Háskóli Íslands

STÆ405G

Töluleg greining

23.mars

Verkefni 2

Höfundar:
Anna Margrét
amb33@hi.is
Pétur Jökull
pth20@hi.is
Marinó Kristjánsson
mak78@hi.is

Kennari:
Birgir Hrafnkelsson

Pakkar

```python
import numpy as np
import numpy.linalg as lin
import itertools
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import math
```

Dæmi 1

Solve the system (4.37) by using Multivariate Newtons Method. Find the receiver position (x, y, z) near earth and time correction d for known, simultaneous satellite positions $(15600, 7540, 20140)$, $(18760, 2750, 18610)$, $(17610, 14630, 13480)$, $(19170, 610, 18390)$ in km, and measured time intervals $0.07074, 0.07220, 0.07690, 0.07242$ in seconds, respectively. Set the initial vector to be $(x_0, y_0, z_0, d_0) = (0, 0, 6370, 0)$. As a check, the answers are approximately $(x, y, z) = (-41.77271, -16.78919, 6370.0596)$, and $d = -3.201566 \times 10^{-3}$ seconds.

Lausn:

Fáum Multivariate Newtons Method gefna í bók:

> **Multivariate Newton's Method**
>
> $$\begin{aligned} x_0 &= \text{initial vector} \\ x_{k+1} &= x_k - (DF(x_k))^{-1} F(x_k) \quad \textbf{for } k = 0, 1, 2, \ldots. \end{aligned}$$

Í fyrirmælum er gefið að þetta gildi um rétta skurðpunktinn

$$
\begin{aligned}
r_1(x, y, z, d) &= \sqrt{(x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2} - c(t_1 - d) = 0 \\
r_2(x, y, z, d) &= \sqrt{(x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2} - c(t_2 - d) = 0 \\
r_3(x, y, z, d) &= \sqrt{(x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2} - c(t_3 - d) = 0 \\
r_4(x, y, z, d) &= \sqrt{(x - A_4)^2 + (y - B_4)^2 + (z - C_4)^2} - c(t_4 - d) = 0 \qquad (4.37)
\end{aligned}
$$

Skilgreining fylkisins F(x)

$$\begin{bmatrix} (x-A_1)^2+(y-B_1)^2+(z-C_1)^2-c^2(r_4-d)^2 \\ (x-A_2)^2+(y-B_2)^2+(z-C_2)^2-c^2(r_4-d)^2 \\ (x-A_3)^2+(y-B_3)^2+(z-C_3)^2-c^2(r_4-d)^2 \\ (x-A_4)^2+(y-B_4)^2+(z-C_4)^2-c^2(r_4-d)^2 \end{bmatrix}$$

Útleiðsla á DF(x)

Við staðsetningar gervihnettina sem

$r_1 = [A_1, B_1, C_1]$
$r_2 = [A_2, B_2, C_2]$
$r_3 = [A_3, B_3, C_3]$
$r_4 = [A_4, B_4, C_4]$
þá er DF(x) Jacobi fylki F(x)

$$\begin{bmatrix} (x-A_1)*2+(y-B_1)*22+(z-C_1)*2-2*c^2(r_4-d) \\ (x-A_2)*2+(y-B_2)*2+(z-C_2)*2-2*c^2(r_4-d) \\ (x-A_3)*2+(y-B_3)*2+(z-C_3)*2-2*c^2(r_4-d) \\ (x-A_4)*2+(y-B_4)*2+(z-C_4)*2-2*c^2(r_4-d) \end{bmatrix}$$

Leysum svo verkefnið í python með 10 ýtrunum:

