Gaussian's Sensitivity: 0.0779



More oscillatory kernels with $q = 0.2$:
Boxcar's Sensitivity : 0.0566
Gaussian's Sensitivity: 0.0779

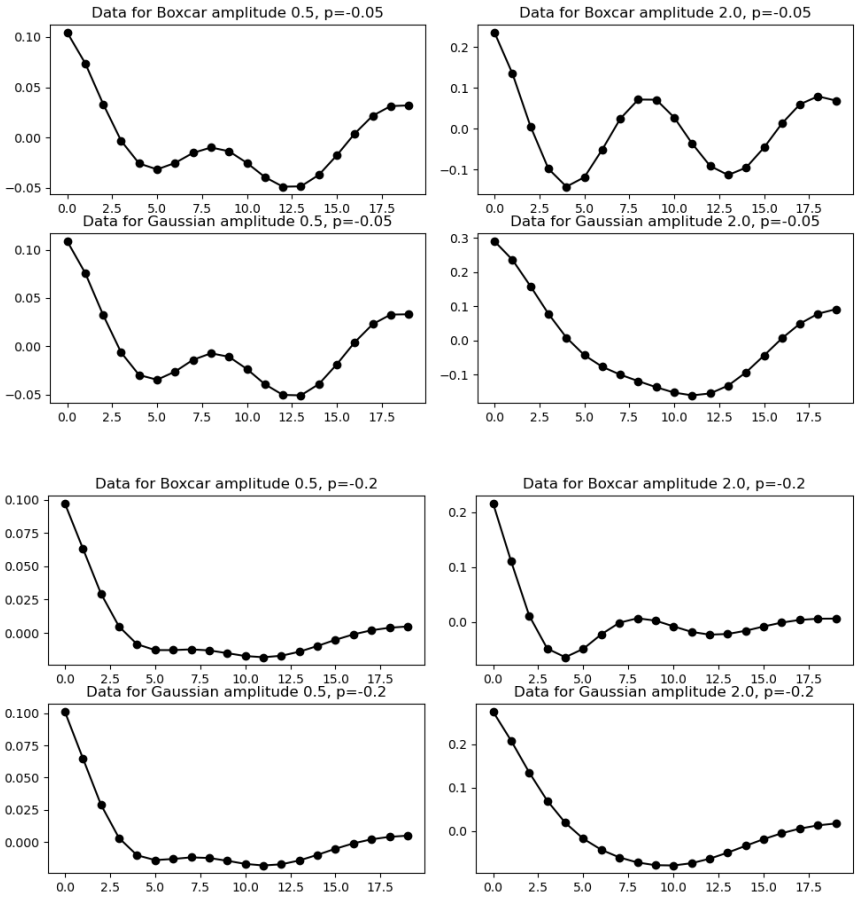


f. Set $q = 0.1$ again. Finally, create some examples that show how p impacts sensitivity. Change p to make the kernels decay more rapidly, and then less rapidly and describe how the data changes with changes in the model parameters in each scenario.

Parameter p controls the decay rate of the kernels, so when p is increased the kernels decay more rapidly. In this case, increasing the amplitude of the boxcar or Gaussian has a lower impact on the data due to the rapid decay of the kernels. The data is flatter and less sensitive to changes in the model amplitude when the kernels decay more rapidly. On the other hand, when the kernels decay less rapidly, the data is more sensitive to changes in the model parameters and the data is more oscillatory.

Sensitivity when p = -0.05 when changing the amplitude of Boxcars is 0.062 and for Gaussian is 0.095
Similarly when p = -0.2, the sensitivity when changing the amplitude of Boxcars is 0.032 and for Gaussian is 0.064.

In the right side panels of the plots below, the amplitude of either the boxcar or Gaussian is increased by a factor of 4 for parameter p = -0.05 and p = -0.2 respectively to show the effect described above.



Q2. In this question, we will set up the inverse problem. We will be solving a problem of the form

$$\min_{\mathbf{m}} \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m}) \tag{5}$$

where ϕ_d is the data misfit, given by

$$\phi_d(\mathbf{m}) = \|\mathbf{W}_d(\mathbf{G}\mathbf{m} - \mathbf{d}^{obs})\|^2 \tag{6}$$

with $\mathbf{W}_d = \text{diag}(1/\epsilon)$ where $\epsilon = [\epsilon_1, \epsilon_2, ...\epsilon_N]^\top$ is a vector containing the standard deviations of each datum. The model norm we will use consists of two parts, a "smallness" term and a first-order "smoothness" term. The continuous version is written as

$$\phi_m(m) = \alpha_s \int_x (m - m_{ref})^2 dx + \alpha_x \int_x \left(\frac{dm}{dx}\right)^2 dx \tag{7}$$

and the discrete version can be written as

$$\phi_m(\mathbf{m}) = \alpha_s \|\mathbf{W}_s(\mathbf{m} - \mathbf{m}_{ref})\|^2 + \alpha_x \|\mathbf{W}_x(\mathbf{m})\|^2 \tag{8}$$

a. We will start by setting up all of the pieces that we need, beginning with the $\mathbf{W}_d$, $\mathbf{W}_s$ and $\mathbf{W}_x$ matrices. We programmed everything for $\mathbf{W}_d$ and $\mathbf{W}_s$ in class, and we talked about what the structure of $\mathbf{W}_x$ should look like. Write a small function to generate $\mathbf{W}_d$ given values for the standard deviations of the data. Write functions to generate $\mathbf{W}_s$ and $\mathbf{W}_x$ given the locations of the nodes of the mesh (e.g. the `x_nodes` vector we created in class). Test the creation of the matrices with a very small mesh (e.g. 4 cells) and very small number of data (e.g. 2 data points). Describe the shapes of each matrix and print the entries that they contain.
The shapes and entries of the weight matrices for mesh of 4 cells and 2 data points are:

```
Wd shape: (2, 2)
Ws shape: (4, 4)
Wx shape: (4, 4)
Wd entries:
[[10.  0.]
 [ 0. 10.]]
Ws entries:
[[0.5 0.  0.  0. ]
 [0.  0.5 0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  0.5]]
Wx entries:
[[ 1. -1.  0.  0.]
 [ 0.  1. -1.  0.]
 [ 0.  0.  1. -1.]
 [ 0.  0.  0.  1.]]
```

b. Next, we will derive the system of equations that need to be solved to solve the inverse problem and recover a model ($\tilde{\mathbf{m}}$). For simplicity, you can assume $\mathbf{m}_{ref} = \mathbf{0}$ (as a bonus, keep $\mathbf{m}_{ref}$ in your derivation). We know that for a convex function, the minimum occurs when $\nabla\phi = 0$. Take the derivative of the objective function in equation 6 and set that equal to 0. Derive the linear equations that need to be solved to estimate $\mathbf{m}$.
The objective function is given by:

$$\phi = \phi_d(\mathbf{m}) + \phi_m(\mathbf{m}) = \frac{1}{2}||\mathbf{W_d}(\mathbf{Gm} - \mathbf{d})||^2 + \frac{1}{2}||\mathbf{W_s}(\mathbf{m} - \mathbf{m_{ref}})||^2 + \frac{1}{2}||\mathbf{W_x}(\mathbf{m})||^2 \quad (9)$$

