https://arxiv.org/pdf/1610.05708

$$f(x) = \frac{1}{4}\|Ex\|_2^2 + \frac{1}{4}\|Ax-b\|_4^4 + \frac{1}{2}\|Cx-d\|_2^2$$

is relatively smooth and strongly convex with

$$h(x) = \frac{1}{4}\|x\|_2^4 + \frac{1}{2}\|x\|_2^2$$

Is

$$f_1(x,y) = f(x) - f(y) + x^\top By$$

relative strongly monotone

$$h_1(x,y) = h(x) - h(y)?$$

Then

$$\nabla^2 f(x) = \|Ex\|_2^2 E^\top E + 2E^\top Exx^\top E^\top E + 3A^\top D(x)^2 A + C^\top C; D(x) = Diag(Ax-b)$$

we have

$$L\nabla^2 h(x) \succeq \nabla^2 f(x) \succeq \mu\nabla^2 h(x)$$

Thus we also have,

$$-L\nabla^2 h(y) \preceq -\nabla^2 f(y) \preceq -\mu\nabla^2 h(y)$$

Now we have,

$$\frac{\nabla F(x,y) + \nabla F(x,y)^\top}{2} = \begin{bmatrix} \nabla^2 f(x) & 0 \\ 0 & -\nabla^2 f(y) \end{bmatrix}$$

$$\frac{\nabla H(x,y) + \nabla H(x,y)^\top}{2} = \begin{bmatrix} \nabla^2 h(x) & 0 \\ 0 & -\nabla^2 h(y) \end{bmatrix}$$

where

$$F(x,y) = (\nabla f(x), -\nabla f(y)) \; and \; H(x,y) = (\nabla h(x), -\nabla h(y)).$$

$$\frac{\nabla F(x,y) + \nabla F(x,y)^\top}{2} - \mu\frac{\nabla H(x,y) + \nabla H(x,y)^\top}{2}$$
$$= \begin{bmatrix} \nabla^2 f(x) - \mu\nabla^2 h(y) & 0 \\ 0 & \mu\nabla^2 h(y) - \nabla^2 f(y) \end{bmatrix}$$

since diagonal blocks are PSD and NSD we have that F is μ-strongly monotone with respect to H.

Similarly we find that F is L relatively smooth with respect to H. By observation a bilinear term $x^\top Ay$ can also be added.

$$z_{k+\frac{1}{2}} = z' \text{ s.t.,}$$

```
    $$\langle{F(z_k)+L(H(z')-H(z_k)),z'-z\rangle}\leq 0~ \forall z$$
    $$z_{k+1} = z' \ \textrm{s.t.,}\ $$
    $$\langle F(z_{k+\frac{1}{2}})+L(H(z')-H(z_k))+m(H(z')-
H(z_{k+\frac{1}{2}})),z'-z \rangle\leq 0~ \forall z$$
```

We have,

$$H(x,y) = ((\|x\|^2 x + x), -(\|y\|^2 y + y))$$

and

$$F(x,y) = ((\|Ex\|^2 E^\top Ex + A^\top[(Ax-b)_i^3]_{i=1}^n) + (Cx-d) + By,$$
$$-(\|Ey\|^2 E^\top Ey + A^\top[(Ay-b)_i^3]_{i=1}^n) + (Cy-d) + B^\top x)$$

.

First do mirror prox on :

$$F(z_k) + L(H(z') - H(z_k)) = O_k(z')$$

output

$$z_{k+\frac{1}{2}}$$

and do mirror prox on,

$$F(z_{k+\frac{1}{2}}) + L(H(z') - H(z_k)) + m(H(z') - H(z_{k+\frac{1}{2}})) = O_{k+\frac{1}{2}}(z')$$

