Introduction to space Geodesy

Assignment 2

```
clc;
clear all;
close all;
```

Exercise 1: Write a function that computes a basic 3d-rotation matrix using the equations in attachment. The first line could read `function` $R_i = \text{rot3d}(\text{angle}, \text{Axis})$ where angle denotes the rotation angle (unit?) and axis specifies the rotation axis ($e_i,\ i \in \{1,2,3\}$) . Both input arguments are scalar. Output argument is a (3x3) matrix R_i. Check the correctness of the function by testing the examples $R_i(2\pi) = R_i(0) = R_i(\pi)\ R_i(\pi) = I$, where I is the (3x3) identity matrix. Are the basic rotation matrices providing clockwise $\left(e_1 = R_3\left(\frac{\pi}{2}\right)e_2\right)$ or anti-clockwise $\left(e_2 = R_3\left(\frac{\pi}{2}\right)e_1\right)$ rotation in case of positive angle when taking the point of view from the origin?

```
function R_i = rot3d(angle, axis)
    c = cos(angle);
    s = sin(angle);

    switch axis
        case 1
            R_i = [1 0 0;
                0 c s;
                0 -s c];
        case 2
            R_i = [c 0 -s;
                    0 1 0;
                    s 0 c];
        case 3
            R_i = [c s 0;
                    -s c 0;
                    0 0 1];
        otherwise
            error('Choose 1, 2, or 3.');
    end
end
```

For checking the correctness

```
% to-do

fprintf(' Exercise 1 tests\n');
```

```
 Exercise 1 tests
```

```
I = eye(3);
```

```
threshold = 1e-12;

for ax = 1:3
    R0 = rot3d(0, ax);
    R2pi = rot3d(2*pi, ax);
    Rpi = rot3d(pi, ax);

    fprintf('Axis %d: \n', ax)
    disp('R(2π) = I?');
    disp(norm(R2pi - I) < threshold);

    disp('R(0) = I?');
    disp(norm(R0 - I) < threshold);

    disp('R(pi)R(pi) = I ?');
    disp(norm(Rpi*Rpi - I) < threshold);

end
```

```
Axis 1:
R(2π) = I?
   1
R(0) = I?
   1
R(pi)R(pi) = I ?
   1
Axis 2:
R(2π) = I?
   1
R(0) = I?
   1
R(pi)R(pi) = I ?
   1
Axis 3:
R(2π) = I?
   1
R(0) = I?
   1
R(pi)R(pi) = I ?
   1
```

Exercise 2: Write a function that computes a 3d-reflection matrix using the equations in attachment. The first line could read `function` $M\_i = \mathrm{mir3d(axis)}$ where axis specifies the mirror axis ($e_i,\ i \in \{1,2,3\}$) . The input argument is scalar. Output argument is a (3x3) matrix M_i. Check the correctness of the function by testing M_i $M_i = I$, $M_i\, M_j\, M_k = -I$ with $i \neq j \neq k \in \{1,2,3\}$ and I as of (exercise 1)

```
% to-do
%
function M_i = mir3d(axis)

    switch axis
```

```
        case 1
            M_i = [-1 0 0;
                0 1 0;
                0 0 1];
        case 2
            M_i = [1 0 0;
                0 -1 0;
                0 0 1];
        case 3
            M_i = [1 0 0;
                0 1 0;
                0 0 -1];
        otherwise
            error('Choose 1, 2 or 3');
    end

end
```

```
fprintf('\n\nExercise 2 test\n')
```

```
Exercise 2 test
```

```

for ax = 1:3
    Mi = mir3d(ax);

    fprintf('Axis %d: \n', ax);

    disp('Mi*Mi = I?');
    disp(norm(Mi*Mi - I) < threshold);

end
```

```
Axis 1:
Mi*Mi = I?
   1
Axis 2:
Mi*Mi = I?
   1
Axis 3:
Mi*Mi = I?
   1
```

```

M1 = mir3d(1);
M2 = mir3d(2);
M3 = mir3d(3);

disp((M1 * M2 * M3));
```

```
    -1     0     0
     0    -1     0
     0     0    -1
```

```
disp("Mi Mj Mk = -I");
```

```
Mi Mj Mk = -I
```

```
disp(norm(M1 * M2 * M3 + I) < threshold);
```

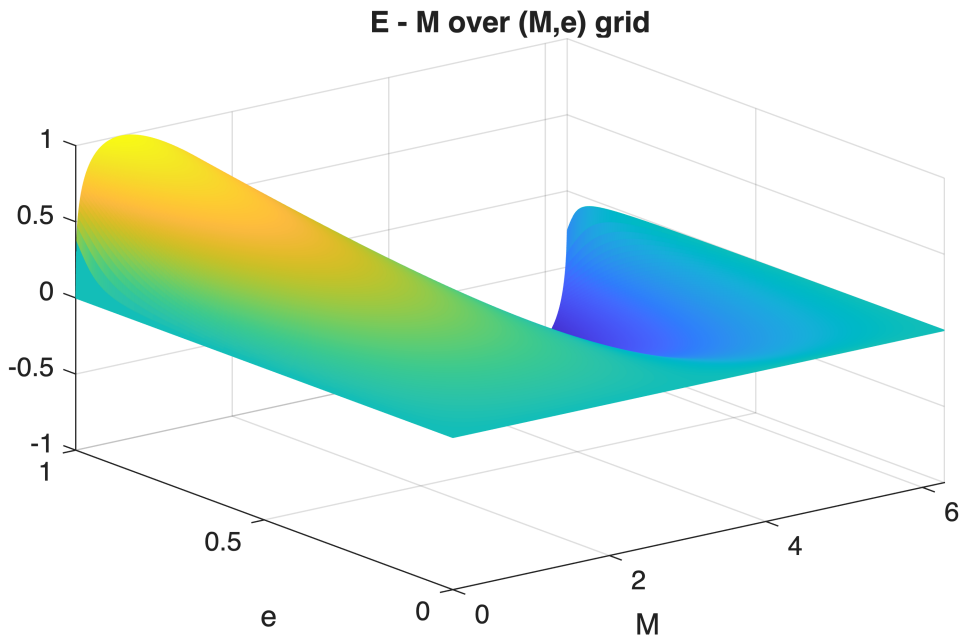
```
   1
```

```
% disp
```

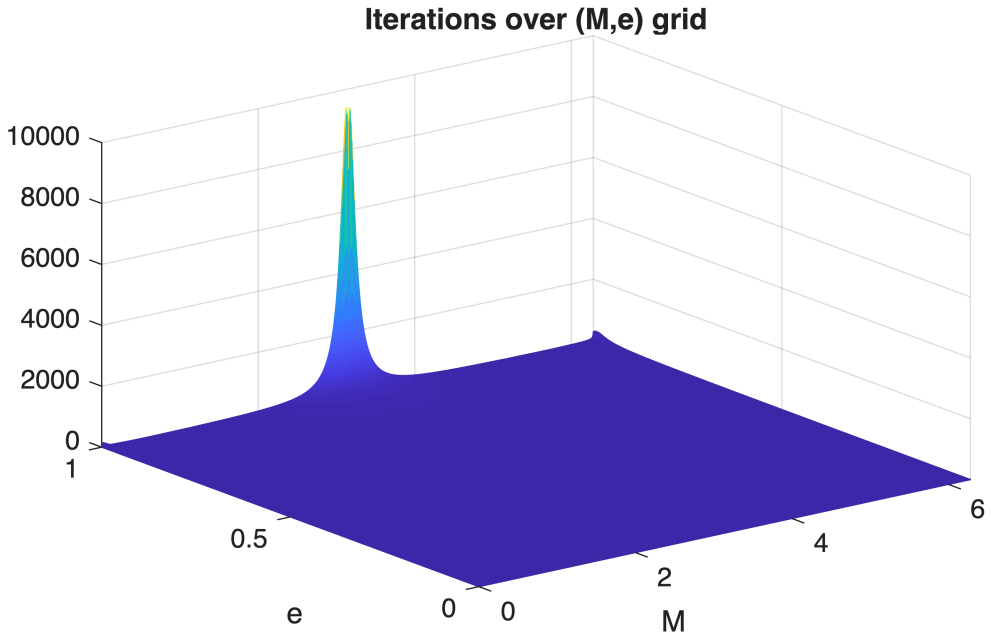
Exercise 3: In the theory of satellite orbits, a common problem is the KEPLER equation: $M = e - e \sin E$. It relates the mean anomaly $M \in [0; 2\pi[$ to the eccentric anomaly $E \in [0; 2\pi[$ with given numerical eccentricity $e \in [0; 1[$. For E, the equation has no closed solution. To compute E from a given M one can iterate: $E_{i+1} = M + e \sin E_i$ e.g. with starting value $E_{i=0} = M$. Write a function that computes E from given M and e iteratively with the stop criterion $| E_{i+1} - E_i | < 10^{-6}$. Next, extend the number of output arguments so that the function returns the number of iterations as well, e.g.: `function` $[E_{\text{an}}, i_{\text{ter}}] = \text{kepler}(M_{\text{an}}, \text{ecc}_1)$. What variable could be introduced to the list of input arguments additionally? Test the correctness of the function on a $[M \times e]$-grid with resolution 0.01 for M and 0.001 for e. For each grid point, compute the difference between the eccentric anomaly and the mean anomaly $(E - M)$ and the number of iterations. Plot the two quantities over the test grid with `mesh` each into an own figure. Where do the anomalies differ maximally and minimally? Where does the solution need the maximal or minimal number of iterations? Compute the median of the number of iterations?

```
% to-do
%

function [E_an, i_ter] = kepler(M_an, ecc)
    tol = 1e-6;
    Eold = M_an;
    i_ter = 0;

    while true
        E_an = M_an + ecc * sin(Eold);
        i_ter = i_ter + 1;
        if abs(E_an - Eold) < tol
            break;
        end
        Eold = E_an;
    end
end
```

```
% to-do

Mgrid = 0:0.01:2*pi;
egrid = 0:0.001:1-0.001;

[Mmat, Emat] = meshgrid(Mgrid, egrid);

Esol = zeros(size(Mmat));
IT   = zeros(size(Mmat));

for i = 1:numel(Mmat)
    [Ei, it] = kepler(Mmat(i), Emat(i));
    Esol(i) = Ei;
    IT(i) = it;
end

Diff = Esol - Mmat;

figure; mesh(Mgrid, egrid, Diff);
title('E - M over (M,e) grid'); xlabel('M'); ylabel('e');
```




```
figure; mesh(Mgrid, egrid, IT);
title('Iterations over (M,e) grid'); xlabel('M'); ylabel('e');
```




```
[maxDiff, idx1] = max(Diff(:));
[minDiff, idx2] = min(Diff(:));

fprintf('Max difference at M=%f, e=%f\n', ...
        Mmat(idx1), Emat(idx1));
```

```
Max difference at M=0.570000, e=0.999000
```

```
fprintf('Min difference at M=%f, e=%f\n', ...
        Mmat(idx2), Emat(idx2));
```

```
Min difference at M=5.710000, e=0.999000
```

```
fprintf('Median iterations = %.2f\n', median(IT(:)));
```

```
Median iterations = 11.00
```

