```
clc, clear,close all
```

In '*myTesi.m*', come già specificato, sono raccolte le funzioni più utilizzate.

```
f = myTesi;
data = importdata("gtoc9_debris.xlsx");
data = data.data.gtoc9_debris(:,1:end-1);

% Importazione dei dati:
ddata = mat2cell(data,size(data,1),ones(1,size(data,2)));
```

Dopo aver lanciato un po' di volte il codice per la ricerca delle possibili sequenze, sono state salvate le migliori in diversi file.mat. Qui ne viene caricata una e si fa un controllo per assicurarsi che nelle sequenze salvate non si ripeta nessun detrito.

In *storia* sono salvate le sequenze che mostrano i detriti con il rispettivo istante temporale in cui vengono raccolti.

```
load tatoo2
vabene = true(1,length(tatoo));
for j=1:length(tatoo)
   if length(unique(tatoo{j}))~=length(tatoo{j})
       vabene(j)  = false;
   end
end

tatoo = tatoo(vabene);
storia = storia(vabene);

disp(length(tatoo))
```

```
    18
```

```
disp(tatoo)
```

```
    {1×94  uint8}
    {1×102 uint8}
    {1×90  uint8}
    {1×91  uint8}
    {1×93  uint8}
    {1×94  uint8}
    {1×91  uint8}
    {1×97  uint8}
    {1×91  uint8}
    {1×93  uint8}
    {1×98  uint8}
    {1×89  uint8}
    {1×102 uint8}
    {1×96  uint8}
    {1×97  uint8}
    {1×92  uint8}
    {1×90  uint8}
    {1×94  uint8}
```

```
idx = 2; % l'indice 2 corrisponde alla missione in tatoo che in questo caso raccoglie 102 detri
```

```
tatoo = tatoo(idx);
storia = storia(idx);

c.start_ep = 23467; % MJD2000
c.stop_ep  = 26419; % MJD2000
c.n  = 123; % numero detriti
c.Mdry  = 2000; % dry mass
c.Mde = 30; % massa del kit di de - orbit
c.G0 = 9.80665; % accelerazione di gravità
c.Isp = 340; % impulso specifico
```

Come già detto, si suddivide la linea atemporale di 8 anni in un numero finito di frames (300).

```
frames = 300;
t = linspace(c.start_ep,c.stop_ep,frames);
disp(storia)
```

```
||34.001_39.004_82.007_63.009_104.012_115.013_23.014_96.017_17.018_119.021_58.022_97.023_122.026_51.028_24.031_118.0
```

Conversione dai frames ai giorni:

```
st = strrep(storia,'||','_');
st = strsplit(st,'_');
st(1) = [];
istanti = t(getTime(st));

for i=2:length(istanti)
   if istanti(i)<=istanti(i-1)
    istanti(i) = istanti(i-1)+5.1;
   end
end

deltaT = diff([c.start_ep istanti]);
disp(deltaT)
```

```
  Columns 1 through 6

         0   29.6187   29.6187   19.7458   29.6187    9.8729

  Columns 7 through 12

    9.8729   29.6187    9.8729   29.6187    9.8729    9.8729

  Columns 13 through 18

   29.6187   19.7458   29.6187   29.6187   29.6187    9.8729

  Columns 19 through 24

    9.8729   29.6187   29.6187   29.6187    9.8729    9.8729

  Columns 25 through 30

   29.6187   19.7458   39.4916   19.7458    9.8729    9.8729

  Columns 31 through 36
```

```
   29.6187   29.6187    9.8729   29.6187   29.6187    9.8729

  Columns 37 through 42

    9.8729   19.7458    9.8729   29.6187    9.8729   19.7458

  Columns 43 through 48

   19.7458   29.6187   29.6187   29.6187    9.8729    9.8729

  Columns 49 through 54

    9.8729   39.4916   29.6187    9.8729   29.6187   19.7458

  Columns 55 through 60

    9.8729   19.7458   19.7458   29.6187   29.6187    9.8729

  Columns 61 through 66

   29.6187   19.7458   19.7458   29.6187   29.6187   19.7458

  Columns 67 through 72

   39.4916    9.8729   29.6187   29.6187    9.8729    9.8729

  Columns 73 through 78

   29.6187   29.6187   39.4916    9.8729    9.8729    9.8729

  Columns 79 through 84

    9.8729    9.8729    9.8729   19.7458   29.6187   29.6187

  Columns 85 through 90

    9.8729    9.8729    9.8729   98.7291   29.6187    9.8729

  Columns 91 through 96

    9.8729    9.8729   29.6187    9.8729  276.4415    9.8729

  Columns 97 through 102

   19.7458   29.6187  335.6789    9.8729   29.6187    9.8729
```

```
subs = strsplit(storia,'||');
subs(1) = [];
nodi = arrayfun(@(x) strsplit(x,'_'),subs,'UniformOutput', false);
```

La missione globale si suddivide in sottomissioni. Dal grafo *DV* si prende il peso in termini in ΔV, in modo da calcolare la massa iniziale di ogni sottmissione.

```
deltaV = cell(length(nodi),1);
massa = zeros(1,length(nodi));
for i = 1:length(nodi)
    idxOut = findedge(DV,nodi{i}(1:end-1),nodi{i}(2:end));
    w = DV.Edges.Weight(idxOut);
    deltaV{i} = w;
```

```
    massa(i) = massaMission(c.Mdry, c.Mde, deltaV{i}, c.G0, c.Isp);
end
disp(round(massa))
```

```
  Columns 1 through 5

       19165       13275        8613        3817        6012

  Columns 6 through 8

        3566        2688        2755
```

Se la massa della sottomissione è maggiore di *maxMassa,* la sottomissione viene divisa in due sottomissioni.

```
mission = struct('n',[]);
mission.list = getNode(st);
mission.pat = {};
maxMassa = 8000;
for i = 1:length(nodi)
if massa(i)<maxMassa
 mission.n = cat(2,mission.n,length(nodi{i}));
 mission.pat{end+1} = nodi{i};
else
 a = tagliaLegame(nodi{i}); %fix(length(nodi{i})/2);
 b = length(nodi{i})-a;
 mission.n = cat(2,mission.n,[a b]);
 mission.pat{end+1} = nodi{i}(1:a);
 mission.pat{end+1} = nodi{i}(a+1:end);
 disp('mission Splitted')
end
end
```

```
mission Splitted
mission Splitted
mission Splitted
```

```
fprintf('detriti di partenza: %d\n',sum(mission.n));
```

```
detriti di partenza: 102
```

mission è un oggetto con cui monitorare la suddivisione delle sottomissioni, il numero detriti, l'identificativo di ogni detrito, i tempi in cui vengono raccolti, i costi dei passaggi orbitali, le masse, il costo in MEUR e il rispetto delle condizioni di massa (che si trova nella property *feas*).

```
mission.feas = ones(length(mission.pat),1);
```

Qui vengono creati i vincoli di ottimizzazione (la funzione *creaConstraints* verrà poi trattata in un altro momento):

```
vals = [max(deltaT)+10 10 30 deltaT(1) mean(deltaT(2:end))];
[lb,ub,~,~,~] = f.creaConstraints(vals,c,mission.n);

    deltaT(deltaT'<lb) = lb(deltaT'<lb);
```

```
    deltaT(deltaT'>ub) = ub(deltaT'>ub);
```

```
clear DV
t = deltaT;
[~, memo] = f.theMission(t,ddata,mission,c); % calcola il dv a seconda della lista di detriti,
% restituisce le proprietà di mission
M = memo;
attuali = sum(M.n);
cerca = true;
%memo.estM
```

L'obiettivo ora è incrementare la sequenza, cercare di prendere più detriti. Il passo successivo sarà quello di ottimizzare i costi.

Per cercare di incrementare la sequenza si usa la funzione *tabbaBuchi* (che si trova in fondo alla pagina). Il processo termina se tutti i detriti vengono raccolti o se la funzione non trova il modo di inserirne altri. L'aggiunta dei detriti procede con miglioramenti localizzati senza ottimizzare globalmente la sequenza.

```
while cerca
    [t,M] = tappaBuchi(t,M,ddata,f,c);
    [~, M] = f.theMission(t,ddata,M,c);
    cerca = sum(M.n)>attuali && sum(M.n)<123;
    attuali = sum(M.n);
    fprintf('detriti attuali: %d\n', attuali);
end
```

```
sto cercando di ampliare la missione prima del nodo #73
metto nella 11° submission il detrito #50
prevedo 3171.9 [kg]
detriti attuali: 103
sto cercando di ampliare la missione prima del nodo #50
metto nella 11° submission il detrito #16
prevedo 3459.7 [kg]
detriti attuali: 104
sto cercando di ampliare la missione prima del nodo #74
metto nella 10° submission il detrito #111
prevedo 3487.9 [kg]
detriti attuali: 105
sto cercando di ampliare la missione prima del nodo #111
metto nella 10° submission il detrito #116
prevedo 4412.0 [kg]
detriti attuali: 106
sto cercando di ampliare la missione prima del nodo #116
metto nella 10° submission il detrito #98
prevedo 4739.7 [kg]
detriti attuali: 107
sto cercando di ampliare la missione prima del nodo #9
metto nella 9° submission il detrito #77
prevedo 5026.9 [kg]
detriti attuali: 108
sto cercando di ampliare la missione prima del nodo #77
non riesco ad introdurre detriti: dV troppo alti
introduco una submission dopo la 10°
prevedo 5595 kg per 5 detriti
detriti attuali: 113
sto cercando di ampliare la missione prima del nodo #98
non riesco ad introdurre detriti: dV troppo alti
```

```
introduco una submission alla fine
prevedo 5552 kg per 6 detriti
detriti attuali: 119
sto cercando di ampliare la missione prima del nodo #49
non riesco ad introdurre detriti: dV troppo alti
introduco una submission dopo la 9°
prevedo 4346 kg per 4 detriti
detriti attuali: 123
```

