Inverse Problems Exercises: 2024s s01 (all)

https://www.umm.uni-heidelberg.de/miism/

Notes

- Please **DO NOT** change the name of the `.ipynb` file.
- Please **DO NOT** import extra packages to solve the tasks.
- Please put the `.ipynb` file directly into the `.zip` archive without any intermediate folder.

Please provide your personal information

- full name (Name):

YOUR ANSWER HERE

I05: Parabolic trajectory problem

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Forward problem

Assume that we observe a ball thrown perpendicular to the earth's surface to the sky. By ignoring friction, we can write the equation of motion as the height g as a function of time t as

$$g_i = g(t_i) = f_0 + f_1 t_i + \frac{1}{2} f_2 t_i^2 = \begin{bmatrix} 1 & t_i & \frac{1}{2}t_i^2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}.$$

- Given $f_{\text{true}} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 25 \\ -4 \end{bmatrix}$
- Given $t_i = (0.0, 0.5, 1.0, \ldots, 9.0, 9.5)$
- Calculate $g_i = g(t_i)$
- Save the output in the variables `f_true`, `t` and `g`, respectivly (as `numpy.array`)
- Plot `g` versus `t` in the axes `ax`

In [ ]:
```python
# YOUR CODE HERE
raise NotImplementedError()

fig, ax = plt.subplots()  # Create a figure and an axes.

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```python
# This cell contains hidden tests.
```

In [ ]:
```python
# This cell contains hidden tests.
```

Inverse problem

Collecting all measurements, we obtain the system of equations

$$g = \begin{bmatrix} g_1 \\ \dots \\ g_n \end{bmatrix} = \begin{bmatrix} 1 & t_1 & \frac{1}{2}t_1^2 \\ \dots & \dots & \dots \\ 1 & t_n & \frac{1}{2}t_n^2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} = Af.$$

When the number of measurements is the same as the number of parameters, A is square. The solution involves the inverse as

$$\hat{f} = A^{-1}g.$$

- Take the last 3 elements from `t` and `g`
- Calculate A and $\hat{f}$
- Save the output in the variables `A` and `f_est`, respectivly (as `numpy.array`)

In [ ]:
```python
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```python
# This cell contains tests.


print(A)
print(f_est)
```

Measurement errors

The real measurements usually contain noise. In the case of additive Gaussian noise, the problem is formulated as

$$g' = g + \epsilon = Af + \epsilon,$$

where ϵ is a random variable with Gaussian distribution with mean 0 and variance σ^2, i.e. $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The solution involves the inverse as

$$\hat{f} = A^{-1} g'.$$

- Given $\sigma = 2$
- Calculate g'
- Save the output in the variable `g_n` (as `numpy.array`)
- Plot `g` versus `t` in the axes `ax`
- Plot `g_n` versus `t` in the axes `ax` as well
- Show the legend in the axes `ax`
- Take the last 3 elements from `t` and `g_n`
- Calculate A and $\hat{f}$
- Save the output in the variables `A` and `f_est`, respectivly (as `numpy.array`)

In [ ]:
```
# YOUR CODE HERE
raise NotImplementedError()

fig, ax = plt.subplots()  # Create a figure and an axes.

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# This cell contains hidden tests.
```

In [ ]:
```
# This cell contains hidden tests.
```

In [ ]:
```
# This cell contains tests.


print(f_est)
```

Question: Noise

- How does the result depend on the noise?

YOUR ANSWER HERE

Pseudo-inverse

When the number of measurements is larger than the number of parameters, A is generally not invertible. The solution involves the pseudo-inverse as

$$\hat{f} = (A^T A)^{-1} A^T g = A^{PI} g'.$$

- Take all elements from `t` and `g_n`
- Calculate A and $\hat{f}$
- Save the output in the variables `A` and `f_est`, respectivly (as `numpy.array`)

In [ ]:
```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# This cell contains tests.


print(A)
print(f_est)
```

In [ ]:
```
# This cell contains hidden tests.
```

Question: Data

- How does the result depend on the number of considered data?

YOUR ANSWER HERE