Some MATLAB programs for approximating integrals in multivariable calculus

John D. Towers
MiraCosta College

Draft Version of May 19, 2009 for use in the MiraCosta course Math 260

Warning: These notes are incomplete and almost certainly contain errors. They are made available primarily for the use of students in my classes at MiraCosta College. Please contact me for other uses. jtowers@miracosta.edu

Preface

These notes describe computer programs that are designed for use with Math 260 at MiraCosta College. Among other things, this third semester calculus course covers the usual multidimensional integrals. The idea behind the notes is to give students some practical computational experience. There are a number of computer exercises, each of which requires the student to modify one of the computer programs to solve a problem involving an integral. These exercises are meant to complement the standard pencil and paper approach.

I chose MATLAB as the computational environment simply because I have some familiarity with it. The goal of these notes is not to provide extensive training in MATLAB. In fact, I hope that without being MATLAB experts, students will be able to readily get answers to integration problems of the type discussed here.

The algorithms that produce the integral approximations are based on research that I have done over the past few years related to discretizing Heaviside functions and Dirac delta functions. In these notes the emphasis is on showing how to use the algorithms. I do not go into much detail about the algorithms themselves - more information can be found in references [6], [7], and [8].

The mathematical approach in these notes is motivated by the level set method [3], [4], [5]. The level set method is both a point of view and a collection of computational techniques. For area and volume integrals, this level set point of view is fairly compatible with the traditional calculus textbook approach.

However, for arc length integrals, line integrals, and surface integrals, the level set approach turns out to be somewhat different from the traditional textbook approach. The difference is that the traditional approach relies on a parametric representation of the domain of integration, while the level set approach represents the domain of integration using one or more level sets. For any given problem, one method or the other might be superior, depending on whether it is easier to represent the region of integration parametrically or using level sets.

A comment on notation: In order to maintain consistency between the text and the MATLAB code, I often use notation like R_1 instead of the

more customary R_1.

These notes were produced while I was on a one-semester sabbatical from my usual teaching duties at MiraCosta College. I am grateful to the college for granting this sabbatical. This project would never have made it off of my "to do" list without the sabbatical.

Contents

Chapter 1

Area Integrals: simple regions

1.1 Introduction

In this chapter we are interested in numerical approximations for area integrals (also called double integrals) of the form

$$\int\int_R f(x,y)\,dA. \tag{1.1}$$

I will assume that you have already learned how to express integrals like this as iterated integrals:

$$\int_a^b \int_{g(y)}^{h(y)} f(x,y)\,dx\,dy \text{ or } \int_a^b \int_{g(x)}^{h(x)} f(x,y)\,dy\,dx \tag{1.2}$$

Once you have the integral in one of these forms, you would use pencil and paper techniques from single-variable calculus to complete the calculation.

For example, if you were asked to use a double integral (in rectangular coordinates) to find the area of a circle of radius one, you would take $f(x,y) = 1$, and express the integral (1.1) in the form

$$\int\int_R f(x,y)\,dA = \int_{-1}^{1} \int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} dx\,dy$$

or

$$\int\int_R f(x,y)\,dA = \int_{-1}^{1} \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy\,dx,$$

and then calculate the answer by computing the one-dimensional integrals, working from right to left.

The examples and exercises in the notes will introduce you to an alternative approach for problems like this. We are going to use a computer program to find approximations to integrals like (1.1). Sometimes the computer approach is the only practical way to solve problems like this. In other situations, the paper and pencil approach is more convenient. If you are really determined to make sure your answer is correct, you might use both methods. If the answers agree, there is a decent chance that you got the right answer.

1.2 R is a simple region

Suppose we want to use an area integral to find the area of a circle of radius 1. The region of integration R is the region bounded by the circle

$$x^2 + y^2 = 1, \tag{1.3}$$

and the integrand f is just the constant function $f(x, y) = 1$. For the purpose of specifying the region of integration to our computer program, we will use a **level set function**. A level set function $u(x, y)$ has the following properties:

1. $u(x, y) > 0$ for (x, y) in the region R
2. $u(x, y) = 0$ for (x, y) on the boundary of R
3. $u(x, y) < 0$ for (x, y) not in R or on the boundary of R
4. $||\nabla u(x, y)|| \neq 0$ for (x, y) on the boundary of R

Speaking of the boundary of R, we will often use the notation ∂R to denote it. The symbol ∂ used here is the same one that you learned for partial derivatives.

It is easy to see that for our example,

$$u(x, y) = 1 - \sqrt{x^2 + y^2} \tag{1.4}$$

satisfies the requirements for a level set function. Take a look at Figure 1.1 to see what the graph of this function looks like. To check that the inside of the circle corresponds to $u(x, y) > 0$, just pick a point that is inside the

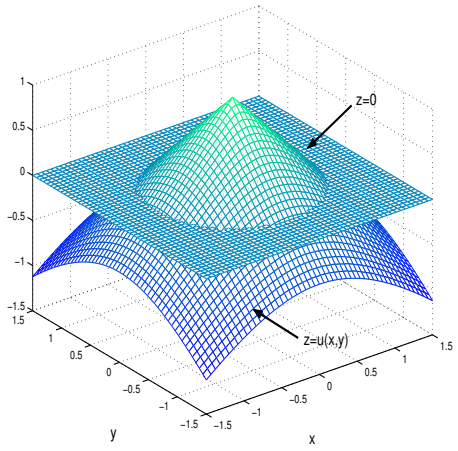

Figure 1.1: Graph of the level set function $z = u(x, y) = 1 - \sqrt{x^2 + y^2}$. The plane $z = 0$ is also plotted. The circle where the two surfaces intersect gives the zero level set, which defines the boundary of the circular region R.

circle, and evaluate u at that point. Take for example $(x, y) = (0, 0)$. Clearly, $u(0, 0) = 1$, and that's greater than zero. If you pick a point outside of the circle, like $(2, 0)$, you get $u(2, 0) = -1$. For the gradient condition, note that

$$\nabla u = -\frac{1}{\sqrt{x^2 + y^2}}(x, y),$$

and for (x, y) on ∂R,

$$||\nabla u|| = 1.$$

There are numerous other level set functions that would work here. Here are a few examples:

$$u(x, y) = 1 - (x^2 + y^2), \qquad u(x, y) = 25\left(1 - \sqrt{x^2 + y^2}\right). \tag{1.5}$$

The function $u(x, y) = 1 - \sqrt{x^2 + y^2}$ happens to have the property that $||\nabla u|| = 1$. A level set function with this property is called a signed distance function. For technical reasons, we try to use signed distance functions whenever convenient.

For this particular problem, i.e., where R is the interior of a circle, we are able to use a single smooth level set function u. When this is the case

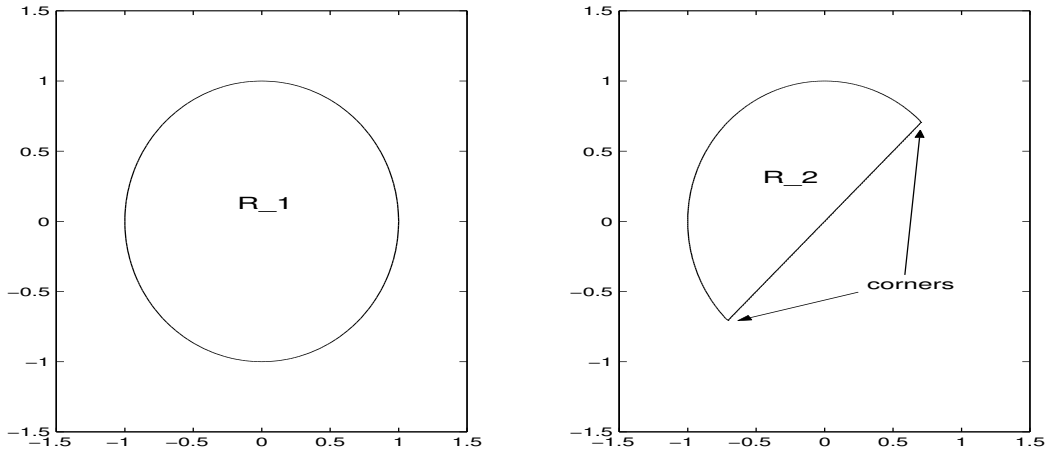

Figure 1.2: R_1 is a simple region. R_2 is not. Note that ∂R_2 has corners. ∂R_2 is not smooth.

we will call R a **simple region**. Figure 1.2 shows both a simple region, and one that is not simple.