Tutti i detriti ora sono stati raccolti. Si ottimizza ora il costo con **fmincon**, una funzione di MATLAB per l'ottimizzazione di problemi non lineari soggetti a vincoli (https://www.mathworks.com/help/optim/ug/fmincon.html).

```
vals = [max(t)+10 10 30 t(1) mean(t(2:end))];
[lb,ub,~,Adis,Bdis] = f.creaConstraints(vals,c,M.n);

t(t'<lb) = lb(t'<lb);
t(t'>ub) = ub(t'>ub);

%[lb,ub,t0,t']

opt_fun = @(t_vet) f.theMission(t_vet,ddata,M,c);
options = optimoptions('fmincon','Display','iter',...
               'MaxFunctionEvaluations',9e5,...
               'MaxIter',500,'HonorBounds',false);

[t_sol, ~, ~] = fmincon(opt_fun,t,Adis,Bdis,[],[],lb,ub,[],options);
```

```
                                          First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0     124    9.793454e+02    0.000e+00    1.919e+00
    1     251    9.786334e+02    0.000e+00    2.942e+00    8.804e-01
    2     376    9.737371e+02    0.000e+00    2.534e+00    2.228e+00
    3     500    9.671984e+02    0.000e+00    6.133e-01    2.531e+00
    4     624    9.575258e+02    0.000e+00    1.097e+00    7.439e+00
    5     748    9.510515e+02    0.000e+00    1.203e+00    7.821e+00
    6     872    9.456523e+02    0.000e+00    9.306e-01    5.874e+00
    7     996    9.401849e+02    0.000e+00    3.800e-01    4.783e+00
    8    1120    9.366868e+02    0.000e+00    2.632e-01    4.848e+00
    9    1244    9.344622e+02    0.000e+00    3.135e-01    3.823e+00
   10    1368    9.329316e+02    0.000e+00    3.413e-01    3.579e+00
   11    1492    9.299672e+02    0.000e+00    3.159e-01    8.284e+00
   12    1616    9.292746e+02    0.000e+00    3.414e-01    1.615e+00
   13    1740    9.269940e+02    0.000e+00    2.166e-01    6.873e+00
   14    1864    9.260984e+02    0.000e+00    2.225e-01    3.309e+00
   15    1988    9.254323e+02    0.000e+00    2.347e-01    2.344e+00
   16    2112    9.249984e+02    0.000e+00    2.576e-01    2.862e+00
   17    2236    9.236941e+02    0.000e+00    2.270e-01    3.959e+00
   18    2360    9.227876e+02    0.000e+00    3.756e-01    3.751e+00
   19    2484    9.204549e+02    0.000e+00    6.502e-01    8.082e+00
   20    2608    9.180935e+02    0.000e+00    9.009e-01    8.720e+00
   21    2732    9.165480e+02    0.000e+00    8.423e-01    4.856e+00
   22    2856    9.161248e+02    0.000e+00    3.904e-01    3.884e+00
   23    2980    9.154207e+02    0.000e+00    1.578e-01    2.365e+00
   24    3104    9.152851e+02    0.000e+00    1.756e-01    1.326e+00
   25    3228    9.149494e+02    0.000e+00    2.568e-01    2.159e+00
   26    3352    9.143253e+02    0.000e+00    2.995e-01    3.831e+00
   27    3476    9.135148e+02    0.000e+00    2.174e-01    4.544e+00
   28    3600    9.130839e+02    0.000e+00    1.553e-01    1.932e+00
```

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
29	3724	9.123654e+02	0.000e+00	1.931e-01	3.473e+00
30	3848	9.120606e+02	0.000e+00	1.548e-01	2.322e+00
31	3972	9.115486e+02	0.000e+00	2.076e-01	4.732e+00
32	4096	9.114325e+02	0.000e+00	1.781e-01	2.460e+00
33	4220	9.111749e+02	0.000e+00	1.766e-01	3.750e+00
34	4344	9.109485e+02	0.000e+00	1.734e-01	2.687e+00
35	4468	9.105909e+02	0.000e+00	2.302e-01	3.276e+00
36	4592	9.101797e+02	0.000e+00	2.347e-01	3.934e+00
37	4716	9.097909e+02	0.000e+00	2.342e-01	4.519e+00
38	4840	9.094955e+02	0.000e+00	1.502e-01	4.504e+00
39	4964	9.093275e+02	0.000e+00	1.887e-01	3.312e+00
40	5088	9.092709e+02	0.000e+00	1.730e-01	1.883e+00
41	5212	9.092001e+02	0.000e+00	1.386e-01	2.145e+00
42	5336	9.090800e+02	0.000e+00	1.114e-01	2.315e+00
43	5460	9.089295e+02	0.000e+00	1.376e-01	1.939e+00
44	5584	9.087767e+02	0.000e+00	1.000e-01	1.495e+00
45	5708	9.062595e+02	0.000e+00	2.308e-01	5.944e+00
46	5832	9.045648e+02	0.000e+00	1.506e-01	2.609e+00
47	5956	9.043560e+02	0.000e+00	1.062e-01	1.335e+00
48	6080	9.039958e+02	0.000e+00	1.136e-01	1.571e+00
49	6204	9.038233e+02	0.000e+00	1.220e-01	9.546e-01
50	6328	9.036044e+02	0.000e+00	9.031e-02	1.530e+00
51	6452	9.035395e+02	0.000e+00	6.070e-02	8.749e-01
52	6576	9.034299e+02	0.000e+00	5.391e-02	1.283e+00
53	6700	9.033562e+02	0.000e+00	7.162e-02	1.024e+00
54	6824	9.032073e+02	0.000e+00	6.741e-02	1.748e+00
55	6948	9.031944e+02	0.000e+00	5.853e-02	5.141e-01
56	7072	9.031420e+02	0.000e+00	3.111e-02	9.883e-01
57	7196	9.031217e+02	0.000e+00	3.647e-02	8.491e-01
58	7320	9.030745e+02	0.000e+00	4.700e-02	1.196e+00
59	7444	9.030467e+02	0.000e+00	4.138e-02	8.120e-01
60	7568	9.030061e+02	0.000e+00	3.050e-02	1.001e+00
61	7692	9.029645e+02	0.000e+00	4.274e-02	1.014e+00
62	7816	9.029408e+02	0.000e+00	3.979e-02	7.176e-01
63	7940	9.029262e+02	0.000e+00	2.976e-02	6.260e-01
64	8064	9.029047e+02	0.000e+00	3.955e-02	8.514e-01
65	8188	9.028718e+02	0.000e+00	6.589e-02	1.166e+00
66	8312	9.028313e+02	0.000e+00	7.278e-02	1.230e+00
67	8436	9.028146e+02	0.000e+00	3.374e-02	5.889e-01
68	8560	9.028071e+02	0.000e+00	2.829e-02	3.393e-01
69	8684	9.027969e+02	0.000e+00	2.985e-02	2.421e-01
70	8808	9.027845e+02	0.000e+00	2.605e-02	2.980e-01
71	8932	9.027751e+02	0.000e+00	2.051e-02	3.619e-01
72	9056	9.027652e+02	0.000e+00	2.900e-02	5.080e-01
73	9180	9.027532e+02	0.000e+00	2.856e-02	5.545e-01
74	9304	9.027357e+02	0.000e+00	2.559e-02	6.884e-01
75	9428	9.027113e+02	0.000e+00	2.516e-02	9.463e-01
76	9552	9.026844e+02	0.000e+00	4.147e-02	1.027e+00
77	9676	9.026629e+02	0.000e+00	4.263e-02	8.830e-01
78	9800	9.026353e+02	0.000e+00	3.123e-02	1.104e+00
79	9924	9.025804e+02	0.000e+00	6.895e-02	1.927e+00
80	10048	9.024922e+02	0.000e+00	1.019e-01	2.412e+00
81	10172	9.024079e+02	0.000e+00	1.027e-01	1.767e+00
82	10296	9.023335e+02	0.000e+00	7.869e-02	1.342e+00
83	10420	9.023257e+02	0.000e+00	3.417e-02	6.033e-01
84	10544	9.023131e+02	0.000e+00	3.375e-02	7.893e-01
85	10668	9.023064e+02	0.000e+00	4.365e-02	3.885e-01
86	10792	9.022870e+02	0.000e+00	5.369e-02	6.318e-01
87	10916	9.022801e+02	0.000e+00	3.962e-02	3.481e-01

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
88	11040	9.022728e+02	0.000e+00	2.000e-02	3.981e-01
89	11164	9.012952e+02	0.000e+00	2.541e-02	2.675e+00
90	11288	9.011845e+02	0.000e+00	3.137e-02	1.163e+00
91	11412	9.011621e+02	0.000e+00	2.364e-02	3.687e-01
92	11536	9.011482e+02	0.000e+00	3.076e-02	4.520e-01
93	11660	9.011190e+02	0.000e+00	3.349e-02	1.015e+00
94	11784	9.011016e+02	0.000e+00	2.579e-02	1.150e+00
95	11908	9.010816e+02	0.000e+00	2.937e-02	1.404e+00
96	12032	9.010748e+02	0.000e+00	2.328e-02	4.129e-01
97	12156	9.010601e+02	0.000e+00	1.262e-02	4.508e-01
98	12280	9.010540e+02	0.000e+00	1.537e-02	2.778e-01
99	12404	9.010448e+02	0.000e+00	1.703e-02	3.541e-01
100	12528	9.010362e+02	0.000e+00	1.503e-02	5.256e-01
101	12652	9.010300e+02	0.000e+00	1.533e-02	5.900e-01
102	12776	9.010261e+02	0.000e+00	2.146e-02	4.588e-01
103	12900	9.010196e+02	0.000e+00	3.262e-02	6.309e-01
104	13024	9.010097e+02	0.000e+00	3.310e-02	7.497e-01
105	13148	9.009990e+02	0.000e+00	2.447e-02	6.603e-01
106	13272	9.009900e+02	0.000e+00	1.665e-02	4.722e-01
107	13396	9.009826e+02	0.000e+00	2.852e-02	3.677e-01
108	13520	9.009773e+02	0.000e+00	2.981e-02	2.917e-01
109	13644	9.009735e+02	0.000e+00	1.816e-02	3.368e-01
110	13768	9.009706e+02	0.000e+00	1.088e-02	3.858e-01
111	13892	9.009672e+02	0.000e+00	2.354e-02	4.221e-01
112	14016	9.009633e+02	0.000e+00	2.706e-02	4.562e-01
113	14140	9.009581e+02	0.000e+00	1.946e-02	5.277e-01
114	14264	9.009521e+02	0.000e+00	1.029e-02	5.446e-01
115	14388	9.009482e+02	0.000e+00	1.721e-02	3.340e-01
116	14512	9.009463e+02	0.000e+00	1.762e-02	1.956e-01
117	14636	9.009451e+02	0.000e+00	1.366e-02	2.002e-01
118	14760	9.009434e+02	0.000e+00	1.033e-02	2.609e-01
119	14884	9.009416e+02	0.000e+00	1.821e-02	3.269e-01
120	15008	9.009400e+02	0.000e+00	1.996e-02	3.158e-01
121	15132	9.009378e+02	0.000e+00	1.007e-02	2.952e-01
122	15256	9.009356e+02	0.000e+00	1.113e-02	3.028e-01
123	15380	9.009345e+02	0.000e+00	1.540e-02	2.510e-01
124	15504	9.009333e+02	0.000e+00	1.339e-02	2.994e-01
125	15628	9.009326e+02	0.000e+00	9.482e-03	2.038e-01
126	15752	9.009324e+02	0.000e+00	7.143e-03	1.289e-01
127	15876	9.009318e+02	0.000e+00	7.255e-03	1.250e-01
128	16000	9.009310e+02	0.000e+00	9.885e-03	1.505e-01
129	16124	9.009305e+02	0.000e+00	1.081e-02	1.923e-01
130	16248	9.009296e+02	0.000e+00	1.159e-02	2.088e-01
131	16372	9.009295e+02	0.000e+00	7.898e-03	1.982e-01
132	16496	9.009289e+02	0.000e+00	9.406e-03	1.950e-01
133	16620	9.009284e+02	0.000e+00	1.224e-02	2.217e-01
134	16744	9.009270e+02	0.000e+00	1.399e-02	3.109e-01
135	16868	9.009257e+02	0.000e+00	1.255e-02	3.677e-01
136	16992	9.009237e+02	0.000e+00	1.601e-02	4.120e-01
137	17116	9.009216e+02	0.000e+00	1.770e-02	4.083e-01
138	17240	9.009190e+02	0.000e+00	1.942e-02	5.146e-01
139	17364	9.009153e+02	0.000e+00	1.654e-02	6.296e-01
140	17488	9.009112e+02	0.000e+00	2.587e-02	7.042e-01
141	17612	9.009068e+02	0.000e+00	3.046e-02	6.827e-01
142	17736	9.009030e+02	0.000e+00	2.154e-02	5.743e-01
143	17860	9.008995e+02	0.000e+00	1.676e-02	4.681e-01
144	17984	9.008972e+02	0.000e+00	1.472e-02	3.879e-01
145	18108	9.008951e+02	0.000e+00	1.783e-02	3.370e-01
146	18232	9.008944e+02	0.000e+00	1.219e-02	1.896e-01