Taking the derivative of the objective function with respect to m and setting it to zero, we get:

$$\frac{\partial\phi}{\partial\mathbf{m}} = \mathbf{G}^T{\mathbf{W_d}}^T\mathbf{W_d}(\mathbf{Gm} - \mathbf{d}) + {\mathbf{W_s}}^T\mathbf{W_s}(\mathbf{m} - \mathbf{m_{ref}}) + {\mathbf{W_x}}^T\mathbf{W_x}m = 0 \quad (10)$$

Rearranging the terms, we get:

$$(\mathbf{G}^T{\mathbf{W_d}}^T\mathbf{W_d}\mathbf{G})\mathbf{m} + \beta({\mathbf{W_s}}^T\mathbf{W_s} + {\mathbf{W_x}}^T\mathbf{W_x})\mathbf{m} = \mathbf{G}^T{\mathbf{W_d}}^T\mathbf{W_d}\mathbf{d} + \beta{\mathbf{W_s}}^T\mathbf{W_s}\mathbf{m_{ref}} \quad (11)$$

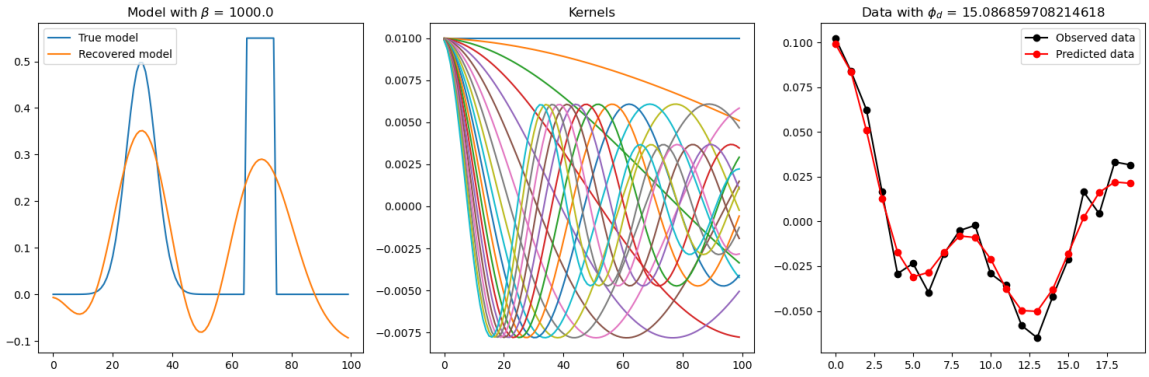
The above equation can be written as a linear equation which is solved to estimate m.
The equation is given by

$$\mathbf{Am} = \mathbf{b} \quad (12)$$

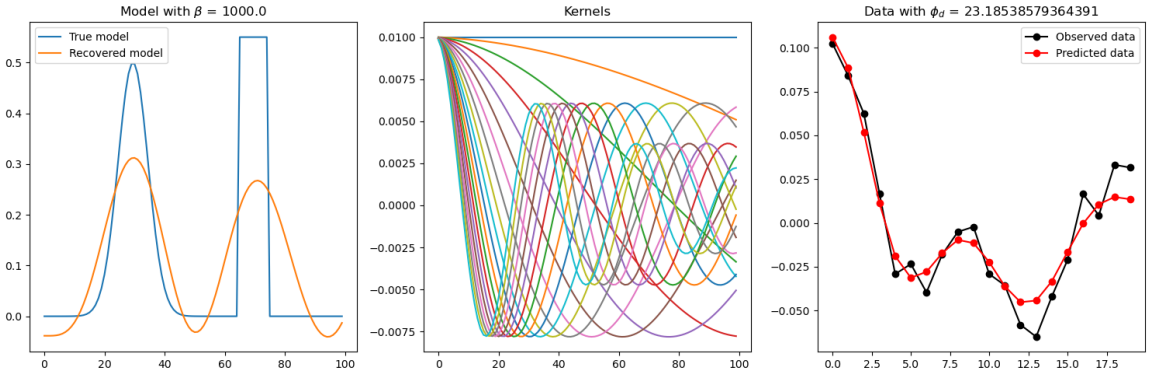
where,
$\mathbf{A} = \mathbf{G}^T{\mathbf{W_d}}^T\mathbf{W_d}\mathbf{G} + \beta({\mathbf{W_s}}^T\mathbf{W_s} + {\mathbf{W_x}}^T\mathbf{W_x})$,
$\mathbf{m}$ = model parameters,
$\mathbf{b} = \mathbf{G}^T{\mathbf{W_d}}^T\mathbf{W_d}\mathbf{d} + \beta{\mathbf{W_s}}^T\mathbf{W_s}\mathbf{m_{ref}}$

c. Now, we will program a function to estimate a model given a observed data, estimates of the uncertainties on those data, and the parameters $\beta, \alpha_s, \alpha_x$. It is also a good idea to have this function take the matrix $\mathbf{G}$ so that it is easy to later play with the kernels. Create this function. Now we will use it to recover a model. Use the parameters for the kernels described in Q1c and add Gaussian random noise with a standard deviation of 10^{-2}. Try two different scenarios:

- assign the correct standard deviations to the observed data and use $\alpha_s = 1$ and $\alpha_x = 0$. Choose a β-value that gives you $\phi_d \sim N$ where N is the number of data.



- assign the correct standard deviations to the observed data, $\alpha_s = 0$ and $\alpha_x = 1$. Choose a β-value that gives you $\phi_d \sim N$ where N is the number of data.



Generate a plot with three panels that shows: (a) the true and recovered models, (b) the kernels, and (c) the observed and predicted data for each of these scenarios (it will be useful if you create a function for generating these plots that takes the true and recovered models, and $\mathbf{G}$ matrix). Specify the choice of β and the resultant ϕ_d-value (here, I am only looking for something that is the right order of magnitude, e.g. try $\beta = 10^2, 10^3, 10^4, ...$ We will get more specific in the next questions.)

$\beta = 10^3$ produces good results. Plots attached above.

Q3. In this question, we will plot and explore the Tikhonov curves. You will generate a plot that has 6 panels. One row should show the Tikhonov curves: ϕ_d vs. β, ϕ_m vs. β, and ϕ_d v.s. ϕ_m (be sure to plot β as decreasing to the right, just as we did in class), and another row should show the true and recovered model, the kernels, and the predicted and observed data. It should look something like Figure 1 (note that your model might be different than mine, so the curves, model and data will all look different.)

