```
clear;
clc;
close all;
mag = @(vector) sqrt(vector(1)^2 + vector(2)^2 + vector(3)^2); % magnitude of a 3
element vector
% constants for the sun
duS = 1.4959965e8; % conversion from du sun to km
au = duS; % a sun du is sometimes called an au
tuS = 5.0226757e6; % converson from tu sun to seconds
muS = 1.3271544e11; % mu for sun km^3/s^2
vuS = duS/tuS;
% constants for earth
duE = 6378.136; % conversion from du earth to km
tuE = 806.8118744; % converson from tu earth to seconds
muE = 3.986004418e5; % mu for earth
% constants for mars
muM = 4.305e4
```

```
muM = 43050
```

```
duM = 3380;
rj = 778.5e6;
muJ = 126.687e6;

vj = sqrt(muS/rj)
```

```
vj = 13.0566
```

```

%problem1

phi = asin(0.7)
```

```
phi = 0.7754
```

```
v2infin = sqrt(8.6488^2+vj^2-2*8.6488*vj*cos(phi))
```

```
v2infin = 9.1646
```

```
vph1 = sqrt(9.049^2+2*muE/(duE+200))
```

```
vph1 = 14.2504
```

```
vce = sqrt(muE/(duE+200))
```

```
vce = 7.7843
```

```
Vph2 = sqrt(v2infin^2 + 2*muJ/(71492+200))
```

```
Vph2 = 60.1514
```

```
vcj = sqrt(muJ/71492)
```

```
vcj = 42.0957
```

```
dv1 = vph1 - vce
```

```
dv1 = 6.4661
```

```
dv2 = Vph2 - vcj
```

```
dv2 = 18.0557
```

```
dvtotal = dv1 + dv2
```

```
dvtotal = 24.5219
```

```

%problem2

v2prob2 = sqrt(2*muS/rj - 2*muS/(au + rj))
```

```
v2prob2 = 7.4133
```

```


vinfprob2 = vj - v2prob2
```

```
vinfprob2 = 5.6433
```

```
vescape = sqrt(2*muS/rj)
```

```
vescape = 18.4649
```

```
sdelta = 2*asind(1/(1+71492*(vinfprob2^2)/muJ))
```

```
sdelta = 158.4357
```

```
vmax = vinfprob2 + vj
```

```
vmax = 18.6999
```

```

%problem3

tof3 = 210*24*60*60/tuS;

r1 = [-0.707 -0.707 0]; % AU
v1 = [1.0423 -0.4124 0]; % AU/TU


[r2,v2] = timeOfFlight(r1,v1,tof3)
```

```
check = 1.0000
r2 = 1×3
    0.0000    1.6654         0
v2 = 1×3
   -0.6175    0.2751         0
```

```
elementsproblem3b = rv2elements(r2,v2,1)
```

```
elementsproblem3b = struct with fields:
```

```
          a: 1.3444
          e: 0.4617
          i: 0
   bigOmega: NaN
      omega: NaN
         nu: 142.2054
          u: NaN
```

```
vinf1 = (mag(v1) - 1)*vuS
```

```
vinf1 = 3.6016
```

```
vpeh = sqrt(vinf1^2+2*muE/(duE+200))
```

```
vpeh = 11.5828
```

```
vce = sqrt(muE/(duE+200))
```

```
vce = 7.7843
```

```
deltav1 = vpeh - vce
```

```
deltav1 = 3.7985
```

```
vinf2 = abs(mag(v2)*vuS - sqrt((2*muS)/249.3e6 - (2*muS)/(249.3e6+206.7e6)))
```

```
vinf2 = 1.8321
```

```
delta1 = 3522.2/vinf2*sqrt(vinf2^2 + 2*muM/3522.2)
```

```
delta1 = 1.0137e+04
```

```
delta2 = 3380/vinf2*sqrt(vinf2^2 + 2*muM/3380)
```

```
delta2 = 9.9056e+03
```

```
deltadelta = delta1 - delta2