So, the main things that you have to specify are the level set function u, and the integrand f. You must also specify which algorithm you want to use. You do this by setting fdm=1 or fdm=2. Those are the only choices. fdm=1 runs faster, but fdm=2 is more accurate. I generally use fdm=2, but there are times when fdm=1 is a a more natural choice.

There are a few other parameters that you have to determine before you can run the computer program. First you have to find a computational box that bounds the region R. For this, you need to specify four numbers:

$$\begin{aligned} \text{x\_lo} &= \text{smallest } x, \qquad & \text{x\_hi} &= \text{largest } x, \\ \text{y\_lo} &= \text{smallest } y, \qquad & \text{y\_hi} &= \text{largest } y. \end{aligned} \tag{1.6}$$

In what follows, we will use the symbol B to denote the bounding computational box determined by these four numbers. For our circle example, I chose

$$\text{x\_lo} = \text{y\_lo} = -1.5, \qquad \text{x\_hi} = \text{y\_hi} = 1.5.$$

With these values the circular region R fits completely in the bounding box

$$\text{x\_lo} \le x \le \text{x\_hi}, \qquad \text{y\_lo} \le y \le \text{y\_hi},$$

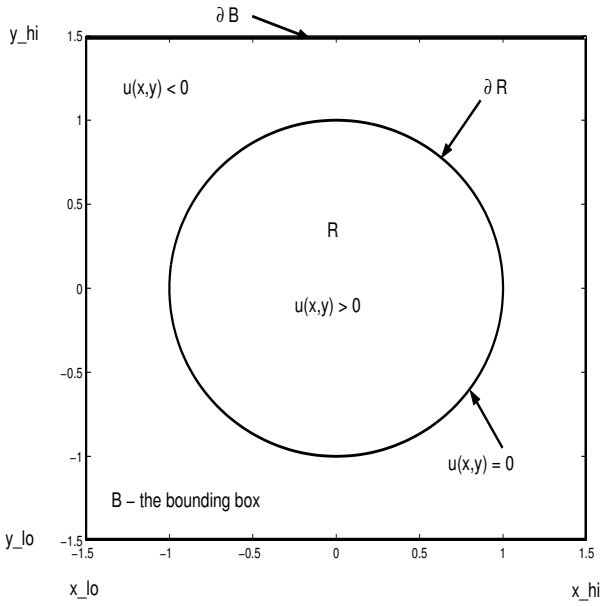

Figure 1.3: The computational setup for area integrals.

and there is some room to spare. Some extra room is required in order for the computer program to work correctly. At this point, I suggest that you have a look at Figure 1.3, which should help to clarify some of the preceding discussion.

Finally, there are two more parameters that have to be specified. They are the number of grid points in the x and y directions, nx and ny. With small values of nx and ny, the computer program will run quickly, but the answer may not be very accurate. With large values, the answer will be more accurate, but it might take a long time for the computer to complete the computation. As a rule of thumb, I generally start with nx = ny = 51, and then try nx = ny = 101, etc.

Based on the numbers nx and ny, the computer program divides the computational domain, which is basically the the region inside the bounding box, into little rectangles, with a grid point (also called mesh point) at the center of each rectangle. All quantities that the program deals with will be defined at those points, and only at those points. Figure 1.4 shows the grid that results when nx = ny = 15.

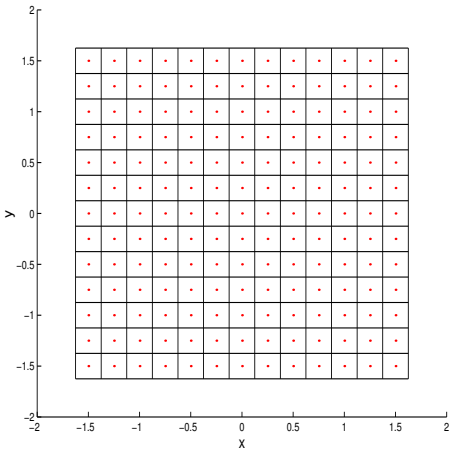

Figure 1.4: The computational grid (or mesh) for area integrals. The dots mark the locations of grid points, which are also called mesh points.

1.2.1 Summary for using area_simple.m

Here is a checklist of the items you must determine before you can run the program area_simple.m. A listing of the program area_simple.m follows the checklist. You should convince yourself that you see where each of these objects is being set up in the program.

1. Identify the level set function $u(x, y)$.
2. Identify the integrand $f(x, y)$.
3. Choose fdm=1 or fdm=2.
4. Identify the four numbers x_lo, x_hi, y_lo, y_hi that define the bounding box B.
5. Choose the numbers nx and ny.

The program area_simple.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%area_simple.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-1.5; % smallest y value
```

```
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=1; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       f(j,k) = 1; % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the Heaviside function for u
area_make_H;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*H))  %this is the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u',[0 0]); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,H');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Take a look at Figure 1.5. These plots are generated by the lines of code at the end of area_simple.m. The plot on the left gives the circular level set (or contour) $u(x, y) = 0$. The second plot is the function which is 1 at points (x, y) where $u(x, y) > 0$, and 0 elsewhere - this is called a Heaviside function. (Stay tuned for more information about Heaviside functions.) From both of these plots it is clear that the region of integration R lies completely within the computational rectangle B, and there is some room to spare.

Suggestion: It is a good idea to make some sort of plot (it could be like one of the ones that I generated, but anything of this sort will work) in order to verify that the region R lies within the bounding box B. If your plot indicates that B is not big enough, you can enlarge B by modifying one or more of the parameters x_lo, x_hi, y_lo, y_hi. Conversely, if B is excessively large, there is a loss of efficiency, because then you have to use more grid points (larger nx and ny) for a given level of accuracy. This is not overly

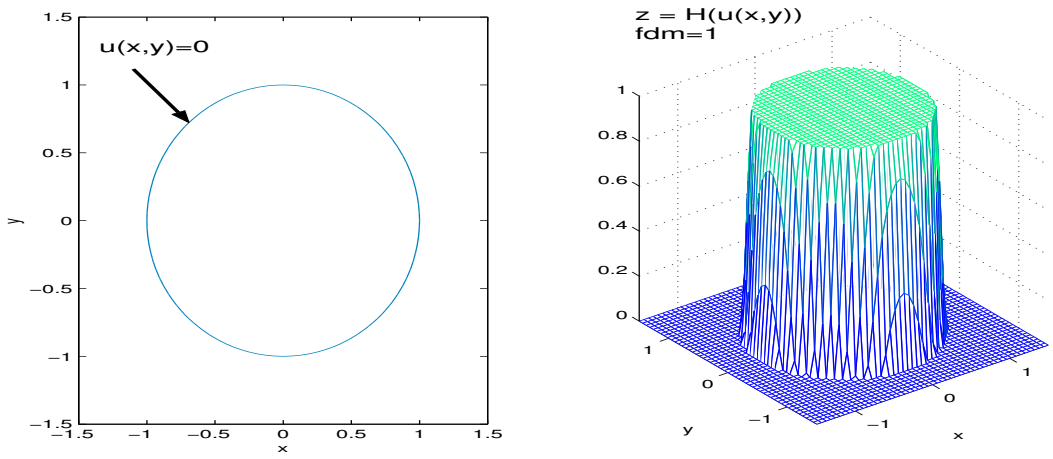

Figure 1.5: Plots for area_simple.m.

critical for two-dimensional problems, but for three dimensional problems it is an important consideration.

