hw4_code

March 10, 2021

[4]:
```
# reference code for HW4 P1
import numpy as np
import numpy.linalg as la
# set random seed for generating matrices
np.random.seed(10)
```

[141]:
```
def conj_grad(A, b, x0, n=100, tol=1e-4):
    # (7.31) from B&F
    """ conjugate gradient method without preconditioning
    A   (numpy.ndarray)    (N x N) matrix
    b   (numpy.ndarray)    (N x 1) vector
    x0  (numpy.ndarray)    (N x 1) initial guess
    n   (int)              number of iterations allowed
    tol (float)            tolerance, default 1e-6
    """
    k = 0
    r_new = b - A@x0
    xk = x0
    vk = r_new
    r_old = r_new
    all_err = np.array([])
    while k <= n:
        k += 1
        if la.norm(r_new) < tol:
            return xk, all_err
        else:
            all_err = np.append(all_err, la.norm(r_new, float('inf')))
            tk = r_old.dot(r_old)/vk.dot(A@vk)
            r_new = r_new - tk*(A@vk)
            sk = r_new.dot(r_new)/r_old.dot(r_old)
            xk = xk + tk*vk
            vk = r_new + sk*vk
            r_old = r_new
    raise FloatingPointError("CG could not converge")
```

[142]:
```
A = np.array([
    [4, 1],
```

```
    [1, 3]
])
b = np.array([1,2])
x0 = np.array([2, 1])
x_exact = la.solve(A, b)
x_cg, all_err = conj_grad(A, b, x0)
```

[143]:
```
all_err
```

[143]:
```
array([8.        , 0.74924471])
```

[144]:
```
# exact
la.inv(A)@b
```

[144]:
```
array([0.09090909, 0.63636364])
```

[145]:
```
# CG should converge in 2 steps, as shown in len(all_err)
x_cg
```

[145]:
```
array([0.09090909, 0.63636364])
```