Kóði fyrir dæmi 1

```
r1 = {'x':15600, 'y':7540 ,'z':20140,'d':0.07074 }#Gervitungl 1
r2 = {'x':18760, 'y':2750 ,'z':18610,'d':0.07220 }#Gervitungl 2
r3 = {'x':17610, 'y':14630,'z':13480,'d':0.07690 }#Gervitungl 3
r4 = {'x':19170, 'y':610  ,'z':18390,'d':0.07242 }#Gervitungl 4
r0 = {'x':0,'y':0,'z':6370,'d':0} #upphafsvigur
```

```python
c=299792.458 #ljóshraði km/s

for i in range(10):
 F = np.array([
 [(r0['x'] - r1['x'])**2 + (r0['y'] - r1['y'])**2 + (r0['z'] - r1['z'])**2 - (c*(r1['d'] - r0['d']))**2],
 [(r0['x'] - r2['x'])**2 + (r0['y'] - r2['y'])**2 + (r0['z'] - r2['z'])**2 - (c*(r2['d'] - r0['d']))**2],
 [(r0['x'] - r3['x'])**2 + (r0['y'] - r3['y'])**2 + (r0['z'] - r3['z'])**2 - (c*(r3['d'] - r0['d']))**2],
 [(r0['x'] - r4['x'])**2 + (r0['y'] - r4['y'])**2 + (r0['z'] - r4['z'])**2 - (c*(r4['d'] - r0['d']))**2]
 ])

 DF = np.array([
 [(r0['x'] - r1['x'])*2 , (r0['y'] - r1['y'])*2 , (r0['z'] - r1['z'])*2 , (2*c**2)*(r1['d'] - r0['d'])],
 [(r0['x'] - r2['x'])*2 , (r0['y'] - r2['y'])*2 , (r0['z'] - r2['z'])*2 , (2*c**2)*(r2['d'] - r0['d'])],
 [(r0['x'] - r3['x'])*2 , (r0['y'] - r3['y'])*2 , (r0['z'] - r3['z'])*2 , (2*c**2)*(r3['d'] - r0['d'])],
 [(r0['x'] - r4['x'])*2 , (r0['y'] - r4['y'])*2 , (r0['z'] - r4['z'])*2 , (2*c**2)*(r4['d'] - r0['d'])]
 ])


 res = lin.solve(DF,F)
 r0['x'] = r0['x'] - res[0][0]
 r0['y'] = r0['y'] - res[1][0]
 r0['z'] = r0['z'] - res[2][0]
 r0['d'] = r0['d'] - res[3][0]
print(r0)
```

```
C:\Users\Dev\Documents\hi\toluleg_greining\verkefni2>python d1.py
{'x': -41.772709570873225, 'y': -16.78919410653207, 'z': 6370.05955922334, 'd': -0.0032015658295942427}
```

Mynd 1: Reiknuð staðsetning auk tímamismuns er sú sama og gefin er í dæminu

Dæmi 2

Write a MATLAB program to carry out the solution via the quadratic formula. Hint: Subtracting the last three equations of (4.37) from the first yields three linear equations in the four unknowns $x\vec{u}_x + y\vec{u}_y + z\vec{u}_z + d\vec{u}_d + \vec{w} = 0$, expressed in vector form. A formula for x in terms of d can be obtained from

$$0 = \det[\vec{u}_y|\vec{u}_z|x\vec{u}_x + y\vec{u}_y + z\vec{u}_z + d\vec{u}_d + \vec{w}],$$

noting that the determinant is linear in its columns and that a matrix with a repeated column has determinant zero. Similarly, we can arrive at formulas for y and z, respectively, in terms of d, that can be substituted in the first quadratic equation of (4.37), to make it an equation in one variable.

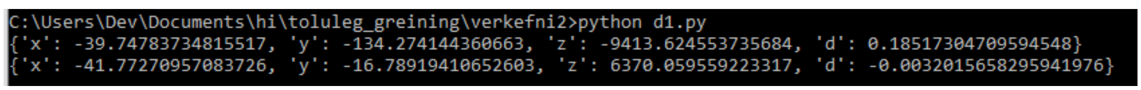
Lausn:

Fylgjum ábendingunum sem eru gefin í dæminu og drögum seinni þrjár jöfnunar frá fyrstu og fáum þannig þrjár línulegar jöfnur. Kóðann fyrir dæmi 2 má finna á næstu blaðsíðu.

Kóði fyrir dæmi 2

```
def daemi2(r0,r1,r2,r3,r4,c):
 ux = [2*(r2['x']-r1['x']),2*(r3['x']-r1['x']),2*(r4['x']-r1['x'])]
 uy = [2*(r2['y']-r1['y']),2*(r3['y']-r1['y']),2*(r4['y']-r1['y'])]
 uz = [2*(r2['z']-r1['z']),2*(r3['z']-r1['z']),2*(r4['z']-r1['z'])]
 ud = [2*(c**2)*(r1['d']-r2['d']),2*(c**2)*(r1['d']-r3['d']),2*(c**2)*(r1['d']-r4['

 w = [
 (r1['x']**2 - r2['x']**2)+(r1['y']**2 - r2['y']**2)+(r1['z']**2 - r2['z']**2) - (c**2)*(r1['d']**2 - r2['d']**2),
 (r1['x']**2 - r3['x']**2)+(r1['y']**2 - r3['y']**2)+(r1['z']**2 - r3['z']**2) - (c**2)*(r1['d']**2 - r3['d']**2),
 (r1['x']**2 - r4['x']**2)+(r1['y']**2 - r4['y']**2)+(r1['z']**2 - r4['z']**2) - (c**2)*(r1['d']**2 - r4['d']**2)
 ]

 s1 = - (lin.det([uy,uz,ud])/lin.det([uy,uz,ux]))
 s2 = (lin.det([uy,uz,w])/lin.det([uy,uz,ux]))
 s3 = - (lin.det([ux,uz,ud])/lin.det([ux,uz,uy]))
 s4 = (lin.det([ux,uz,w])/lin.det([ux,uz,uy]))
 s5 = - (lin.det([ux,uy,ud])/lin.det([ux,uy,uz]))
 s6 = (lin.det([ux,uy,w])/lin.det([ux,uy,uz]))

 p1 = (s1**2 + s3**2 + s5**2 - c**2)
 p2 =  2*((c**2)*r1['d'] - s1*(s2+r1['x']) - s3*(s4+r1['y']) - s5*(s6+r1['z']))
 p3 = (s2 + r1['x'])**2 + (s4 + r1['y'])**2 + (s6 + r1['z'])**2 - (c*r1['d'])**2

```

```python
    d = np.roots([p1,p2,p3])

    x = d*s1 - s2
    y = d*s3 - s4
    z = d*s5 - s6

    svar1 = {'x':x[0],'y':y[0],'z':z[0],'d':d[0]}
    svar2 = {'x':x[1],'y':y[1],'z':z[1],'d':d[1]}

    return svar1,svar2

svar1,svar2 = daemi2(r0,r1,r2,r3,r4,c)

print(svar1)
print(svar2)
```



```
C:\Users\Dev\Documents\hi\toluleg_greining\verkefni2>python d1.py
{'x': -39.74783734815517, 'y': -134.274144360663, 'z': -9413.624553735684, 'd': 0.18517304709594548}
{'x': -41.77270957083726, 'y': -16.78919410652603, 'z': 6370.059559223317, 'd': -0.0032015658295941976}
```

Mynd 2: Keyrsla á dæmi 2

Dæmi 3

Lausn:

```python
from sympy import symbols , Eq , solve
import numpy as np

c = 299792.458  # Ljoshradi km/s
r1 = {'x':15600,'y':7540 ,'z':20140,'d':0.07074 }#Gervitungl 1
r2 = {'x':18760,'y':2750 ,'z':18610,'d':0.07220 }#Gervitungl 2
r3 = {'x':17610,'y':14630,'z':13480,'d':0.07690 }#Gervitungl 3
r4 = {'x':19170,'y':610  ,'z':18390,'d':0.07242 }#Gervitungl 4
r0 = {'x':0,'y':0,'z':6370,'d':0} #upphafsvigur

x,y,z,d = symbols('x y z d')

eq1 = Eq((x-r1['x'])**2 + (y-r1['y'])**2 + (z-r1['z'])**2 - (c*(r1['d'] - d))**2, 0)
eq2 = Eq((x-r2['x'])**2 + (y-r2['y'])**2 + (z-r2['z'])**2 - (c*(r2['d'] - d))**2, 0)
eq3 = Eq((x-r3['x'])**2 + (y-r3['y'])**2 + (z-r3['z'])**2 - (c*(r3['d'] - d))**2, 0)
eq4 = Eq((x-r4['x'])**2 + (y-r4['y'])**2 + (z-r4['z'])**2 - (c*(r4['d'] - d))**2, 0)

sol = solve((eq1,eq2,eq3,eq4),(x,y,z,d))

for s in sol:
 print(s)
```

```
C:\Users\Dev\Documents\GitHub\tolgrVerkefni2>python d3.py
(-41.7727095708173, -16.7891941065185, 6370.05955922335, -0.00320156582959409)
(-39.7478373482208, -134.274144360683, -9413.62455373582, 0.185173047095946)
```

Dæmi 4

Now set up a test of the conditioning of the GPS problem. Define satellite positions (A_i, B_i, C_i) from spherical coordinates (ρ, ϕ_i, θ_i) as

$$\begin{aligned} A_i &= \rho \cos\phi_i \cos\theta_i \\ B_i &= \rho \cos\phi_i \sin\theta_i \\ C_i &= \rho \sin\phi_i, \end{aligned}$$

where $\rho = 26570$ km is fixed, while $0 \le \phi_i \le \pi/2$ and $0 \le \theta_i \le 2\pi$ for $i = 1, \ldots, 4$ are chosen arbitrarily. The ϕ coordinate is restricted so that the four satellites are in the upper hemisphere. Set $x = 0$, $y = 0$, $z = 6370$, $d = 0.0001$, and calculate the corresponding satellite ranges $R_i = \sqrt{A_i^2 + B_i^2 + (C_i - 6370)^2}$ and travel times $t_i = d + R_i/c$.

We will define an error magnification factor specially tailored to the situation. The atomic clocks aboard the satellites are correct up to about 10 nanoseconds, or 10^{-8} second. Therefore, it is important to study the effect of changes in the transmission time of this magnitude. Let the backward, or input error be the input change in meters. At the speed of light, $\Delta t_i = 10^{-8}$ second corresponds to $10^{-8}c \approx 3$ meters. Let the forward, or output error be the change in position $||(\Delta x, \Delta y, \Delta z)||_\infty$, caused by such a change in t_i, also in meters. Then we can define the dimensionless

$$\text{error magnification factor} = \frac{||(\Delta x, \Delta y, \Delta z)||_\infty}{c||(\Delta t_1, \ldots, \Delta t_m)||_\infty},$$

and the condition number of the problem to be the maximum error magnification factor for all small Δt_i (say, 10^{-8} or less).

Change each t_i defined in the foregoing by $\Delta t_i = +10^{-8}$ or -10^{-8}, not all the same. Denote the new solution of the equations (4.37) by $(\overline{x}, \overline{y}, \overline{z}, \overline{d})$, and compute the difference in position $||(\Delta x, \Delta y, \Delta z)||_\infty$ and the error magnification factor. Try different variations of the Δt_i's. What is the maximum position error found, in meters? Estimate the condition number of the problem, on the basis of the error magnification factors you have computed.

Lausn:

Við veljum sjálf gildin á θ_i og ρ_i þar sem ρ_i skal vera í efra hálfhveli en $\phi = 26570$ km er fast. Veljum

$$\overrightarrow{\theta} = (0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{4})^T$$

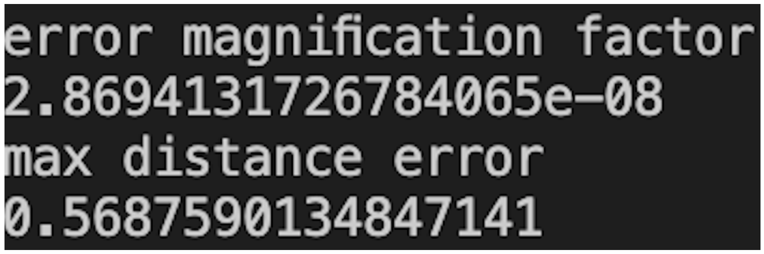
$$\overrightarrow{\rho} = (0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})^T$$

Kóði fyrir dæmi 4

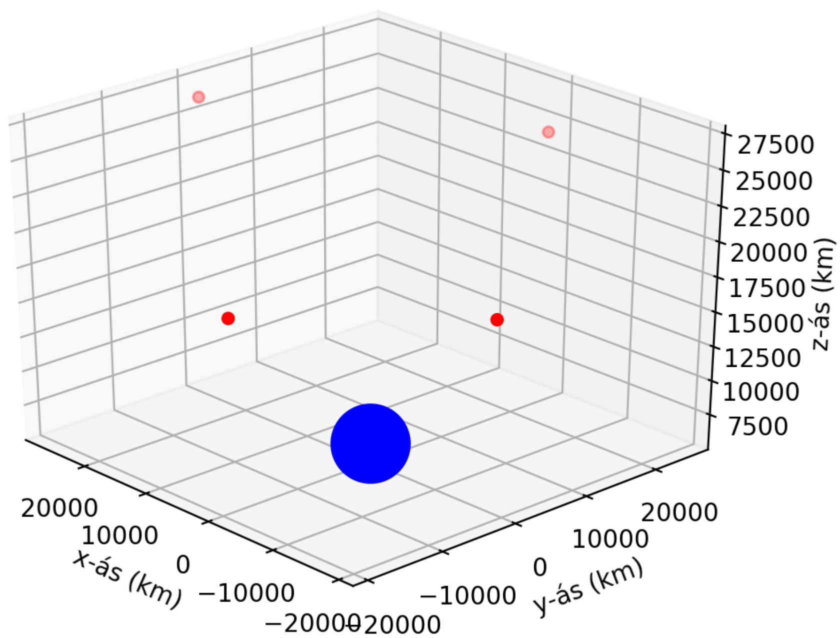
```python
c = 299792.458  # Ljoshradi km/s
r1 = [15600, 7540, 20140, 0.07074]  # Gervitungl 1
r2 = [18760, 2750, 18610, 0.07220]  # Gervitungl 2
r3 = [17610, 14630, 13480, 0.07690]  # Gervitungl 3
r4 = [19170, 610, 18390, 0.07242]  # Gervitungl 4
r0 = [0, 0, 6370, 0.0001]  # hnit mottakara
rho = 26570  # km
phi = [0, math.pi/8, math.pi/4, math.pi/4]
theta = [0, math.pi/2, math.pi, 3*math.pi/2]
R = np.zeros(4)
t_m = np.zeros(4)
A = [r1[0], r2[0], r3[0], r4[0]]
B = [r1[0], r2[1], r3[2], r4[3]]
C = [r1[0], r2[1], r3[2], r4[3]]
t_m = [r1[3], r2[3], r3[3], r4[3]]

#Reikna hnit gervihnattanna
for i in range(4):
    A[i] = rho*math.cos(phi[i])*math.cos(theta[i])
    B[i] = rho*math.cos(phi[i])*math.sin(theta[i])
    C[i] = rho*math.cos(phi[i])
    R[i] = math.sqrt(A[i]**2 + B[i]**2 + (C[i]-r0[2])**2)
    t_m[i] = r0[3] + R[i]/c
print('A',A, '\n', 'B',B, '\n', 'C',C, '\n', 'R',R)
teikna = True
if teikna is True:
```

```
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(r0[0], r0[1], r0[2], c='b', s=1000)
    ax.scatter(A,B,C, c='r')
    ax.set_xlabel('x-ás (km)')
    ax.set_ylabel('y-ás (km)')
    ax.set_zlabel('z-ás (km)')

    ax.invert_xaxis()

    plt.show()

dt = 10**-8  # nákvæmni klukku í gervihnöttum
errorcoef = [[0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 0], [0, 0, 1, 1], [0, 1, 0, 0], [0, 1, 0, 1], [0, 1, 1, 0], [
    0, 1, 1, 1], [1, 0, 0, 0], [1, 0, 0, 1], [1, 0, 1, 0], [1, 0, 1, 1], [1, 1, 0, 0], [1, 1, 0, 1], [1, 1, 1, 0], [1, 1, 1, 1]]
errorcoef = np.array(errorcoef)
errormagcoef = 0
maxFE = 0

for i in range(0, 16):

    t_new = [
        t_m[0] + dt*errorcoef[i, 0],
        t_m[1] + dt*errorcoef[i, 1],
        t_m[2] + dt*errorcoef[i, 2],
        t_m[3] + dt*errorcoef[i, 3]
    ]

    for j in range(0, 10):
        DF = np.array([
            [(r0[0] - A[0])*2, (r0[1] - B[0])*2,
                (r0[2] - C[0])*2, (2*c**2)*(t_new[0] - r0[3])],
            [(r0[0] - A[1])*2, (r0[1] - B[1])*2,
                (r0[2] - C[1])*2, (2*c**2)*(t_new[1] - r0[3])],
            [(r0[0] - A[2])*2, (r0[1] - B[2])*2,
```

```
                (r0[2] - C[2])*2, (2*c**2)*(t_new[2] - r0[3])],
            [(r0[0] - A[3])*2, (r0[1] - B[3])*2,
                (r0[2] - C[3])*2, (2*c**2)*(t_new[3] - r0[3])]
        ])

        F = np.array([
        [(r0[0] - A[0])**2 + (r0[1] - B[0])**2 +
            (r0[2] - C[0])**2 - (c**2)*(t_new[0] - r0[3])**2],
        [(r0[0] - A[1])**2 + (r0[1] - B[1])**2 +
            (r0[2] - C[1])**2 - (c**2)*(t_new[1] - r0[3])**2],
        [(r0[0] - A[2])**2 + (r0[1] - B[2])**2 +
            (r0[2] - C[2])**2 - (c**2)*(t_new[2] - r0[3])**2],
        [(r0[0] - A[3])**2 + (r0[1] - B[3])**2 +
            (r0[2] - C[3])**2 - (c**2)*(t_new[3] - r0[3])**2]
        ])


        res = lin.solve(DF, F)
        for i in range(len(r0)):
            r0[i] = r0[i] - res[i,0]


    forwarderror = lin.norm(np.asarray([r0[0], r0[1], r0[2]-6370]), ord=2)
    backwarderror = c*lin.norm([t_new[0]-t_m[0], t_new[1] -
                                t_m[1], t_new[2]-t_m[2], t_new[3]-t_m[3]], ord=2)

    if maxFE <= np.absolute(forwarderror):
        maxFE = np.absolute(forwarderror)

if errormagcoef <= np.absolute(forwarderror/backwarderror):
    errormagcoef = np.absolute(forwarderror/backwarderror)
```



Mynd 3: Maximum position error og Error magnification factor



Mynd 4: Staða gervihnatta miðað við jörðu í dæmi 4.

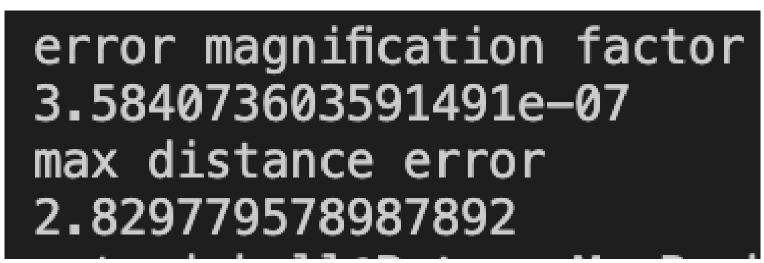
Dæmi 5

Now repeat Step 4 with a more tightly grouped set of satellites. Choose all ϕ_i within 5 percent of one another and all θ_i within 5 percent of one another. Solve with and without the same input error as in Step 4. Find the maximum position error and error magnification factor. Compare the conditioning of the GPS problem when the satellites are tightly or loosely bunched.