Exercise 1.1 In this exercise you will get some experience running the program area_simple.m.

1. At the command line type in "format short e". This puts the numerical output in a convenient type of scientific notation.

2. Run the program with nx = ny = 101, and fdm=1. Record the output. It should be an approximation to the number π. After you run the program, type in " Intgrl - pi". You will get the error in the calculation.

3. Change nx and ny to 201, and repeat Step 2. Compute the error in the calculation. You should see a reduction in the error.

3. Repeat Step 2., except change fdm=1 to fdm=2. Compare the error to what you got in Step 2.

4. Repeat Step 2., but change the bounding box. Specifically, use

$$\text{x\_lo} = \text{y\_lo} = -1.25, \qquad \text{x\_hi} = \text{y\_hi} = 1.25.$$

5. Repeat Step 2., but use the level set function $u(x, y) = 1 - x^2 - y^2$.

1.3 Estimating accuracy

In the circle example, we knew the answer, so it was easy to determine how accurate the results were. Usually we won't know the answer, but we will still want to know how accurate our calculation is.

To see how to estimate accuracy, let's continue with our circle example. Pretend that we do not know that the answer is π. We will run the version of area_simple.m shown in the listing, so fdm=1, and nx = ny = 51. Also, we will type in the command "format long" before running the program. This will allow us to see a large number of digits of the answer. After running the program with nx = ny = 51, we run it again with nx = ny = 101. The results are

$$\begin{aligned} \text{nx} = 51,\ \text{ny} = 51: &\quad \text{Intgrl=3.14179503921833} \\ \text{nx} = 101,\ \text{ny} = 101: &\quad \text{Intgrl=3.14158296375963} \end{aligned} \tag{1.7}$$

We compare the two answers and look for the leading digits that don't change - those are the digits that we assume are correct. So, using these two runs of the program, we can report that the answer is 3.141, and that all of these digits are correct. This means that the error in the approximation 3.141 is no greater than $0.00099999.... = 10^{-3}$ in absolute value. So, we could report our answer as 3.141 ± 10^{-3}, or we could say that the error in our answer of 3.141 is no larger than 10^{-3}.

Exercise 1.2 Modify the program area_simple.m in order to compute the integral

$$\int\int_R y^2 \cos(\pi x)\, dA, \tag{1.8}$$

where R is the region bounded by the ellipse

$$\frac{x^2}{4} + \frac{y^2}{25} = 1. \tag{1.9}$$

Your approximation should have an error not exceeding 10^{-2}, i.e., there should be two correct digits to the right of the decimal point.

1.4 The basic idea of the computer program

This section gives some information about the math behind the computer program. This quickly becomes technical, so I will try to just give you what you need to know. More details can be found in [8].

We will use a certain piecewise defined function called the Heaviside function, which is defined by

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases} \tag{1.10}$$

Note that we did not define H at $x = 0$. If this bothers you, you can define $H(0) = 1/2$, but it turns out not to matter.

Remark: You may have already encountered the Heaviside function, for example in a course on differential equations. In those courses, it sometimes denoted $u(x)$ instead of $H(x)$, and called the unit step function instead of the Heaviside function.

Recall the bounding box B that contains (with some room to spare) the region R. Here is a simple (but useful) observation:

$$\int\int_R f(x,y)\,dA = \int\int_B H(u(x,y))f(x,y)\,dA. \tag{1.11}$$

What we did here in going from the integral on the left to the one on the right was enlarge the region of integration from R to B, and then, to account for that, we multiplied the integrand by $H(u)$. The key thing to note here is that

$$H(u(x,y)) = \begin{cases} 1, & \text{if } (x,y) \text{ is in } R \\ 0, & \text{if } (x,y) \text{ is not in } R \end{cases} \tag{1.12}$$

See Figure 1.5 for an example of such a Heaviside function, with $u(x,y) = 1 - \sqrt{x^2+y^2}$.

The computer program specifically exploits the representation (1.11). In slightly more detail, what it does it compute an approximation to $H(u(x,y))$. This approximation is done in such a way that the integral of interest can be approximated with respectable accuracy without having to use an excessive number of grid points, i.e, nx and ny do not have to be too large.

Referrring back to the program listing for area_simple.m, the Heaviside function is built via the invocation of the script file area_make_H.m. A listing of area_make_H.m is included in Chapter 7.

One final item worth mentioning is that the plots of the Heaviside functions produced by fdm=1 and fdm=2 differ slightly in appearance. Specifically the graph produced by fdm=2 has a ragged-looking "overshoot" near ∂R. See Figure 1.7.

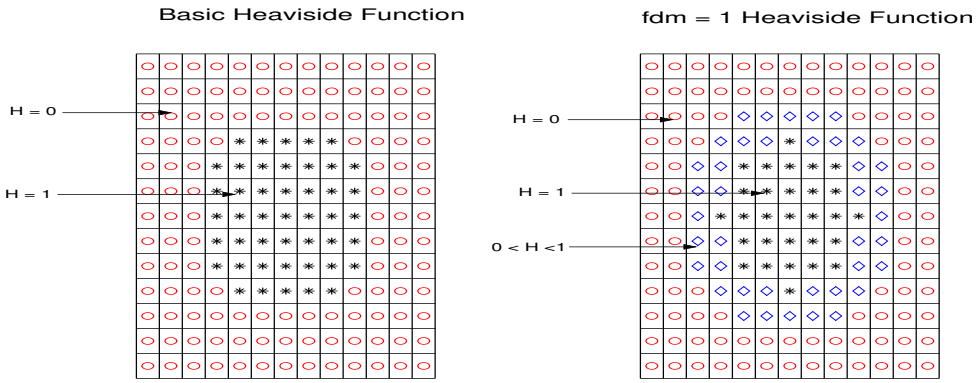

Figure 1.6:

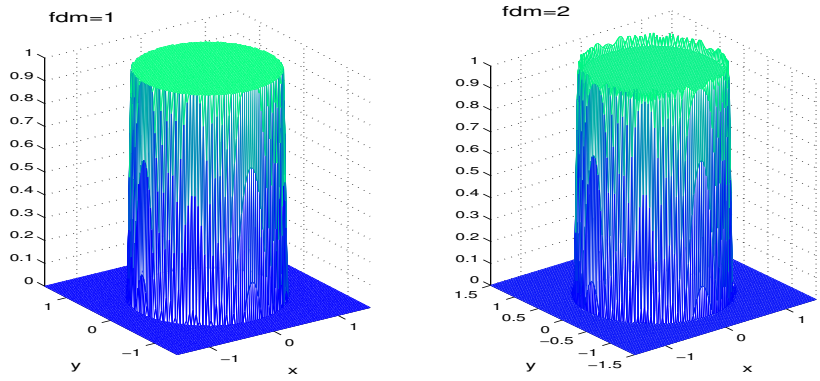

Figure 1.7: Comparison of Heaviside functions for fdm=1 and fdm=2 produced by area_simple.m.

Chapter 2

Area integrals: regions that are not simple

In this section we extend the technique of the previous section to the situation where there is no single smooth level set function. In other words, R is not what we have been calling a simple region. We will need a little bit of set theory.

Suppose that R_1 and R_2 are two sets of points in the plane. We will be dealing with the following three types of set operations.

