## CP.2.7.5.a, Sauer3

Use multivariate Newton's Method to find the two points in common of the three given spheres in three-dimensional space.

a. Each sphere has radius 1, with centers $\begin{pmatrix} 1, & 1, & 0 \end{pmatrix}$, $\begin{pmatrix} 1, & 0, & 1 \end{pmatrix}$, and $\begin{pmatrix} 0, & 1, & 1 \end{pmatrix}$.

The answers are $\begin{pmatrix} 1, & 1, & 1 \end{pmatrix}$ and $\begin{pmatrix} \frac{1}{3}, & \frac{1}{3}, & \frac{1}{3} \end{pmatrix}$.



## CP.2.7.5.a, Sauer3, solution, Langou

Colab: `https://colab.research.google.com/drive/1zxXgVys2AVjzDmmWcek6XDOHchHJGssY`

We want points $(x_0, x_1, x_2)$ that are on the three spheres.

a. For $(x_0, x_1, x_2)$ to be on the sphere of radius 1 with center $\begin{pmatrix} 1, & 1, & 0 \end{pmatrix}$, we have

$$(x_0 - 1)^2 + (x_1 - 1)^2 + x_2^2 = 1^2.$$

So we have

$$(x_0 - 1)^2 + (x_1 - 1)^2 + x_2^2 - 1 = 0.$$

b. For $(x_0, x_1, x_2)$ to be on the sphere of radius 1 with center $\begin{pmatrix} 1, & 0, & 1 \end{pmatrix}$, we have

$$(x_0 - 1)^2 + x_1^2 + (x_2 - 1)^2 = 1^2.$$

So we have

$$(x_0 - 1)^2 + x_1^2 + (x_2 - 1)^2 - 1 = 0.$$

c. For $(x_0, x_1, x_2)$ to be on the sphere of radius 1 with center $\begin{pmatrix} 0, & 1, & 1 \end{pmatrix}$, we have

$$x_0^2 + (x_1 - 1)^2 + (x_2 - 1)^2 = 1^2.$$

So we have

$$x_0^2 + (x_1 - 1)^2 + (x_2 - 1)^2 - 1 = 0.$$

So, if we define

$$\left( \begin{array}{cccc} F: & \mathbb{R}^3 & \to & \mathbb{R}^3 \\ & \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} & \mapsto & \begin{pmatrix} (x_0 - 1)^2 + (x_1 - 1)^2 + x_2^2 - 1 \\ (x_0 - 1)^2 + x_1^2 + (x_2 - 1)^2 - 1 \\ x_0^2 + (x_1 - 1)^2 + (x_2 - 1)^2 - 1 \end{pmatrix} \end{array} \right)$$

We see that finding the two intersections of the three spheres is the same as finding the roots of $F$.

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \text{ is on the three spheres } \iff F(\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

We will use Newton's method to find the two roots of $F$. (Which are the two intersections of the three spheres.)

To use Newton's method, we need the Jacobian of $F$. The Jacobian of $F$ is

$$DF(\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} 2(x_0-1) & (x_1-1) & 2x_2 \\ 2(x_0-1) & 2x_1 & 2(x_2-1) \\ 2x_0 & 2(x_1-1) & 2(x_2-1) \end{pmatrix}$$

And, from there, we pretty much need to

a. set an initial guess, for example `x = np.array( [ 0, -0.2, 0.3] )`,

b. iterate with Newton's method scheme

$$x \longleftarrow x - (DF(x))^{-1}(F(x))$$

In Python: `x = x - np.linalg.solve( DF(x), F(x) )`.