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
147	18356	9.008937e+02	0.000e+00	1.031e-02	1.056e-01
148	18480	9.008929e+02	0.000e+00	6.954e-03	1.274e-01
149	18604	9.008920e+02	0.000e+00	8.857e-03	1.108e-01
150	18728	9.008912e+02	0.000e+00	9.583e-03	9.444e-02

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
151	18852	9.008902e+02	0.000e+00	8.160e-03	1.062e-01
152	18976	9.008897e+02	0.000e+00	7.421e-03	1.049e-01
153	19100	9.008891e+02	0.000e+00	5.839e-03	8.107e-02
154	19224	9.008886e+02	0.000e+00	5.566e-03	7.287e-02
155	19348	9.008885e+02	0.000e+00	4.659e-03	4.913e-02
156	19472	9.008885e+02	0.000e+00	4.409e-03	6.568e-02
157	19596	9.008883e+02	0.000e+00	4.000e-03	6.142e-02
158	19720	9.006516e+02	0.000e+00	7.846e-03	1.586e+00
159	19844	9.006320e+02	0.000e+00	1.184e-02	3.081e-01
160	19968	9.006294e+02	0.000e+00	1.380e-02	1.680e-01
161	20092	9.006288e+02	0.000e+00	9.797e-03	6.620e-02
162	20216	9.006277e+02	0.000e+00	4.279e-03	1.468e-01
163	20340	9.006274e+02	0.000e+00	7.104e-03	1.249e-01
164	20464	9.006271e+02	0.000e+00	7.039e-03	1.082e-01
165	20588	9.006267e+02	0.000e+00	4.578e-03	6.381e-02
166	20712	9.006265e+02	0.000e+00	2.448e-03	4.568e-02
167	20836	9.006263e+02	0.000e+00	4.774e-03	4.958e-02
168	20960	9.006261e+02	0.000e+00	6.255e-03	6.172e-02
169	21084	9.006258e+02	0.000e+00	5.888e-03	7.877e-02
170	21208	9.006256e+02	0.000e+00	3.287e-03	7.822e-02
171	21332	9.006255e+02	0.000e+00	2.353e-03	6.624e-02
172	21456	9.006255e+02	0.000e+00	4.784e-03	5.459e-02
173	21580	9.006254e+02	0.000e+00	5.211e-03	5.360e-02
174	21704	9.006252e+02	0.000e+00	3.606e-03	6.040e-02
175	21828	9.006251e+02	0.000e+00	2.162e-03	5.348e-02
176	21952	9.006250e+02	0.000e+00	1.562e-03	3.491e-02
177	22076	9.006248e+02	0.000e+00	1.872e-03	2.853e-02
178	22200	9.006248e+02	0.000e+00	1.959e-03	3.218e-02
179	22324	9.006247e+02	0.000e+00	1.656e-03	4.393e-02
180	22448	9.006247e+02	0.000e+00	1.852e-03	5.316e-02

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
181	22572	9.006247e+02	0.000e+00	1.948e-03	5.179e-02
182	22696	9.006246e+02	0.000e+00	1.277e-03	3.124e-02
183	22820	9.006246e+02	0.000e+00	1.151e-03	2.655e-02
184	22944	9.006245e+02	0.000e+00	1.864e-03	3.181e-02
185	23068	9.006246e+02	0.000e+00	2.518e-03	3.811e-02
186	23192	9.006245e+02	0.000e+00	2.342e-03	3.923e-02
187	23316	9.006245e+02	0.000e+00	1.453e-03	3.901e-02
188	23440	9.006245e+02	0.000e+00	1.256e-03	3.652e-02
189	23564	9.006244e+02	0.000e+00	1.356e-03	1.430e-02
190	23688	9.006245e+02	0.000e+00	8.000e-04	2.277e-02
191	23812	9.005729e+02	0.000e+00	4.675e-03	4.222e-01
192	23936	9.005719e+02	0.000e+00	3.542e-03	1.453e-01
193	24060	9.005718e+02	0.000e+00	2.235e-03	4.710e-02
194	24184	9.005717e+02	0.000e+00	1.017e-03	1.384e-02
195	24308	9.005717e+02	0.000e+00	2.069e-03	2.764e-02
196	24432	9.005717e+02	0.000e+00	2.520e-03	2.530e-02
197	24556	9.005716e+02	0.000e+00	2.431e-03	3.990e-02
198	24680	9.005716e+02	0.000e+00	1.752e-03	3.439e-02
199	24804	9.005715e+02	0.000e+00	6.998e-04	4.443e-02
200	24928	9.005715e+02	0.000e+00	9.632e-04	2.567e-02
201	25052	9.005715e+02	0.000e+00	1.067e-03	2.886e-02
202	25176	9.005715e+02	0.000e+00	8.469e-04	1.991e-02
203	25300	9.005715e+02	0.000e+00	6.294e-04	1.964e-02
204	25424	9.005715e+02	0.000e+00	1.016e-03	2.899e-02
205	25548	9.005715e+02	0.000e+00	1.041e-03	2.371e-02

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
206	25672	9.005714e+02	0.000e+00	7.425e-04	3.032e-02
207	25796	9.005714e+02	0.000e+00	7.553e-04	2.398e-02
208	25920	9.005714e+02	0.000e+00	1.106e-03	2.728e-02
209	26044	9.005714e+02	0.000e+00	1.070e-03	2.754e-02
210	26168	9.005714e+02	0.000e+00	6.294e-04	1.982e-02
211	26292	9.005713e+02	0.000e+00	7.921e-04	2.044e-02
212	26416	9.005714e+02	0.000e+00	7.143e-04	6.914e-03
213	26540	9.005714e+02	0.000e+00	5.159e-04	1.049e-02
214	26664	9.005713e+02	0.000e+00	5.334e-04	9.609e-03
215	26788	9.005713e+02	0.000e+00	9.940e-04	1.255e-02
216	26912	9.005713e+02	0.000e+00	1.237e-03	2.157e-02
217	27036	9.005713e+02	0.000e+00	1.033e-03	1.912e-02
218	27160	9.005713e+02	0.000e+00	4.991e-04	1.252e-02
219	27284	9.005713e+02	0.000e+00	4.369e-04	6.648e-03
220	27408	9.005713e+02	0.000e+00	7.403e-04	9.388e-03
221	27532	9.005713e+02	0.000e+00	6.819e-04	7.908e-03
222	27656	9.005713e+02	0.000e+00	4.261e-04	1.946e-02
223	27780	9.005713e+02	0.000e+00	3.466e-04	5.691e-03
224	27904	9.005713e+02	0.000e+00	8.017e-04	2.237e-02
225	28028	9.005713e+02	0.000e+00	6.441e-04	5.620e-03
226	28152	9.005713e+02	0.000e+00	3.800e-04	2.101e-02
227	28276	9.005713e+02	0.000e+00	2.640e-04	1.212e-02
228	28400	9.005713e+02	0.000e+00	4.713e-04	2.910e-02
229	28524	9.005713e+02	0.000e+00	3.381e-04	7.560e-03
230	28648	9.005713e+02	0.000e+00	2.396e-04	8.587e-03
231	28772	9.005713e+02	0.000e+00	2.726e-04	8.398e-03
232	28896	9.005713e+02	0.000e+00	5.090e-04	2.451e-03
233	29022	9.005713e+02	0.000e+00	1.600e-04	9.052e-03
234	29146	9.005609e+02	0.000e+00	2.321e-03	1.280e-01
235	29270	9.005607e+02	0.000e+00	1.199e-03	6.506e-02
236	29394	9.005607e+02	0.000e+00	2.377e-03	4.041e-02
237	29518	9.005606e+02	0.000e+00	1.899e-03	5.527e-02
238	29642	9.005606e+02	0.000e+00	1.754e-03	6.125e-02
239	29766	9.005606e+02	0.000e+00	9.064e-04	2.090e-02
240	29890	9.005606e+02	0.000e+00	2.659e-04	1.522e-02
241	30014	9.005606e+02	0.000e+00	2.958e-04	3.454e-03
242	30138	9.005606e+02	0.000e+00	2.469e-04	1.294e-02
243	30262	9.005606e+02	0.000e+00	3.404e-04	4.924e-03
244	30386	9.005606e+02	0.000e+00	3.289e-04	8.822e-03
245	30510	9.005606e+02	0.000e+00	2.220e-04	7.730e-03
246	30634	9.005606e+02	0.000e+00	4.271e-04	7.225e-03
247	30758	9.005606e+02	0.000e+00	4.395e-04	9.833e-03
248	30882	9.005606e+02	0.000e+00	3.599e-04	8.777e-03
249	31006	9.005606e+02	0.000e+00	2.583e-04	6.861e-03
250	31130	9.005606e+02	0.000e+00	3.481e-04	1.710e-02
251	31254	9.005606e+02	0.000e+00	2.109e-04	6.562e-03
252	31378	9.005606e+02	0.000e+00	1.164e-04	5.648e-03
253	31502	9.005606e+02	0.000e+00	1.624e-04	3.581e-03
254	31627	9.005606e+02	0.000e+00	1.425e-04	1.126e-02
255	31751	9.005606e+02	0.000e+00	1.579e-04	4.437e-03
256	31875	9.005606e+02	0.000e+00	2.562e-04	1.315e-02
257	31999	9.005606e+02	0.000e+00	1.007e-04	3.402e-03
258	32123	9.005606e+02	0.000e+00	1.328e-04	6.508e-03
259	32247	9.005606e+02	0.000e+00	2.440e-04	2.461e-03
260	32371	9.005606e+02	0.000e+00	2.211e-04	1.716e-03
261	32496	9.005606e+02	0.000e+00	1.665e-04	8.207e-03
262	32620	9.005606e+02	0.000e+00	9.093e-05	3.044e-03
263	32745	9.005606e+02	0.000e+00	9.472e-05	3.634e-03
264	32869	9.005606e+02	0.000e+00	1.156e-04	1.035e-03