- The **intersection** of R_1 and R_2, denoted $R\_1 \cap R\_2$, is the set of points that are in both R_1 and R_2.
- The **union** of R_1 and R_2, denoted $R\_1 \cup R\_2$, is the set of points that are in at least one of R_1 and R_2.
- The **difference** of R_1 and R_2, denoted $R\_1 - R\_2$, is the set of points that are in R_1, but not R_2.

Figure 2.1 should help clarify these three operations.

2.1 R is an intersection of simple regions

In this section we will see how to deal with problems where R is the intersection of simple regions. For example, consider the integral

$$\int\int_R e^x \cos y \, dA, \tag{2.1}$$

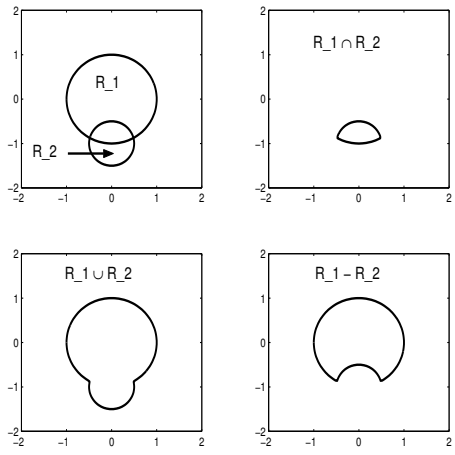

Figure 2.1: Two circular regions - their intersection, union, and difference.

where R is the region bounded above by the circle $x^2+y^2=1$, and the below by the parabola $y=x^2$. R is not a simple region. There is no single smooth level set function $u(x,y)$ that will work for this problem.

Nevertheless, we want to find a Heaviside function, which we will call H_R, for the region R. In other words, $H\_R(x,y)$ should be equal to one if (x,y) is in both R_1 and R_2, but zero otherwise.

We start by observing that $R = R\_1 \cap R\_2$, where R_1 is the region with level set function $u\_1(x,y) = 1 - \sqrt{x^2+y^2}$, and R_2 is the region with level set function $u\_2(x,y) = y - x^2$.

Here is the key idea, and this will always work if we are dealing with a situation where $R = R\_1 \cap R\_2$: We can get the Heaviside function we are looking for by simply forming the product of the Heaviside functions for R_1 and R_2:

$$H\_R(x,y) = H(u\_1(x,y)) \cdot H(u\_2(x,y)). \tag{2.2}$$

At this point you should convince yourself that you see why $H\_R(x,y) = 1$ if (x,y) is in both R_1 and R_2, but is zero otherwise

Using our new Heaviside funtion H_R, and with $f(x,y) = e^x \cos y$, we can write the integral in the form

$$\int\int_R f(x,y)\,dA = \int\int_B H\_R(x,y) f(x,y)\,dA, \tag{2.3}$$

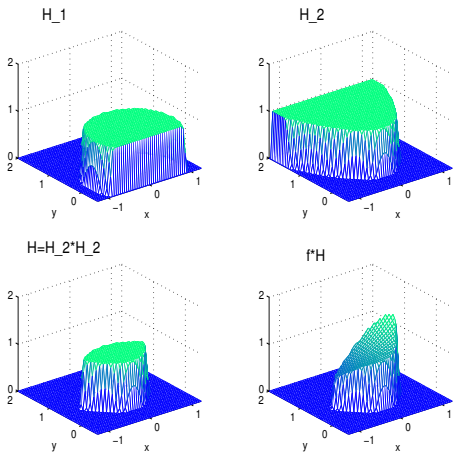

Figure 2.2: Plots produced by area_circle_parabola.m. R = intersection of circle and parabola.

and the integral on the right side of this equation is the type that we dealt with in the previous chapter.

Take a look at the listing of the program area_circle_parabola.m. It shows the process described above being implemented. Notice that the script file area_make_H.m is invoked twice, once for each of the Heaviside functions H_1 and H_2. Figure 2.3 shows the various functions that are constructed by the program. The value of the integral is ≈ 1.0901.

The program area_circle_parabola.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%area_circle_parabola.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-1.25;  % smallest x value
x_hi = 1.25;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-.5; % smallest y value
y_hi = 2.0; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
```

```
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=1; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       u_2(j,k) = y(k) - x(j)^2;
       f(j,k) = exp(x(j)*y(k)); % integrand
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the Heaviside function for u_1
u=u_1;
area_make_H;
H_1 = H;

%make the Heaviside function for u_2
u=u_2;
area_make_H;
H_2 = H;

%Make the combined Heaviside function for the region R
H = H_1.*H_2;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*H))  %this is the answer, which will be output to the screen

subplot(2,2,1);mesh(x,y,H_1'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);
subplot(2,2,2);mesh(x,y,H_2'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);
subplot(2,2,3);mesh(x,y,H'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);
subplot(2,2,4);mesh(x,y,(f.*H)'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 2.1 Approximate the integral

$$\int\int_R e^{-(x+y)}\, dA,$$

where R is the region bounded by the curves $y = x^3$ and $x = y^3$ in the first quadrant. The error in your approximation should not exceed 10^{-6}, i.e., there should be six correct digits to the right of the decimal point.

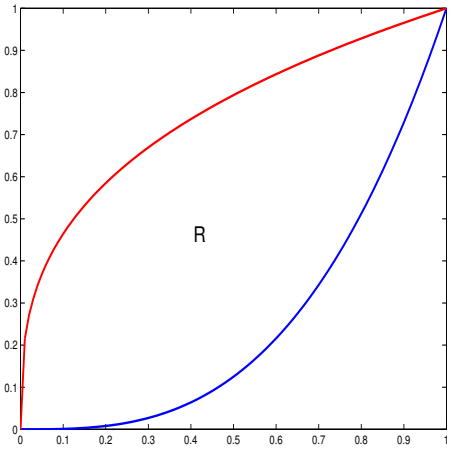

Figure 2.3: The region R for Exercise 2.1.

2.1.1 Integral over a rectangle

Suppose that we want to compute the integral

$$\int\int_R f(x,y)\,dA, \tag{2.4}$$

where R is the rectangle

$$A < x < B, \quad C < y < D.$$

This would require four different level set functions:

$$\begin{aligned} u\_1(x,y) &= x - A, \quad u\_2(x,y) = B - x, \\ u\_3(x,y) &= y - C, \quad u\_4(x,y) = D - y. \end{aligned} \tag{2.5}$$

To construct a Heaviside function for the rectangular region R, we would form the product

$$H\_R(x,y) = H(u\_1(x,y)) \cdot H(u\_2(x,y)) \cdot H(u\_3(x,y)) \cdot H(u\_4(x,y)).$$

This is a generalization of formula (2.2) for the Heaviside function of an intersection of regions.

To see a MATLAB example where a Heaviside function for a rectangle is constructed, take a look at the program area_dahlquist.m in Section 2.5.

2.2 R is a union of simple regions

Suppose that R is the union of a pair of simple regions R_1 and R_2. For example, suppose that we want to approximate

$$\int\int_R f(x,y)\,dA, \tag{2.6}$$

where $f(x,y) = 3x^2 - 2y^3$ and $R = R\_1 \cup R\_2$, where R_1 is the region bounded by the circle $(x-1/2)^2 + y^2 = 1$, and R_2 is the region bounded by the circle $(x+1/2)^2 + y^2 = 1$. In other words, the region R is formed by a pair of ovelapping circles. Let u_1 and u_2 be level set functions for the regions R_1 and R_2

$$u\_1(x,y) = 1 - \sqrt{(x-1/2)^2 + y^2}, \qquad u\_1(x,y) = 1 - \sqrt{(x+1/2)^2 + y^2}. \tag{2.7}$$

As a Heaviside function for R, we can use

$$H(u\_1(x,y)) + H(u\_2(x,y)) - H(u\_1(x,y))H(u\_1(x,y)). \tag{2.8}$$

Note that if we did not subtract the term $H(u\_1(x,y))H(u\_1(x,y))$, any point that is in both R_1 and R_2 would be counted twice, and the Heaviside function would have a value of 2 at that point. Subtracting $H(u\_1(x,y))H(u\_1(x,y))$ takes care of this double counting.

