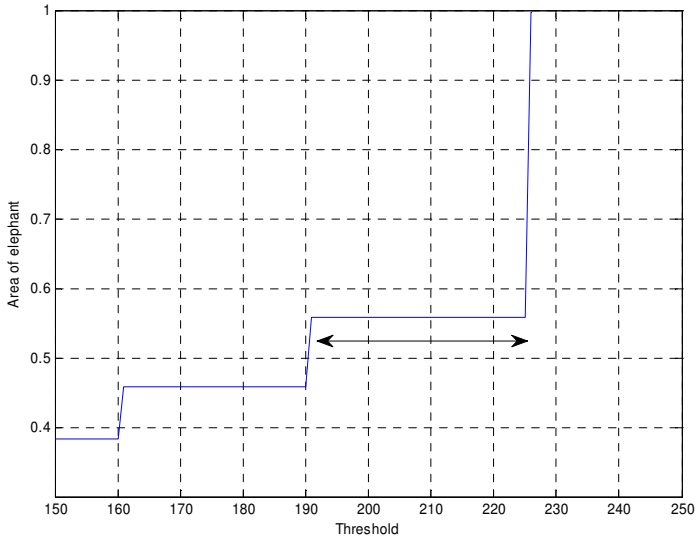
CMSC/AMSC 460 Fall 2007
Homework 3

We want to estimate the area of the elephant "elephant.tif" using three approaches. For this purpose, we define a function f(x,y)=1 if (x,y) is inside the elephant and zero otherwise. Then, we find the area of the elephant using double integral over f(x,y) for x=[0,289] and y=[0,417].

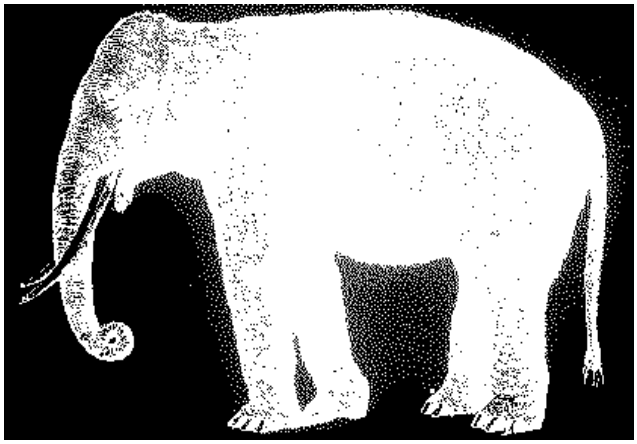
To construct such f(x,y) function we need to define a proper threshold to separate the pixels of the elephant image in two groups of background and elephant area.

Following m-file gives us a sense of a proper threshold by plotting the area of elephant vs. threshold. (The area of elephant for each threshold is defined by summation over f(x,y))

```
clear all
clc
I = imread('elephant.tif');
k=1;
for Th = 150:250;
    f_xy = I;
    f_xy(I<Th) = 1;
    f_xy(I>=Th)= 0;
    ElephantArea (k) = sum(sum(f_xy))/(289*417);
    k=k+1;
end
plot(150:250,ElephantArea3)
```



The range shown by arrow is proper for defining threshold and results in the following separated image. I chose Th= 200.



As can be seen in this image, there are some pixels that have been misclassified. This would be a source of error in our area estimation

Now, we write our program for area estimation.

```
% This program estimates the area of the elephant.tif
% I(f), using three approaches:
%   1- nested calls to quad
%   2- dblquad
%   3- summation over the function f(x,y)
%       f(x,y) = 1  if (x; y) is inside the elephant
%              = 0  otherwise.
%
% Written by Sima Taheri
% Date: 10/12/07
%
% Input : --
% Output: I_f: Estimates area of elephant by three methods
%         t  : Elapsed time for each estimation


clc
close all
clear all

% Load the elephant picture stored in elephant.tif
% and define it as a global variable
global I
I = imread('elephant.tif');

% Measure cost time using tic and toc
% 1- nested calls to quad
tic
q1 = quad (@g,1,289);
ElephantArea1 = q1/(289*417);
t1 = toc;
```

```
% 2- dblquad
tic
q2 = dblquad (@f,1,289,1,417);
ElephantArea2 = q2/(289*417);
t2 = toc;

% 3- Simple summation
tic
Th = 200;

% Define f(x,y)
f_xy = I;
f_xy (I<=Th) = 1;
f_xy (I>Th ) = 0;

ElephantArea3 = sum(sum(f_xy))/(289*417);
t3 = toc;

disp('Area of the elephant and the elapsed time')
disp('for three different methods')
disp(' ')
disp('       Method           Area     Elapsed time')
disp('-----------------------------------------------')
disp(sprintf(' Nested quad    %7.3e    %7.3e n',[ElephantArea1; t1]))
disp(sprintf(' Dblquad      %7.3e   %7.3e \n',[ElephantArea2; t2]))
disp(sprintf(' Summation    %7.3e   %7.3e \n',[ElephantArea3; t3]))

%-------------------------------------------------------------------
% Define function g(x) for nested quad
function t=g(x)
global myx

for i=1:length(x)
    myx=x(i);
    t(i) = quad (@h2,1,417);
end

%-------------------------------------------------------------------
% Define function h(y) for nested quad
function t = h(y)
global myx I

Th = 200;
t = zeros(1,length(y));
for i = 1:length(y)
    if I(round(myx),round(y(i))) < Th
        t(i) = 1;
    else
        t(i) = 0;
    end
end
```

```
%-------------------------------------------------------------------
% Define function f(x,y) for dblquad
function t = f(x,y)
global I

Th = 200;

t = zeros(1,length(y));
for i=1:length(x)
    if I(round(x(i)),round(y)) < Th
        t(i) = 1;
    else
        t(i)=0;
    end
end
```