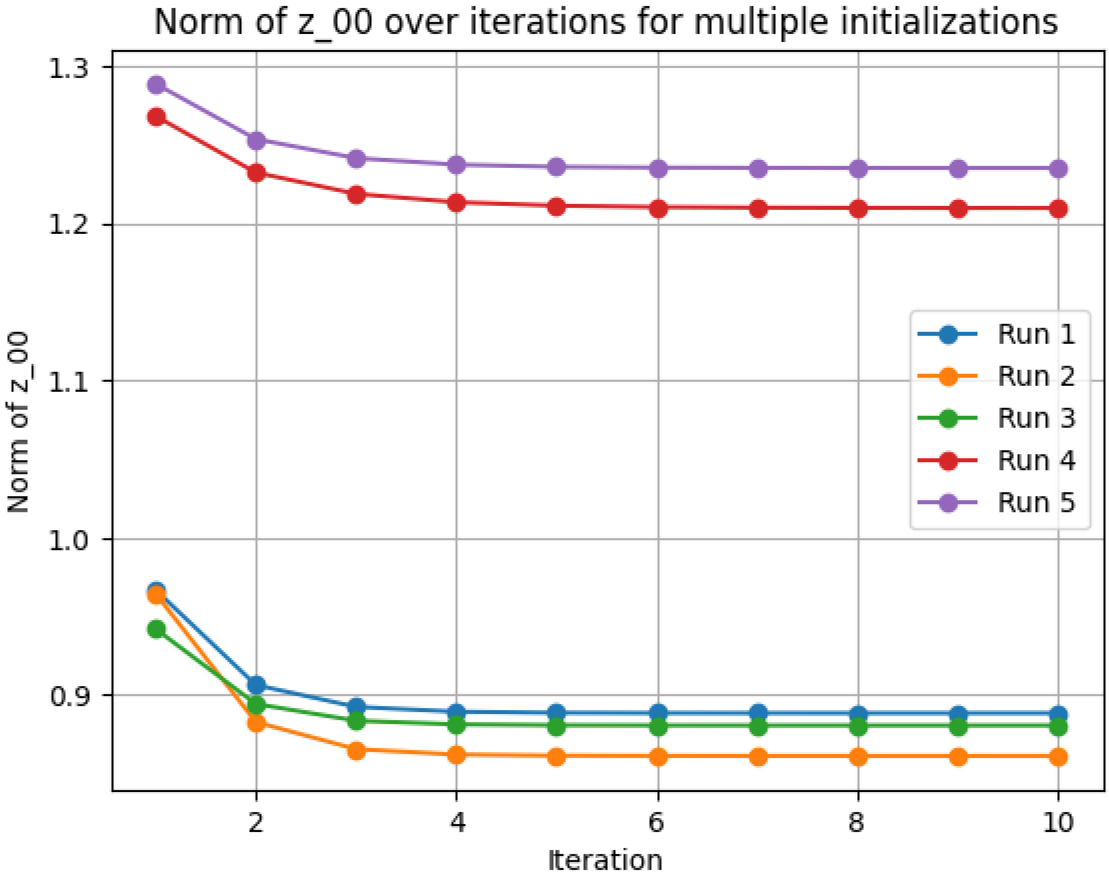
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

# Constants
matrixSize = 3
L = 10
m = 0.1
Lp = 500
inner_iter = 3
outer_iter = 10
num_runs = 5

# Random matrices
def create_sym_matrix(size):
    mat = np.random.rand(size, size)
    return np.dot(mat, mat.T)

E = create_sym_matrix(matrixSize)
C = create_sym_matrix(matrixSize)
B = np.random.rand(matrixSize, matrixSize)
B = np.zeros((matrixSize, matrixSize))
A = np.random.rand(matrixSize, matrixSize)
b = np.random.rand(matrixSize, 1)
d = np.random.rand(matrixSize, 1)
```

```python
def mirrorfreeMP(z_0, z_00):
    norm_z_00 = []
    for outer in range(outer_iter):
        z_12 = MP1(z_00, z_0)
        z_00 = MP2(z_00, z_12, z_0)
        norm_z_00.append(np.linalg.norm(z_00))
        print(f"After outer iteration {outer + 1}: z_00 norm = {np.linalg.norm(z_00)}")
    return z_00, norm_z_00

def MP1(z_k, z_0):
    for _ in range(inner_iter):
        z_half = z_0 - oracleOk(z_0, z_k) / Lp
        z_0 = z_half - oracleOk(z_half, z_k) / Lp
    return z_0

def MP2(z_k, z_half, z_0):
    for _ in range(inner_iter):
        z_half = z_0 - oracleOk12(z_0, z_half, z_k) / Lp
        z_0 = z_half - oracleOk12(z_0, z_half, z_k) / Lp
    return z_0

def oracleOk(zp, zk):
    term1 = oracleF(zk)
    term2 = L * (oracleH(zp) - oracleH(zk))
    return term1 + term2

def oracleOk12(zp, zk12, zk):
    term1 = oracleF(zk12)
    term2 = L * (oracleH(zp) - oracleH(zk))
    term3 = m * (oracleH(zp) - oracleH(zk12))
    return term1 + term2 + term3

def oracleF(z):
    x, y = z[:matrixSize], z[matrixSize:]
    gx = (np.linalg.norm(np.dot(E, x))**2 * np.dot(E, np.dot(E, x)) +
          np.dot(A, (np.dot(A, x) - b)**3) + np.dot(C, x) - d + np.dot(B, y))
    gy = -(np.linalg.norm(np.dot(E, y))**2 * np.dot(E, np.dot(E, y)) +
            np.dot(A, (np.dot(A, y) - b)**3) + np.dot(C, y) - d + np.dot(B.T, x))
    return np.concatenate([gx, -gy])

def oracleH(z):
    (x, y) = (z[:matrixSize], z[matrixSize:])
    gx = np.linalg.norm(x)**2 * x + x
    gy = -np.linalg.norm(y)**2 * y - y
    result = np.concatenate([gx, -gy])
    return result