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
265	32993	9.005606e+02	0.000e+00	2.752e-04	3.385e-03
266	33117	9.005606e+02	0.000e+00	9.485e-05	1.704e-03
267	33242	9.005606e+02	0.000e+00	1.853e-04	2.541e-03
268	33366	9.005606e+02	0.000e+00	1.495e-04	1.643e-03
269	33490	9.005606e+02	0.000e+00	1.186e-04	2.778e-03
270	33615	9.005606e+02	0.000e+00	2.661e-04	2.785e-03
271	33739	9.005606e+02	0.000e+00	1.287e-04	1.842e-03
272	33863	9.005606e+02	0.000e+00	1.135e-04	2.820e-03
273	33987	9.005606e+02	0.000e+00	2.092e-04	1.100e-03
274	34112	9.005606e+02	0.000e+00	1.004e-04	1.730e-03
275	34236	9.005606e+02	0.000e+00	3.498e-04	9.585e-04
276	34360	9.005606e+02	0.000e+00	2.124e-04	2.270e-03
277	34484	9.005606e+02	0.000e+00	9.498e-05	1.307e-03
278	34616	9.005606e+02	0.000e+00	5.106e-05	1.290e-03
279	34744	9.005606e+02	0.000e+00	5.321e-05	1.296e-03
280	34868	9.005606e+02	0.000e+00	1.818e-04	1.848e-03
281	34992	9.005606e+02	0.000e+00	1.310e-04	1.563e-03
282	35118	9.005606e+02	0.000e+00	6.319e-04	2.071e-03
283	35243	9.005606e+02	0.000e+00	2.836e-04	1.027e-03
284	35367	9.005606e+02	0.000e+00	9.644e-05	1.378e-03
285	35491	9.005606e+02	0.000e+00	8.707e-05	1.035e-03
286	35616	9.005606e+02	0.000e+00	8.003e-05	1.352e-03
287	35740	9.005606e+02	0.000e+00	2.883e-04	1.083e-03
288	35864	9.005606e+02	0.000e+00	9.673e-05	1.129e-03
289	35988	9.005606e+02	0.000e+00	2.798e-04	1.650e-03
290	36112	9.005606e+02	0.000e+00	1.497e-04	1.697e-03
291	36236	9.005606e+02	0.000e+00	1.547e-04	1.509e-03
292	36360	9.005606e+02	0.000e+00	2.005e-04	1.341e-03
293	36484	9.005606e+02	0.000e+00	1.451e-04	8.809e-04
294	36608	9.005606e+02	0.000e+00	2.573e-04	1.168e-03
295	36740	9.005606e+02	0.000e+00	4.556e-05	1.192e-03
296	36864	9.005606e+02	0.000e+00	1.683e-04	1.328e-03
297	36988	9.005606e+02	0.000e+00	8.646e-05	1.457e-03
298	37113	9.005606e+02	0.000e+00	1.029e-04	1.729e-03
299	37238	9.005606e+02	0.000e+00	1.194e-04	1.319e-03
300	37362	9.005606e+02	0.000e+00	1.070e-04	1.410e-03

Iter	F-count	f(x)	Feasibility	First-order optimality	Norm of step
301	37486	9.005606e+02	0.000e+00	1.871e-04	4.949e-04
302	37610	9.005606e+02	0.000e+00	5.703e-05	7.355e-04
303	37735	9.005606e+02	0.000e+00	9.518e-05	7.660e-04
304	37860	9.005606e+02	0.000e+00	4.309e-04	1.370e-03
305	37984	9.005606e+02	0.000e+00	7.368e-05	7.143e-04
306	38110	9.005606e+02	0.000e+00	7.636e-05	4.979e-04
307	38235	9.005606e+02	0.000e+00	9.422e-05	7.457e-04
308	38367	9.005606e+02	0.000e+00	4.335e-05	4.506e-04
309	38492	9.005606e+02	0.000e+00	4.824e-05	1.119e-04
310	38620	9.005606e+02	0.000e+00	3.218e-05	8.645e-04
311	38745	9.005606e+02	0.000e+00	3.662e-04	3.607e-04
312	38869	9.005606e+02	0.000e+00	1.114e-04	8.352e-04
313	38994	9.005606e+02	0.000e+00	1.599e-04	5.353e-04
314	39127	9.005606e+02	0.000e+00	4.358e-05	3.689e-04
315	39252	9.005606e+02	0.000e+00	3.586e-04	6.021e-05
316	39380	9.005606e+02	0.000e+00	3.430e-05	7.278e-04
317	39505	9.005606e+02	0.000e+00	1.137e-04	2.643e-04
318	39630	9.005606e+02	0.000e+00	1.250e-04	8.891e-04
319	39757	9.005606e+02	0.000e+00	4.247e-04	5.517e-04
320	39883	9.005606e+02	0.000e+00	1.995e-04	4.802e-04
321	40010	9.005606e+02	0.000e+00	2.453e-04	3.799e-04
322	40134	9.005606e+02	0.000e+00	7.120e-05	6.751e-04
323	40258	9.005606e+02	0.000e+00	1.077e-04	8.505e-04

```
 324   40392    9.005606e+02    0.000e+00    3.227e-05    2.364e-04
 325   40517    9.005606e+02    0.000e+00    3.270e-04    6.486e-05
 326   40646    9.005606e+02    0.000e+00    3.390e-05    4.673e-04
 327   40770    9.005585e+02    0.000e+00    2.289e-03    9.013e-02
 328   40894    9.005585e+02    0.000e+00    1.777e-03    7.968e-02
 329   41018    9.005585e+02    0.000e+00    9.833e-04    9.336e-03
 330   41142    9.005585e+02    0.000e+00    6.430e-04    9.662e-03

                                           First-order      Norm of
Iter F-count            f(x)  Feasibility   optimality         step
 331   41266    9.005585e+02    0.000e+00    5.035e-04    8.407e-03
 332   41390    9.005585e+02    0.000e+00    6.332e-04    1.450e-02
 333   41514    9.005585e+02    0.000e+00    4.924e-04    6.794e-03
 334   41638    9.005585e+02    0.000e+00    6.812e-04    9.037e-03
 335   41762    9.005585e+02    0.000e+00    7.570e-04    3.843e-03
 336   41886    9.005585e+02    0.000e+00    6.542e-04    8.121e-03
 337   42010    9.005585e+02    0.000e+00    5.420e-04    2.355e-03
 338   42134    9.005585e+02    0.000e+00    3.988e-04    6.381e-03
 339   42258    9.005585e+02    0.000e+00    6.866e-04    3.165e-03
 340   42382    9.005585e+02    0.000e+00    6.763e-04    1.162e-02
 341   42506    9.005585e+02    0.000e+00    4.356e-04    3.337e-03
 342   42630    9.005585e+02    0.000e+00    4.500e-04    7.549e-03
 343   42754    9.005584e+02    0.000e+00    4.638e-04    2.770e-03
 344   42878    9.005584e+02    0.000e+00    4.348e-04    6.962e-03
 345   43002    9.005584e+02    0.000e+00    3.997e-04    2.452e-03
 346   43126    9.005584e+02    0.000e+00    4.897e-04    5.053e-03
 347   43250    9.005584e+02    0.000e+00    5.233e-04    4.139e-03
 348   43374    9.005584e+02    0.000e+00    4.496e-04    2.058e-03
 349   43498    9.005584e+02    0.000e+00    3.293e-04    5.365e-03
 350   43622    9.005584e+02    0.000e+00    1.690e-04    1.134e-03
 351   43746    9.005584e+02    0.000e+00    1.350e-04    4.307e-03
 352   43870    9.005584e+02    0.000e+00    1.604e-04    1.435e-03
 353   43994    9.005584e+02    0.000e+00    2.886e-04    1.719e-03
 354   44118    9.005584e+02    0.000e+00    3.298e-04    2.555e-03
 355   44242    9.005584e+02    0.000e+00    2.816e-04    3.835e-03
 356   44366    9.005584e+02    0.000e+00    2.393e-04    2.840e-03
 357   44490    9.005584e+02    0.000e+00    1.898e-04    1.810e-03
 358   44614    9.005584e+02    0.000e+00    1.583e-04    2.741e-03
 359   44738    9.005584e+02    0.000e+00    4.634e-04    1.541e-03
 360   44862    9.005584e+02    0.000e+00    2.995e-04    2.297e-03

                                           First-order      Norm of
Iter F-count            f(x)  Feasibility   optimality         step
 361   44986    9.005584e+02    0.000e+00    2.943e-04    3.048e-03
 362   45110    9.005584e+02    0.000e+00    3.310e-04    8.942e-04
 363   45235    9.005584e+02    0.000e+00    4.531e-04    2.162e-03
 364   45359    9.005584e+02    0.000e+00    2.238e-04    1.065e-03
 365   45483    9.005584e+02    0.000e+00    2.955e-04    2.676e-03
 366   45607    9.005584e+02    0.000e+00    5.217e-04    1.230e-03
 367   45731    9.005584e+02    0.000e+00    3.920e-04    3.312e-03
 368   45855    9.005584e+02    0.000e+00    2.320e-04    1.032e-03
 369   45979    9.005584e+02    0.000e+00    2.103e-04    4.229e-03
 370   46103    9.005584e+02    0.000e+00    2.119e-04    1.660e-03
 371   46227    9.005584e+02    0.000e+00    1.829e-04    8.868e-03
 372   46351    9.005584e+02    0.000e+00    5.188e-04    4.254e-03
 373   46475    9.005584e+02    0.000e+00    2.038e-04    3.315e-03
 374   46599    9.005584e+02    0.000e+00    2.525e-04    1.242e-03
 375   46725    9.005584e+02    0.000e+00    3.105e-04    6.603e-03
 376   46849    9.005584e+02    0.000e+00    2.749e-04    2.317e-03
 377   46973    9.005584e+02    0.000e+00    2.527e-04    3.532e-03
 378   47097    9.005584e+02    0.000e+00    3.520e-04    8.215e-04
 379   47222    9.005584e+02    0.000e+00    3.749e-04    4.103e-03
 380   47346    9.005584e+02    0.000e+00    2.223e-04    1.870e-03
 381   47470    9.005584e+02    0.000e+00    2.146e-04    1.966e-03
 382   47594    9.005584e+02    0.000e+00    2.757e-04    2.378e-03
```

```
 383   47718    9.005584e+02    0.000e+00    2.280e-04    2.263e-03
 384   47842    9.005584e+02    0.000e+00    2.369e-04    3.196e-03
 385   47966    9.005584e+02    0.000e+00    5.718e-04    2.550e-03
 386   48090    9.005584e+02    0.000e+00    3.270e-04    2.132e-03
 387   48214    9.005584e+02    0.000e+00    3.544e-04    1.796e-03
 388   48338    9.005584e+02    0.000e+00    2.738e-04    2.373e-03
 389   48462    9.005584e+02    0.000e+00    3.210e-04    8.478e-04
 390   48588    9.005584e+02    0.000e+00    1.724e-04    4.184e-03

                                            First-order      Norm of
Iter F-count            f(x)  Feasibility   optimality         step
 391   48712    9.005584e+02    0.000e+00    1.495e-04    1.124e-03
 392   48836    9.005584e+02    0.000e+00    1.342e-04    3.066e-03
 393   48960    9.005584e+02    0.000e+00    2.204e-04    1.386e-03
 394   49084    9.005584e+02    0.000e+00    1.783e-04    9.158e-04
 395   49208    9.005584e+02    0.000e+00    2.701e-04    3.727e-03
 396   49332    9.005584e+02    0.000e+00    1.768e-04    2.206e-03
 397   49458    9.005584e+02    0.000e+00    2.701e-04    4.573e-03
 398   49590    9.005584e+02    0.000e+00    5.635e-05    1.523e-03
 399   49714    9.005584e+02    0.000e+00    2.292e-04    3.012e-03
 400   49841    9.005584e+02    0.000e+00    1.888e-04    5.350e-04
 401   49970    9.005584e+02    0.000e+00    5.340e-05    2.746e-03
 402   50099    9.005584e+02    0.000e+00    6.942e-05    1.193e-03
 403   50224    9.005584e+02    0.000e+00    1.205e-04    4.686e-04
 404   50348    9.005584e+02    0.000e+00    1.128e-04    2.002e-03
 405   50472    9.005584e+02    0.000e+00    1.495e-04    1.347e-03
 406   50596    9.005584e+02    0.000e+00    2.520e-04    1.164e-03
 407   50722    9.005584e+02    0.000e+00    1.574e-04    5.388e-04
 408   50846    9.005584e+02    0.000e+00    1.879e-04    6.911e-04
 409   50970    9.005584e+02    0.000e+00    2.115e-04    1.527e-03
 410   51095    9.005584e+02    0.000e+00    2.269e-04    3.338e-03
 411   51219    9.005584e+02    0.000e+00    2.542e-04    8.745e-04
 412   51344    9.005584e+02    0.000e+00    2.318e-04    2.419e-03
 413   51468    9.005584e+02    0.000e+00    3.871e-04    1.257e-03
 414   51592    9.005584e+02    0.000e+00    1.703e-04    2.320e-03
 415   51716    9.005584e+02    0.000e+00    6.771e-04    6.179e-04
 416   51849    9.005584e+02    0.000e+00    5.964e-05    1.112e-03
 417   51973    9.005584e+02    0.000e+00    3.891e-04    8.104e-04
 418   52099    9.005584e+02    0.000e+00    1.511e-04    2.655e-03
 419   52223    9.005584e+02    0.000e+00    3.204e-04    2.734e-03
 420   52347    9.005584e+02    0.000e+00    3.815e-04    1.248e-03

                                            First-order      Norm of
Iter F-count            f(x)  Feasibility   optimality         step
 421   52473    9.005584e+02    0.000e+00    8.147e-05    1.168e-03
 422   52597    9.005584e+02    0.000e+00    1.450e-04    7.870e-04
 423   52721    9.005584e+02    0.000e+00    3.662e-04    7.044e-04
 424   52846    9.005584e+02    0.000e+00    1.837e-04    7.138e-04
 425   52970    9.005584e+02    0.000e+00    2.418e-04    1.214e-03
 426   53094    9.005584e+02    0.000e+00    5.168e-04    1.264e-03
 427   53218    9.005584e+02    0.000e+00    7.686e-04    8.423e-04
 428   53344    9.005584e+02    0.000e+00    1.582e-04    1.006e-03
 429   53469    9.005584e+02    0.000e+00    4.038e-04    5.780e-04
 430   53593    9.005584e+02    0.000e+00    2.566e-04    1.367e-03
 431   53717    9.005584e+02    0.000e+00    3.253e-04    9.836e-04
 432   53841    9.005584e+02    0.000e+00    2.026e-04    1.389e-03
 433   53965    9.005584e+02    0.000e+00    5.077e-04    7.294e-04
 434   54089    9.005584e+02    0.000e+00    1.528e-04    8.909e-04
 435   54213    9.005584e+02    0.000e+00    4.300e-04    9.199e-04
 436   54337    9.005584e+02    0.000e+00    1.589e-04    9.731e-04
 437   54461    9.005584e+02    0.000e+00    2.138e-04    7.645e-04
 438   54585    9.005584e+02    0.000e+00    1.260e-04    8.570e-04
 439   54712    9.005584e+02    0.000e+00    3.211e-04    7.133e-04
 440   54838    9.005584e+02    0.000e+00    1.609e-04    4.083e-04
 441   54962    9.005584e+02    0.000e+00    1.539e-04    7.477e-04
```