c. monitor convergence with $\|F(x)\|$. In Python: `np.linalg.norm( F(x), np.infty)`.

```
import numpy as np
import scipy
from scipy.optimize import fsolve
```

```
F = lambda x : np.array( [
 ( x[0] - 1. )**2 + ( x[1] - 1. )**2  + ( x[2]      )**2  - 1.,
 ( x[0] - 1. )**2 + ( x[1]      )**2  + ( x[2] - 1. )**2  - 1.,
 ( x[0] - 0. )**2 + ( x[1] - 1. )**2  + ( x[2] - 1. )**2  - 1. ] )
```

```
# Per Sauer, we have two exact solution x1 and 2 such that
x1 = np.array( [ 1., 1., 1. ] )
x2 = np.array( [ 1./3, 1./3., 1./3. ] )

# Let us check that || F(x1) ||_oo and || F(x2) ||_oo are small
print( "|| F(x1) ||_oo =", f"{np.linalg.norm(F(x1),np.infty):.2E}" )
print( "|| F(x2) ||_oo =", f"{np.linalg.norm(F(x2),np.infty):.2E}" )
```

```
|| F(x1) ||_oo = 0.00E+00
|| F(x2) ||_oo = 2.22E-16
```

```
# using scipy.optimize.fsolve to find two approximate solutions

# take the initial guess (0,0,0)
# we get an approximate solution, we call it x1

x = np.array( [ 0., 0., 0. ] )

x = scipy.optimize.fsolve( F, x )

print( "\nx = [", f"{x[0]:+20.16f}", "]")
print( "    [", f"{x[1]:+20.16f}", "]")
print( "    [", f"{x[2]:+20.16f}", "]")
print( "backward error: || F(x) ||_oo   = ",\
 f"{np.linalg.norm(F(x),np.infty):7.2e}")
```

```
x1 = x

# take another initial guess, so we take (4,-1,3) to be fancy
# we get an approximate solution, we call it x2
# !!! we make sure that x2 is not x1 !!!
# (if x2 is the same x1, pick anothe initial
# guess until you find something different)

x = np.array( [ 4., -1., 3. ] )

x = scipy.optimize.fsolve( F, x )

print( "\nx = [", f"{x[0]:+20.16f}", "]")
print( "    [", f"{x[1]:+20.16f}", "]")
print( "    [", f"{x[2]:+20.16f}", "]")
print( "backward error: || F(x) ||_oo   = ",\
 f"{np.linalg.norm(F(x),np.infty):7.2e}")

x2 = x
```

```
x = [  +0.3333333333333334 ]
    [  +0.3333333333333333 ]
    [  +0.3333333333333334 ]
backward error: || F(x) ||_oo   =  0.00e+00

x = [  +1.0000000000000000 ]
    [  +1.0000000000000000 ]
    [  +1.0000000000000000 ]
backward error: || F(x) ||_oo   =  0.00e+00
```

```
# this is Jacobian of F

DF = lambda x : np.array([
  [ 2. * ( x[0] - 1. ), 2. * ( x[0] - 1. ), 2. * x[2] ],
  [ 2. * ( x[0] - 1. ), 2. * x[1], 2. * ( x[2] - 1. ) ],
  [ 2. * x[0], 2. * ( x[0] - 1. ), 2. * ( x[1] - 1. ) ] ] )

for x, x_ in zip( [ np.array( [ 0, -0.2, 0.3 ] ),\
                    np.array( [ 0.1, 1.5, 1.5 ] ) ],
                  [ x1, x2 ] ):

  print( "**", "[", f"{x_[0]:+18.14f}", f"{x_[1]:+18.14f}",\
        f"{x_[2]:+18.14f}", "]", "*******", "*******" )

  print( f"{0:2d}", "[", f"{x[0]:+18.14f}", f"{x[1]:+18.14f}",\
         f"{x[2]:+18.14f}",\
       "]", f"{np.linalg.norm( x - x_, np.infty):4.1e}", \
      f"{np.linalg.norm( F(x), np.infty):4.1e}" )

# this is the Newton's method loop, 8 iterations should be enough
  for i in range(1,8):