```

```
deltadelta = 230.9589
```

```
vpmh = sqrt(vinf2^2 + 2*muM/(3380+400))
```

```
vpmh = 5.1122
```

```
vcm = sqrt(muM/(3380+400))
```

```
vcm = 3.3747
```

```
deltavm = vpmh - vcm
```

```
deltavm = 1.7374
```

```

%problem4

re0 = [0.9422 -0.3614 -1.49e-7]; % AU

ve0 = [0.3419 0.93 8.543e-7]; % AU/TU

```

```
rm0 = [1.3871 0.1765 -0.0304]; % AU

vm0 = [-0.0714 0.8766 0.0201]; % AU/TU

% departure time in tu
ti = 706*24*60*60/tuS;

% time of flight in tu
tof = 210*24*60*60/tuS;

% departure position and velocity of mars in heliocentric frame
[rd,vd] = timeOfFlight(rm0,vm0,ti)
```

```
check = 1.0000
rd = 1×3
    1.3386    0.4525   -0.0234
vd = 1×3
   -0.2292    0.8403    0.0232
```

```
% arrival time from start
ta = ti + tof;

% departure position and velocity of earth in heliocentric frame
[ra,va] = timeOfFlight(re0,ve0,ta)
```

```
check = 1.0000
ra = 1×3
   -0.9591    0.2526    0.0000
va = 1×3
   -0.2711   -0.9711   -0.0000
```

```
% earth nu
earthElements = rv2elements(ra,va,1)
```

```
earthElements = struct with fields:
           a: 0.9999
           e: 0.0167
           i: 5.0010e-05
    bigOmega: 348.7544
       omega: 114.5398
          nu: 61.9497
           u: 176.4894
```

```
earthElements.nu
```

```
ans = 61.9497
```

```
% mars nu
marsElements = rv2elements(rd,vd,1);
marsElements.nu
```

```
ans = 42.5160
```

```
% phase 2: we have to r vectors find v vectors with gauss's
[v1,v2] = gaussSolution(rd,ra,tof)
```

```
check = 1.0000
check = 1.0000
v1 = 1×3
   -0.2313    0.7202   -0.0192
v2 = 1×3
   -0.1696   -1.0696    0.0324
```

```
transferElements = rv2elements(rd,v1,1)
```

```
transferElements = struct with fields:
           a: 1.1866
           e: 0.1916
           i: 1.7241
    bigOmega: 165.2437
       omega: 37.2303
          nu: 176.2133
           u: 213.4436
```

```
transferElements = rv2elements(ra,v2,1)
```

```
transferElements = struct with fields:
           a: 1.1866
           e: 0.1916
           i: 1.7241
    bigOmega: 165.2437
       omega: 37.2303
          nu: 322.7698
           u: 1.0177e-04
```

```
vinf1 = mag(v1 - vd)*vuS;

vinf2 = mag(v2 - va)*vuS;

vph1 = sqrt(vinf1^2+2*muM/(duM+500));

vph2 = sqrt(vinf2^2+2*muE/(duE+300));
vcm = sqrt(muM/(duM+500));
vce = sqrt(muE/(duE+300));
deltav1 = vph1 - vcm;
deltav2 = vph2 - vce;
totaldeltav = deltav1 + deltav2
```

```
totaldeltav = 6.7415
```

```
function [V1s,V2s,V1l,V2l] = gaussSolution(r1,r2,tof)
    % this function finds the solution to gausses problem when given values
    % for position in DU and a TOF in TU. The s are the short way and
    % the l is the long way.

