prb4_p

March 27, 2022

[2]:
```
import numpy as np
from numpy import testing as testing
from matplotlib import pyplot as plt
```

1 Sheet 4, Exercise 1

[33]:
```
def conjugate_gradient(A:np.ndarray, b:np.ndarray, steps: int) -> np.ndarray:
    x_k = np.zeros(b.shape)
    r_k = b - (A @ x_k)
    p_k = r_k
    for k in range(steps):
         _k = (p_k @ r_k) / (p_k @ A @ p_k)
        x_k = x_k + ( _k * p_k)      # x_k+1
        r_k = r_k -  _k * (A @ p_k) # r_k+1
        if not np.any(r_k) or :          # stop if r_k+1 = 0
            break
         _k = ((A @ p_k) @ r_k) / ((A @ p_k) @ p_k)
        p_k = r_k - ( _k * p_k)
    return x_k

def poisson_mat(n:int, m : int =None) -> np.ndarray:
    return 2 * np.eye(n, m) + (-1) * np.eye(n, m, k=1) + (-1) * np.eye(n, m,␣
 ↪k=-1)
```

[34]:
```
for n in range(3, 21):
    Q = poisson_mat(n)
    b = np.ones(n)
    x = np.linalg.inv(Q) @ b
    x_k = conjugate_gradient(Q, b, 10)
    testing.assert_allclose(x_k, x, rtol=1e-5, err_msg=f'CG failed at dim -␣
 ↪{n}')
```

What happends to a matrix that is not positive definite? Consider the system

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \tag{1}$$

Because A is indefinite, the coefficients α_k are undefined, i.e. there is an $r^k \neq 0$ such that $(r^k)^T A r^k = 0$.

[35]:
```
A = np.array([[1, 0 , 0], [0, 1, 0], [0, 0, -2]])
b = np.ones(3)
x_k = conjugate_gradient(A, b , 10)
testing.assert_allclose(x_k, b, rtol=1e-5 ,err_msg=f'CG failed for non-definite␣
 ↪matrix')
```

```
/tmp/ipykernel_217820/1775264609.py:6: RuntimeWarning: divide by zero
encountered in double_scalars
  _k = (p_k @ r_k) / (p_k @ A @ p_k)
/tmp/ipykernel_217820/1775264609.py:12: RuntimeWarning: invalid value
encountered in subtract
  p_k = r_k - ( _k * p_k)
/tmp/ipykernel_217820/1775264609.py:6: RuntimeWarning: invalid value encountered
in matmul
  _k = (p_k @ r_k) / (p_k @ A @ p_k)
/tmp/ipykernel_217820/1775264609.py:8: RuntimeWarning: invalid value encountered
in matmul
  r_k = r_k -  _k * (A @ p_k) # r_k+1
/tmp/ipykernel_217820/1775264609.py:11: RuntimeWarning: invalid value
encountered in matmul
  _k = ((A @ p_k) @ r_k) / ((A @ p_k) @ p_k)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Input In [35], in <cell line: 4>()
      2 b = np.ones(3)
      3 x_k = conjugate_gradient(A, b , 10)
----> 4␣
 ↪testing.assert_allclose(x_k, b, rtol=1e-5 ,err_msg=f'CG failed for non-defini e matrix')

    [… skipping hidden 2 frame]

File ~/.local/lib/python3.10/site-packages/numpy/testing/_private/utils.py:745,
 ↪in assert_array_compare.<locals>.func_assert_same_pos(x, y, func, hasval)
    740 if bool_(x_id == y_id).all() != True:
    741     msg = build_err_msg([x, y],
    742                         err_msg + '\nx and y %s location mismatch:'
    743                         % (hasval), verbose=verbose, header=header,
    744                         names=('x', 'y'), precision=precision)
--> 745     raise AssertionError(msg)
    746 # If there is a scalar, then here we know the array has the same
    747 # flag as it everywhere, so we should return the scalar flag.
    748 if isinstance(x_id, bool) or x_id.ndim == 0:

AssertionError:
```

```
Not equal to tolerance rtol=1e-05, atol=0
CG failed for non-definite matrix
x and y nan location mismatch:
 x: array([nan, nan, nan])
 y: array([1., 1., 1.])
```

2 Sheet 4, Exercise 4 1) check

[36]:
```
A = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])
b = np.array([4, 0, 0])
x = conjugate_gradient(A, b, 10)
print(x)
```

```
[3. 2. 1.]
```

[ ]:
```
np.any(x_k)
```

[ ]:

[ ]: