Computational Analysis and Conditioning of Navigational Systems using Nonlinear Least Squares

Nathan Tamiru
Advisor: Dr. Bedřich Sousedík

University of Maryland, Baltimore County

Spring 2023

Objectives

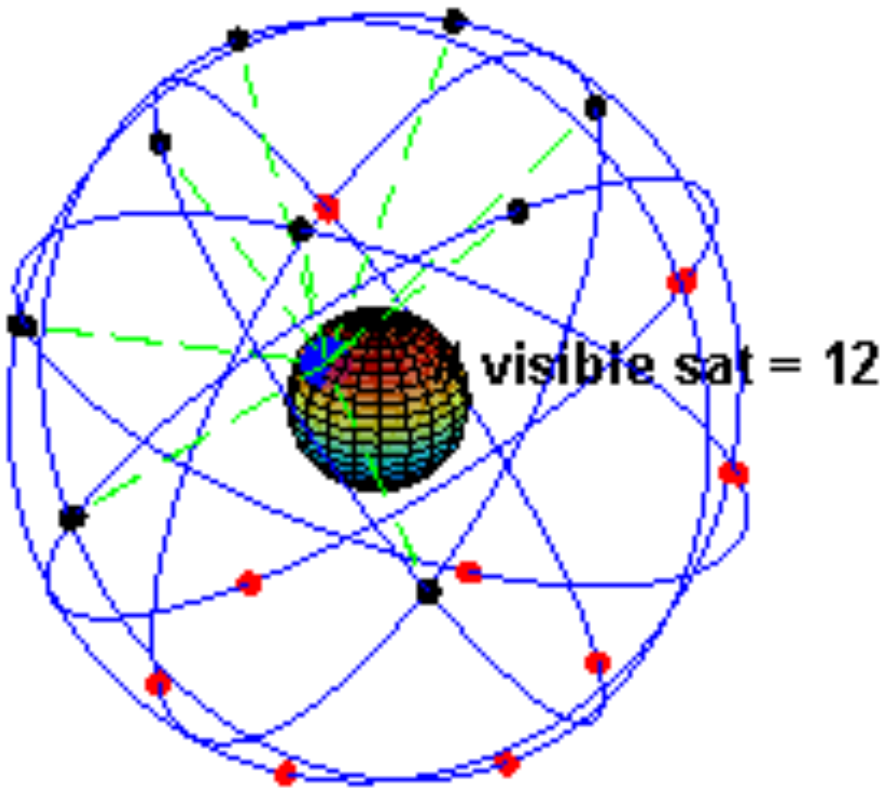
- Introduce the concept of computational analysis, conditioning, and numerical methods in GPS
- Investigate the occurrence of errors in GPS satellites and methods for mitigating their impact on the accuracy of GPS output.
- Explain how nonlinear least squares can be used to model GPS data
- Demonstrate the effectiveness of the approach using experimental data

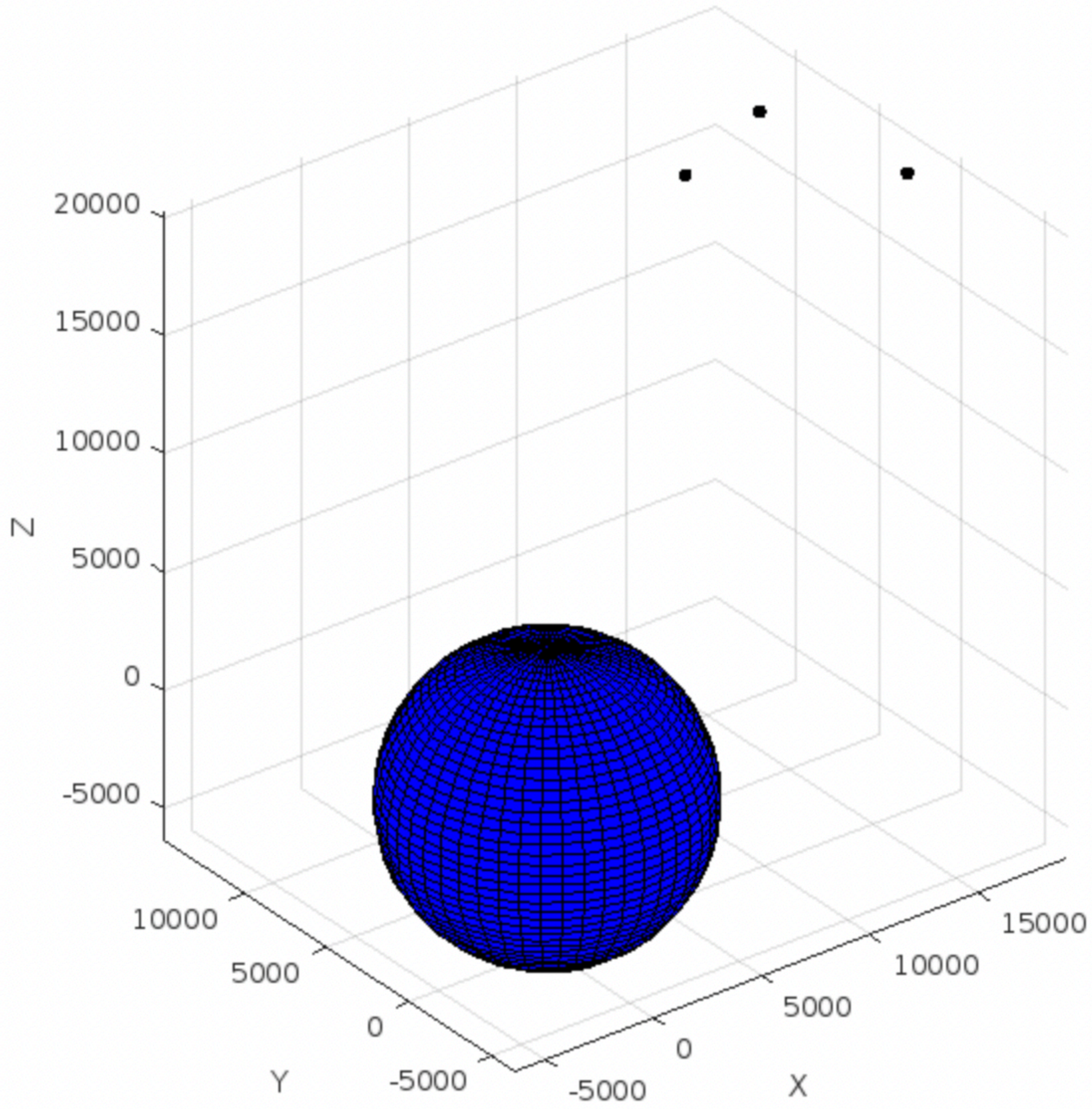
Global Navigation Satellite Systems

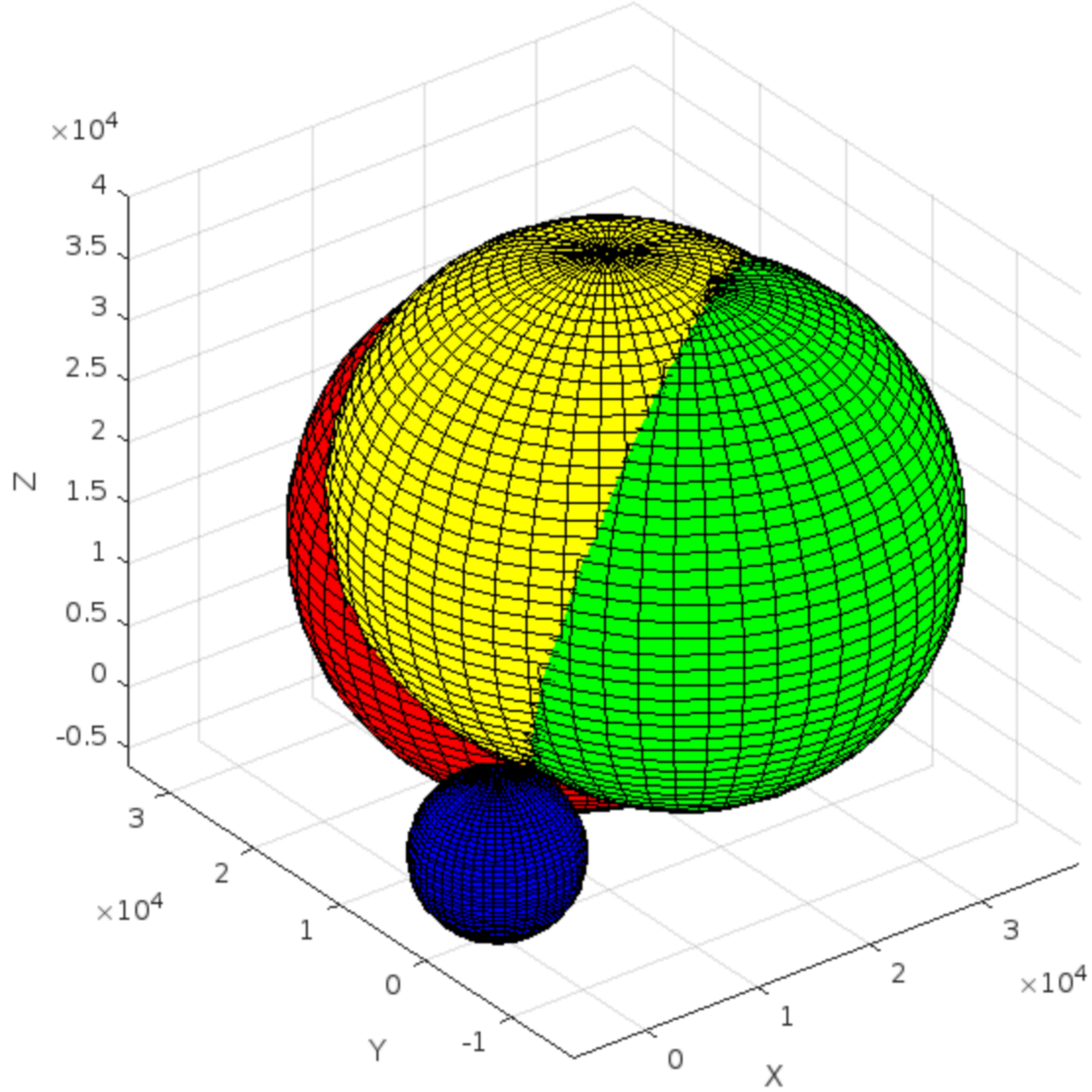
- Global Positioning System (GPS)
  - U.S.
  - Global coverage
  - 31 operational satellites
- Glonass
  - Russia
  - Global coverage
  - 27 operational satellites
- Galileo
  - EU
  - Focused on Asia and Europe
  - 26 operational satellites
- BeiDou
  - China
  - Focused on Asia and Europe
  - 35 operational satellites

GPS : Signals and Equipment

- Consists of of 24 satellite on 6 orbits around Earth
- Satellites are equipped with:
  - Antennas
    - Transmitter
    - Receiver
  - Clocks
    - Quartz-crystal
    - Atomic




20000
15000
10000
5000
0
-5000
Z
10000
5000
0
-5000
Y
-5000
0
5000
10000
15000
X


×10^4
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
Z
3
2
1
0
-1
×10^4
Y
0
1
2
3
×10^4
X

Three Satellite System

- We will need to solve for the following:

$$f_1 = \sqrt{(x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2} - ct_1 = 0$$
$$f_2 = \sqrt{(x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2} - ct_2 = 0$$
$$f_3 = \sqrt{(x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2} - ct_3 = 0$$

- $[A_i, B_i, C_i]$ are the $[x, y, z]$ coordinates of of our satellites from Earth's frame of reference.
- t_i is the time it took for our satellite signal to reach the receiver.

Error in GPS

- Error during transmission (multiphase interference)
- Error in clocks
  - Environmental change (Temperature or pressure) in quartz crystal clocks
  - Noises in atomic clocks

Clocks

- An ideal clock is an oscillator that generates a sinusoidal signal

$$u(t) = U_0 sin(2\pi v_0 t)$$

- Quartz crystal
~35kHz = 1 sec
- Rubidium clock
~9MHz = 1 sec

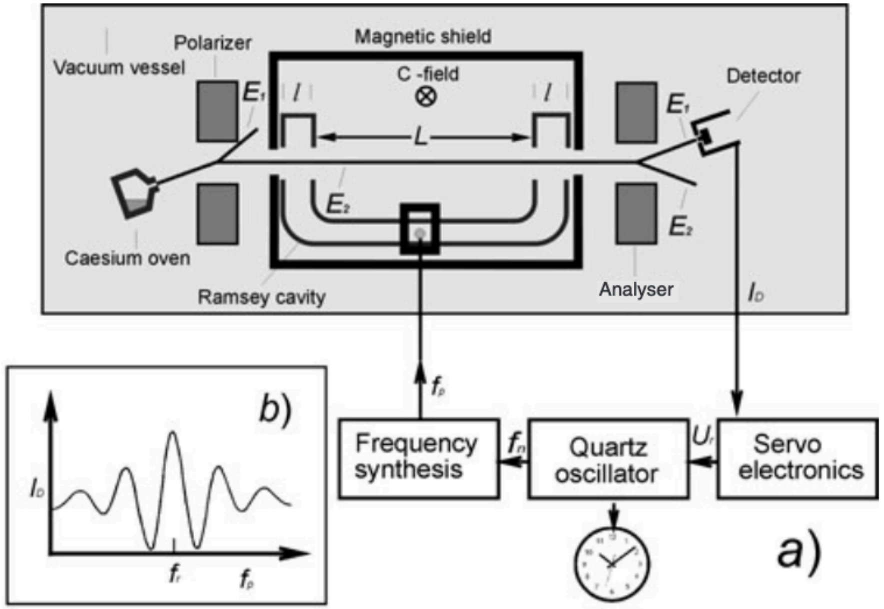
	Piezoelectric	Atomic
Functionality	Use the conversion of mechanical energy into electrical energy	Based onthe oscillations of atoms, typically using the transitions between energy levels of atoms or ions
Accuracy	Accuracy within 10^-6 sec	Accuracy between 10^-18 to 10^-14
Error	Susceptible to error due to environmental changes (Temperature,	Brownian, white noise

Atomic Clocks

- Allan deviation- Error signal feedback loop:

$$\sigma_y(\tau) = \sqrt{\frac{1}{2(N-1)\tau^2}\sum_{i=1}^{N-1}(y_{i+1}-y_i)^2}$$

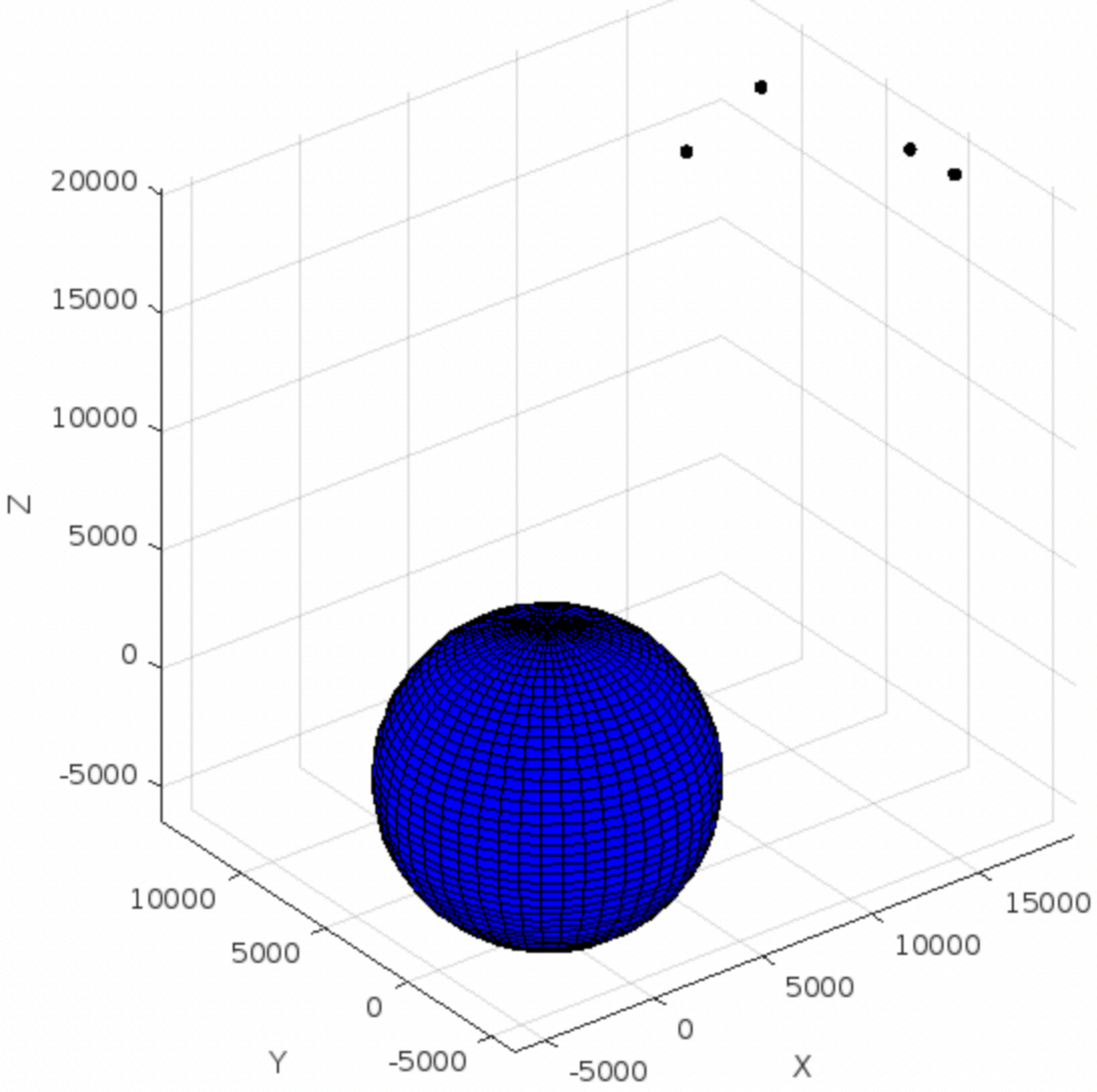
  - N - the number of data points used in the calculation
  - y_i- the i-th clock's frequency error or deviation from the ideal frequency.
  - τ - the averaging time over which the Allan deviation is calculated
- Measure the difference between the clock's output frequency and a reference frequency Adjust the frequency of the quartz oscillator to match the absorbed radiation frequency. $\Delta f = f_{oscillator} - \nu_0$
- Count the number of cycles of the frequency to calculate the output frequency of the atomic clock: $f_{out} = N_{cycles} \times \nu_0$

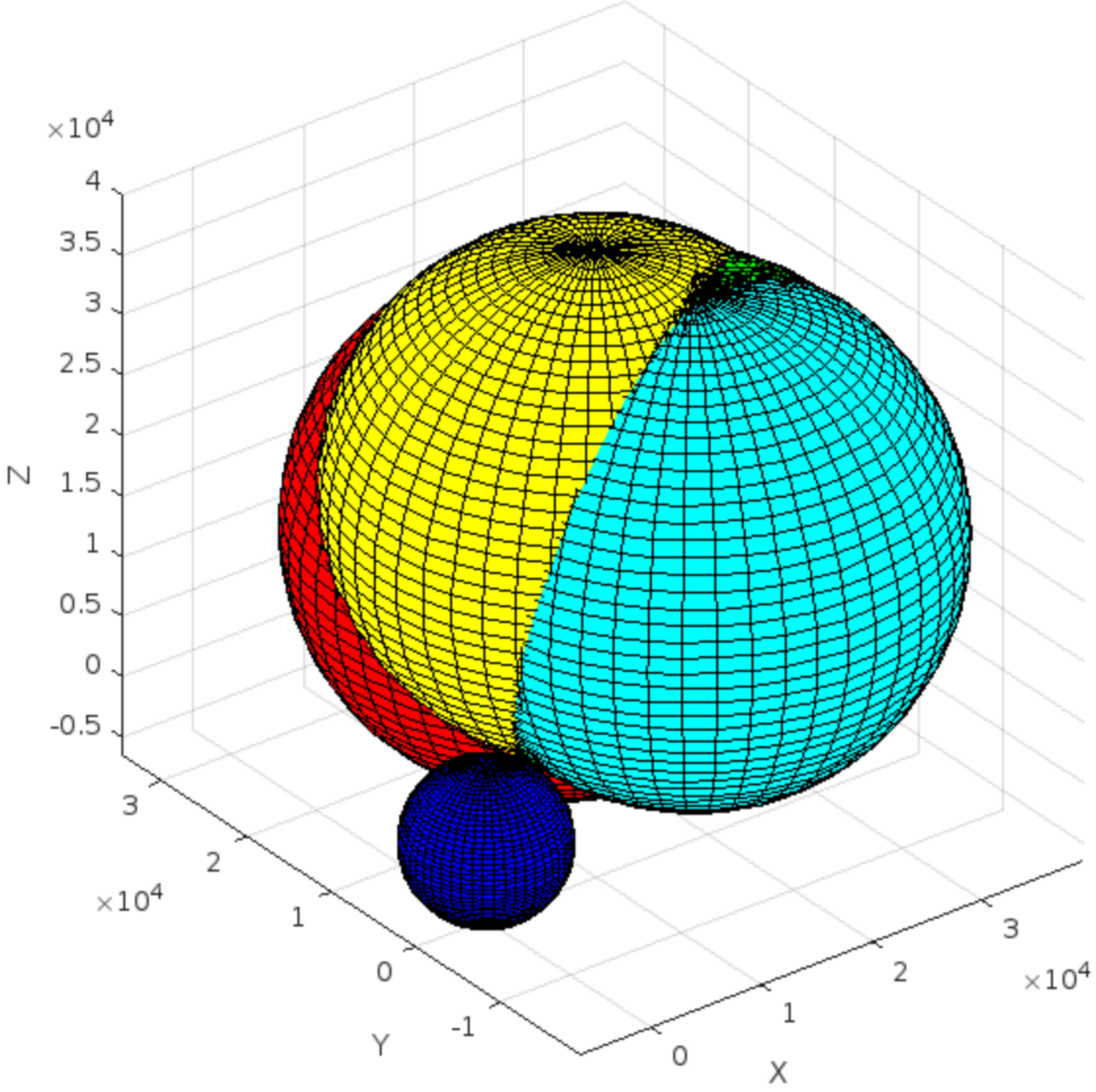


Four Satellite System

$$f_1 = \sqrt{(x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2} - c(t_1 - d) = 0$$
$$f_2 = \sqrt{(x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2} - c(t_2 - d) = 0$$
$$f_3 = \sqrt{(x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2} - c(t_3 - d) = 0$$

- $f_4 = \sqrt{(x - A_4)^2 + (y - B_4)^2 + (z - C_4)^2} - c(t_4 - d) = 0$
- $[A_i, B_i, C_i]$ are the $[x, y, z]$ coordinates of our satellites from Earth's frame of reference.
- t_i is the time it took for the satellite signal to reach the receiver.
- d is the difference between the atomic clocks on the and the receiver


20000
15000
10000
Z
5000
0
-5000
10000
5000
0
Y
-5000
-5000
0
5000
X
10000
15000


×10^4
4
3.5
3
2.5
2
Z
1.5
1
0.5
0
-0.5
3
2
×10^4
1
0
-1
Y
0
1
2
3
×10^4
X

Computing Coordinates: Algebraic Manipulation

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

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

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

$$Where$$

$$\vec{w} = \begin{pmatrix} -c^2 * (t_1^2 - t_4^2) + A_1^2 - A_4^2 + B_1^2 - B_4^2 + C_1^2 - C_4^2 \\ -c^2 * (t_1^2 - t_3^2) + A_1^2 - A_3^2 + B_1^2 - B_3^2 + C_1^2 - C_3^2 \\ -c^2 * (t_1^2 - t_2^2) + A_1^2 - A_2^2 + B_1^2 - B_2^2 + C_1^2 - C_2^2 \end{pmatrix} \vec{u_x} = \begin{pmatrix} 2 * (A_4 - A_1) \\ 2 * (A_3 - A_1) \\ 2 * (A_2 - A_1) \end{pmatrix}$$

$$\vec{u_y} = \begin{pmatrix} 2 * (B_4 - B_1) \\ 2 * (B_3 - B_1) \\ 2 * (B_2 - B_1) \end{pmatrix}$$

$$\vec{u_z} = \begin{pmatrix} 2 * (C_4 - C_1) \\ 2 * (C_3 - B_1) \\ 2 * (C_2 - B_1) \end{pmatrix}$$

$$\vec{u_d} = \begin{pmatrix} -2 * c^2 * (t_4 - t_1) \\ -2 * c^2 * (t_3 - t_1) \\ -2 * c^2 * (t_2 - t_1) \end{pmatrix}$$

MATLAB Snippet

```
w_1 = -c^2*(t_1^2 - t_4^2)  + A_1^2 - A_4^2 + B_1^2 - B_4^2 + C_1^2 - C_4^2;
w_2 = -c^2*(t_1^2 - t_3^2)  + A_1^2 - A_3^2 + B_1^2 - B_3^2 + C_1^2 - C_3^2;
w_3 = -c^2*(t_1^2 - t_2^2)  + A_1^2 - A_2^2 + B_1^2 - B_2^2 + C_1^2 - C_2^2;


u_x_1 = 2*(A_4 - A_1);
u_x_2 = 2*(A_3 - A_1);
u_x_3 = 2*(A_2 - A_1);

u_y_1 = 2*(B_4 - B_1);
u_y_2 = 2*(B_3 - B_1);
u_y_3 = 2*(B_2 - B_1);

u_z_1 = 2*(C_4 - C_1);
u_z_2 = 2*(C_3 - C_1);
u_z_3 = 2*(C_2 - C_1);

u_d_1 = -2*c^2*(t_4 - t_1);
u_d_2 = -2*c^2*(t_3 - t_1);
u_d_3 = -2*c^2*(t_2 - t_1);

w = [w_1;w_2;w_3];

syms x y z d;

ux = [u_x_1;u_x_2 ;u_x_3];
uy= [u_y_1;u_y_2;u_y_3];
uz = [u_z_1;u_z_2;u_z_3];
ud = [u_d_1;u_d_2;u_d_3];

mx = [uy,uz, x*ux+y*uy+z*uz+d*ud+w];
x_in_d = det(mx)==0;

my = [ux,uz, x*ux+y*uy+z*uz+d*ud+w];
y_in_d = det(my)==0;

mz = [ux,uy, x*ux+y*uy+z*uz+d*ud+w];
z_in_d = det(mz)==0;


sx = solve(x_in_d,x);
sy = solve(y_in_d,y);
sz = solve(z_in_d,z);


f1 = (sx - sat1coord(1,1))^2 + (sy - sat1coord(2,1))^2 + (sz - sat1coord(3,1))^2 - (c*(t_1 - d))^2 ==0;

sd = vpasolve(f1,d);
sx = subs(sx, d,sd(1,1));
sy = subs(sy, d,sd(1,1));
sz = subs(sz, d,sd(1,1));
soln= [sx;sy;sz;sd(1,1)]
```

Computing Coordinates: Multivariate Newton's Method

$$f_1 = (x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2 - (c(t_1 - d))^2 = 0$$
$$f_2 = (x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2 - (c(t_2 - d))^2 = 0$$
$$f_3 = (x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2 - (c(t_3 - d))^2 = 0$$
$$f_4 = (x - A_4)^2 + (y - B_4)^2 + (z - C_4)^2 - (c(t_4 - d))^2 = 0$$

$$\alpha_k = (x_k, y_k, z_k, d_k)$$

$$F(\alpha) = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}$$

$$DF(\alpha) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial t} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} & \frac{\partial f_2}{\partial t} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} & \frac{\partial f_3}{\partial t} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial z} & \frac{\partial f_4}{\partial t} \end{pmatrix} = \begin{pmatrix} 2(x - A_1) & 2(y - B_1) & 2(z - C_1) & 2c^2(t_1 - d) \\ 2(x - A_2) & 2(y - B_2) & 2(z - C_2) & 2c^2(t_2 - d) \\ 2(x - A_3) & 2(y - B_3) & 2(z - C_3) & 2c^2(t_3 - d) \\ 2(x - A_4) & 2(y - B_4) & 2(z - C_4) & 2c^2(t_4 - d) \end{pmatrix}$$

$$DF(\alpha_k)s = -F(\alpha_k)$$
$$\alpha_{k+1} = \alpha_k + s$$

$$\text{for } k = 0, 1, \ldots, n$$

MATLAB Snippet

```
format long
c = 299792.458;

sat1coord = [15600; 7540; 20140];
sat2coord = [18760; 2750; 18610];
sat3coord = [17610; 14630; 13480];
sat4coord = [19170; 610; 18390];

t1 = 0.07074;
t2 = 0.07220;
t3 = 0.07690;
t4 = 0.07242;


res = [0; 0; 6370; 0];

for i = 1:100

    x = res(1,1);
    y = res(2,1);
    z = res(3,1);
    d = res(4,1);

    f1 = (x - sat1coord(1,1))^2 + (y - sat1coord(2,1))^2 + (z - sat1coord(3,1))^2 - (c*(t1 - d))^2;
    f2 = (x - sat2coord(1,1))^2 + (y - sat2coord(2,1))^2 + (z - sat2coord(3,1))^2 - (c*(t2 - d))^2;
    f3 = (x - sat3coord(1,1))^2 + (y - sat3coord(2,1))^2 + (z - sat3coord(3,1))^2 - (c*(t3 - d))^2;
    f4 = (x - sat4coord(1,1))^2 + (y - sat4coord(2,1))^2 + (z - sat4coord(3,1))^2 - (c*(t4 - d))^2;

    F = [f1; f2; f3; f4];

    Dr = [2*(x - sat1coord(1,1)), 2*(y - sat1coord(2,1)), 2*(z - sat1coord(3,1)), 2*c^2*(t1 - d);
            2*(x - sat2coord(1,1)), 2*(y - sat2coord(2,1)), 2*(z - sat2coord(3,1)), 2*c^2*(t2 - d);
            2*(x - sat3coord(1,1)), 2*(y - sat3coord(2,1)), 2*(z - sat3coord(3,1)), 2*c^2*(t3 - d);
            2*(x - sat4coord(1,1)), 2*(y - sat4coord(2,1)), 2*(z - sat4coord(3,1)), 2*c^2*(t4 - d)];

    s = Dr \ (-F);

    res = res + s;
end
```

Comparison of Newton's method vs. MATLAB's *fsolve*

Initial Values for Newton-Raphson and Matlab "fsolve"

$x_s^{(0)}$, m		$y_s^{(0)}$, m	$z_s^{(0)}$, m	$\delta_t^{(0)}$, m
6731110		-2269170	2313	20
Max. Iteration	100		Tolerance for Convergence	10^{-6}

Real Values for LEO Satellite at Startup

$x_s^{(0)}$, m	$y_s^{(0)}$, m	$z_s^{(0)}$, m	$\delta_t^{(0)}$, m
6731110.93468	-2269170.47424	2313.80490	25

Parameters

Latitude	*Longitude*	*Altitude (km)*
14.175	78.643	20474.394
-9.498	31.381	20275.821
30.177	-11.786	20198.588
38.790	43.273	20087.115

Standard Deviation Values Between Real Data and Results

Method	σ_x	σ_y	σ_z	σ_δ
Newton-Raphson	40.36462	28.53368	35.59287	14.05249
MATLAB "fsolve"	44.35108	29.89649	36.92109	53.85707

Algorithm that is Used	***Computation Time***
Newton-Raphson	0.27243437899553 sec
MATLAB "fsolve"	18.5056278105048 sec

Conditioning:Error Magnification

$$\text{emf} = \frac{||\Delta x, \Delta y, \Delta z||_\infty}{c||\Delta t_1, \Delta t_2, \Delta t_3, \Delta t_4||_\infty}$$

- Allows us to quantify the impact of errors in the input on the accuracy and reliability of the output.
- By minimizing the error magnification factor, we can improve the overall performance and accuracy of the GPS system.

Conditioning:Maximum Error

- Define satellite positions (A_i, B_i, C_i) from spherical coordinates ρ, ϕ, θ as:

$$A_i = \rho cos(\phi_i) cos(\theta_i)$$
$$B_i = \rho cos(\phi_i) sin(\theta_i)$$
$$C_i = \rho sin(\phi_i)$$

- Take a constant input error: d = 10^{-8}

$$\text{emf} = \frac{||\Delta x, \Delta y, \Delta z||_\infty}{c||\Delta t_1 \Delta t_2 \Delta t_3 \Delta t_4||_\infty} = \frac{||\Delta x, \Delta y, \Delta z||_\infty}{cd}$$

- Calculate the maximum EMF after combining non-erroneous and erroneous t (+10^{-8}or- 10^{-8}) for all our satellites- 81 possible combinations

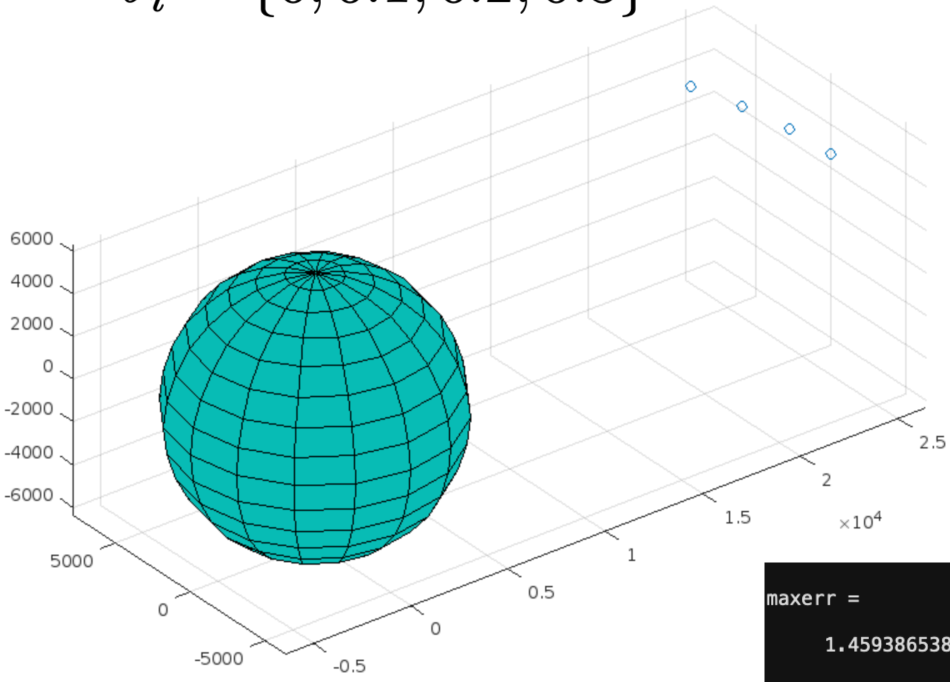
Conditioning:Effect of distance

$$\phi_i = \{0.1, 0.1, 0.1, 0.1\}$$
$$\theta_i = \{0, 0.1, 0.2, 0.3\}$$

$$A_i = \rho cos(\phi_i)cos(\theta_i)$$
$$B_i = \rho cos(\phi_i)sin(\theta_i)$$
$$C_i = \rho sin(\phi_i)$$

$$\rho = 26570$$

$$\phi_i = \{0, 0.1, 0.1, 1.1\}$$
$$\theta_i = \{0, \pi/2, \pi, 3\pi/2\}$$

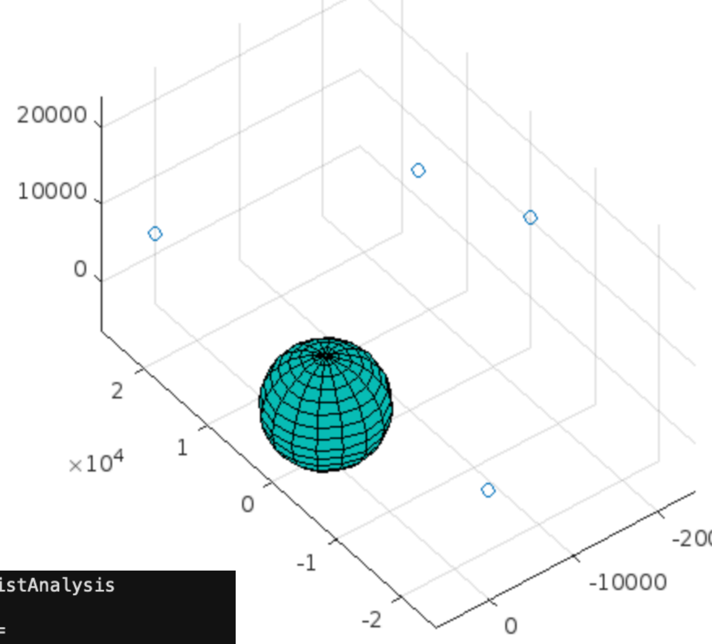



```
maxerr =

    1.459386538707454e+14

EMF =

    4.867989503283147e+16
```




```
>> satDistAnalysis

maxerr =

   0.007661877434657

EMF =

   2.555727214010524
```