    zguess = 10;
    mag = @(vector) sqrt(vector(1)^2 + vector(2)^2 + vector(3)^2); % magnitude of a
3 element vector

    dtheta = acos(dot(r1,r2)/(mag(r1)*mag(r2))); % the angle between the position
vectors

    A = (sqrt(mag(r1)*mag(r2))*sin(dtheta))/sqrt(1-cos(dtheta));


    error = 1;
    znplus1 = zguess;

    while error > 1*10^(-5)
        [s,c] = fofZ(znplus1);

        y = mag(r1) + mag(r2) - A*(1-znplus1*s)/sqrt(c);

       x = sqrt(y/c);

       t = x^3*s + A*sqrt(y);

       error = abs(tof - t);


        dsdz = (c - 3*s)/2*znplus1;

        dcdz = (1 - znplus1*s - 2*c)/2*znplus1;

       dtdz = x^3 * (dsdz - 3*s*dcdz/(2*c)) + A/8 * (3*s*sqrt(y)/c + A/x);

      znplus1 = znplus1 + (tof-t)/dtdz;

    end

    f = 1 - y/mag(r1);
    fdot = -x/(mag(r1)*mag(r2))*(1-znplus1*s);
    g = A*sqrt(y);
    gdot = 1 - y/mag(r2);

    V1s = (r2 - f*r1)/g;
    V2s = (gdot*r2 - r1)/g;


    check = f*gdot-fdot*g

    dthetal = 2*pi-dtheta; % the angle between the position vectors

    Al = (sqrt(mag(r1)*mag(r2))*sin(dthetal))/sqrt(1-cos(dthetal));
    error = 1;
    znplus1 = zguess;

    while error > 1*10^(-5)
        [s,c] = fofZ(znplus1);

        y = mag(r1) + mag(r2) - Al*(1-znplus1*s)/sqrt(c);

       x = sqrt(y/c);

       t = x^3*s + Al*sqrt(y);

       error = abs(tof - t);

        dsdz = (c - 3*s)/2*znplus1;

        dcdz = (1 - znplus1*s - 2*c)/2*znplus1;

       dtdz = x^3 * (dsdz - 3*s*dcdz/(2*c)) + Al/8 * (3*s*sqrt(y)/c + Al/x);

      znplus1 = znplus1 + (tof-t)/dtdz;

    end

    f = 1 - y/mag(r1);
    fdot = -x/(mag(r1)*mag(r2))*(1-znplus1*s);
    g = Al*sqrt(y);
    gdot = 1 - y/mag(r2);

    V1l = (r2 - f*r1)/g;
    V2l = (gdot*r2 - r1)/g;


    check = f*gdot - fdot*g

    function [s,c] = fofZ(z) % this function finds the s and c values for a given z

    if z > 0.1
        s = (sqrt(z)-sin(sqrt(z)))/z^(3/2);
        c = (1-cos(sqrt(z)))/z;
    elseif z < -0.1
        s = (sinh(-z)-sqrt(-z))/(-z)^(3/2);
        c = (1 - cosh(-z))/z;
    else
        s = 1/6 - z/factorial(5) + z^2/factorial(7) - z^3/factorial(9);
        c = 1/2 - z/factorial(4) + z^2/factorial(6) - z^3/factorial(8);

    end
    end
```

```
end

function [r1,v1] = timeOfFlight(r0,v0,tof)
% this function finds some new r and v vectors from an initial vector and a
% time of flight everything must be in canonical units

    xguess = 1000;

    mag = @(vector) sqrt(vector(1)^2 + vector(2)^2 + vector(3)^2); % magnitude of a
3 element vector

    elements = rv2elements(r0,v0,1);



    error = 1;
    x = xguess;

    while error > 1*10^(-5)

        znplus1 = x^2/elements.a;

        [s,c] = fofZ(znplus1);

        t = x^3*s + dot(r0,v0)*x^2*c+mag(r0)*x*(1-znplus1*s);

        rn = x^2*c+dot(r0,v0)*x*(1-znplus1*s)+mag(r0)*(1-znplus1*c);

        error = abs(tof - t);

        dtdx = rn;

        x = x + (tof-t)/dtdx;


    end

    f = 1 - elements.a/mag(r0)*(1 - cos(x/sqrt(elements.a)));

    fdot = -sqrt(elements.a)/(mag(r0)*rn)*sin(x/sqrt(elements.a));

    g = (elements.a)^2/sqrt(elements.a)*(dot(r0,v0)/sqrt(elements.a)*(1-cos(x/
sqrt(elements.a)))+mag(r0)/elements.a*sin(x/sqrt(elements.a)));

    gdot = 1 - elements.a/rn + elements.a/rn*cos(x/sqrt(elements.a));

    check = f*gdot-fdot*g

    r1 = f*r0 + g*v0;
    v1 = fdot*r0 + gdot*v0;
    function [s,c] = fofZ(z) % this function finds the s and c values for a given z

    if z > 0.1
        s = (sqrt(z)-sin(sqrt(z)))/z^(3/2);
        c = (1-cos(sqrt(z)))/z;
    elseif z < -0.1
        s = (sinh(-z)-sqrt(-z))/(-z)^(3/2);
        c = (1 - cosh(-z))/z;
    else
        s = 1/6 - z/factorial(5) + z^2/factorial(7) - z^3/factorial(9);
        c = 1/2 - z/factorial(4) + z^2/factorial(6) - z^3/factorial(8);

    end
    end

end
```