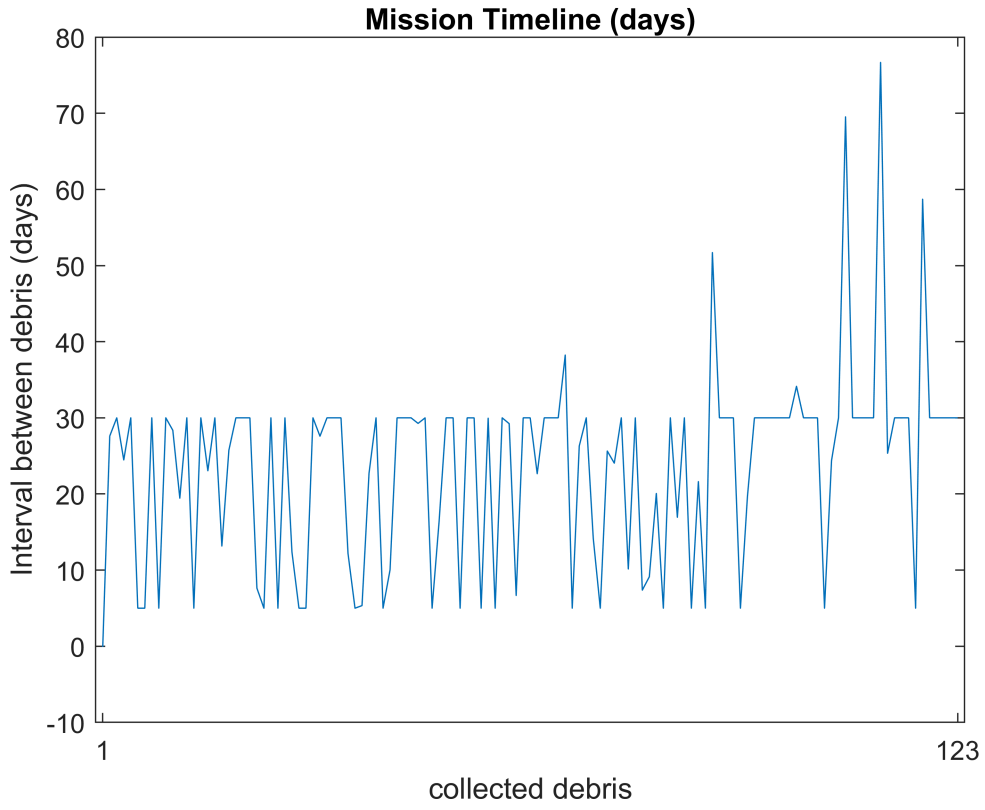
```
 442   55086    9.005584e+02    0.000e+00    5.805e-04    1.477e-03
 443   55210    9.005584e+02    0.000e+00    1.865e-04    1.097e-03
 444   55334    9.005584e+02    0.000e+00    2.448e-04    7.440e-04
 445   55459    9.005584e+02    0.000e+00    2.545e-04    8.063e-04
 446   55583    9.005584e+02    0.000e+00    1.717e-04    4.724e-04
 447   55707    9.005584e+02    0.000e+00    5.291e-04    8.403e-04
 448   55832    9.005584e+02    0.000e+00    1.153e-04    9.579e-04
 449   55956    9.005584e+02    0.000e+00    1.412e-04    4.759e-04
 450   56080    9.005584e+02    0.000e+00    1.824e-04    3.274e-04

                                           First-order      Norm of
Iter F-count            f(x)  Feasibility   optimality         step
 451   56204    9.005584e+02    0.000e+00    6.873e-04    4.938e-04
 452   56328    9.005584e+02    0.000e+00    8.487e-05    7.102e-04
 453   56452    9.005584e+02    0.000e+00    1.533e-04    6.689e-04
 454   56576    9.005584e+02    0.000e+00    2.143e-04    6.445e-04
 455   56700    9.005584e+02    0.000e+00    4.203e-04    4.787e-04
 456   56824    9.005584e+02    0.000e+00    4.279e-04    1.055e-03
 457   56948    9.005584e+02    0.000e+00    1.245e-04    6.659e-04
 458   57072    9.005584e+02    0.000e+00    2.448e-04    6.413e-04
 459   57196    9.005584e+02    0.000e+00    6.873e-04    5.230e-04
 460   57320    9.005584e+02    0.000e+00    2.447e-04    7.772e-04
 461   57444    9.005584e+02    0.000e+00    1.602e-04    5.349e-04
 462   57569    9.005584e+02    0.000e+00    9.587e-05    1.660e-03
 463   57693    9.005584e+02    0.000e+00    9.821e-05    6.677e-04
 464   57817    9.005584e+02    0.000e+00    1.141e-04    1.265e-03
 465   57941    9.005584e+02    0.000e+00    2.595e-04    1.058e-03
 466   58065    9.005584e+02    0.000e+00    1.469e-04    9.786e-04
 467   58189    9.005584e+02    0.000e+00    5.265e-04    5.476e-04
 468   58316    9.005584e+02    0.000e+00    1.367e-04    5.524e-04
 469   58440    9.005584e+02    0.000e+00    1.086e-04    3.077e-04
 470   58565    9.005584e+02    0.000e+00    1.088e-04    1.229e-03
 471   58689    9.005584e+02    0.000e+00    7.566e-05    3.887e-04
 472   58814    9.005584e+02    0.000e+00    1.014e-04    5.859e-04
 473   58938    9.005584e+02    0.000e+00    3.070e-04    5.497e-04
 474   59062    9.005584e+02    0.000e+00    1.208e-04    5.026e-04
 475   59187    9.005584e+02    0.000e+00    1.334e-04    3.835e-04
 476   59311    9.005584e+02    0.000e+00    1.422e-04    3.658e-04
 477   59437    9.005584e+02    0.000e+00    1.879e-04    2.548e-04
 478   59563    9.005584e+02    0.000e+00    2.191e-04    3.680e-04
 479   59694    9.005584e+02    0.000e+00    5.164e-05    4.831e-04
 480   59818    9.005584e+02    0.000e+00    3.430e-04    1.109e-03

                                           First-order      Norm of
Iter F-count            f(x)  Feasibility   optimality         step
 481   59942    9.005584e+02    0.000e+00    7.098e-04    4.865e-04
 482   60066    9.005584e+02    0.000e+00    3.050e-04    8.533e-04
 483   60190    9.005584e+02    0.000e+00    1.135e-04    3.436e-04
 484   60314    9.005584e+02    0.000e+00    1.750e-04    9.769e-04
 485   60438    9.005584e+02    0.000e+00    7.995e-05    5.289e-04
 486   60562    9.005584e+02    0.000e+00    1.587e-04    2.809e-04
 487   60688    9.005584e+02    0.000e+00    3.468e-04    3.735e-04
 488   60812    9.005584e+02    0.000e+00    1.635e-04    4.321e-04
 489   60936    9.005584e+02    0.000e+00    7.459e-05    2.648e-04
 490   61062    9.005584e+02    0.000e+00    4.240e-04    9.954e-05
 491   61189    9.005584e+02    0.000e+00    1.249e-04    9.685e-05
 492   61314    9.005584e+02    0.000e+00    1.188e-04    2.905e-04
 493   61438    9.005584e+02    0.000e+00    1.116e-04    6.084e-04
 494   61563    9.005584e+02    0.000e+00    8.145e-05    4.102e-04
 495   61687    9.005584e+02    0.000e+00    1.090e-04    6.179e-04
 496   61811    9.005584e+02    0.000e+00    1.122e-04    2.497e-04
 497   61935    9.005584e+02    0.000e+00    2.550e-04    3.402e-04
 498   62062    9.005584e+02    0.000e+00    2.261e-04    1.321e-04
 499   62194    9.005584e+02    0.000e+00    3.452e-05    2.727e-04
 500   62318    9.005584e+02    0.000e+00    2.899e-04    2.714e-04
```

```
Solver stopped prematurely.

fmincon stopped because it exceeded the iteration limit,
options.MaxIterations = 5.000000e+02.
```

```
[~, memo] = f.theMission(t_sol,ddata,M,c); % missione ottimizzata
memo.timeline = cumsum(t_sol);
memo.interval = t_sol;
verdetto = rispettaMassa(t_sol,memo,f,ddata,c); % vedo se rispetto le masse; tendenzialmente l'

disp([round(memo.estM)', verdetto'])
```

```
        4828           1
        6364           1
        5184           1
        3767           1
        3471           1
        3987           1
        3626           1
        4770           1
        4460           1
        4242           1
        4655           1
        5082           1
        3270           1
        5223           1
```

```
figure
plot(t_sol)
title('Mission Timeline (days)'), ylabel('Interval between debris (days)'), xlabel('collected d
xlim([0 length(t_sol)+1]), xticks([1 length(t_sol)])
```