Exercise 4: Write a function that converts an angle in degree (decimal number) into degree (integer), minute (integer) and second (decimal number). Can you use this function to convert an angle in hour angle (decimal number) to hour, minute and decimal second as well? If so, what would have to be changed? Write a function that does the back conversion: [degree, minute, second] → degree. Make sure the function works for negative angles as well. Test both functions with two own examples by doing a forward and a consecutive backward transformation.

```
% to-do
%
function [d, m, s] = deg2dms(angle)
    sgn = sign(angle);
```

```
    angle = abs(angle);

    d = floor(angle);
    m = floor((angle - d)*60);
    s = (angle - d - m/60)*3600;

    d = sgn*d;
end
```

```
function angle = dms2deg(d,m,s)
    sgn = sign(d);
    d = abs(d);
    angle = sgn * (d+ m/60 + s/3600);
end
```

```
disp('Exercise 4 test');
```

```
Exercise 4 test
```

```


ang_deg1 = 23.2342;
ang_deg2 = -43.2134;

[d1, m1, s1] = deg2dms(ang_deg1);
[d2, m2, s2] = deg2dms(ang_deg2);

back_ang1 = dms2deg(d1, m1, s1)
```

```
back_ang1 =
23.2342
```

```
back_ang2 = dms2deg(d2,m2,s2)
```

```
back_ang2 =
-43.2134
```

Exercise 5: Write a function that computes the Julian Date and Modified Julian Date from a given Gregorian calendar date specified as year, month, day, hour, minute and second.

(a) Compute the Earth Rotation Angle (ERA) at 2025-04-10, 02h:00m:00s CET and specify it in degree, minute and second to one second precision.