Below is a listing of the program area_two_circles.m, which implements this this process. Figure 2.5 shows the various plots produced by the program. The value of the integral is ≈ 9.2060.

The program area_two_circles.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%area_two_circles.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-2;  % smallest x value
x_hi = 2;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-2 % smallest y value
y_hi = 2; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
```

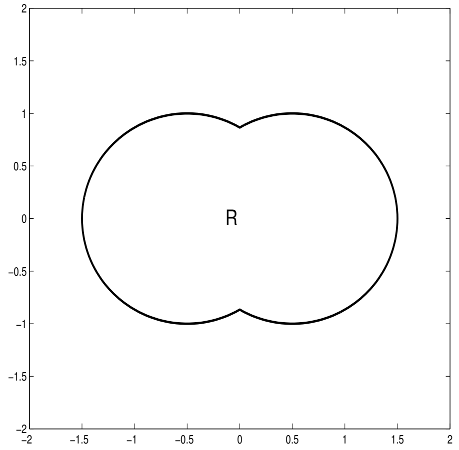

Figure 2.4: The region R formed by a union of two circles.

```
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=1; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = 1 - sqrt((x(j)-1/2)^2 + y(k)^2);  % level set function
       u_2(j,k) = 1 - sqrt((x(j)+1/2)^2 + y(k)^2);  % level set function
       f(j,k) = 3*x(j)^2 - 2*y(k)^3; % integrand
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the Heaviside function for u_1
u=u_1;
area_make_H;
H_1 = H;

%make the Heaviside function for u_2
u=u_2;
area_make_H;
H_2 = H;

%Make the combined Heaviside function for the region R
H = H_1 + H_2 - H_1.*H_2;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*H))  %this is the answer, which will be output to the screen
```

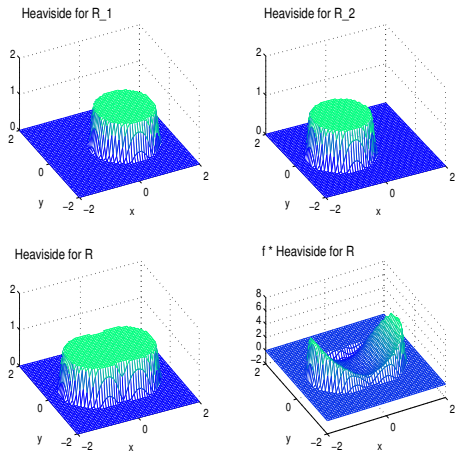

Figure 2.5: Plots produced by area_two_circles.m. R is a union of two circles.

```
subplot(2,2,1);mesh(x,y,H_1'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);colormap(winter)
subplot(2,2,2);mesh(x,y,H_2'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);colormap(winter)
subplot(2,2,3);mesh(x,y,H'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);colormap(winter)
subplot(2,2,4);mesh(x,y,(f.*H)'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 5]);colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.3 R is a difference of simple regions

Suppose that R_1 and R_2 are two simple regions in the plane, and suppose that we want to form the set $R\_1 - R\_2$, which consists of those points that are in R_1 but not R_2.

We can reduce this to a set intersection problem if we first introduce the idea of the complement of a set. Specifically, we will need the complement of R_2, denoted R_2^c, which consists of all points (x, y) that are not in R_2. A moment's consideration should convince you that

$$R\_1 - R\_2 = R\_1 \cap R\_2^c.$$

Next, note that if u_2 is a level set function for the region R_2, then $1 - H(u\_2)$ is a Heaviside function for R_2^c. Recalling how we formed a Heaviside function for the intersection of two regions in Section 2.1, the following formula provides a Heaviside function for the region $R\_1 - R\_2$:

$$H(u\_1(x,y))\,(1 - H(u\_2(x,y)))\,. \tag{2.9}$$

Example 2.1 We want to compute the integral

$$\int\int_R \left(2x^2 + 3y^2\right)\, dA, \tag{2.10}$$

where R is the region in the plane formed by starting with the circular region R_1

$$x^2 + y^2 < 1,$$

and then removing the portion of the circular region R_2

$$x^2 + (y+1)^2 < 1/2$$

that intersects the first circular region R_1. For level set functions we can use

$$u\_1(x,y) = 1 - \sqrt{x^2+y^2}, \qquad u\_2(x,y) = 1/2 - \sqrt{x^2+(y+1)^2}.$$

Then we can form the Heaviside function for the set difference $R\_1 - R\_2$ using formula (2.9). This is implemented in the program area_diff_circles.m, whose listing appears below. See Figure 2.7 for the plots produced by the program. The value of the integral is ≈ 3.2551.

The program area_diff_circles.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%area_diff_circles.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-2;  % smallest x value
x_hi = 2;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-2 % smallest y value
y_hi = 2; % largest y value
```

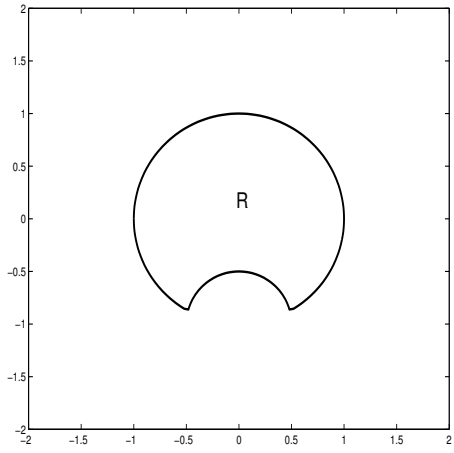

Figure 2.6: Example 2.1. The region R formed by a difference of two circles.

```
dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=1; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       u_2(j,k) = 1/2 - sqrt(x(j)^2 + (y(k)+1)^2);  % level set function
       f(j,k) = 2*x(j)^2 + 3*y(k)^2; % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the Heaviside function for u_1
u=u_1;
area_make_H;
H_1 = H;
%make the Heaviside function for u_2
u=u_2;
area_make_H;
H_2 = 1 - H;
%Make the combined Heaviside function for the region R
H = H_1.*H_2;

%Compute the integral
```

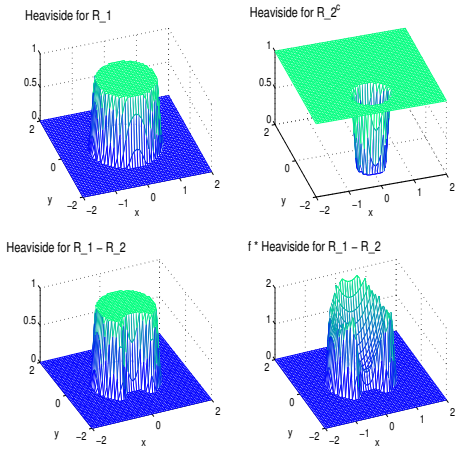

Figure 2.7: Example 2.1. Plots produced by area_diff_circles.m. R is the difference of two circles.

```
Intgrl = dx*dy*sum(sum(f.*H))  %this is the answer, which will be output to the screen

subplot(2,2,1);mesh(x,y,H_1'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);colormap(winter)
subplot(2,2,2);mesh(x,y,H_2'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);colormap(winter)
subplot(2,2,3);mesh(x,y,H'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);colormap(winter)
subplot(2,2,4);mesh(x,y,(f.*H)'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2]);colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 2.2 Approximate the integral

$$\int\int_R \left(3 - \sqrt{x^2 + y^2}\right)\, dA,$$

where R is the region inside the circle $x^2 + y^2 = 4$ and outside of the circle $x^2 + y^2 = 1$. This type of region is called an annulus. Your answer should have an error no larger than 10^{-5}, i.e., there should be 5 correct digits to the right of the decimal point.