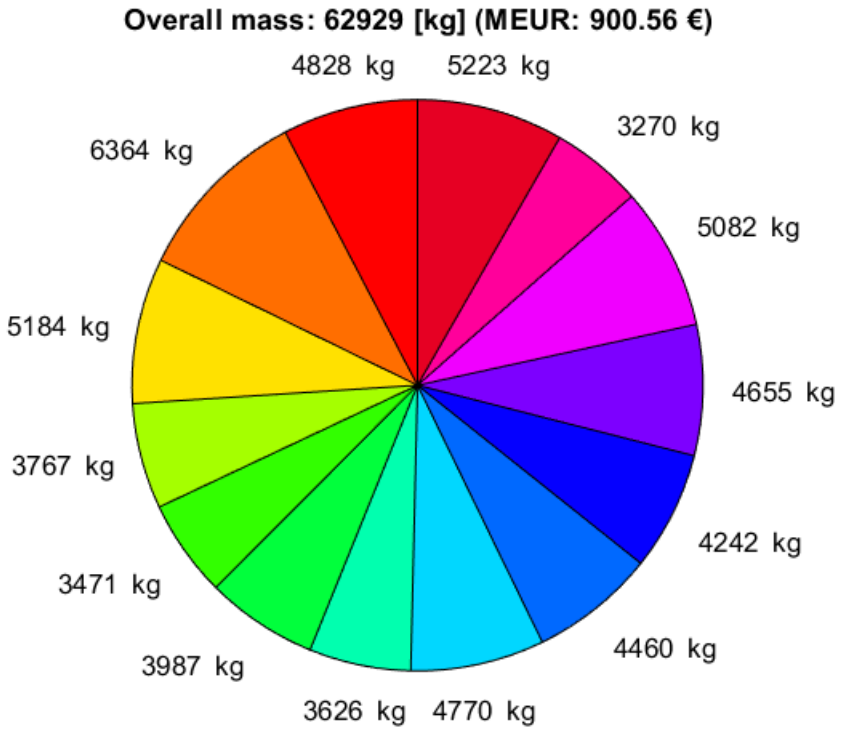


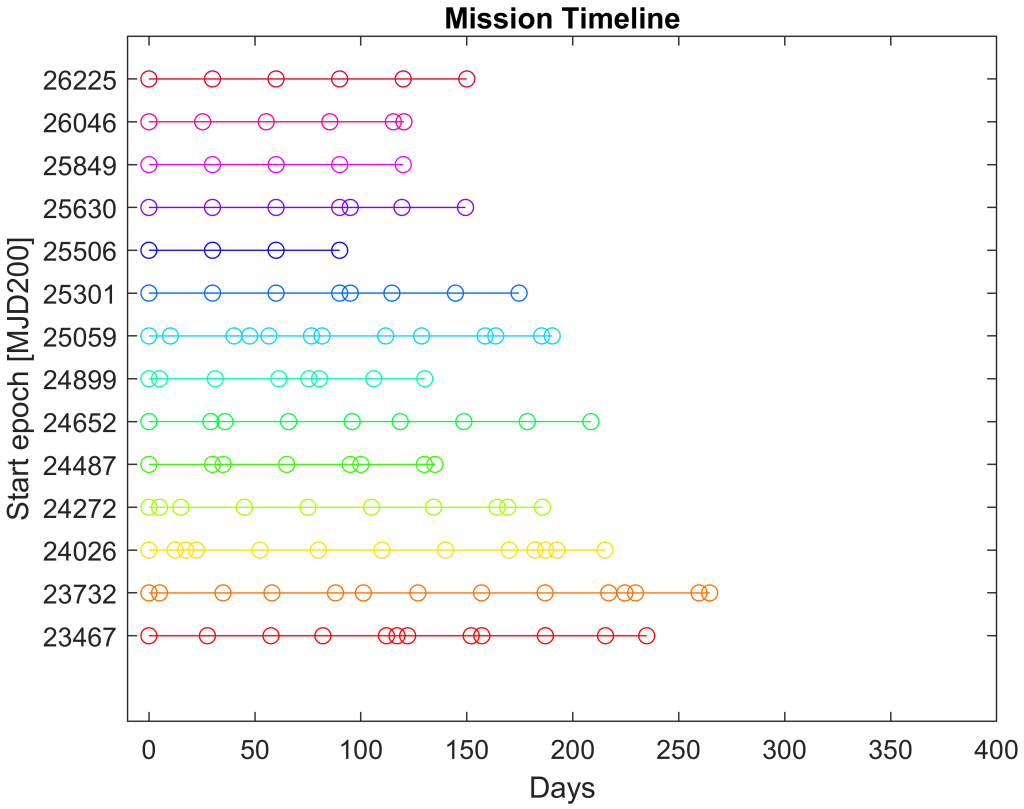

```
vantaggio = c.stop_ep-sum(t_sol)-c.start_ep;
fprintf('la missione si conclude con %.2f giorni di anticipo rispetto alla scadenza\n', vantagg
```

```
la missione si conclude con 44.40 giorni di anticipo rispetto alla scadenza
```

Primi risultati

```
[~,~,tabella,tabella2,tabella3] = f.mostraRisultati(memo);
```






```
disp(tabella)
```

Debris raccolti	Lista debris	Costo dV [m/s

12	"34 39 82 63 104 115 23 96 17 119 58 97"	"355.4 242.9 177.3 354.2 84.9 337.7 207.6 12
14	"122 51 24 118 88 21 85 26 80 114 79 37 56 6"	"154.3 326.7 363.9 306.6 210.5 232.4 395.5 3
13	"40 46 83 8 71 86 48 27 105 3 69 123 1"	"228.3 149.8 181.5 395.4 122.9 118.8 271.5 3
10	"109 68 7 19 12 25 45 121 76 38"	"129.9 132.3 259.5 145.1 397.6 230 106.4 203
8	"53 22 65 31 101 67 20 70"	"279.4 192 208.4 200.2 145.9 154.3 382.9"
9	"94 10 66 91 15 107 29 78 95"	"296.5 138.1 285.5 168.9 284.3 258.1 332.6 2
8	"93 112 13 57 43 72 110 44"	"152.3 298.4 241.2 142.7 198.1 341 344.5"
13	"52 99 60 92 103 55 100 2 36 90 5 41 113"	"163 28.8 213.6 128.8 286.6 139.6 234.8 254.
8	"77 9 28 117 108 62 18 81"	"1154.3 224.9 57.8 172.8 200 432.7 155"
4	"120 33 32 89"	"213.8 1496.5 682.2"
7	"98 116 111 74 87 35 11"	"221.4 746.5 864.8 256.5 199.2 302.4"
5	"61 54 4 84 14"	"474.4 554.9 1713.4 221.3"
6	"16 50 73 64 59 42"	"236.6 508.6 135.2 369.7 180.8"
6	"49 106 47 75 102 30"	"233.7 428.5 332.3 1252.3 795.1"

```
disp(tabella2)
```

Debris raccolti	Lista debris	Transfer Duration [days]
12	"34 39 82 63 104 115 23 96 17 119 58 97"	{["27.6 30 24.47 30 5 5 30 5 30 28.36 19.44'
14	"122 51 24 118 88 21 85 26 80 114 79 37 56 6"	{["5 30 23.05 30 13.16 25.74 30 30 30 7.63 5
13	"40 46 83 8 71 86 48 27 105 3 69 123 1"	{["12.39 5 5 30 27.59 30 30 30 12.19 5 5.34
10	"109 68 7 19 12 25 45 121 76 38"	{["5 10.04 30 30 30 29.27 30 5 16.4"
8	"53 22 65 31 101 67 20 70"	{["30 5 30 29.99 5 29.96 5"
9	"94 10 66 91 15 107 29 78 95"	{["29.24 6.67 30 30 22.66 30 30 30"
8	"93 112 13 57 43 72 110 44"	{["5 26.31 30 14.15 5 25.64 24.05"
13	"52 99 60 92 103 55 100 2 36 90 5 41 113"	{["10.15 30 7.37 9.13 20.07 5 30 16.94 30 5
8	"77 9 28 117 108 62 18 81"	{["30 30 30 5 19.65 30 30"
4	"120 33 32 89"	{["30 30 30"
7	"98 116 111 74 87 35 11"	{["30 30 30 5 24.38 30"
5	"61 54 4 84 14"	{["30 30 30 30"
6	"16 50 73 64 59 42"	{["25.34 30 30 30 5"
6	"49 106 47 75 102 30"	{["30 30 30 30 30"

```
disp(tabella3)
```

Debris raccolti	Lista debris	Start [MJD2000]	Stop [MJD2000]
12	"34 39 82 63 104 115 23 96 17 119 58 97"	23467	23702
14	"122 51 24 118 88 21 85 26 80 114 79 37 56 6"	23732	23996
13	"40 46 83 8 71 86 48 27 105 3 69 123 1"	24026	24242
10	"109 68 7 19 12 25 45 121 76 38"	24272	24457
8	"53 22 65 31 101 67 20 70"	24487	24622
9	"94 10 66 91 15 107 29 78 95"	24652	24861
8	"93 112 13 57 43 72 110 44"	24899	25029
13	"52 99 60 92 103 55 100 2 36 90 5 41 113"	25059	25250
8	"77 9 28 117 108 62 18 81"	25301	25476
4	"120 33 32 89"	25506	25596
7	"98 116 111 74 87 35 11"	25630	25779
5	"61 54 4 84 14"	25849	25969
6	"16 50 73 64 59 42"	26046	26166
6	"49 106 47 75 102 30"	26225	26375

Si ottimizza ora la lista dei nodi per verificare che non ci siano alternative più valide di quelle ottenute aggiungendo i nodi difficili da inserire nel filone. Qui la timeline non viene modificata.

```
% Lanciare questa section più e più volte per ottenere il massimo
% dell'ottimizzazione
```

```
tentativi = 5000;
old = memo;
continua = true;
while continua
new = dubbione(old,tentativi,ddata,c,f);
continua = ~isequal(new,old);
old = new;
end

memo = old; fprintf('Costo totale: %.3f\n',memo.TC)
```

```
Costo totale: 865.829
```

Per ulteriore verifica si tenta di riottimizzare modificando la timeline, ma ci si accorge che si è quasi sempre su un minimo locale della funzione e quella che si ottiene è la soluzione complessiva.