Output:

```
Area of the elephant and the elapsed time
for three different methods

                Method            Area       Elapsed time
            ----------------------------------------------------
Tol = 1e-6      Nested quad       5.217e-001    3.178e+002
Tol = 1e-7 &    Nested quad       5.386e-001    4.717e+002
      1e-8

                Dblquad           5.627e-001    2.777e+002

                Summation         5.593e-001    8.608e-003
```

The default tolerance in both quad and dblquad functions is 1.0e-6. Therefore, the absolute error for dblquad is 1.e-6/(289*417). But for quad, it is a bit more complicated. The outer quad has an absolute error of 1.e-6, but the function evaluation in the inner quad is "noisy", also with errors of size 1.e-6, so the combined error is larger than 1.e-6/(289*417). So, it would be better to ask for higher precision in the inner quad than in the outer, to reduce the effects of the noisy function

For quad we also checked the result when tol is set to 1e-8 in the inner quad and to 1e-7 in the outer quad. As we can see, the new result of quad more similar to the results of dblquad and also the summation function. But smaller tol causes more computation time. Therefore, there is a trade-off between accuracy and cost.
For both quad and dblquad functions the number of function evaluations is 10,000+.

How you decided on the additional method?

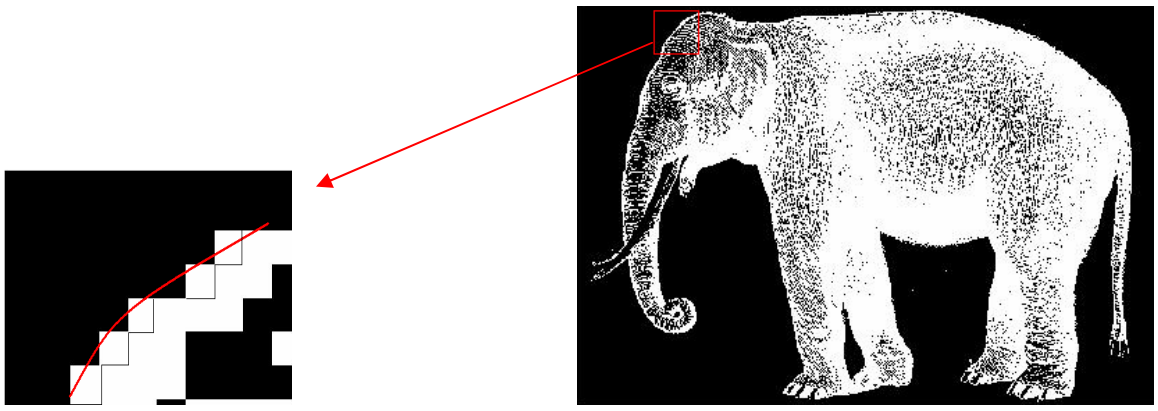
The additional method should be a ground truth for our estimation in order to enable us to evaluate our estimations using quad and dblquad functions. Therefore, I chose it as the summation over the elements of matrix f(x,y). Since f(x,y)=1 inside the elephant and =0 otherwise, summation over it gives us a very good estimation of the area ,considering the thresholding. Now we can compare our results from quad and dblquad methods with this new result.

Why each method works well or does not work well?

Regardless of thresholding error that is almost the same for all methods, among these three methods, third one gives us the best estimation of the area, then dblquad gives us nearest result to the third one and after that is the quad method. Although dblquad uses nested quad in some sense, it works better than nested quad since it has been originally designed for double integral computation. Moreover, as I mentioned before, the absolute error of nested quad is larger than dblquad.

Your assessment of sources of error and which estimate is best?

- One source of error is (as mentioned before) the image thresholding which misclassified some pixels.
- Other source of error is due to the quad and dblquad functions which have error in their integral estimations. This error is related to the tol parameter. Larger values of tol result in fewer function evaluations and faster computation, but less accurate results.
- Function quad is fitting a parabola and assuming that the function is continuous and has continuous derivatives. But our function is discontinuous -- jumping from 0 to 1 at the borders of the elephant --so the error estimate should not be trusted. This is another source of error.
- Because the function is not continuous, we would expect quad8 and quadl to give poor results. That is why the simple summation program is probably more reliable. But even after image manipulation there is error -- we assign each pixel (array value) to be either "in" or "out" of the elephant, and instead, the boundary runs through the pixel, with some percentage "in" and the rest "out". So there is error proportional to the number of pixels on the boundary of the elephant.



Optional

We can also perform some image manipulation in Matlab (and also minor manipulation in paint) to find the area of elephant with better estimation.

```
I = imread('elephant.tif');
temp = I;

Th=200;
I(temp<Th) = 1;
I(temp>=Th) =0;

imshow(I*255)
Area1 = sum(sum(I))

I1 = bwmorph(I,'close');
I2 = bwfill(I1,'holes');
figure(3)
imshow(I2*255)
Area = sum(sum(I2))
```

Real area of elephant = 0.5722