Problem 6. **Min / Max Questions**

France 1 — ITYM 2010

Abstract

This ratio problem was about finding best set disposition and proving that they are the best disposition. As it isn't always possible to prove this kind of thing, we did trivial cases, and then we started to work on upper and lower bounderies for values that the smaller ratio can take. (the bigger ratio can always be infinite).

We have found some ideas in physics or made programs to give us estimates of the optimal ratios.

To begin, I'll introduse a notation so it'll be easier to know what part of the problem is beeing prove and where.

The probleme is composed with 3*6=18 smaller problemes, that can be fixed on a 3*6 tabel. Here it is:

	a)	b)	c)	d)	e)	f)
1)						
2)						
3)						

While we'll go one with the problem, we'll fill this board with equivalences and we'll shadow what is completet.

To begin we'll see some trivial things:
2)a) is the same question than 1)a), so is 2)f) to 1)f).
3)a) is obvious because the triangles we get are always flat, so the ratio is inderterminated(or we can assume it's 1)).
We fill the table with those informations:

	a)	b)	c)	d)	e)	f)
1)						
2)						
3)						

Now we'll continue by proving other equivalences.

We'll show that 1)d), e) and f) is equivalent, and so is 1)d), e) and f), by showing that any set of N points in the whole plan also exist in an infinite triangular grid, and in a n*m rectangular grid.

Here is the formel proof for the triangular grid, the rectangular one is more or less the same because you can take n and m as big as you want it to be.

If we can make a set in the plan, we can mesh this plan with an infinite triangle grid. Then, we can for any ε, as small as we want, make a thicker grid such that any point of the set will be closer than ε to a node of the mesh. It's easy to see if you concider that Q is dense everywhere in R.
In other words, you can "unzoom" the triangular grid such that it'll be almost like the plan, and by using the Q densitiy in R, we have that the length are the same(at least their limits).
So we have that for example, if the ratio for N=4 belong to [sqrt(2);+∞[in the plan, it belong to]sqrt(2);+∞[in a grid.

	a)	b)	c)	d)	e)	f)
1)						
2)						1)f)
3)						

Now we made some equivalences, we'll start answering the questions.
The questions are about what value a ratio can take :
The ratio can always be infinite(if you stick a point to an other, the minimal distance or area tend to zéro), so we'll search the smallest max/min ratio and we are going to call it "the best ratio".

1)a)b)

Here we'll define the smallest distance to be 1(we can do it because we're working on ratio, so a homothetic transformation doesn't change anything). By using pigeon's hole principle it's obvious that the maximal lenght cannot be bigger than N-1.
A case for Lmax/Lmin = N-1 is for the set (0;0),(1;0),(2;0),...,(N-1;0).

	a)	b)	c)	d)	e)	f)
1)						
2)						1)f)
3)						

1)b)
We'll work on a circumference whose radious 1.
It's obvious by pigeon's hole principle that the minimal distance cannot be bigger than "sqrt(2-2*cos(2pi/N))"(lenght of a "2pi/N"-chord).
If we take a regular N-gone, that is circumscibed in our cricumference, we see it cannot be a better ratio than the one of this(using the "ball analogy" we'll use to work on 1)f); with balls of "sqrt(2-2*cos(2pi/N))" radious).
The best ratio that we obtain so is :
sqrt((2-2cos(int(n/2)*2pi/N))/(2-2cos(2pi/N)))

2)b)
Here the length is proportional to the angle. We can chose a point to be the origin(angle 0) and than place the circumference quite. We'll have a 2pi length segment, where the origin is at coordonate 0 and 2pi. It's obvious that the best ratio in this case is int(N/2).

3)b)
Here again the best ratio is the regular N-gone, we can see it by making an orthogonal projection to the Oy axe(taking one on the smaller distance as beeing on the Ox axe).

c)
We can prove that the answer for c) are the same than for b) by using optimisation rules.

	a)	b)	c)	d)	e)	f)
1)						
2)						1)f)
3)						

1)f)