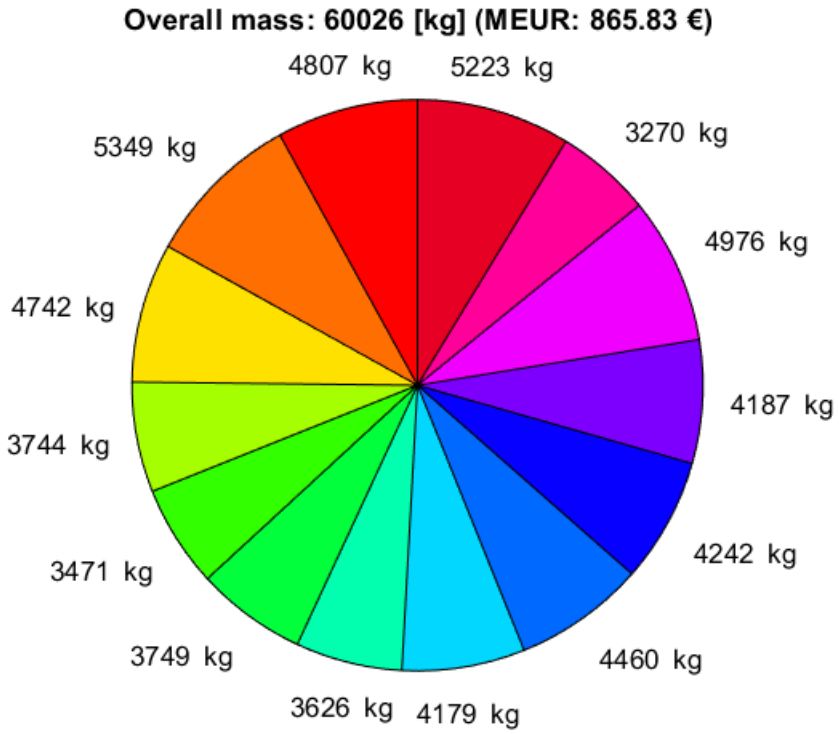
```
t = new.interval;
M = new;
vals = [max(t)+10 10 30 t(1) mean(t(2:end))];
[lb,ub,t0,Adis,Bdis] = f.creaConstraints(vals,c,M.n);

t(t'<lb) = lb(t'<lb);
t(t'>ub) = ub(t'>ub);



opt_fun = @(t_vet) f.theMission(t,ddata,M,c);
options = optimoptions('fmincon','Display','iter',...
                'MaxFunctionEvaluations',9e5,...
                'MaxIter',500,'HonorBounds',false);

[t_sol, cost, flag] = fmincon(opt_fun,t,Adis,Bdis,[],[],lb,ub,[],options);
```

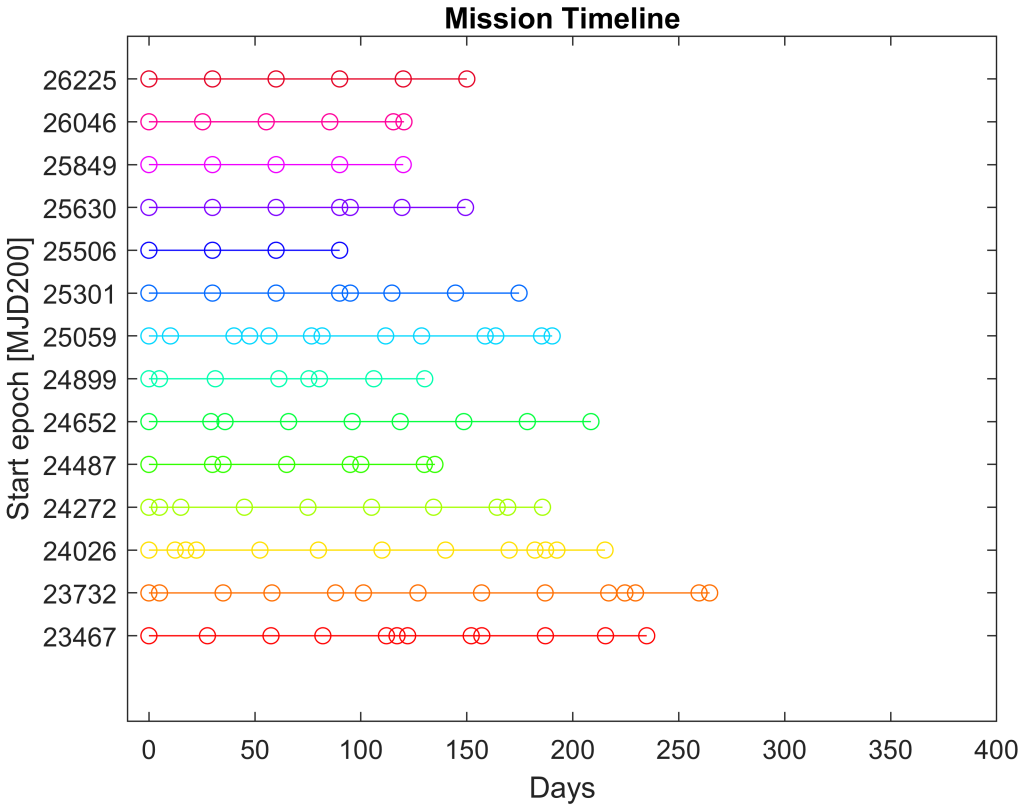
```
                                         First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0     124    8.658289e+02    0.000e+00    0.000e+00

Initial point is a local minimum that satisfies the constraints.

Optimization completed because at the initial point, the objective function is non-decreasing
in feasible directions to within the value of the optimality tolerance, and
constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

```
M.pat = mat2cell(M.list,1,M.n); %(!)
[tc, memo] = f.theMission(t_sol,ddata,M,c);
```

Risultati

```
[f1,f2,tabella,tabella2,tabella3] = f.mostraRisultati(memo);
```






```
disp(tabella)
```

Debris raccolti	Lista debris	Costo dV [m/s

12	"34 5 82 39 104 115 23 96 17 58 24 97"	"161.8 289.3 539.9 53.8 84.9 337.7 207.6 12
14	"122 118 51 119 88 21 85 26 80 114 79 37 56 6"	"63.9 148.8 81.5 273.7 210.5 232.4 395.5 31
13	"40 46 83 8 71 86 48 27 3 105 12 69 1"	"228.3 149.8 181.5 395.4 122.9 118.8 271.5
10	"68 7 109 19 123 25 45 121 76 38"	"124.3 96.4 239 161.1 424.5 230 106.4 203.9
8	"53 22 65 31 101 67 20 70"	"279.4 192 208.4 200.2 145.9 154.3 382.9"
9	"15 10 66 91 94 107 29 78 95"	"85.7 138.1 285.5 131.1 328.5 258.1 332.6 2
8	"93 112 13 57 43 72 110 44"	"152.3 298.4 241.2 142.7 198.1 341 344.5"
13	"52 99 60 92 103 55 100 63 36 2 90 41 113"	"163 28.8 213.6 128.8 286.6 139.6 138.6 253
8	"77 9 28 117 108 62 18 81"	"1154.3 224.9 57.8 172.8 200 432.7 155"
4	"120 33 32 89"	"213.8 1496.5 682.2"
7	"98 116 111 87 74 35 11"	"221.4 746.5 463.8 256.5 241.9 302.4"
5	"61 54 4 14 84"	"474.4 554.9 1642.2 221.3"
6	"16 50 73 64 59 42"	"236.6 508.6 135.2 369.7 180.8"
6	"49 106 47 75 102 30"	"233.7 428.5 332.3 1252.3 795.1"

```
disp(tabella2)
```

Debris raccolti	Lista debris	Transfer Duration [days]
12	"34 5 82 39 104 115 23 96 17 58 24 97"	{["27.6 30 24.47 30 5 5 30 5 30 28.36 19.44
14	"122 118 51 119 88 21 85 26 80 114 79 37 56 6"	{["5 30 23.05 30 13.16 25.74 30 30 30 7.63
13	"40 46 83 8 71 86 48 27 3 105 12 69 1"	{["12.39 5 5 30 27.59 30 30 30 12.19 5 5.34
10	"68 7 109 19 123 25 45 121 76 38"	{["5 10.04 30 30 30 29.27 30 5 16.4"
8	"53 22 65 31 101 67 20 70"	{["30 5 30 29.99 5 29.96 5"
9	"15 10 66 91 94 107 29 78 95"	{["29.24 6.67 30 30 22.66 30 30 30"
8	"93 112 13 57 43 72 110 44"	{["5 26.31 30 14.15 5 25.64 24.05"
13	"52 99 60 92 103 55 100 63 36 2 90 41 113"	{["10.15 30 7.37 9.13 20.07 5 30 16.94 30 5
8	"77 9 28 117 108 62 18 81"	{["30 30 30 5 19.65 30 30"
4	"120 33 32 89"	{["30 30 30"
7	"98 116 111 87 74 35 11"	{["30 30 30 5 24.38 30"
5	"61 54 4 14 84"	{["30 30 30 30"
6	"16 50 73 64 59 42"	{["25.34 30 30 30 5"
6	"49 106 47 75 102 30"	{["30 30 30 30 30"

```
disp(tabella3)
```

Debris raccolti	Lista debris	Start [MJD2000]	Stop [MJD2000]
12	"34 5 82 39 104 115 23 96 17 58 24 97"	23467	23702
14	"122 118 51 119 88 21 85 26 80 114 79 37 56 6"	23732	23996
13	"40 46 83 8 71 86 48 27 3 105 12 69 1"	24026	24242
10	"68 7 109 19 123 25 45 121 76 38"	24272	24457
8	"53 22 65 31 101 67 20 70"	24487	24622
9	"15 10 66 91 94 107 29 78 95"	24652	24861
8	"93 112 13 57 43 72 110 44"	24899	25029
13	"52 99 60 92 103 55 100 63 36 2 90 41 113"	25059	25250
8	"77 9 28 117 108 62 18 81"	25301	25476
4	"120 33 32 89"	25506	25596
7	"98 116 111 87 74 35 11"	25630	25779
5	"61 54 4 14 84"	25849	25969
6	"16 50 73 64 59 42"	26046	26166
6	"49 106 47 75 102 30"	26225	26375

```
writetable(tabella, 'tabella1.xlsx')
writetable(tabella2, 'tabella2.xlsx')
writetable(tabella3, 'tabella3.xlsx')
```

Funzioni ausiliarie