# Running the algorithm for multiple initializations
all_norms = []

for run in range(num_runs):
    z = np.random.rand(2 * matrixSize, 1)
    result, norm_z_00 = mirrorfreeMP(z, z)
    all_norms.append(norm_z_00)
    print(f"Run {run + 1} complete")

# Plotting the norm of z_00 for each run
for i, norms in enumerate(all_norms):
    plt.plot(range(1, outer_iter + 1), norms, marker='o', label=f'Run {i + 1}')

plt.title('Norm of z_00 over iterations for multiple initializations')
plt.xlabel('Iteration')
plt.ylabel('Norm of z_00')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

```
After outer iteration 1: z_00 norm = 0.9668677606687373
After outer iteration 2: z_00 norm = 0.9063465383763782
After outer iteration 3: z_00 norm = 0.8924321749839715
After outer iteration 4: z_00 norm = 0.8893404907204661
After outer iteration 5: z_00 norm = 0.8886538112309523
After outer iteration 6: z_00 norm = 0.8885003740121772
After outer iteration 7: z_00 norm = 0.8884658863585999
After outer iteration 8: z_00 norm = 0.8884580985283924
After outer iteration 9: z_00 norm = 0.8884563337650722
After outer iteration 10: z_00 norm = 0.8884559328221885
Run 1 complete
After outer iteration 1: z_00 norm = 0.9644617317839554
After outer iteration 2: z_00 norm = 0.8827477645598474
After outer iteration 3: z_00 norm = 0.8654982989666862
After outer iteration 4: z_00 norm = 0.8620501885815155
After outer iteration 5: z_00 norm = 0.8613648112967649
After outer iteration 6: z_00 norm = 0.8612280185133416
After outer iteration 7: z_00 norm = 0.8612005782554117
After outer iteration 8: z_00 norm = 0.8611950497119978
After outer iteration 9: z_00 norm = 0.8611939318941235
After outer iteration 10: z_00 norm = 0.8611937052428492
Run 2 complete
After outer iteration 1: z_00 norm = 0.9420531474560809
After outer iteration 2: z_00 norm = 0.8942737555755859
After outer iteration 3: z_00 norm = 0.8835962268252221
After outer iteration 4: z_00 norm = 0.8812743449164225
After outer iteration 5: z_00 norm = 0.8807696120824753
After outer iteration 6: z_00 norm = 0.8806593875261732
After outer iteration 7: z_00 norm = 0.8806352086621023
After outer iteration 8: z_00 norm = 0.8806298860207312
After outer iteration 9: z_00 norm = 0.8806287111888103
After outer iteration 10: z_00 norm = 0.8806284513579092
Run 3 complete
After outer iteration 1: z_00 norm = 1.26831703608505
After outer iteration 2: z_00 norm = 1.232361718586752
After outer iteration 3: z_00 norm = 1.2188247589572923
After outer iteration 4: z_00 norm = 1.2134757884761167
After outer iteration 5: z_00 norm = 1.2113041284878063
After outer iteration 6: z_00 norm = 1.2104096531062292
After outer iteration 7: z_00 norm = 1.2100385181525934
After outer iteration 8: z_00 norm = 1.2098839641080188
After outer iteration 9: z_00 norm = 1.209819486749358
After outer iteration 10: z_00 norm = 1.2097925643539968
Run 4 complete
After outer iteration 1: z_00 norm = 1.2888977938959656
After outer iteration 2: z_00 norm = 1.2536708047879106
After outer iteration 3: z_00 norm = 1.2415675796290975
After outer iteration 4: z_00 norm = 1.2374358312769418
After outer iteration 5: z_00 norm = 1.2360237060580672
After outer iteration 6: z_00 norm = 1.2355395859468439
After outer iteration 7: z_00 norm = 1.2353731568502775
After outer iteration 8: z_00 norm = 1.2353158255362975
After outer iteration 9: z_00 norm = 1.2352960473215315
After outer iteration 10: z_00 norm = 1.2352892170003626
Run 5 complete
```



Norm of z_00 over iterations for multiple initializations
Norm of z_00
Iteration
1.3
1.2
1.1
1.0
0.9
2
4
6
8
10
Run 1
Run 2
Run 3
Run 4
Run 5