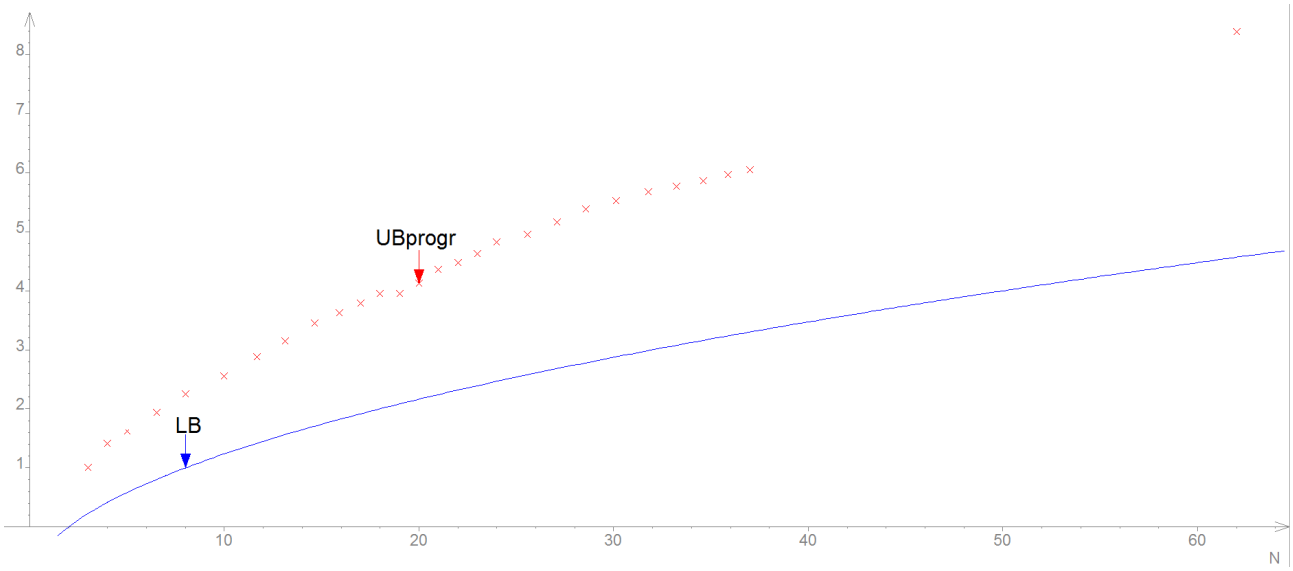
Here come the real question.
We didn't really answered to this question with a explicit "best ratio", but we found an upper and a lower bound for this ratio.

After having tried to make it manually for a while, we decided to search bound of the "bestratio(N)" function using programs(searching very good ratio's, so the best is better than the one we found).

At the beginning, we tried to study the problem by making numerical analysis for cases up to 30 points. Without losing generality of solution, it is possible to fix two points to (0,0) and (0,1) coordinates. For placing remaining points we use positions of previously calculated N-1 point case. The last point is placed at the gravity center of all previously placed points. Then Lmax/Lmin ratio is calculated for this first approximation of N point case. Than all N-2 not-fixed points are randomly moved closely around previous approximation, by using normal(or Gaussian) 2d distribution random number generators. Each time, Lmax/Lmin ratio is calculated. If Lmax/Lmin ratio is smaller than ever before, than this ratio is retained, together with point coordinates. This coordinates will become the middle points for normal 2d distribution random number generators.

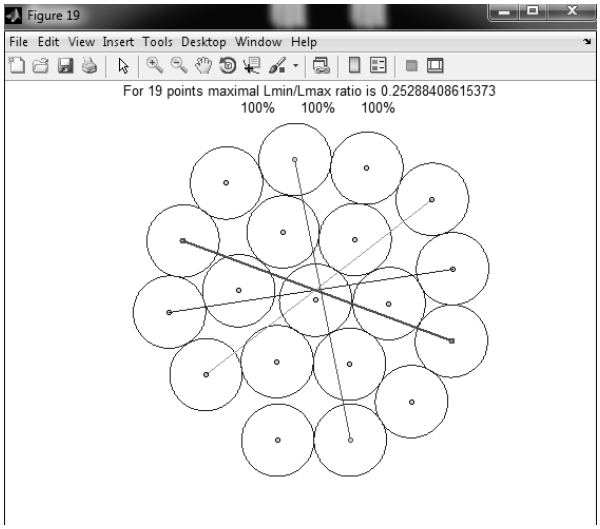
By using this method, effectively, we are simulating N identical marbles, which are placed in a very low profiled dish. Due to the gravity, all marbles will tend to group in the middle of the dish, if friction is equal to zero. Namely, the gravity will force them closer and closer together. If centers of marbles are considered, Lmin would be constant and equal to the diameter of the marbles. As well, Lmax will be very “close” to its minimal value, which would produce minimal Lmax/Lmin ratio.