```
function best = dubbione(memo,tentativi,ddata,c,f)
    best = memo;
    for i = 1:tentativi
        provino = best;
        scambio = randperm(length(best.list),randi([2 6])); % determina una sequenza casuale di
        provino.list(scambio) = provino.list(scambio([2:end 1])); % i detriti slittano di una p
        [punteggio,temp] = f.theMission(provino.interval,ddata,provino,c); % calcolo la nuova m
        if punteggio <= best.TC % controllo se costa meno; se costa meno prendo la nuova missio

             fprintf("permutazione trovata: %s\n",strjoin(string(scambio)))
             fprintf("risparmio: %.3f M€UR\n",best.TC-punteggio)
             best = temp;
        end
    end
end
```

Potrebbe essere utile sottolineare lo schema logico della funzione *tappaBuchi* visto che è una function ricca di comandi. Come funziona:

tappaBuchi viene richiamata in un punto del codice in cui non sono ancora stati raccolti tutti i 123 detriti. La finalità più grande è di ampliare la sequenza. Per farlo si procede per piccoli step:

- si guarda la lista dei detriti che avanzano;
- si calcola il tempo che intercorre tra una sottomissione e la seguente. Si isolano solo le pause maggiori di una certa quantità di tempo, poichè non possono essere inferiori ai 30 gg e al tempo stesso bisogna garantire il trasferimento orbitale con un costo sufficientemnte ridotto.
- **prima modalità di ampliamento**: un determinato detrito viene incluso all'inizio di una sottomissione, a patto che la sua introduzione soddisfi il vincolo sulla massa. Il motivo per cui lo si mette all'inizio e non alla fine è dovuto al fatto che il calcolo della massa iniziale del satellite si svolge a ritroso, quindi meglio non cambiare il termine iniziale della successione dei ΔV.
- se non si può applicare questo metodo, si applica la **seconda modalità di ampliamento**: se esiste una pausa sufficienttemente lunga da poter inserire più trasferimenti orbitali, si tenta di inserire una nuova sottomissione con una porzione dei detriti non ancora raccolti. Se la pausa si trova tra due sottomissioni preesistenti (caso1), bisogna rispettare il vincolo dei 30 giorni di inattività da entrambe. Se la pausa predisposta all'inserimento è quella successiva all'ultima sottomissione preesistente (caso 2) si deve garantire che la missione si chiuda prima della scadenza.

```
function [t,M] = tappaBuchi(t,M,ddata,f,c)
lp = find(t>60); durata = 30; %lp = long pause
if isempty(lp)
    disp('no pause disponibili')
else
    istanti = cumsum(t);
    starter = istanti(lp)-durata;
    grids = f.estimatedV(ddata,starter,c.start_ep,durata);
```

```
    arr = M.list(lp);
    left = setdiff(1:123,M.list);

    dV_pox = zeros(size(grids,3),length(left));
    for i=1:size(grids,3)
        dV_pox(i,:) = grids(left,arr(i),i);
    end

    [dVbest, idx] = min(dV_pox,[],'all','linear','omitnan' );
    [rig, col] = ind2sub(size(dV_pox),idx);
    debris_raccolto = left(col);
    fprintf('sto cercando di ampliare la missione prima del nodo #%d\n',arr(rig));

    for i = 1:length(M.n)
        primo = getNode(M.pat{i}(1));
        if primo==arr(rig)
          epoca = i;
          break
        end
    end

    mass_new = massaMission(c.Mdry, c.Mde, [dVbest M.estdV{epoca}'], c.G0, c.Isp);
    dove = find(M.list==arr(rig));

    if mass_new < 7000
        fprintf('metto nella %d° submission il detrito #%d \n',epoca,debris_raccolto);
        fprintf('prevedo %.1f [kg]\n',mass_new)
        t_new = [t(1:lp(rig)-1) t(lp(rig))-durata durata t(lp(rig)+1:end)];
        M.n(epoca) = M.n(epoca)+1;

        M.list = [M.list(1:dove-1) debris_raccolto M.list(dove:end)];
        [~, M] = f.theMission(t_new,ddata,M,c);
        t = t_new;
        nuovoTime = getTime(M.pat{epoca}(1))-3;
        nuovoNodo = getId(debris_raccolto,nuovoTime);
        M.pat{epoca} = cat(2,nuovoNodo,M.pat{epoca});
    else
        disp('non riesco ad introdurre detriti: dV troppo alti')
        margine = 20;
        [maxPausa,dove] = max([t c.stop_ep-sum(t)-c.start_ep-margine]);

        if maxPausa>80
            if (c.stop_ep-sum(t)-c.start_ep-margine == maxPausa)
                %% CASO 2
                fprintf('introduco una submission alla fine\n');
                % finestra = c.stop_ep+ [max(sum(t)+30,c.stop_ep-c.start_ep-30*8) 0];

                a1 = max(sum(t)+30,c.stop_ep-c.start_ep-30*8-margine);
                a2 =  c.stop_ep-c.start_ep-margine;
                finestra =  [a1 a2];
                [seq,tt] = newSubMission(f,M,ddata,c,finestra);
                M.list = [M.list seq];
                M.n(end+1) = length(seq);
                start = tt(1)-sum(t);
```

```
                t_new = [t start diff(tt)];
                t = t_new;

                nuovoTime = getTime(M.pat{end}(end))+3;
                nuovoNodo = getId(seq,nuovoTime+(1:length(seq))*2);
                M.pat{end+1} = nuovoNodo;
                M.feas(end+1) = 1;
            else
                %% CASO 1
                if t(dove)>80
                    for i = 1:length(M.n)
                        primo = getNode(M.pat{i}(1));
                        if primo==M.list(dove)
                          epoca = i;
                          break
                        end
                    end

                    fprintf('introduco una submission dopo la %d°\n', epoca-1);

                    finestra = [sum(t(1:dove-1)) sum(t(1:dove))]+[30 -30];
                    [seq,tt] = newSubMission(f,M,ddata,c,finestra);

                    M.list = [M.list(1:dove-1) seq M.list(dove:end)];
                    M.n = [M.n(1:epoca-1) length(seq) M.n(epoca:end)];
                    start = tt(1)-sum(t(1:dove-1));
                    %[t(lp(rig)) diff(tt) start 30]

                    %t(lp(rig))==sum(diff(tt))+start+30

                    t_new = [t(1:dove-1) start diff(tt) 30 t(dove+1:end)];
                    t = t_new;
                    nuovoTime = getTime(M.pat{epoca}(1))-3;
                    nuovoNodo = getId(seq,nuovoTime-(length(seq):-1:1)-3);
                    M.pat = [M.pat(1:epoca-1) {nuovoNodo} M.pat(epoca:end)];
                    M.feas(end+1) = 1;
                else
                    disp('slot temporale troppo piccolo')
                end
            end
        end
    end
end
M.left = setdiff(1:123,M.list);
end

function s=getTime(str)
    s = round(rem(double(string(str)),1)*1000);
end

function s = getNode(str)
    s=uint8(double(string(str)));
end
```

```
function s=getId(n,t)
    s=(n+t/1000)+"";
end
```

Dati i frame temporali di una sottomissione troppo lunga per poter essere eseguita con un solo chaser, si individuano i traferimenti orbitali più lunghi (30 giorni = 3 frames). Tra questi si sceglie quello posizionato più vicino al centro della sequenza per una suddivisione equa in due sottomissioni. Il taglio avviene solo sui trasferiemnti da 30 giorni poichè se tagliassimo un trasferimento orbitale più breve introdurremmo con la pausa forzata di 30 giorni uno shift che danneggerebbe tutta la sottomissione.

```
function taglio = tagliaLegame(nodi)
    istanti = getTime(nodi);
    intervalli = diff(istanti);
    spazilunghi = find(intervalli==3);
    x = abs(spazilunghi-length(istanti)/2);
    [~,sep] = min(x);
    taglio = spazilunghi(sep);
end

function verdetto = rispettaMassa(t, mission,f,ddata,c)
    [~, M] = f.theMission(t,ddata,mission,c);
    soglia = c.Mdry+c.Mde.*mission.n+5000;
    verdetto = M.estM<=soglia;
end
```

```
function [sol,tt] = newSubMission(f,M,ddata,c,finestra)
    tt = [];  sol = [];
    n_ipo = floor(diff(finestra)/30+1); % calcolo il numero di detriti che posso raccogliere, i
    n_ipo = min(max(n_ipo,2),length(M.left)); % questo numero non può essere inferiore a 2 nè s
    finestra = linspace(finestra(1),finestra(end),n_ipo); % creo una suddivisione temporale in
    durata = finestra(2) - finestra(1); % tempo che intercorre tra un detrito e l'altro (lo cal
    if durata > 30 % se il passo è superiore a 30 lo reimposto a 30 e gli slot si ricalcolano c
        durata = 30;
        finestra = finestra(end)-durata*(n_ipo-1:-1:0);
    end
    numero_magico = 7; % per determinare la sequenza ottimale di detriti da aggiungere in bloco
    % Estendere il numero_magico a valori più alti aumenta i tempi di
    % calcolo o risulta impossibile per l'utilizzo di RAM.

    if n_ipo > numero_magico % se il numero degli slot temporali è maggiore di 7, allora ci si
       finestra = finestra(end-(numero_magico-1:-1:0));
       n_ipo = numero_magico;
    end

    if n_ipo>1
         dV = f.estimatedV(ddata,finestra(1:end-1),c.start_ep,durata); % calcolo i costi in dV

         p = nchoosek(M.left,n_ipo); % ottengo la lista delle combinazioni scegliendo un numero
```

```
        s = size(dV,3);
        best1 = inf; best2 = inf;
        for j = 1:size(p,1) % per ogni possibile combinazione:
             combo = p(j,:);
             pp = perms(combo); % ottengo la lista delle permutazioni di quella specifica comb
             idx = sub2ind(size(dV),pp(:,1:end-1),pp(:,2:end),repmat(1:s,size(pp,1),1));
             DV = dV(idx); % vado a pescare da dV i trasferimenti orbitali di ogni permutazion
             [~,MASSE] = massaMission(c.Mdry, c.Mde, DV, c.G0, c.Isp); % calcolo la massa dopo
             [rig,col] = find(MASSE<7000); % voglio identificare in ogni permutazione dopo qua
             if ~isempty(rig)
                 parimerito = col==min(col); % tutti i set che prendono più detriti avranno il
                 %pp = pp(parimerito,:);
                 rig = rig(parimerito,:); % isolo i set che ne prendono di più
                 col =  col(parimerito,:);
                 MASSE = MASSE(rig,col(1)); % osservo il valore esatto di massa di ogni set pr
                 [mass,seq] = min(MASSE); % prendo il set con la massa più bassa (a parità di
                  riga_vincente = rig(seq);
             else
                 col(1) = inf;
             end
            if col(1)<best2 % al variare di ogni combinazione si produce un nuovo insieme di p
                best2 = col(1); % se il massimo numero dei detriti è superiore a quello finora
                best1 = mass;
                sol = pp(riga_vincente,best2:end); % --> il set introdotto nella missione è qu
                tt = finestra(best2:end); % --> questi sono gli slot dedicati che occuperà
            elseif col(1)==best2 % se il numero massimo di detriti che puoi ottenere da una co
                %solo se la sua massa è inferiore
            if mass<best1
                 best1 = mass;
                 sol = pp(riga_vincente,best2:end); % --> il set introdotto nella missione è c
                 tt = finestra(best2:end); % --> questi sono gli slot dedicati che occuperà
            end
            end
        end

        fprintf('prevedo %d kg per %d detriti',round(best1),length(sol));
    end
end
```