MATLAB Snippet

```
maxerr_disp = [];
maxerr_vect = [];
maxerr = 0;
for t1_err=-1:1
    for t2_err=-1:1
        for t3_err=-1:1
            for t4_err=-1:1

                t1 = times(1) + t1_err * delt_t;
                t2 = times(2) + t2_err * delt_t;
                t3 = times(3) + t3_err * delt_t;
                t4 = times(4) + t4_err * delt_t;

                init_vec = [100; 100; 6380; 0];

                for i = 1:100
                    x = init_vec(1,1);
                    y = init_vec(2,1);
                    z = init_vec(3,1);
                    d = init_vec(4,1);

                    f1 = (x-satcoord(1,1))^2 + (y-satcoord(1,2))^2 + (z-satcoord(1,3))^2 - (c*(t1-d))^2;
                    f2 = (x-satcoord(2,1))^2 + (y-satcoord(2,2))^2 + (z-satcoord(2,3))^2 - (c*(t2-d))^2;
                    f3 = (x-satcoord(3,1))^2 + (y-satcoord(3,2))^2 + (z-satcoord(3,3))^2 - (c*(t3-d))^2;
                    f4 = (x-satcoord(4,1))^2 + (y-satcoord(4,2))^2 + (z-satcoord(4,3))^2 - (c*(t4-d))^2;

                    F = [f1; f2; f3; f4];

                    Dr = 2*[x - satcoord(1,1), y - satcoord(1,2), z - satcoord(1,3), c^2*(t1 - d);
                               x - satcoord(2,1), y - satcoord(2,2), z - satcoord(2,3), c^2*(t2 - d);
                               x - satcoord(3,1), y - satcoord(3,2), z - satcoord(3,3), c^2*(t3 - d);
                               x - satcoord(4,1), y - satcoord(4,2), z - satcoord(4,3), c^2*(t4 - d);];

                    s = Dr \ (-F);
                    init_vec = init_vec + s;
                end

                err = max (abs([init_vec(1)-true_x, init_vec(2)-true_y, init_vec(3)-true_z]));
                if err > maxerr
                    maxerr_disp = [t1_err, t2_err, t3_err, t4_err] * delt_t;
                    maxerr_vect = init_vec;
                    maxerr = err;
                end
            end
        end
    end
end
```

Least squares

- Method for finding the best fit line or curve through a set of data points
- Involves minimizing the sum of the squared differences between the observed values and the predicted values from the model.
- In linear regression: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$
  - The goal is to find the values of $\boldsymbol{\beta}$ that minimize the sum of the squared residuals:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

  - The solution to this problem can be found by setting the derivative of the objective function with respect to $\boldsymbol{\beta}$ equal to zero:

$$\frac{\partial}{\partial \boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \mathbf{0}$$

  - Solving for $\boldsymbol{\beta}$ yields: $\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Non-linear least squares in GPS

- The non-linearity of the observation equation requires us to use NLLS.

$$f_1 = \sqrt{(x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2} - c(t_1 - d) = 0$$
$$f_2 = \sqrt{(x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2} - c(t_2 - d) = 0$$
$$f_3 = \sqrt{(x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2} - c(t_3 - d) = 0$$
$$f_4 = \sqrt{(x - A_4)^2 + (y - B_4)^2 + (z - C_4)^2} - c(t_4 - d) = 0$$

- The NLLS problem in GPS is typically solved using iterative algorithms such as the Gauss-Newton or Levenberg-Marquardt methods
- Involves linearizing the model and solving a sequence of linear least squares problems

Gauss-Newton NLLS Algorithm

- The Gauss-Newton algorithm for (n $\geq$ 4) satellites:

$$r_k(x,y,z,d) = (x-A_k)^2 + (y-B_k)^2 + (z-C_k)^2 - (c(t_k-d))^2$$

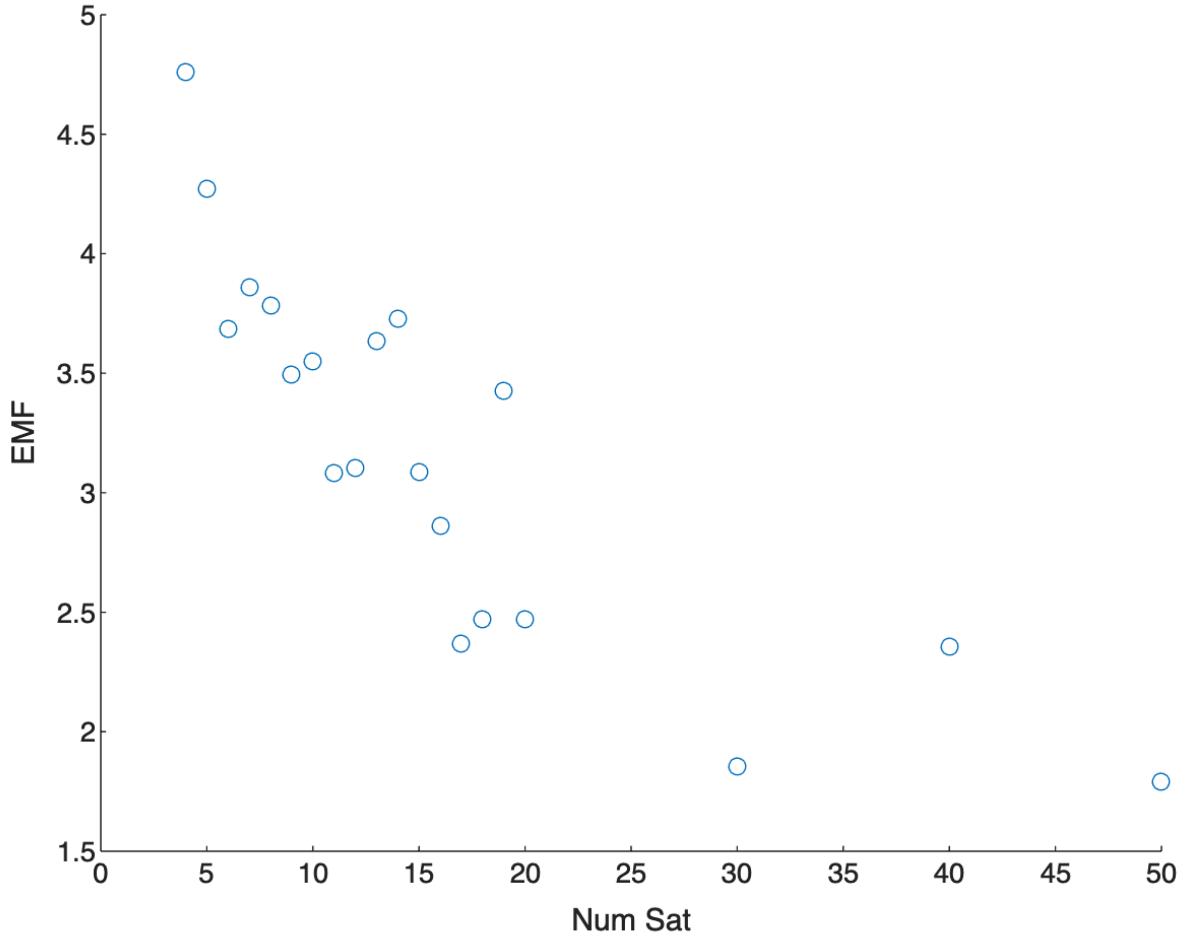
$$r(\alpha_k) = \begin{pmatrix} r_1(x,y,z,d) \\ r_2(x,y,z,d) \\ r_3(x,y,z,d) \\ \vdots \\ r_n(x,y,z,d) \end{pmatrix} \qquad \alpha_k = (x_k, y_k, z_k, d_k)$$

$$\begin{cases} (Dr)^T(Dr)s_k = -(Dr)^T r(\alpha_k) \\ \alpha_{k+1} = \alpha_k + s_k \end{cases} \qquad \text{for } k = 0, 1, \ldots, m.$$

- Dr is the Jacobian

Accuracy and the Number of Satellites

- Initial vector ([0,0,6370,0.001]- north pole)
- Randomizing error in satellites instead of permuting
- Here we let our true solution = [1136.1863; -4813.2651; 4014.2038; 0.0001];
  // This point is UMBC's Math and Psych Building
- An increase in the number of satellites can significantly reduce EMF and maximum error



MATLAB Snippet

```
for i=1:num_of_sat
    sat_A(i) = rho*cos(phi(1,i))*cos(theta(1,i));
    sat_B(i) = rho*cos(phi(1,i))*sin(theta(1,i));
    sat_C(i) = rho*sin(phi(1,i));
    R(i) = sqrt((sat_A(i)-umbc_sol(1))^2 + (sat_B(i) - umbc_sol(2))^2 + (sat_C(i) - umbc_sol(3))^2);
    t(i) = d_1 + R(i)/c;
end

maxerr = 0;
maxposerr = 0;

for randiter = 1:100
    k = 100;
    f = zeros(1,num_of_sat);
    init_vec = [0; 0; 6370; 0];

    %using rand num of errors instead of permuting
    time_error = err_t * ((rand(1, num_of_sat) > 0.5) * 2 - 1);

    for iter=1:k
        x = init_vec(1,1);
        y = init_vec(2,1);
        z = init_vec(3,1);
        d = init_vec(4,1);
        for i=1:num_of_sat
            f(i) = (x - sat_A(i))^2 + (y - sat_B(i))^2 + (z - sat_C(i))^2 - (c*(t(i) + time_error(i) - d))^2;
        end
        F = transpose(f);

        Jacobian = zeros(num_of_sat,4);
        for i=1:num_of_sat
            Jacobian(i,1) = 2*(x - sat_A(i));
            Jacobian(i,2) = 2*(x - sat_B(i));
            Jacobian(i,3) = 2*(x - sat_C(i));
            Jacobian(i,4) = 2*c^2*(t(i) + time_error(i) - d);
        end

        A = transpose(Jacobian)*Jacobian;
        b = -(transpose(Jacobian)*F);
        v = A \ b;

        init_vec = init_vec + v;
    end
    comp_vec = init_vec;

    maxerr = max(maxerr, max(abs(comp_vec - umbc_sol)));
    maxposerr = max(maxposerr, norm(comp_vec - umbc_sol));
end


maxerr_(n_idx,1) = maxposerr;
n_value(n_idx,1) = num_of_sat;
emf(n_idx,1) = maxerr / (c * err_t);
```

Bibliography

- *Allan Deviation of atomic clock frequency corrections: A ... - IEEE xplore*. (n.d.). https://ieeexplore.ieee.org/document/9359763/
- Bagci, M., & Hacizade, C. (2015). Performance analysis of GPS based orbit determination via Numerical Methods for a leo satellite. *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*. https://doi.org/10.1109/rast.2015.7208437
- Bauch1, A. (2003, July 16). *IOPscience*. Measurement Science and Technology. https://iopscience.iop.org/article/10.1088/0957-0233/14/8/301*Caesium atomic clocks: Function, performance and applications - iopscience*. (n.d.) https://iopscience.iop.org/article/10.1088/0957-0233/14/8/301
- Newton-Taylor-Jacobi Nexus. (n.d.). https://mason.gmu.edu/~msulli22/
- Sauer, T. (2019). Least Squares. In *Numerical analysis*. Pearson.
- Wikimedia Foundation. (2015, December 9). *GPS Constellation*. Wikipedia. https://en.wikipedia.org/wiki/Global_Positioning_System/ConstellationGPS.gif