By using this algorithm, we calculated first 30 cases and some others. Figures for all those cases, are presented here, as well as the Lmax/Lmin resuts.(the code of the program is at the end of the document).

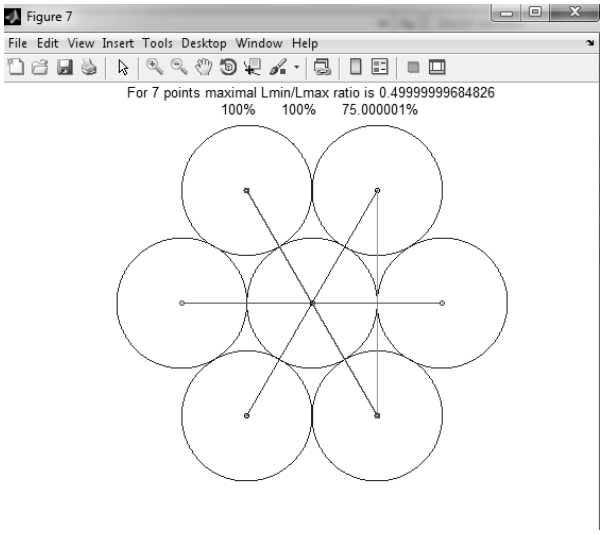


In this plot you have UBprogr, those are result from the program and LB that is a lower bound I'll explain later.

The program's result is given by coordinates, and a figure that look like :



or

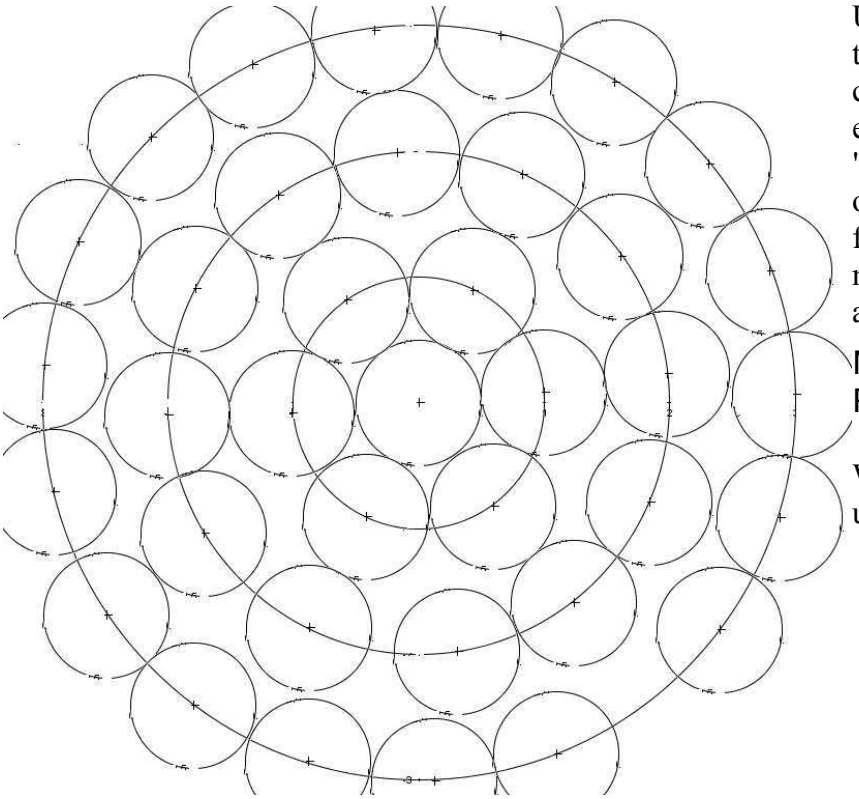


Here we can see the seven points for N=7 with ½Lmin radious circumferences around each one.

Here can see how the problem can be analogued by balls those are triing to be as near as possible.

By using this, analogie we found a lower bound. Oranges cannot be better compiled than if they were crushed, that's why a circle whose area is equal to the sum the area of every orange cannot be obtain, or at least, we cannot find a better ratio than this, that's why an lower bound is :

sqrt(N/2)-1.

