f-17-jupyter-kond

April 6, 2021

1 Eksperiment

Approksimer med polynomier funktionen

$$f(t) = \frac{1}{c} e^{\sin(4t)}$$

på intervallet $[0, 1]$.

Arbejd med polynomier af grad 14.

Del $[0, 1]$ i $m = 100$ punkter.

Problemet svarer til at finde mindste kvadraters løsning til

$$\begin{bmatrix} t_0^{14} & t_0^{13} & \dots & t_0 & 1 \\ t_1^{14} & t_1^{13} & \dots & t_1 & 1 \\ \vdots & & & & \\ t_{m-1}^{14} & t_{m-1}^{13} & \dots & t_{m-1} & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{14} \end{bmatrix} = \begin{bmatrix} f(t_0) \\ f(t_1) \\ \vdots \\ f(t_{m-1}) \end{bmatrix}$$

Jeg har regnet en værdi for c, så at det korrekte svar har $x_0 = 1{,}0$.

[1]:
```python
import numpy as np
```

[2]:
```python
m = 100
cols = 15
t = np.linspace(0, 1, m)
a = np.vander(t, cols)
b = np.exp(np.sin(4*t))[:, np.newaxis] / 2006.787453104852
```

[3]:
```python
u, s, vt = np.linalg.svd(a, full_matrices=False)
```

[4]:
```python
# svd løsning
x = vt.T @ (np.diag(1/s) @ (u.T @ b))
x[0,0]
```

[4]:
```
1.000000068599447
```

[5]:
```python
kappa = s[0] / s[-1]
print(kappa, f'{kappa:e}')
```

```
22717773321.988663 2.271777e+10
```

[6]:
```python
pb = u @ (u.T @ b)
cos_theta = np.linalg.norm(pb) / np.linalg.norm(b)
np.arccos(cos_theta)
```

[6]: 3.745916011291342e-06

[7]:
```python
eta = s[0] * np.linalg.norm(x) / np.linalg.norm(pb)
print(eta)
```

```
210355.96238247075
```

[8]:
```python
print(f'{eta:e}')
```

```
2.103560e+05
```

[9]:
```python
kond_x_A = kappa + (kappa**2 * np.sqrt(1-cos_theta**2) / cos_theta / eta)
print(f'{kond_x_A:e}')
```

```
3.190818e+10
```

[10]:
```python
kond_x_A * np.finfo(float).eps
```

[10]: 7.085039238860051e-06

[11]:
```python
x[0,0]
```

[11]: 1.000000068599447

[12]:
```python
korrekt = 1.0
x[0,0] - korrekt
```

[12]: 6.859944701176346e-08

[13]:
```python
def forbedret_gram_schmidt(a):
    _, k = a.shape
    q = np.copy(a)
    r = np.zeros((k, k))
    for i in range(k):
        r[i, i] = np.linalg.norm(q[:, i])
        q[:, i] /= r[i,i]
        r[[i], i+1:] = q[:, [i]].T @ q[:, i+1:]
        q[:, i+1:] -= q[:, [i]] @ r[[i], i+1:]
    return q, r
```

[14]:
```python
q, r = forbedret_gram_schmidt(a)
```

[15]:
```python
(np.linalg.solve(r, q.T @ b))[0,0]
```

[15]: 1.0007157941525924

[16]:
```python
(np.linalg.solve(a.T @ a, a.T @ b))[0,0]
```

[16]: -0.24673590859846803

[17]:
```python
# konditionstal for a.T @ a
_, sn, _ = np.linalg.svd(a.T @ a, full_matrices=False)
sn[0] / sn[-1]
```

[17]: 7.507812256475365e+17

[19]:
```python
print(f'{1/np.finfo(float).eps:e}')
```

4.503600e+15

[20]:
```python
qnp, rnp = np.linalg.qr(a)
```

[21]:
```python
(np.linalg.solve(rnp, qnp.T @ b))[0,0]
```

[21]: 1.0000001280386777

[22]:
```python
(np.linalg.solve(rnp, qnp.T @ b))[0,0] - korrekt
```

[22]: 1.2803867766031374e-07

[23]:
```python
def house(x):
    u = x / np.linalg.norm(x)
    eps = -1 if u[0] >= 0 else +1
    s = 1 + np.abs(u[0])
    v = - eps * u
    v[0] += 1
    v /= s
    return v, s
```

[24]:
```python
def householder_qr_data(a):
    data = np.copy(a)
    _, k = a.shape
    s = np.empty(k)
    for j in range(k):
        v, s[j] = house(data[j:, [j]])
        data[j:, j:] -= (s[j] * v) @ (v.T @ data[j:, j:])
        data[j+1:, [j]] = v[1:]
    return data, s
```

[25]:
```python
def householder_qr(a):
    data, s = householder_qr_data(a)
    n, k = a.shape
    r = np.triu(data[:k, :k])
    q = np.eye(n, k)
```

```python
    for j in reversed(range(k)):
        x = data[j+1:, [j]]
        v = np.vstack([[1], x])
        q[j:, j:] -= (s[j] * v) @ (v.T @ q[j:, j:])
    return q, r
```

[26]:
```python
def householder_lsq(a, b):
    data, s = householder_qr_data(a)
    k = a.shape[1]
    r = np.triu(data[:k, :k])
    c = np.copy(b)
    for j in range(k):
        x = data[j+1:, [j]]
        v = np.vstack([[1], x])
        c[j:] -= (s[j] * np.vdot(v, c[j:])) * v
    return np.linalg.inv(r) @ c[:k]
```

[27]:
```python
householder_lsq(a, b)[0,0]
```

[27]: 0.9999999555257091

[28]:
```python
householder_lsq(a, b)[0,0] - korrekt
```

[28]: -4.447429091669619e-08

[29]:
```python
np.linalg.lstsq(a, b, rcond=None)[0][0,0]
```

[29]: 1.000000068608618

[30]:
```python
np.linalg.lstsq(a, b, rcond=None)[0][0,0] - korrekt
```

[30]: 6.860861789803607e-08

[ ]:
```python

```