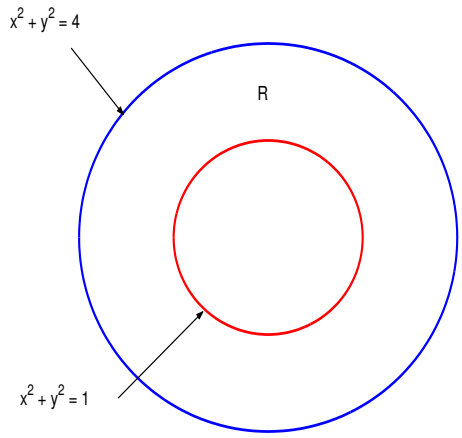

Figure 2.8: The region R for Exercise 2.2.

2.4 Summary of formulas for combining regions

Before proceeding, let's stop collect our formulas for constructing Heaviside functions:

Intersection:

$$\begin{aligned} R &= R\_1 \cap R\_2 \\ H\_R &= H(u\_1) \cdot H(u\_2) \end{aligned} \tag{2.11}$$

Union:

$$\begin{aligned} R &= R\_1 \cup R\_2 \\ H\_R &= H(u\_1) + H(u\_2) - H(u\_1) \cdot H(u\_2) \end{aligned} \tag{2.12}$$

Difference:

$$\begin{aligned} R &= R\_1 - R\_2 \\ H\_R &= H(u\_1) \cdot (1 - H(u\_2)) \end{aligned} \tag{2.13}$$

By using these formulas as many times as necessary, it is possible to construct Heaviside functions for very complicated regions. Moreover, although we have been focused on the two-dimensional setting, these formulas are also valid for three-dimensional problems.

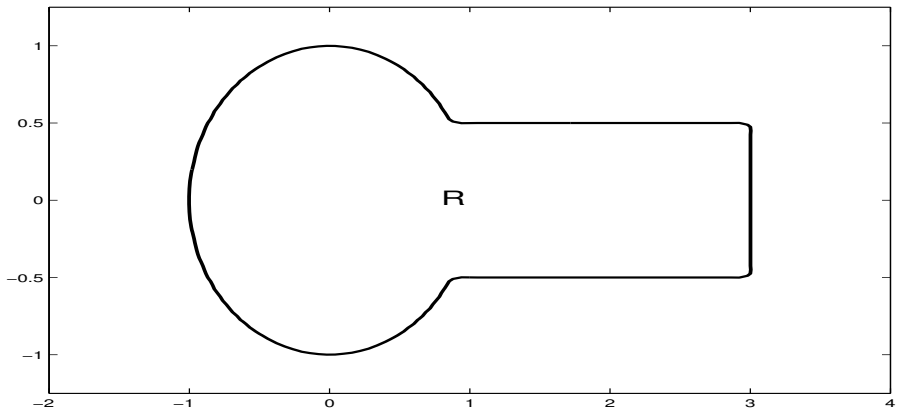

Figure 2.9: The region R formed by a union of a circle and rectangle.

2.5 A combination of operations

Sometimes it is necessary to combine two or more of the operations (intersection, union, difference) that we discussed in the preceeding sections. The following example from [1] illustrates this. The problem is to approximate

$$\int\int_R \sin^2 y \cdot \sin^2 x \cdot \left(1 + x^2 + y^2\right)\, dA. \tag{2.14}$$

Here R is the region formed by joining (i.e., taking the union of) the circular region $x^2 + y^2 < 1$ and the rectangular region

$$0 < x < 3, \qquad -1/2 < y < 1/2.$$

In order to construct a Heaviside function, we first construct a Heaviside function for the rectangle using the formula for the intersection of regions (recall the Example in Section 2.1.1):

$$H\_rectangle = H\_1 \cdot H\_2 \cdot H\_3 \cdot H\_4,$$

where

$$\begin{aligned} &H\_1(x,y) = x, \quad H\_2(x,y) = 3 - x, \\ &H\_3(x,y) = y + 1/2, \quad H\_4(x,y) = 1/2 - y. \end{aligned} \tag{2.15}$$

Then we form the Heaviside function for the region R using the formula for the union of regions:

$$H\_rectangle + H\_5 - H\_rectangle \cdot H\_5.$$

The program area_dahlquist.m, listed below, implements this process, and Figure 2.10 displays the plots that are output by the program.

Reference [1] states that the integral is equal to $.13202 \pm 10^{-5}$. Using fdm=2, nx = ny = 81, and the other parameters as shown in the displayed program listing, the program area_dahlquist.m also gives a value of .13202. Using fdm=1, nx = ny = 401, the program gave a value of .13203, not as accurate as the fdm=2 version with only a 81×81 grid. The fdm=2 algorithm is known to be more accurate than the fdm=1 version, and this example confirms it.

The program area_dahlquist.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%area_dahlquist.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-2;  % smallest x value
x_hi = 4;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-1.25; % smallest y value
y_hi = 1.25; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
   for k = 1:ny
      u_1(j,k) = x(j) - 0;
      u_2(j,k) = 3 - x(j);
      u_3(j,k) = y(k) + 1/2;
      u_4(j,k) = 1/2 - y(k);
      u_5(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
      f(j,k) = sin(y(k))^2*sin(x(j))^2/sqrt(1+x(j)^2+y(k)^2); % integrand
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the Heaviside function for u_1
u=u_1;
area_make_H;
H_1 = H;
%make the Heaviside function for u_2
u=u_2;
```

```
area_make_H;
H_2 = H;
%make the Heaviside function for u_3
u=u_3;
area_make_H;
H_3 = H;
%make the Heaviside function for u_4
u=u_4;
area_make_H;
H_4 = H;
%make the Heaviside function for u_5
u=u_5;
area_make_H;
H_5 = H;

%Make the combined Heaviside function for the region R
H_rectangle = H_1.*H_2.*H_3.*H_4;
H = H_rectangle + H_5 - H_rectangle.*H_5;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*H))  %this is the answer, which will be output to the screen

subplot(2,2,1);mesh(x,y,H_rectangle'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);colormap(winter)
subplot(2,2,2);mesh(x,y,H_5'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);colormap(winter)
subplot(2,2,3);mesh(x,y,H'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 1]);colormap(winter)
subplot(2,2,4);mesh(x,y,(f.*H)'); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 .2]);colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 2.3 Find the volume of the solid that is bounded above by the cone

$$z = 2 - \sqrt{x^2 + y^2},$$

and below by the plane $z = 0$, with a square hole bounded by the four planes $x = \pm 1/2$ and $y = \pm 1/2$. See Figure 2.11 to get an idea of what this object looks like. Your answer should have four correct digits to the right of the decimal point.

2.6 Some technical notes

2.6.1 Other ways to construct Heaviside functions

A different way to form a Heaviside function when $R = R\_1 \cap R\_2$ is to first form a new level set function according to

$$u(x, y) = \min(u\_1(x, y)\,,\, u\_2(x, y)),$$

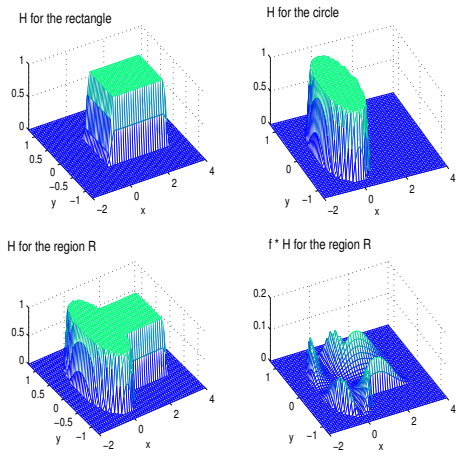

Figure 2.10: Plots produced by area_dahlquist.m. The circle looks like an ellipse in these plots because our plot region is not square.