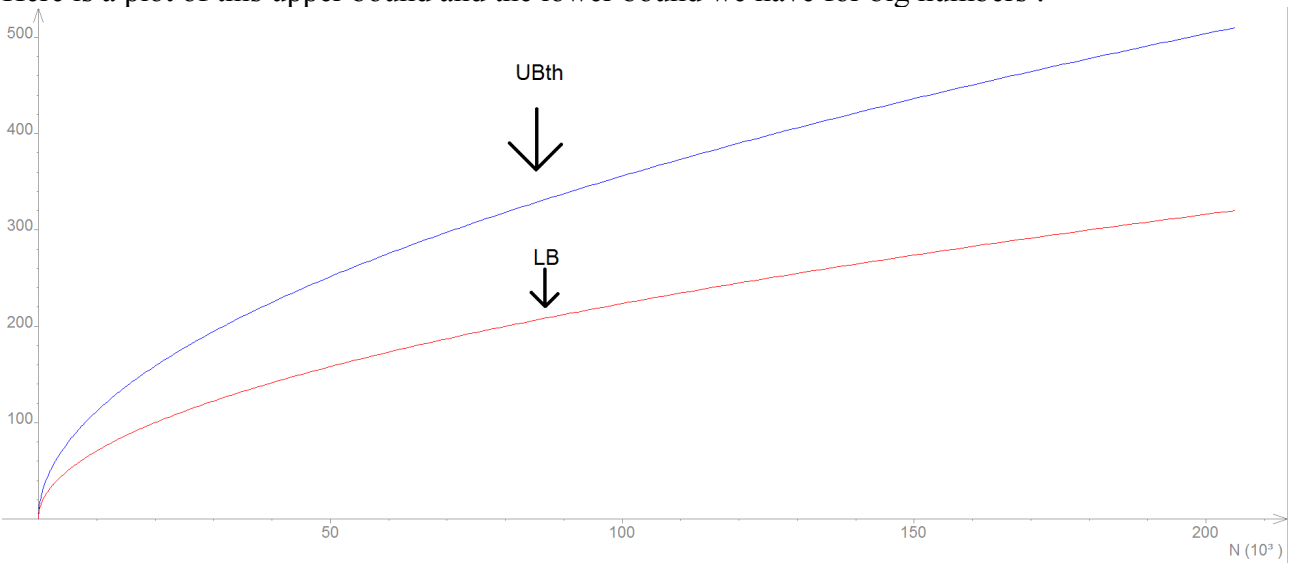


Using this diagram we can see that using cumcentric circumferences can be use full, eacht time we get on the next "level", the newratio is the oldratio+2. So we have as a forumla for upper bound for big numbers that for N those appear as :

N[i]=N[i-1]+int(2*pi*i)
Rmax=2*i

Where N is N and Rmax is the upper bound.

Here is a plot of this upper bound and the lower bound we have for big numbers :



UBth(n)/LB(n)=2 when N tend to $+\infty$.

Those bounderies are for 1)d)e)f) and 2)f).

The program :

Magic13

```
%%% RESOLVE N POINT CASE (Magic Function) %%%
% Nikola Bjelogrlic, June 2010
function [data_xy,maximal_ratio]=magic13(n,in_xy)

% calculate center of gravity
[in_dim,dummy]=size(in_xy);
x_sum=sum(in_xy(:,1));   y_sum=sum(in_xy(:,2));
xc=x_sum/in_dim;   yc=y_sum/in_dim;

n_min=min(in_dim,n);
xy(1:n_min,:)=in_xy(1:n_min,:);   % First two points are always fixed
for ii=in_dim+1:n
   xy(ii,:)=[xc,yc];
end
% Initialization part
minimal_minus_ratio=0.0;    %-0.3891188161;  % Initialize minimum of minus Lmin/Lmax ratio
xy_temp=xy;
sigma1=max(max(in_xy)); % Initialize sigma for normal distribution random number generator
nkk=1;        % Initialize "rate of sigma1 reduction" counter
mu1=0;        % Initialize mean/median/mode for normal distribution
for nk=1:40000 % This loop is fine serch, i.e. mu is moving
   % Points (X,Y) are generated by normally distributed random number generator
   random_Nx2=normrnd(mu1,sigma1,n-2,2); % Generate [(N-2)x2] normal distribution numbers
   xy_temp(3:n,:)=xy(3:n,:)+random_Nx2;  % Center normal distribution
   xy_temp(:,2)=abs(xy_temp(:,2));       % Only positive values for Y
   % Calculate (X,Y) points where is minimal minus Lmin/Lmax ratio
   [xy0,fval,exitflag,output]=fminsearch(@mM,xy_temp(3:n,:), ...
       optimset('MaxFunEvals',80000,'MaxIter',60000,'TolX',1e-14),xy);
   if fval<minimal_minus_ratio  % Test if it is smallest until now
   % -Lmin/Lmax is the smallest until now, so,
   % save those (X,Y) coordinates as the best
      minimal_minus_ratio=fval  % Keep this value as the best Lmin/Lmax ratio
      xy(3:n,:)=xy0;    % Preserve (X,Y) coordinates, as well
      data_xy=[xy]      % Prepare [X,Y] matrix for output
      sigma1=0.95*sigma1 % Makes sigma1 smaller and smaller
      code=show_points(xy,-minimal_minus_ratio);
   end
   nkk=nkk+1;           % Increase "rate of sigma1 reduction" counter
   if nkk>300           % About ??s loop
     nkk=1;             % Reset "rate of sigma1 change" counter
     sigma1=0.6*sigma1  % Makes sigma1 smaller and smaller after certain "time"
   end
   if sigma1<1.0e-08     % How about sigma
     break;             % Sigma is too small stop
   end
end

maximal_ratio=-minimal_minus_ratio;
```