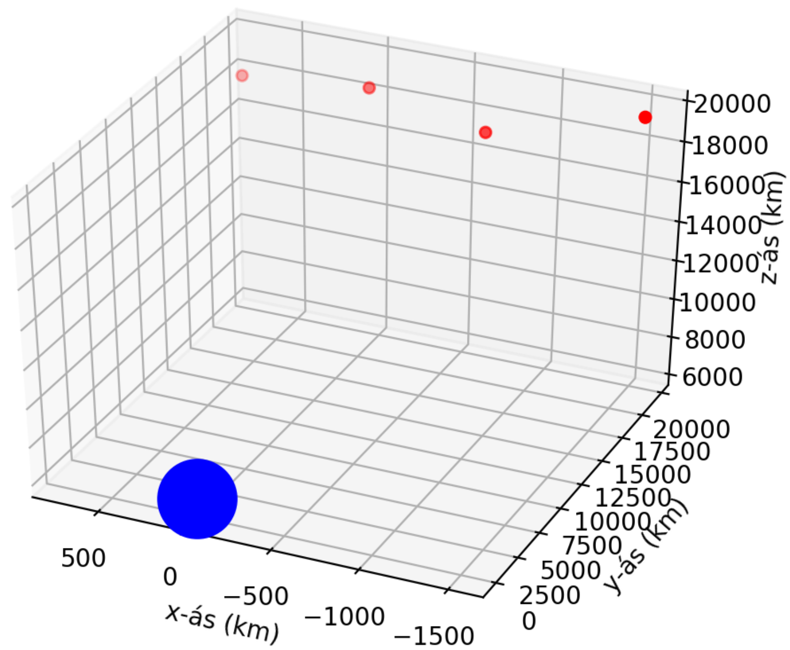
Lausn:

Við breytum aðeins θ_i og ρ_i í þessu dæmi:

```
phi = [math.pi/4 +  math.pi/80, math.pi/4 - math.pi/80, math.pi/4 + math.pi/160, math.pi/4]
theta = [math.pi/2 + math.pi/80, math.pi/2+math.pi/40, math.pi/2 -math.pi/80, math.pi/2]
```



Mynd 5: Maximum position error og Error magnification factor



Mynd 6: Staða gervihnatta miðað við jörðu í dæmi 5.

Dæmi 6

Decide whether the GPS error and condition number can be reduced by adding satellites. Return to the unbunched satellite configuration of Step 4, and add four more. (At all times and at every position on earth, 5 to 12 GPS satellites are visible.) Design a Gauss–Newton iteration to solve the least squares system of eight equations in four variables (x, y, z, d). What is a good initial vector? Find the maximum GPS position error, and estimate the condition number. Summarize your results from four unbunched, four bunched, and eight unbunched satellites. What configuration is best, and what is the maximum GPS error, in meters, that you should expect solely on the basis of satellite signals?

Lausn:

> **Gauss–Newton Method**
>
> To minimize
>
> $$r_1(x)^2 + \cdots + r_m(x)^2.$$
>
> Set $x^0 =$ initial vector,
> **for** $k = 0, 1, 2, \ldots$
>
> $$A = Dr(x^k) \tag{4.33}$$
> $$A^T A v^k = -A^T r(x^k)$$
> $$x^{k+1} = x^k + v^k \tag{4.34}$$
>
> **end**