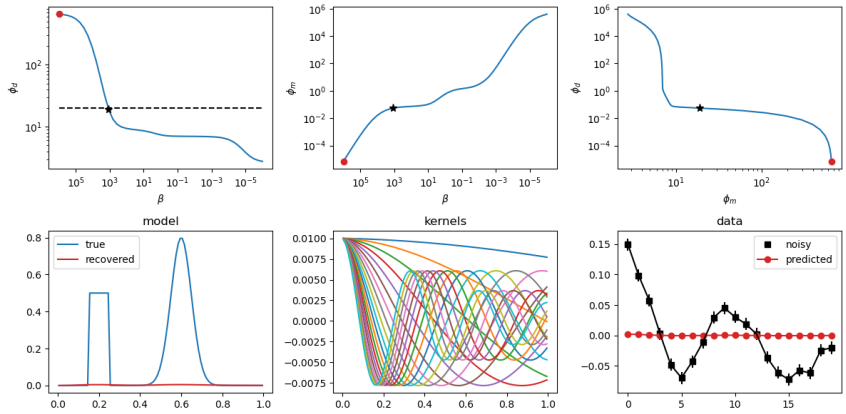
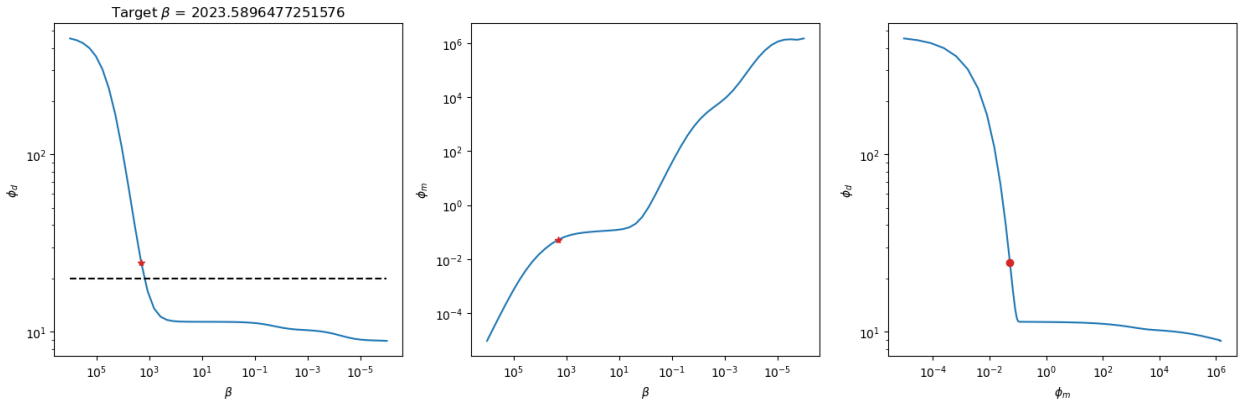
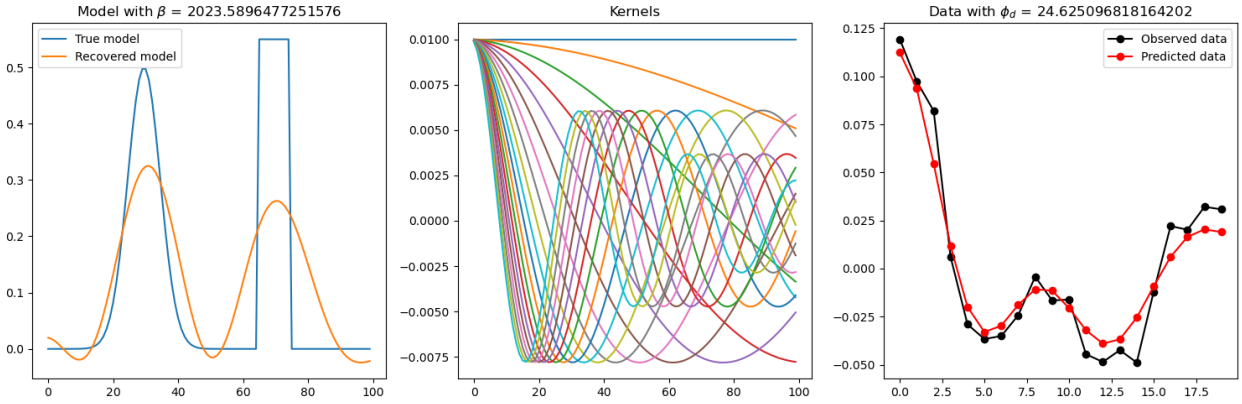


Figure 1: Example plot showing Tikhonov curves and the recovered model, kernels and data.

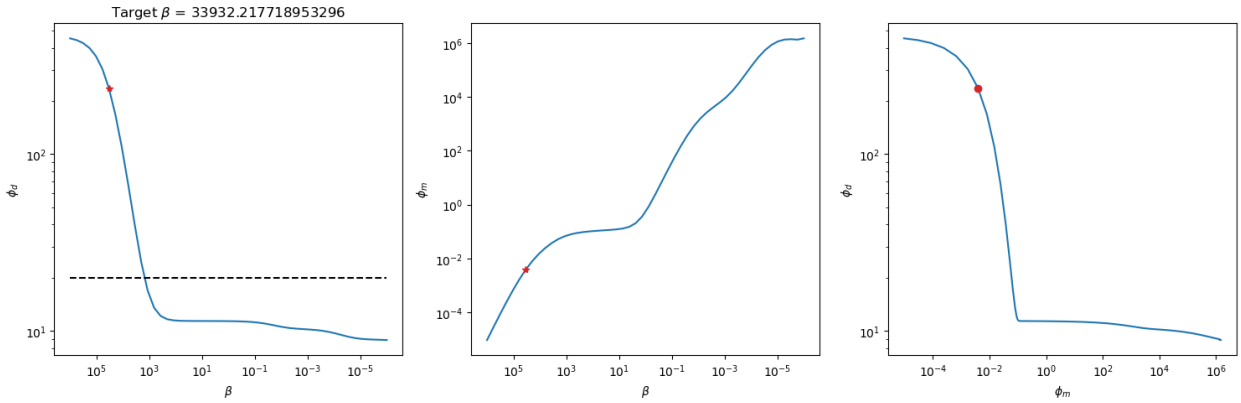
a. Create a vector that contains 50 values of β that are logarithmically spaced between $\beta = 10^6$ and $\beta = 10^{-6}$. Set $\alpha_s = 1$, $\alpha_x = 0$ and plot the Tikhonov curves. On the plots for the Tikhonov curves, plot a line where the target misfit is and use a star to indicate the β that corresponds to the largest β where we have reached the target misfit (e.g. the largest beta where $\phi_d \leq N$, similar to Figure 1). Choose 3 locations along the curves to show: one that is underfit, one that is overfit and one that acceptably fits the data. Choose examples that aren't at the total extremes. Describe which is which.

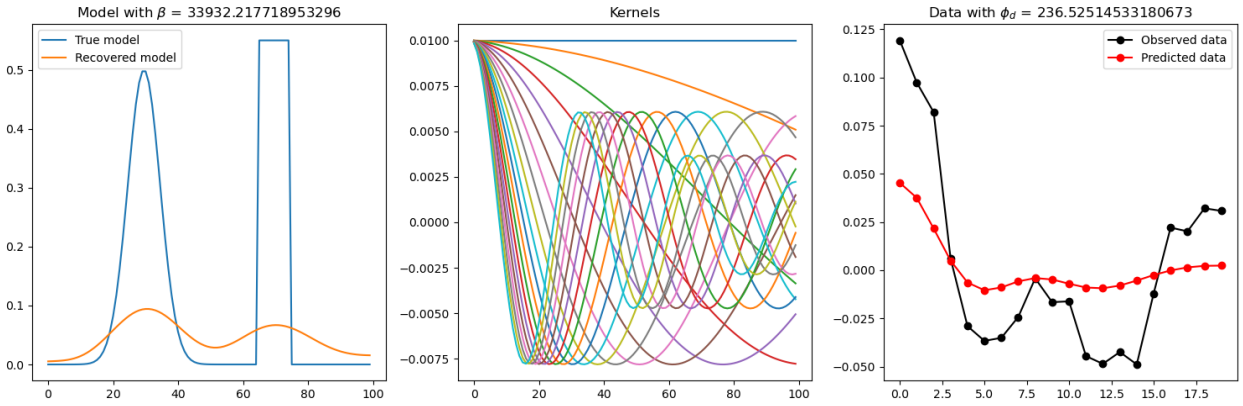
Case with optimal β:



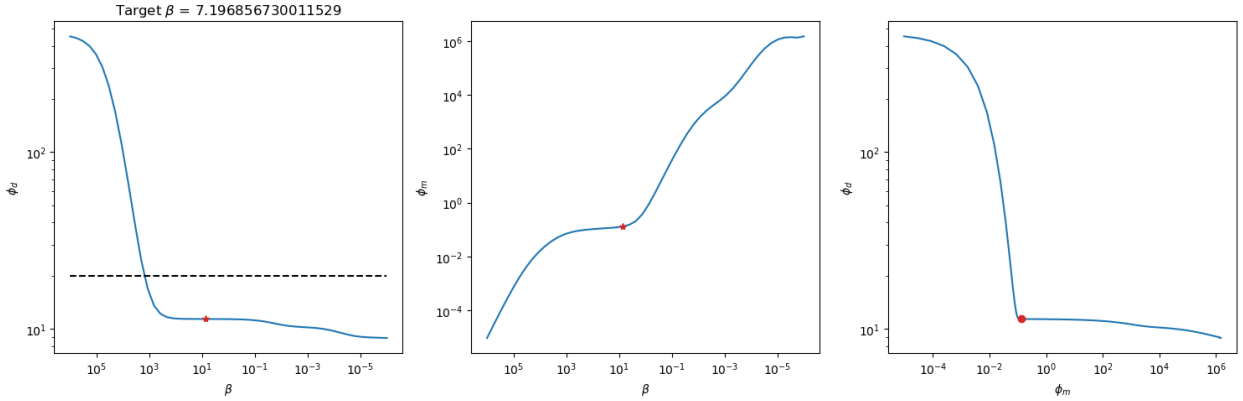


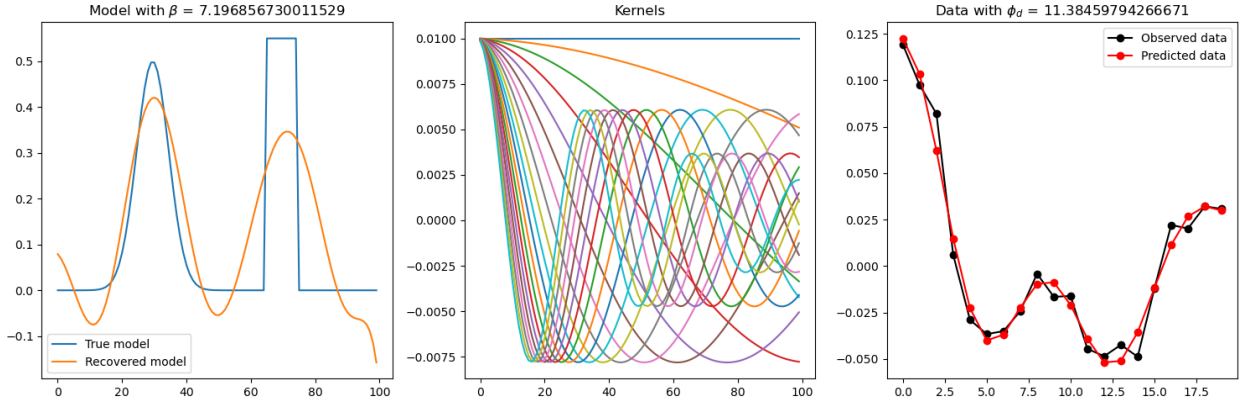
Case with underfit β:





Case with overfit β:

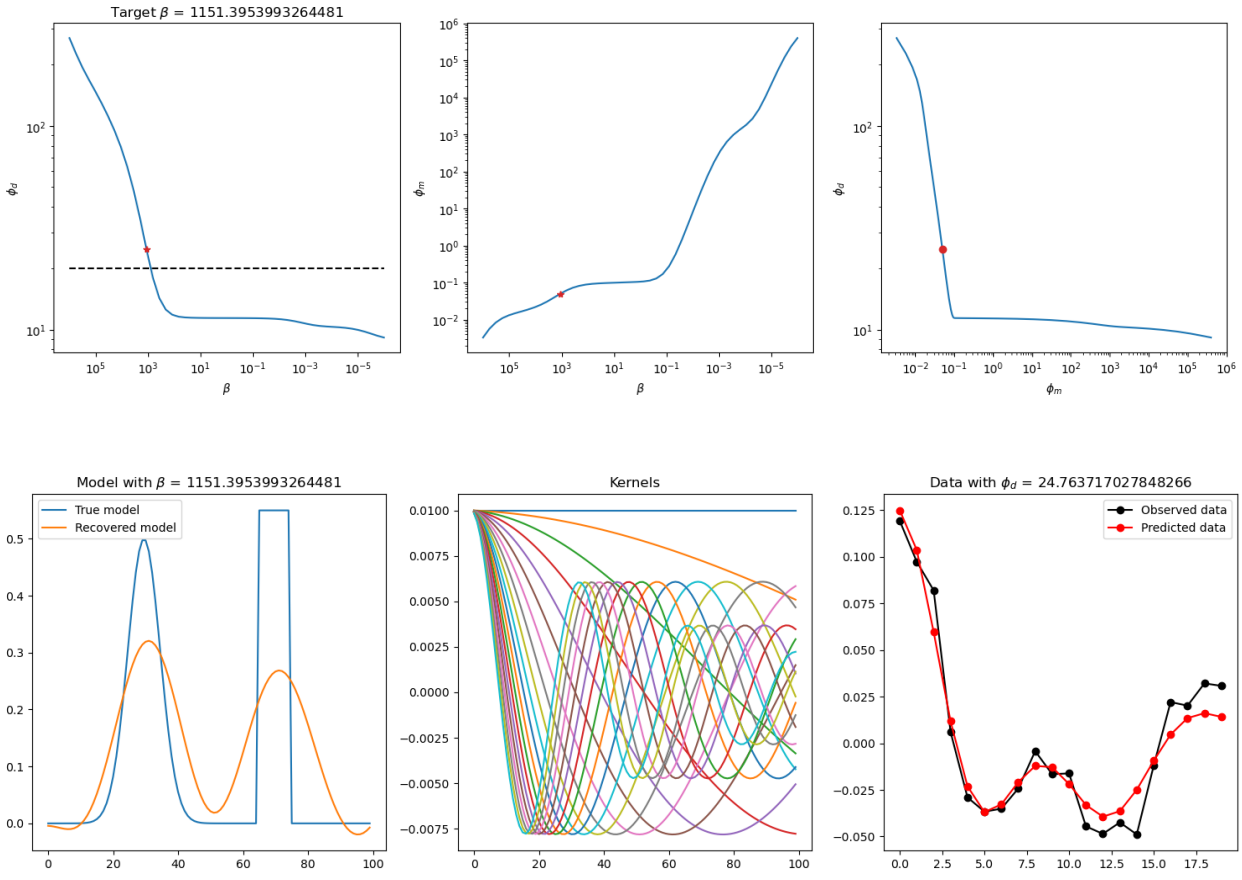




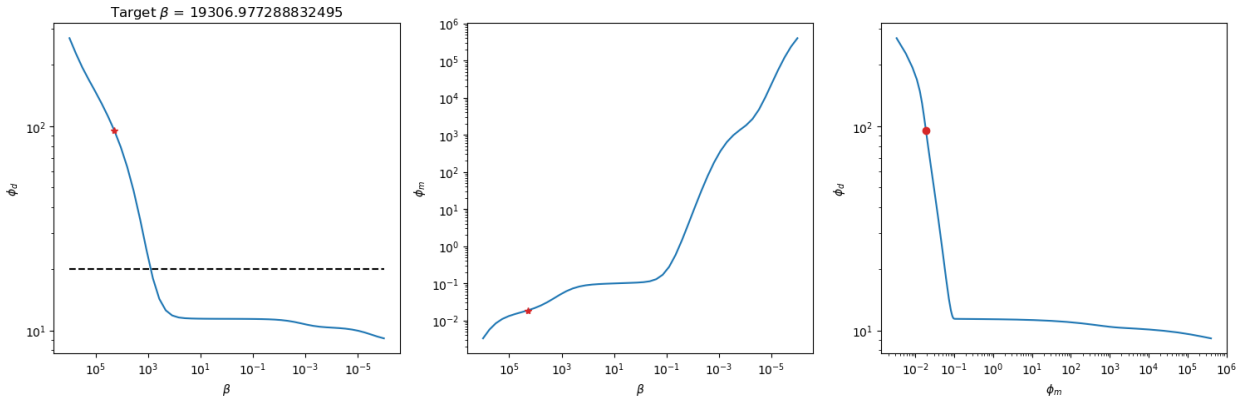
b. Repeat the exercise from Q3a, but now set $\alpha_s = 0$, $alpha_x = 1$. Again, choose 3 examples, one that underfits the data, one that overfits the data and one that acceptably fits the data. Try to choose these models such that the ϕ_d values are similar to those for the choices you made in part a. Describe what is similar / different between the two scenarios.

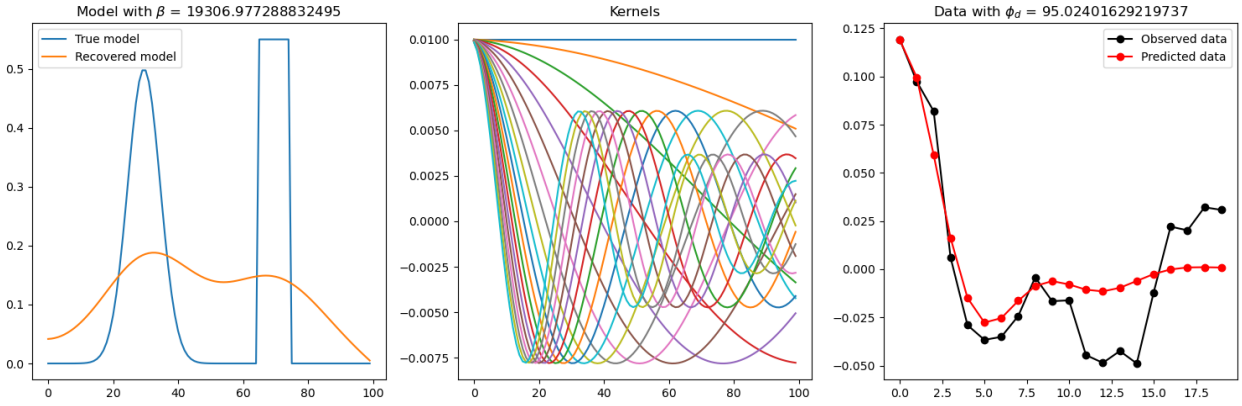
The recovered model for the optimal beta is similar for both scenarios. The recovered model for the underfitting example is also similar for both scenarios. However, the recovered model for the overfitting example is different for the two scenarios. The recovered model for the overfitting example with $\alpha_s = 1$ and $\alpha_x = 0$ is more oscillatory than the recovered model for the overfitting example with $\alpha_s = 0$ and $\alpha_x = 1$. The recovered model for the overfitting example with $\alpha_s = 0$ and $\alpha_x = 1$ is smoother than the recovered model for the overfitting example with $\alpha_s = 1$ and $\alpha_x = 0$.

Case with optimal β:

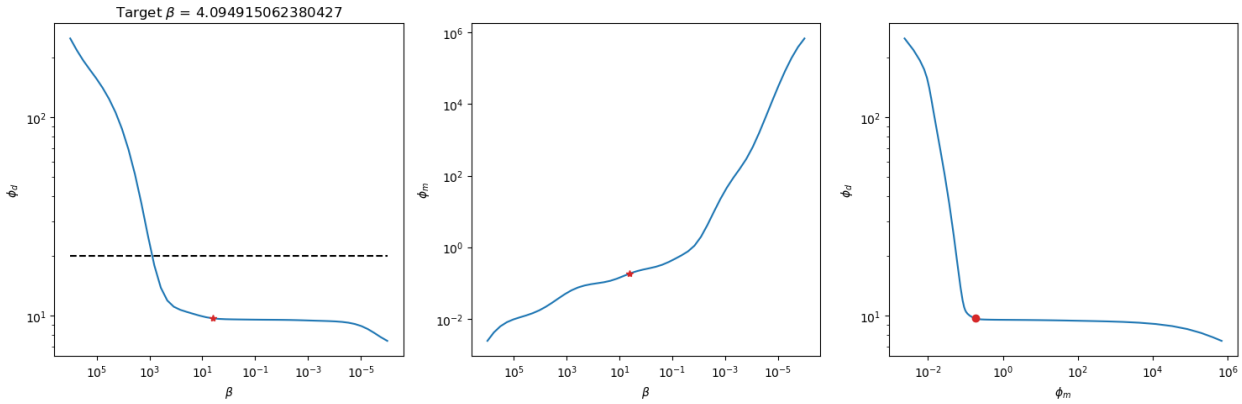


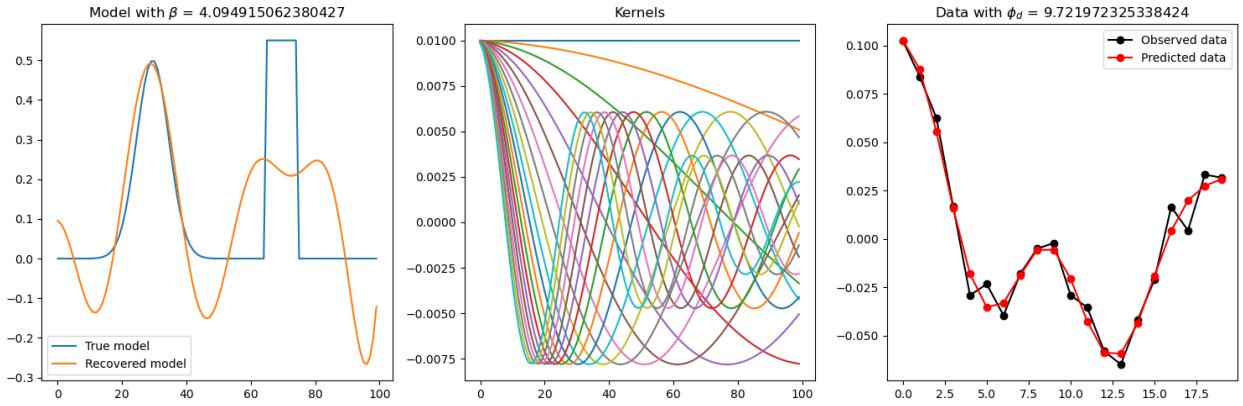
Case with underfit β:





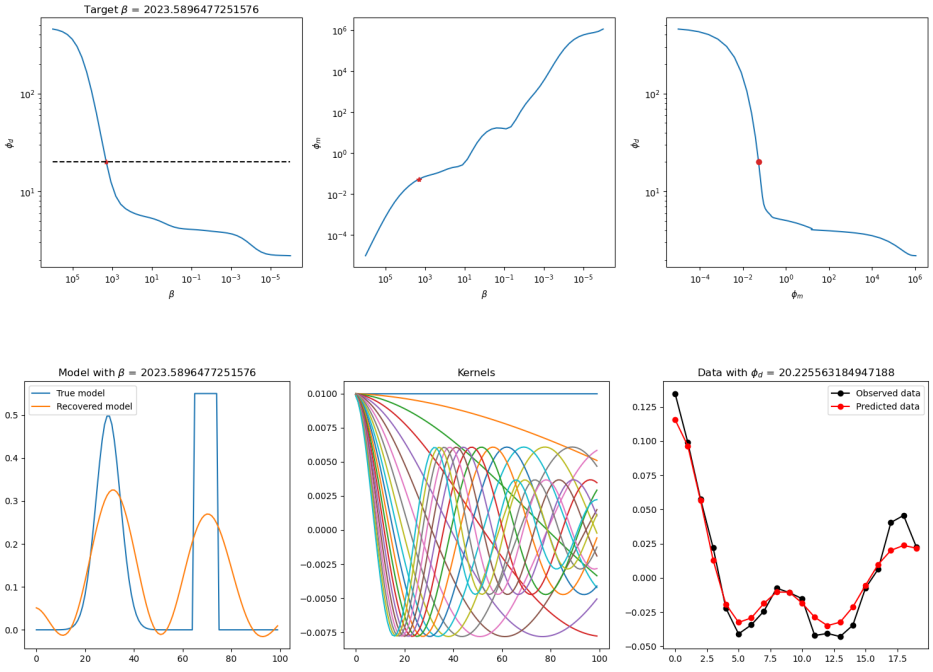
Case with overfit β:



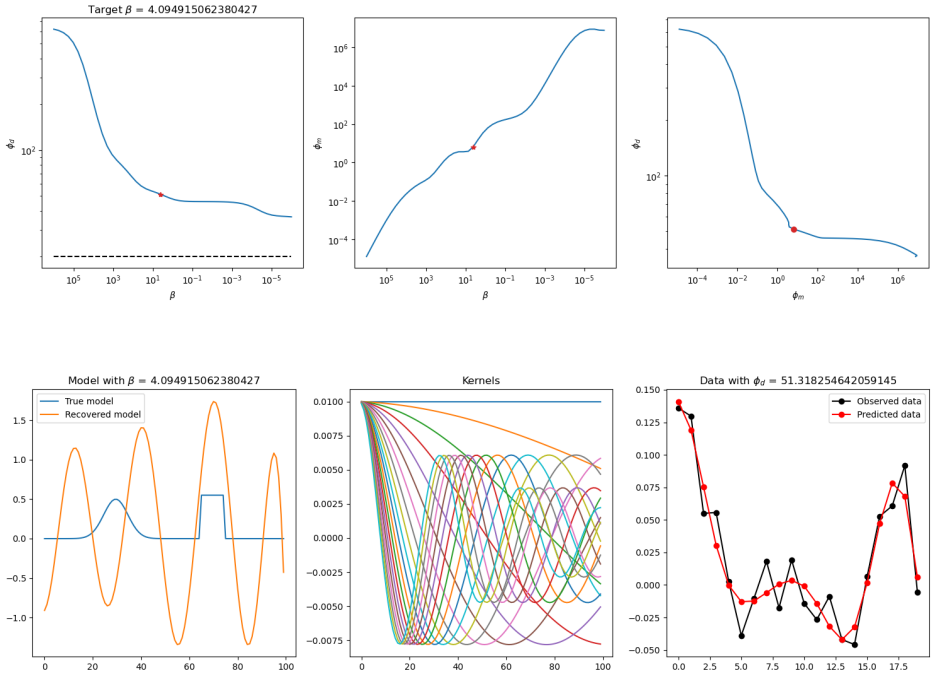


c. Time to play! Increase or decrease the noise, and discuss how this changes the Tikhonov curves, where the target misfit is and the nature of the model. Or what happens if you change the kernels and make them more / less oscillitory? or what if the decay more / less rapidly. Pick something to explore and generate a few plots to illustrate what you learned. I am expecting something no more that 2-3 plots that illustrate whatever you choose to explore.
Let's compare the results for different standard deviations of the data. I use the same model and the same number of kernels and change the standard deviation of the data. I use $\alpha_s = 1$ and $\alpha_x = 0$.

In the first case below a standard deviation of 0.01 is used:



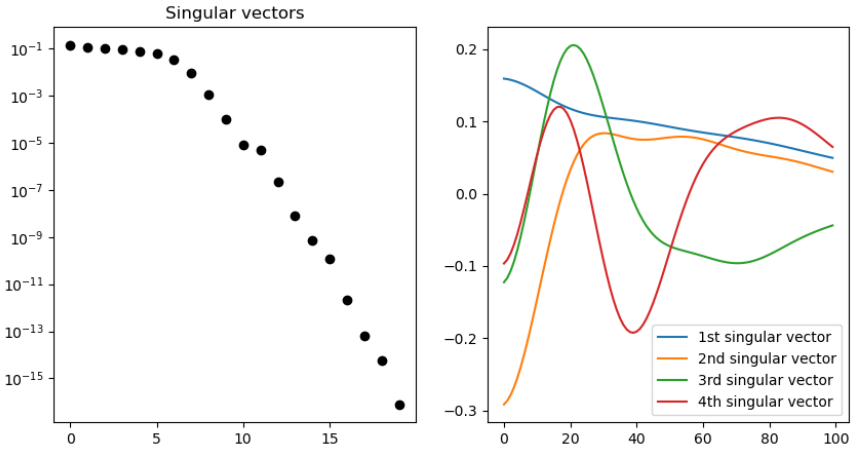
In the second case below a standard deviation of 0.03 is used:



Changing the standard deviation from 1% to 3% of the observed data has a large effect on the inversion results. The inversion fails to hit our target misfit for the larger standard deviation and the recovered model is also more oscillatory for the larger standard deviation.

Q4. Finally, we will set up and solve the inverse problem using Singular Value Decomposition (SVD).