min_MAX

```
%%% RESOLVE N POINT CASE WITH MAGIC13 FUNCTION %%%
% Nikola Bjelogrlic, June 2010
% This program numerically calculate maximal Lmin/Lmax ratio for N points
warning off all
results=zeros(50,2*50);

if (exist('initial_guess_for_x_y')==0) % If not initialized, i.e. if it is the first run after "clear"
   initial_guess_for_x_y=[0 0; 0.1 0];
end

for n=10:16    % n must be greater or equal to 3
   results(1,2*n-5)=n;
   % Calculate maximal Lmin/Lmax ratio for N points
   [solution_for_x_y,maximal_ratio]=magic13(n,initial_guess_for_x_y)
   initial_guess_for_x_y=solution_for_x_y;
   results(1,2*n-4)=maximal_ratio;
   results(2:n+1,(2*n-5):(2*n-4))=solution_for_x_y;
   code=show_points(solution_for_x_y,maximal_ratio);
end

%warning on all
```

mM

```
%%% CALCULATE MINIMAL MINUS Lmin/Lmax RATIO FOR GIVEN SET OF POINTS %%
%
% Nikola Bjelogrlic, June 2010
function f=mM(xy,xy_ini)
% INPUT DATA
% XY a matrix of size (M-2)x2 containing M-2 point coordinates X,Y
% These data are introduced Matlab-like as
% [X(1) Y(1) ; X(2) Y(2) ; .... ; X(M-2) Y(M-2)]
% There is no need to give explicitly the value of M-2,
% because Matlab computes it automatically.
% First two points are always fixed to (0,0) and (2,0)
% OUTPUT DATA
% F a scalar containing calculated minimal minus Lmin/Lmax ratio
[m,dummy]=size(xy); % Read size of input initial coordinate approximation matrix
m=m+2;
x(3:m)=xy(:,1);   y(3:m)=xy(:,2);
x(1:2)=xy_ini(1:2,1);   y(1:2)=xy_ini(1:2,2);  % First two points are fixed
best_min=999999;
best_MAX=0;
% Calculate all distances, and keep only min and max distances
for i=1:m     % Loop for all points
  for j=i+1:m   % Loop for all remaining points
    temp_distance_squared=(x(i)-x(j))^2+(y(i)-y(j))^2;
    if temp_distance_squared < best_min
      best_min = temp_distance_squared;
    end
```

```
    if temp_distance_squared > best_MAX
      best_MAX = temp_distance_squared;
    end
  end
end

% Calculate minimal minus Lmin/Lmax ratio for given set of points
f=-sqrt(best_min/best_MAX);
```