Kóði fyrir dæmi 6

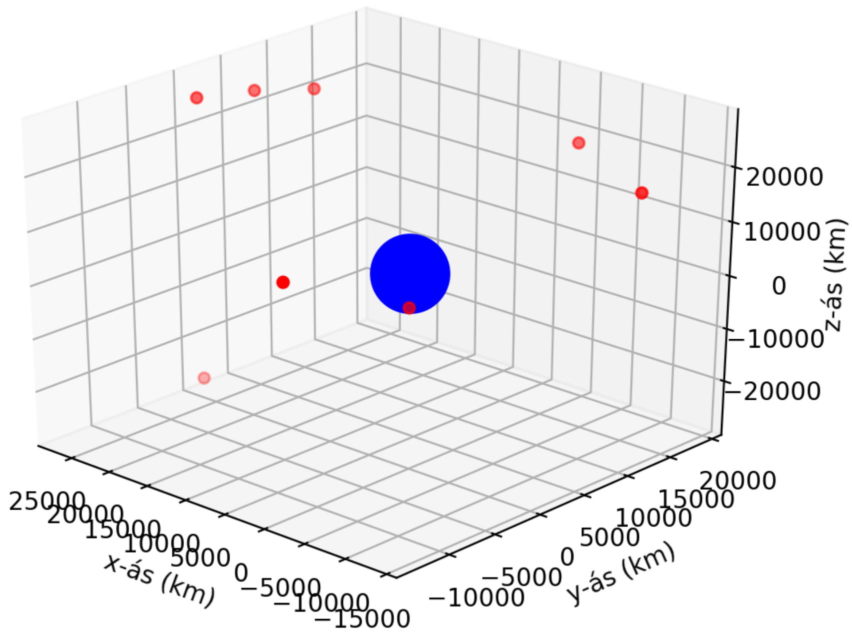
```
c = 299792.458  # Ljoshradi km/s
r0 = np.array([0, 0, 6370, 0.0001])  # hnit mottakara
print(r0[2])
rho = 26570  # km
phi = [0, math.pi/12, math.pi/8, math.pi/4,math.pi/4,
    math.pi/2, math.pi/3,math.pi]
```

```
theta = [0, math.pi/16, math.pi/8, 3*math.pi/4,math.pi/2, math.pi/3, 3*math.pi/2,math.pi]
# Aðferð til að fá hnit dreifð jafnt yfir jörðina
def sample_spherical(npoints, ndim=3):
    vec = np.random.randn(ndim, npoints)
    vec /= np.linalg.norm(vec, axis=0)
    return vec.T

R = sample_spherical(8)

dist=np.zeros(len(R))


A = R[:,0].T # x-hnit gervihnatta
B = R[:,1].T # y-hnit gervihnatta
C = R[:,2].T # z-hnit gervihnatta

for i in range(len(A)):
    A[i] = rho*math.cos(phi[i])*math.cos(theta[i])
    B[i] = rho*math.cos(phi[i])*math.sin(theta[i])
    C[i] = rho*math.cos(phi[i])
for i in range(len(R)):
    temp = np.sqrt(((R[i,0]**2 + R[i,1]**2 + (R[i,2]-6370)**2)))/c
    dist[i] = temp
t_m = dist.T # fjarlægðar fylki

#### TEIKNA HER ####
teikna = True
if teikna is True:
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(r0[0], r0[1], r0[2], c='b', s=1000) #jörðin
    ax.scatter(A, B, C, c='r') # gervihnettir
    ax.set_xlabel('x-ás (km)')
    ax.set_ylabel('y-ás (km)')
    ax.set_zlabel('z-ás (km)')