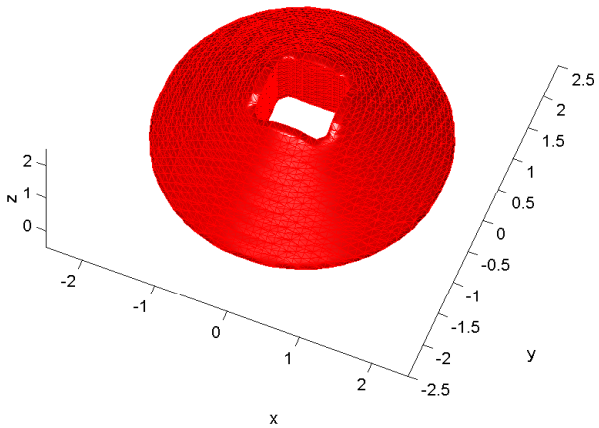

Figure 2.11: Exercise 2.3. A cone with a square hole.

and then use $H\_R = H(u)$. Similarly if $R = R\_1 \cup R\_2$, one could proceed in a similar manner, this time using

$$u(x, y) = \max(u\_1(x, y)\,,\, u\_2(x, y)),$$

and if $R = R\_1 - R\_2$, one could use

$$u(x, y) = \min(u\_1(x, y)\,,\, 1 - u\_2(x, y)).$$

Since this approach involves computing only one Heaviside function, it actually runs faster. The reason that we use the approach described in the previous sections is that for the fdm=2 algorithm, that approach is often much more accurate. In fact it is so much more accurate that the extra time spent computing multiple Heaviside functions is worthwhile. For the fdm=1 algorithm, there does not seem to be much difference in accuracy between the two approaches.

It would be interesting to know why the one method of forming Heaviside functions for fdm=2 results in so much more accurate results.

2.6.2 How much "room to spare" is required?

We said that there should be some "room to spare" between the boundary of R and the edge of the bounding box B. You might ask how much is actually required. The answer is that the fdm=1 algorithm requires that there be two grid points between ∂R and ∂B. For fdm=2, the number is 6. For fdm=2, I have noticed that the algorithm still gives useful results if this requirement is violated slightly. But 6 gridpoints is what you need for the algorithm to give its best results.

2.6.3 Quadrature as a subject and some remarks about efficiency

The computer programs you have been using (and modifying if you have been doing the exercises) perform a process called quadrature. Quadrature consists of using computational techniques to approximate integrals. Maybe you learned the midpoint rule, the trapezoidal rule, or Simpson's rule. These are sometimes called quadrature rules. The subject of quadrature has a long history (going back at least to Archimedes, I guess), and is highly developed

for the cases where the region R has a lot of symmetry. Very efficient quadrature rules for things like rectangles, triangles, and circles are abundant. By efficient, I mean that an accurate approximation is possible using a small number of evaluations of the integrand $f(x, y)$. For less symmetric regions R, like some of the ones that we have dealt with in the preceeding examples and exercises, the subject is much less developed.

I mentioned efficiency above. Let me emphasize that the algorithms that we have been using would not be considered efficient by any quadrature expert's definition. These algorithms use a large number of function evaluations. The advantage of the approach that we are pursuing is that it makes it easy for the user (you and me) to get a moderately accurate answer without too much thought or computer programming. We are offloading as much of the labor as is practically possible onto the computer. Fortunately, with the power of today's computers, this is quite feasible, especially for problems of the two-dimensional variety that we have been looking at.

As an example, recall the example in Section 2.5, which we borrowed from [1]. From the description in that book, one infers that approximately 1840 function evaluations are required to achieve an accuracy of 10^{-5}, using the method described there. With our method, (specifically using fdm=2 and nx = ny = 81), approximately 6561 function evaluations are needed to achieve this type of accuracy. Clearly our method is much les efficient. On the other hand, the method described in [1] is more problem-dependent. For example, the grid is constructed specifically so that there is a grid point located at the points where the boundary of the circle meets the boundary of the rectangle.

What would be an example where efficiency (small number of function evaluations) becomes the most important consideration? Suppose for example that each function evaluation is actually a measurement of some physical quantity, and there is a cost associated with each measurement. I have in mind things like weather stations, or pressure sensors in wind tunnels. Also, we have been doing two-dimensional problems. When we start to do three-dimensional problems in the next chapter, we will see that our computer program slows down significantly. It becomes impossible to achieve the same level of accuracy that we got on two dimensional problems. This is called the "curse of dimensionality", and would be another possible reason for developing more efficient quadrature methods.

2.6.4 Oliver Heaviside (1850-1925)

The Heaviside function is named after Oliver Heaviside. Wikipedia has an interesting sketch of his life and scientific accomplishments. Evidently Heaviside was to a large extent self-educated in mathematics and science. This is somewhat astonishing considering the very technical areas to which he made important contributions. If you search the web using "wikipedia heaviside", you will find the article.

Chapter 3

Volume integrals

3.1 Introduction

In this chapter we will extend our method from the two-dimensional setting to three-dimensional space. We will compute approximations to volume integrals of the form

$$\int\int\int_R f(x,y,z)\,dV. \tag{3.1}$$

You probably know how to evaluate an integral like (3.1) by first writing it as an iterated integral such as

$$\int_a^b \int_{p(z)}^{q(z)} \int_{g(y,z)}^{h(y,z)} f(x,y,z)\,dx\,dy\,dz.$$

If so, you already know that visualization is much harder in three dimensions. Fortunately, all of the two-dimensional methods from the previous two chapters generalize directly to the three-dimensional setting, and not too much visualization is required using our level set / Heaviside approach.

3.2 Volume integrals: simple regions

We start with a the three-dimensional version of the circle problem that we looked at in Section 1.2. The problem is to compute the volume of the interior of the sphere

$$x^2 + y^2 + z^2 = 1.$$

The integrand is just the constant function $f(x, y, z) = 1$ For a level set function, we use $u(x, y, z) = 1 - \sqrt{x^2 + y^2 + z^2}$, and the region of integration R is the set of points (x, y, z) where $u(x, y, z) > 0$. If you take a look at the program volume_simple.m, you will see that it is completely analogous to the program area_simple.m that you are already familiar with.

Here is a checklist of items that you have to specify in order to run volume_simple.m:

1. Identify the level set function $u(x, y, z)$.
2. Identify the integrand $f(x, y, z)$.
3. Choose fdm=1 or fdm=2.
4. Identify the six numbers x_lo, x_hi, y_lo, y_hi, z_lo, z_hi that define the bounding box B.
5. Choose the numbers nx, ny, and nz.

If you compare this with the check list that we used for the program area_simple_m, you will see that we need an additional two numbers, z_lo and z_hi, to specify the bounding box B. Also, in addition to the numbers nx and ny that you had to specify for the two-dimensional problems, you must provide a third number nz, which specifies the number of grid points in the z-direction.

Figure 3.1 shows the isosurface plot that is output by the program. We can use plots like this to check that the region R fits within the bounding box. Using fdm=2, and nx = ny = nz = 51, the program gives an estimate of the volume of the sphere of ≈ 4.18876. The error is $\approx 3 \times 10^{-5}$. (I used the formula for the volume of a sphere, $(4/3)\pi r^3$, with $r = 1$, to compute the error.)