#   this is the Newton's method update
```

```
    x = x - np.linalg.solve( DF(x), F(x) )

    print( f"{i:2d}", "[", f"{x[0]:+18.14f}",\
           f"{x[1]:+18.14f}", f"{x[2]:+18.14f}",\
           "]", f"{np.linalg.norm( x - x_, np.infty):4.1e}",\
          f"{np.linalg.norm( F(x), np.infty):4.1e}" )

  print( "\nx = [", f"{x[0]:+18.14f}", "]"\
        "      x* = [", f"{x_[0]:+18.14f}", "]")
  print( "    [", f"{x[1]:+18.14f}", "]"\
         "            [", f"{x_[1]:+18.14f}", "]")
  print( "    [", f"{x[2]:+18.14f}", "]"\
         "            [", f"{x_[2]:+18.14f}", "]")
  print( "forward  error: || x - x* ||_oo = ",\
 f"{np.linalg.norm(x-x_,np.infty):7.2e}",\
         "\nbackward error: || F(x) ||_oo    = ",\
 f"{np.linalg.norm(F(x),np.infty):7.2e}")
  print("\n")
```

```
** [  +0.33333333333333  +0.33333333333333  +0.33333333333333 ] *******  *******
 0 [  +0.00000000000000  -0.20000000000000  +0.30000000000000 ] 5.3e-01 1.5e+00
 1 [  +0.20204081632653  +0.34336734693878  +0.23469387755102 ] 1.3e-01 3.4e-01
 2 [  +0.31916154492549  +0.33118220523107  +0.32946012142850 ] 1.4e-02 2.3e-02
 3 [  +0.33321526225350  +0.33319751472520  +0.33322174007604 ] 1.4e-04 2.6e-04
 4 [  +0.33333331262528  +0.33333331233253  +0.33333330992246 ] 2.3e-08 4.5e-08
 5 [  +0.33333333333333  +0.33333333333333  +0.33333333333333 ] 8.3e-16 1.6e-15
 6 [  +0.33333333333333  +0.33333333333333  +0.33333333333333 ] 1.7e-16 0.0e+00
 7 [  +0.33333333333333  +0.33333333333333  +0.33333333333333 ] 1.7e-16 0.0e+00

x = [  +0.33333333333333 ]      x* = [  +0.33333333333333 ]
    [  +0.33333333333333 ]           [  +0.33333333333333 ]
    [  +0.33333333333333 ]           [  +0.33333333333333 ]
forward  error: || x - x* ||_oo =  1.67e-16
backward error: || F(x) ||_oo    =  0.00e+00


** [  +1.00000000000000  +1.00000000000000  +1.00000000000000 ] *******  *******
 0 [  +0.10000000000000  +1.50000000000000  +1.50000000000000 ] 9.0e-01 2.3e+00
 1 [  +1.96666666666667  +1.69444444444444  +1.96666666666667 ] 9.7e-01 4.3e+00
 2 [  +1.26266611750264  +1.30814430800942  +1.41330378959451 ] 4.1e-01 1.2e+00
 3 [  +1.04770781471528  +1.09231114986505  +1.08249544838128 ] 9.2e-02 2.0e-01
 4 [  +1.00456473370184  +1.00757186106901  +1.00379220617049 ] 7.6e-03 1.5e-02
 5 [  +1.00003728589643  +1.00004567195735  +1.00002303974810 ] 4.6e-05 9.1e-05
 6 [  +1.00000000214170  +1.00000000200329  +1.00000000162030 ] 2.1e-09 4.3e-09
 7 [  +1.00000000000000  +1.00000000000000  +1.00000000000000 ] 0.0e+00 0.0e+00

x = [  +1.00000000000000 ]      x* = [  +1.00000000000000 ]
    [  +1.00000000000000 ]           [  +1.00000000000000 ]
    [  +1.00000000000000 ]           [  +1.00000000000000 ]
forward  error: || x - x* ||_oo =  0.00e+00
backward error: || F(x) ||_oo    =  0.00e+00
```