(b) Compute the Greenwich Apparent Sidereal Time (GAST) at 2025-04-10, 02h:00m:00s CET and specify it in hour angle, minute and second to one second precision.

```
% to-do
%
```

```
function [JD, MJD] = julianDate(year, month, day, hour, minute, second)

    if month <= 2
        year = year - 1;
        month = month + 12;
    end

    A = floor(year / 100);
    B = 2 - A + floor(A / 4); % Gregorian calendar correction

    JDN = floor(365.25 * (year + 4716)) + floor(30.6001 * (month + 1)) +
day + B - 1524.5;

    % Convert time to fractional day (UT1/UTC)
    fractional_day = (hour + minute / 60 + second / 3600) / 24;

    % Julian Date
    JD = JDN + fractional_day;

    % Modified Julian Date (MJD = JD - 2400000.5)
    MJD = JD - 2400000.5;

end
```

```
function era = era_angle(JD)
    JD_0 = 2451545.0;
    delta_t = JD - JD_0;

    era = 2*pi*(0.7790572732640 + 1.00273781191135448 * delta_t);
    era = mod(era, 2*pi);
end
```

```
function gast = gast_angle(JD)
    T = (JD - 2451545.0)/36525;
    GMST = mod( 67310.54841 + ...
           (876600*3600 + 8640184.812866)*T + ...
           0.093104*T.^2 - 6.2e-6*T.^3 , 86400 );

    GMST = GMST * (pi/43200);

    eqeq = 0; % small term, ignored in simple model
    gast = mod(GMST + eqeq, 2*pi);
end
```

```
disp('Exercise 5 test');
```

```
Exercise 5 test
```

```
[JD, MJD] = julianDate(2025,4,10, 1,0,0);   % CET = UTC+1
ERA = era_angle(JD);
GAST = gast_angle(JD);

[Ed,Em,Es] = deg2dms(rad2deg(ERA));
fprintf('ERA = %d° %d'' %.1f"\n', Ed,Em,Es);
```

```
ERA = 213° 11' 45.3"
```

```
Gdeg = rad2deg(GAST)/15;   % convert rad → hours
[Hd,Hm,Hs] = deg2dms(Gdeg);
fprintf('GAST = %dh %dm %.1fs\n', Hd,Hm,Hs);
```

```
GAST = 14h 14m 4.7s
```

Exercise 6 (optional): The baseline length of the Westford – Wettzell baseline is about 6000 km in ITRF. At 2025-04-10, 12:00:00 UTC, a VLBI group delay Δ�� in TT is measured to be 20 ms on this baseline. Compute the difference of the delay in TT vs. TCB compatible seconds using the TT to TCG and TCG to TCB coordinate time transformations. Then specify the delay difference in meter. What solar system bodies are significant for the general relativistic transformation when taking 99.99% significance level into consideration? Show that the gravitational relativistic term is about twice as large as the special relativistic term.

```
% to-do
%
function dt = TT_to_TCG(TTsec)
    LG = 6.969290134e-10;
    dt = TTsec * LG;
end
```

```
function dt = TCG_to_TCB(TCGsec)
    LB = 1.550519768e-8;
    dt = TCGsec * LB;
end
```

```
TT = 0.020; % 20 ms
dt_TT_TCG = TT_to_TCG(TT);
dt_TCG_TCB = TCG_to_TCB(TT);
```

```
total = dt_TT_TCG + dt_TCG_TCB;

fprintf('Total offset = %.3e s\n', total);
```

```
Total offset = 3.240e-10 s
```

```
fprintf('In meters = %.3f m\n', total * 299792458);
```

```
In meters = 0.097 m
```

Attachment:

3D rotation Matrix

$$R_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}, \quad R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_1(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D reflexion Matrix

$$M_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad M_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

body (**A**)	$GM_A(m^3, s^{-2})$	mean distance to earth $\mid \vec{x_e} - \vec{x_a} \mid$ (km)
Sun	$1.32712440018 \bullet 10^{20}$	149,597,870.7 km (= 1 AU)
Mercury	$2.2032 \cdot 10^{13}$	91,691,000 km
Venus	$3.24859 \cdot 10^{14}$	41,400,000 km
Moon	$4.9048695 \cdot 10^{12}$	385,000.6 km
Mars	$4.282837 \cdot 10^{13}$	78,340,000 km
Ceres	$6.26325 \cdot 10^{10}$	414,000,000 km
Jupytar	$1.26686534 \cdot 10^{17}$	628,730,000 km
Saturn	$3.7931187 \cdot 10^{16}$	1,275,000,000 km
Uranus	$5.793939 \cdot 10^{15}$	2,723,950,000 km
Neptune	$6.836529 \cdot 10^{15}$	4,351,400,000 km
Pluto	$8.71 \cdot 10^{11}$	5,890,000,000 km
Eris	$1.108 \cdot 10^{12}$	96.1 AU

Use $30\ \mathrm{km}\ s^{-1}$ for the magnitude of the barycentric velocity of Earth v_e.