Show_points

```
%%% RESOLVE N POINT CASE WITH MAGIC13 FUNCTION %%%
% Nikola Bjelogrlic, June 2010
function exit_code=show_points(in_xy,maximal_ratio)
% This function show N points ...
[n,dummy]=size(in_xy);
tt = 0:pi./1000:2.*pi;
% Show N points
best_min=999999;
best_max(1:4)=0;  i_max(1:4)=1;  j_max(1:4)=1;
x=in_xy(:,1);   y=in_xy(:,2);
for i=1:n     % Loop for all points
  for j=i+1:n   % Loop for all remaining points
    temp_distance_squared=(x(i)-x(j))^2+(y(i)-y(j))^2;
    if temp_distance_squared < best_min
      best_min = temp_distance_squared; i_min=i; j_min=j;
    end
    if temp_distance_squared > best_max(1)
      best_max(4)=best_max(3);  i_max(4)=i_max(3); j_max(4)=j_max(3);
      best_max(3)=best_max(2);  i_max(3)=i_max(2); j_max(3)=j_max(2);
      best_max(2)=best_max(1);  i_max(2)=i_max(1); j_max(2)=j_max(1);
      best_max(1)=temp_distance_squared; i_max(1)=i; j_max(1)=j;
    else
      if temp_distance_squared > best_max(2)
      best_max(4)=best_max(3);  i_max(4)=i_max(3); j_max(4)=j_max(3);
      best_max(3)=best_max(2);  i_max(3)=i_max(2); j_max(3)=j_max(2);
      best_max(2)=temp_distance_squared; i_max(2)=i; j_max(2)=j;
      else
        if temp_distance_squared > best_max(3)
          best_max(4)=best_max(3);  i_max(4)=i_max(3); j_max(4)=j_max(3);
          best_max(3)=temp_distance_squared; i_max(3)=i; j_max(3)=j;
        else
          if temp_distance_squared > best_max(4)
            best_max(4) = temp_distance_squared; i_max(4)=i; j_max(4)=j;
          end
        end
      end
    end
  end
end
```

```
figure(n);
clf(n);
hold on;
axis square
axis equal;
axis off;
set(gcf,'Color','w')  % white background for this image

for i=1:n     % Loop and plot all points, and blue cercles around each points
   plot(in_xy(i,1),in_xy(i,2), ...
       '-mo','LineWidth',1,'MarkerEdgeColor','k',...
       'MarkerFaceColor',[.49 1 .63],'MarkerSize',2)
   rr=0.5*sqrt(best_min);
   x0=in_xy(i,1);  y0=in_xy(i,2);
   xx=x0-rr.* cos(tt);
   yy=y0-rr.* sin(tt);
   % dessin de la cercle : trait blue de largeur 1
   plot(xx,yy,'b','LineWidth',1)
end

mmax=zeros(2); mmin=zeros(2);
mmax(1,1)=in_xy(i_max(4),1);  mmax(1,2)=in_xy(i_max(4),2);
mmax(2,1)=in_xy(j_max(4),1);  mmax(2,2)=in_xy(j_max(4),2);
plot(mmax(:,1),mmax(:,2), ...
    '-go','LineWidth',1,'MarkerEdgeColor','g',...
    'MarkerFaceColor',[.49 1 .63],'MarkerSize',1)
mmax(1,1)=in_xy(i_max(3),1);  mmax(1,2)=in_xy(i_max(3),2);
mmax(2,1)=in_xy(j_max(3),1);  mmax(2,2)=in_xy(j_max(3),2);
plot(mmax(:,1),mmax(:,2), ...
    '-bo','LineWidth',1,'MarkerEdgeColor','b',...
    'MarkerFaceColor',[.49 1 .63],'MarkerSize',1)
mmax(1,1)=in_xy(i_max(2),1);  mmax(1,2)=in_xy(i_max(2),2);
mmax(2,1)=in_xy(j_max(2),1);  mmax(2,2)=in_xy(j_max(2),2);
plot(mmax(:,1),mmax(:,2), ...
    '-ro','LineWidth',1,'MarkerEdgeColor','r',...
    'MarkerFaceColor',[.49 1 .63],'MarkerSize',2)
mmax(1,1)=in_xy(i_max(1),1);  mmax(1,2)=in_xy(i_max(1),2);
mmax(2,1)=in_xy(j_max(1),1);  mmax(2,2)=in_xy(j_max(1),2);
plot(mmax(:,1),mmax(:,2), ...
    '-ro','LineWidth',2,'MarkerEdgeColor','r',...
    'MarkerFaceColor',[.49 1 .63],'MarkerSize',2)

r1=100*best_max(2)/best_max(1); r2=100*best_max(3)/best_max(1);
r3=100*best_max(4)/best_max(1);
title({['For ', num2str(n),' points maximal Lmin/Lmax ratio is ', num2str(maximal_ratio,14)]; ...
      ['    ', num2str(r1,8),'%      ', num2str(r2,8),'%      ', num2str(r3,8),'%']});

hold off;

figure(1)
close(1);
exit_code=0;
```