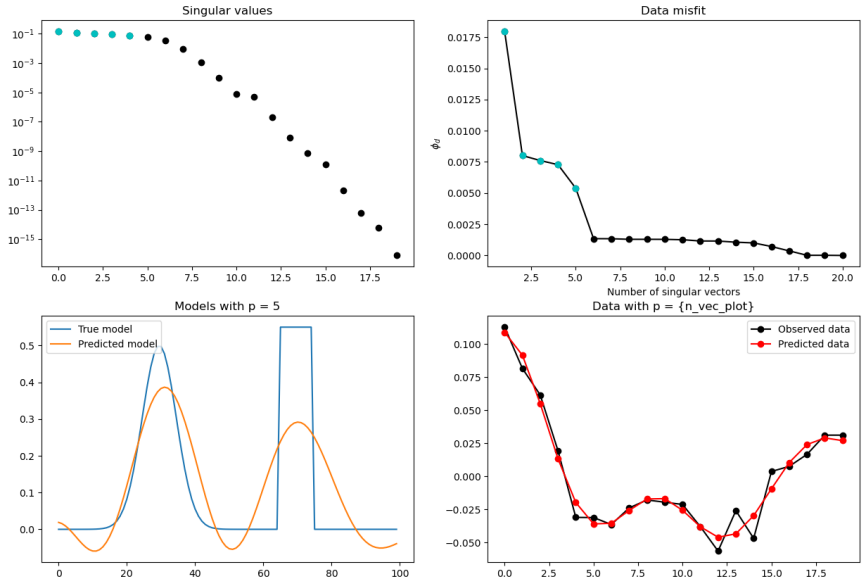
a. Perform the SVD decomposition of the system matrix $\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$. Generate a plot with 2 panels, show the singular values (on a log scale) and plot the first few singular vectors (use different colors to highlight which vectors correspond to each singular value)



b. As we discussed in class, the model can be constructed from the singular values and vectors according to

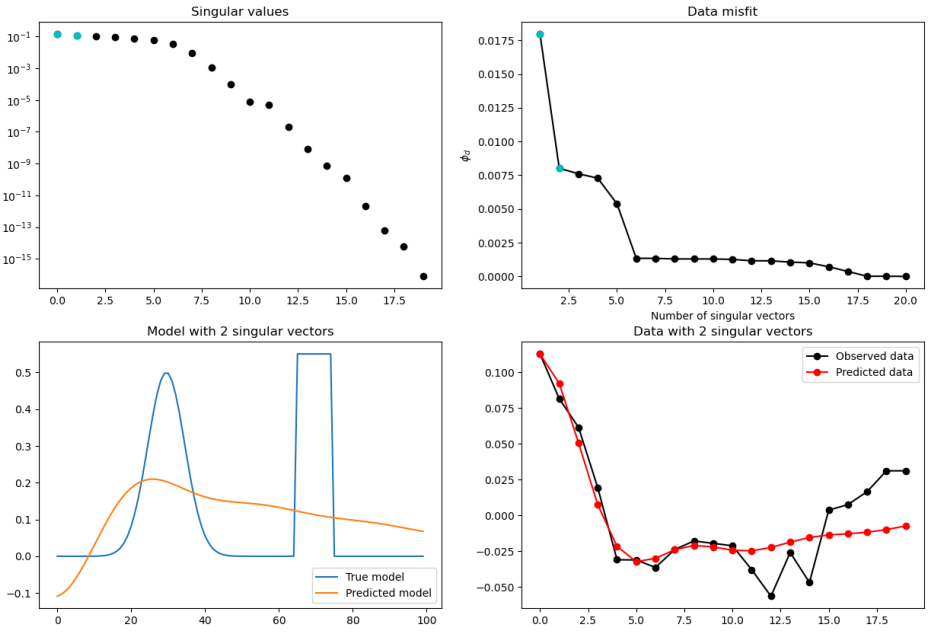
$$\mathbf{m}_c = \sum_{i=1}^{p} \left(\frac{\mathbf{u_i}^\top \mathbf{d}^{obs}}{\lambda_i} \mathbf{v_i} \right) \tag{13}$$

Create a function that returns $\mathbf{m}_c$ given the matrix $\mathbf{G}$ and a choice of the number of singular vectors, p. Generate a plot that shows: (a) the singular values, (b) the data misfit as a function of the number of singular vectors used to construct the solution (similar to the top left plot in Figure 1, but replacing β on the x axis with p, (c) the true and predicted models, and (d) predicted and observed data.

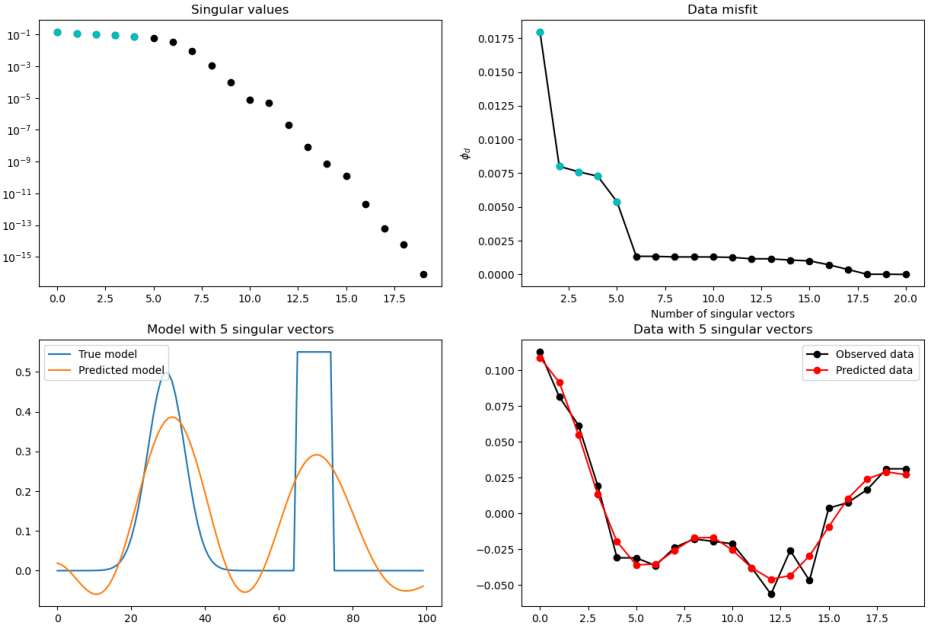


c. Using the code you wrote for Q4b, show 3 different choices of the number of singular values: one that underfits the data, one that overfits the data, and one that acceptably fits the data. Specify which is which.

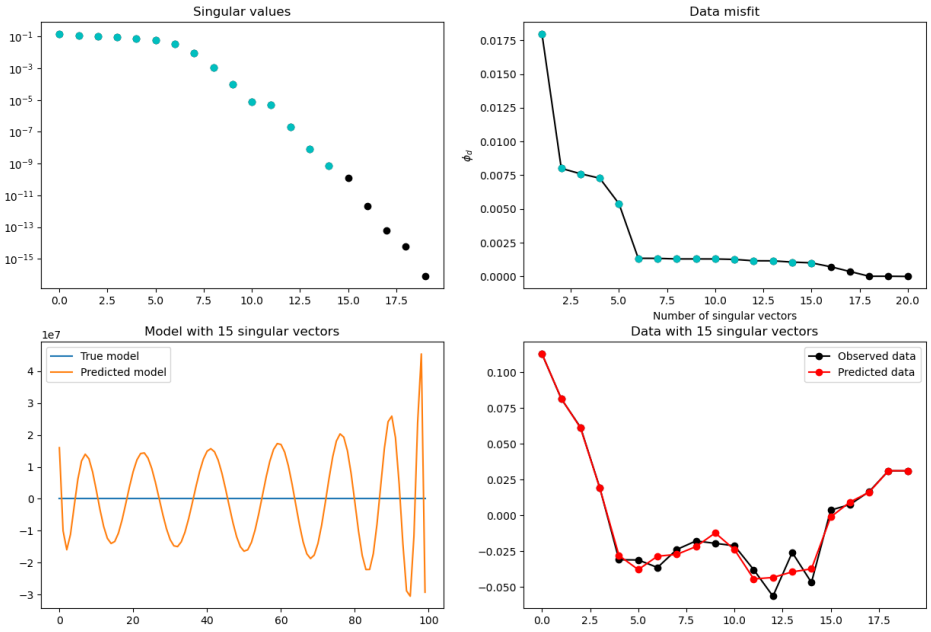
Underfit plot with using 2 singular values:



Example of optimal fit plot with using 5 singular values:

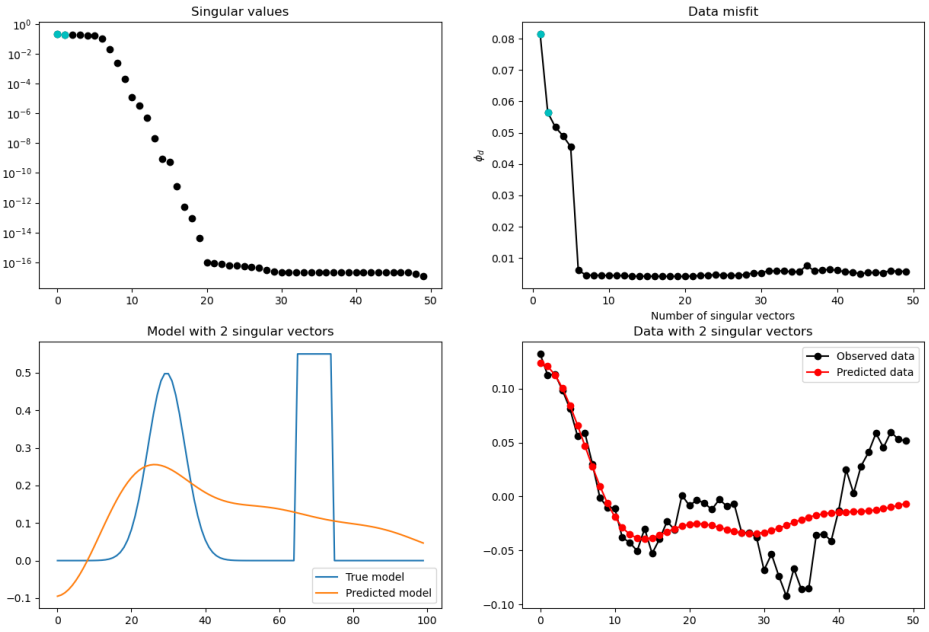


Example of overfit with using 15 singular values:

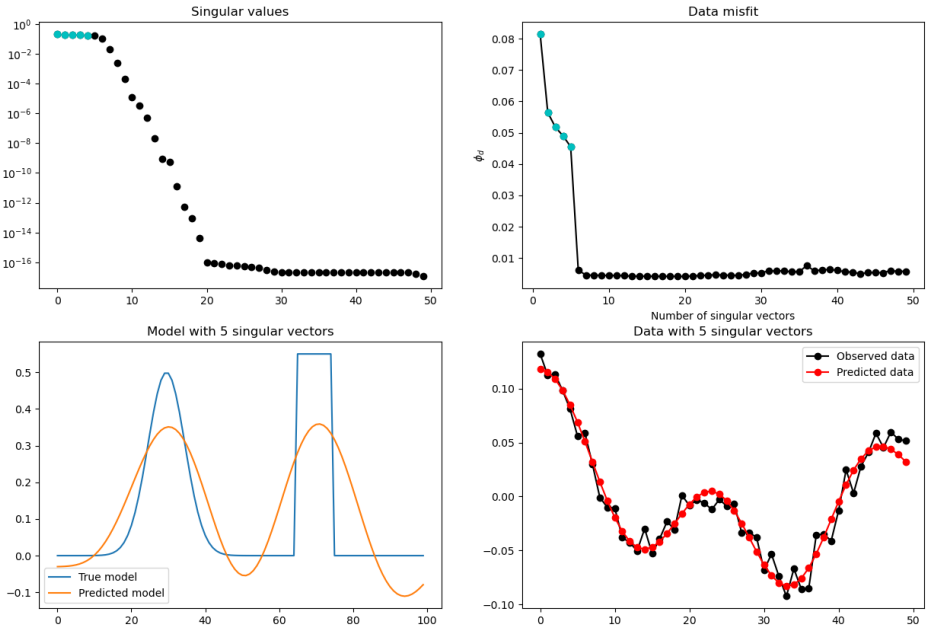


d. Time to play! Change the kernels: what happens if you change the kernels and make them more / less oscillatory? or what if the decay more / less rapidly. Pick something to explore and generate a few plots to illustrate what you learned. I am expecting something no more that 2-3 plots that illustrate whatever you choose to explore.

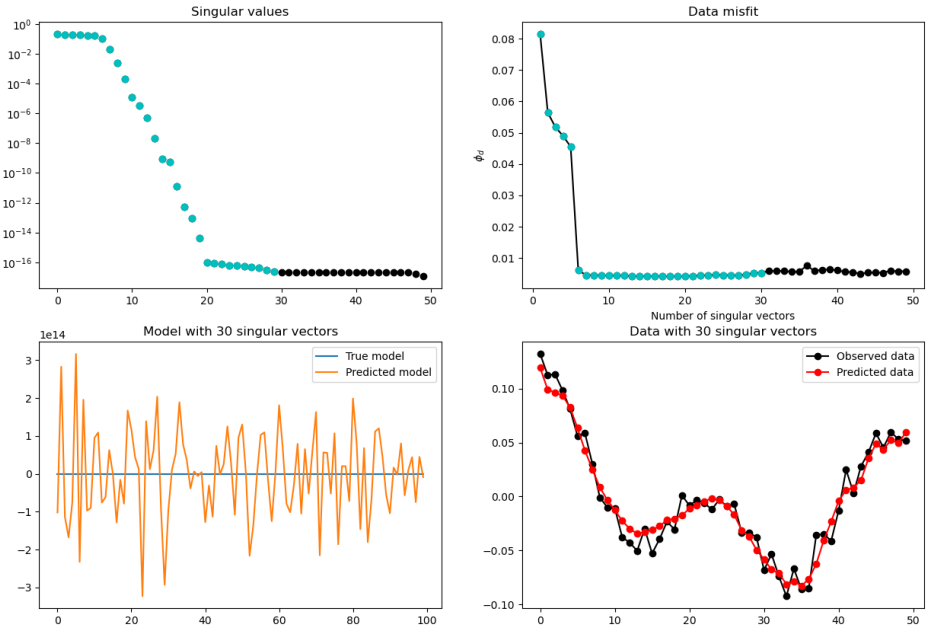
I explored the effect of changing the number of kernels from 20 to 50. Changing the number of kernels show that even with greater number of kernels and thus data, the first 5 vectors are able to capture the data well. As well, the overfitting when including larger number of vectors is apparent similar to the case where no regularization was included.
Underfit plot with using 2 singular values:



Example of optimal fit plot with using 5 singular values:



Example of overfit with using 30 singular values:


Singular values
Data misfit
Number of singular vectors
Model with 30 singular vectors
1e14
True model
Predicted model
Data with 30 singular vectors
Observed data
Predicted data