```

```
    ax.invert_xaxis()

    plt.show()

####            ####

#errorcoef eru
# allar mögulegu uppraðanir á gervihnöttunum
errorcoef = []
for i in itertools.product([0,1],repeat=8):
    errorcoef.append(i)
errorcoef = (np.asarray(errorcoef))

dt = 10**-8  # nákvæmni klukku í gervihnöttum
errormagcoef = 0
maxFE = 0

for i in range(1,len(errorcoef)):
    t = [t_m[0]+dt*errorcoef[i,0],
    t_m[1]+dt*errorcoef[i,1],
    t_m[2]+dt*errorcoef[i,2],
    t_m[3]+dt*errorcoef[i,3],
    t_m[4]+dt*errorcoef[i,4],
    t_m[5]+dt*errorcoef[i,5],
    t_m[6]+dt*errorcoef[i,6],
    t_m[7]+dt*errorcoef[i,7]
    ]

    t_diff = [
        t[0] - t_m[0],
        t[1] - t_m[1],
        t[2] - t_m[2],
        t[3] - t_m[3],
        t[4] - t_m[4],
        t[5] - t_m[5],
        t[6] - t_m[6],
        t[7] - t_m[7]
    ]
```

```
    for j in range(0,10):
        DF = np.array([
            [(r0[0] - A[0])*2, (r0[1] - B[0])*2,
                (r0[2] - C[0])*2, (2*c**2)*(t[0] - r0[3])],
            [(r0[0] - A[1])*2, (r0[1] - B[1])*2,
                (r0[2] - C[1])*2, (2*c**2)*(t[1] - r0[3])],
            [(r0[0] - A[2])*2, (r0[1] - B[2])*2,
                (r0[2] - C[2])*2, (2*c**2)*(t[2] - r0[3])],
            [(r0[0] - A[3])*2, (r0[1] - B[3])*2,
                (r0[2] - C[3])*2, (2*c**2)*(t[3] - r0[3])],
            [(r0[0] - A[4])*2, (r0[1] - B[4])*2,
                (r0[2] - C[4])*2, (2*c**2)*(t[4] - r0[3])],
            [(r0[0] - A[5])*2, (r0[1] - B[5])*2,
                (r0[2] - C[5])*2, (2*c**2)*(t[5] - r0[3])],
            [(r0[0] - A[6])*2, (r0[1] - B[6])*2,
                (r0[2] - C[6])*2, (2*c**2)*(t[6] - r0[3])],
            [(r0[0] - A[7])*2, (r0[1] - B[7])*2,
                (r0[2] - C[7])*2, (2*c**2)*(t[7] - r0[3])]

        ])

        F = np.array([
        [(r0[0] - A[0])**2 + (r0[1] - B[0])**2 +
            (r0[2] - C[0])**2 - (c**2)*(t[0] - r0[3])**2],
        [(r0[0] - A[1])**2 + (r0[1] - B[1])**2 +
            (r0[2] - C[1])**2 - (c**2)*(t[1] - r0[3])**2],
        [(r0[0] - A[2])**2 + (r0[1] - B[2])**2 +
            (r0[2] - C[2])**2 - (c**2)*(t[2] - r0[3])**2],
        [(r0[0] - A[3])**2 + (r0[1] - B[3])**2 +
            (r0[2] - C[3])**2 - (c**2)*(t[3] - r0[3])**2],
        [(r0[0] - A[4])**2 + (r0[1] - B[4])**2 +
            (r0[2] - C[4])**2 - (c**2)*(t[4] - r0[3])**2],
        [(r0[0] - A[5])**2 + (r0[1] - B[5])**2 +
            (r0[2] - C[5])**2 - (c**2)*(t[5] - r0[3])**2],
        [(r0[0] - A[6])**2 + (r0[1] - B[6])**2 +
            (r0[2] - C[6])**2 - (c**2)*(t[6] - r0[3])**2],
        [(r0[0] - A[7])**2 + (r0[1] - B[7])**2 +
```

```
            (r0[2] - C[7])**2 - (c**2)*(t[7] - r0[3])**2]
        ])

        v = lin.solve(np.dot(DF.T,DF) ,np.dot(-DF.T,F))

        for i in range(len(r0)):
            r0[i] = r0[i] + v[i,0]

    forwarderror = lin.norm(np.asarray([r0[0], r0[1], r0[2]-6370]), ord=2)
    backwarderror = c*lin.norm([t[0]-t_m[0], t[1] -
                                t_m[1], t[2]-t_m[2], t[3]-t_m[3],t[4]-t_m[4],t[5]-t_m[5],t[6]-t_m[6
                                ord=2)

    if maxFE <= np.absolute(forwarderror):
        maxFE = np.absolute(forwarderror)

    if errormagcoef <= np.absolute(forwarderror/backwarderror):
        errormagcoef = np.absolute(forwarderror/backwarderror)
```

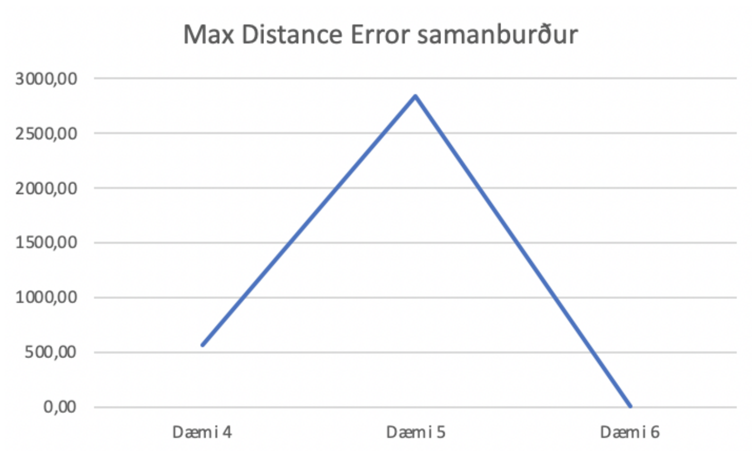


Mynd 7: Staða gervihnatta miðað við jörðu í dæmi 6.

```
error magnification factor
0.7193051696235047
max distance error
0.003267756280126
```

Mynd 8: Niðurstaða í dæmi 6

Ályktun:



Mynd 9: Samanburður á Max Distance Error niðurstöðurnar eru í metrum á töflunni.

Þegar niðurstöðurnar eru settar sjónrænt fram sést greinilega hvaða þættir ráða því hversu árangursríkar mælingarnar eru. Það skiptir höfuðmáli að hafa gervihnettina dreifða og því fleiri, því betra. Þegar bætt er við gervihnöttum verða niðurstöðurnar nákvæmari en þó einungis m.t.t. metra en skekkjan er nokkuð hærri þegar þeir eru fleiri.

Undirskriftir:

Anna Margrét Benediktsdóttir: xxx

Pétur Jökull Þorvaldsson:xxx

Marinó Kristjánsson: xxx