The program volume_simple.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%volume_simple.m
clear; clf;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;    %number of points in the x direction
x_lo =-1.5; %smallest x value
x_hi = 1.5; %largest  x value
ny = 51;    %number of points in the y direction
y_lo =-1.5; %smallest y value
y_hi = 1.5; %largest  y value
nz = 51;    %number of points in the z direction
z_lo =-1.5; %smallest z value
z_hi = 1.5; %largest  z value

dx = (x_hi - x_lo)/(nx-1);
```

```
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
dz = (z_hi - z_lo)/(nz-1);
l=1:nz;
z= z_lo + (l-1)*dz;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm = 1; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the integrand f
for j = 1:nx
   for k = 1:ny
      for l = 1:nz
       u(j,k,l) = 1 - sqrt(x(j)^2 + y(k)^2 + z(l)^2);
       f(j,k,l) = 1;
      end
   end
end

%make the Heaviside function for u
volume_make_H;

%Compute the integral
Intgrl = dx*dy*dz*sum(sum(sum(f.*H))) %the answer, which will be output to the screen

%Plot the surface
p=patch(isosurface(x,y,z,H,1/2));
axis([x_lo x_hi y_lo y_hi z_lo z_hi]);
xlabel('x'); ylabel('y'); zlabel('z');
set(p, 'FaceColor', 'red', 'EdgeColor', 'red');
       daspect([1 1 1])
       view(3)
       camlight
       lighting phong
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 3.1 The equation

$$u(x,y,z) = 1 - z^2 - \left(\sqrt{x^2+y^2} - 2\right)^2 \tag{3.2}$$

(which I got from Reference [2]) is a level set function for a torus. A torus is a geometric object shaped like a donut. (Take a look at Figure 3.2.) Modify the program volume_simple.m to find the volume of this torus, with an error of no more than 10^{-2}. **Suggestion:** Start with nx = ny = nz = 51 and a large bounding box B. Use the isosurface plot that is produced by volume_simple.m to determine an effective bounding box. Once you have

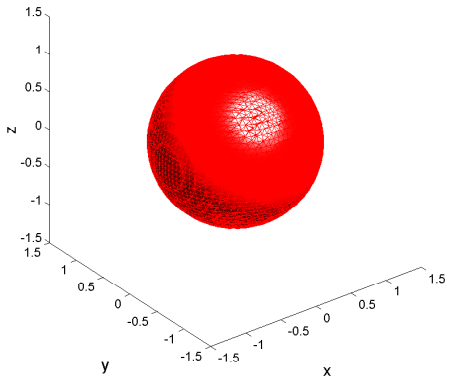

Figure 3.1: Isosurface plot for volume_simple.m.

a good bounding box, you should be able get the required accuracy with nx = ny = nz = 101.

Exercise 3.2 The moment of intertia about the z-axis of a solid occupying the three-dimensional region R is given by

$$I_z = \int\int\int_R (x^2 + y^2)\rho(x, y, z)\, dV, \tag{3.3}$$

where ρ is the density. This quantity is useful in describing the rotational motion of a solid about the z-axis. Find the moment of inertia about the z-axis of the region R bounded by the sphere $(x-1)^2 + y^2 + z^2 = 1$. Assume that $\rho(x, y, z) = 1$. Your answer should have three correct digits to the right of the decimal point.

3.3 Volume integrals: regions that are not simple

3.3.1 Intersection of regions

We want to find the area of the three dimensional region R that is bounded by the cylinders

$$x^2 + y^2 = 1, \quad x^2 + z^2 = 1.$$

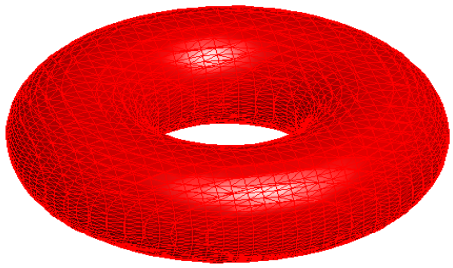

Figure 3.2: Exercise 3.1. A torus is shaped like a donut.

This is a standard problem found in many calculus textbooks. Letting R_1 denote the region $x^2 + y^2 < 1$, and R_2 denote the region $x^2 + z^2 < 1$, the region R is just the intersection of R_1 and R_2; in symbols $R = R\_1 \cap R\_2$. With this in mind, we can simply proceed as we did when working with intersections of two-dimensional regions. For level set functions, we use

$$u\_1(x, y, z) = 1 - \sqrt{x^2 + y^2}, \qquad u\_2(x, y, z) = 1 - \sqrt{x^2 + z^2},$$

then form the Heaviside functions $H(u\_1)$ and $H(u\_2)$, and finally form the Heaviside function for the region R according to

$$H\_R(x, y, z) = H(u\_1(x, y, z)) \cdot H(u\_2(x, y, z)).$$

If you look at the listing for the program volume_two_cylinders.m, you will see that it implements the process that we just described. With fdm=2, and the other parameters as shown in the listing of volume_two_cylinders, the program gives an answer of 5.3289, yielding an error of 4.4×10^{-3}. (The exact value is $16/3 = 5.33333...$) With nx = ny = nz = 101, the approximation is 5.3322, with an error of 1.1×10^{-3}.

Figure 3.3 will help you to see what the region R looks like; but notice that we did not have to visualize R in order to get an answer.

The program volume_two_cylinders.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%volume_two_cylinders.m
clear; clf;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;    %number of points in the x direction
x_lo =-1.5; %smallest x value
x_hi = 1.5; %largest  x value
ny = 51;    %number of points in the y direction
y_lo =-1.5; %smallest y value
y_hi = 1.5; %largest  y value
nz = 51;    %number of points in the z direction
z_lo =-1.5; %smallest z value
z_hi = 1.5; %largest  z value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
dz = (z_hi - z_lo)/(nz-1);
l=1:nz;
z= z_lo + (l-1)*dz;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm = 2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the integrand f
for j = 1:nx
   for k = 1:ny
      for l = 1:nz
         u_1(j,k,l) = 1 - sqrt(x(j)^2 + y(k)^2);
         u_2(j,k,l) = 1 - sqrt(x(j)^2 + z(l)^2);
         f(j,k,l) = 1;
      end
   end
end

%make the Heaviside function for u_1
u=u_1;
volume_make_H;
H_1 = H;
%make the Heaviside function for u_2
u=u_2;
volume_make_H;
H_2 = H;
%Make the combined Heaviside function for the region R
H = H_1.*H_2;

%Compute the integral
Intgrl = dx*dy*dz*sum(sum(sum(f.*H))) %the answer, which will be output to the screen
```

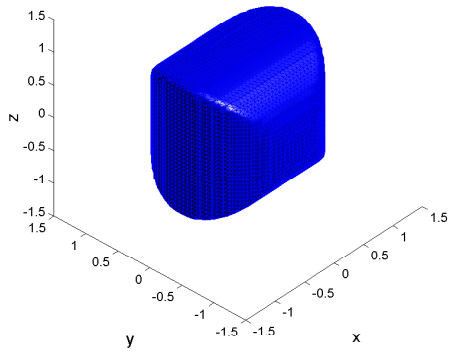

Figure 3.3: Isosurface plot for volume_two_cylinders.m. The solid region is the intersection of a pair of cylinders.

```
%Plot the surface
p=patch(isosurface(x,y,z,H,1/2));
axis([x_lo x_hi y_lo y_hi z_lo z_hi]);
xlabel('x'); ylabel('y'); zlabel('z');
set(p, 'FaceColor', 'blue', 'EdgeColor', 'blue');
       daspect([1 1 1])
       view(3)
       camlight
       lighting phong
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 3.3 Find the volume of the solid that is bounded above by the sphere $x^2 + y^2 + (z-4)^2 = 16$, on the sides by the cone $z = \sqrt{x^2+y^2}$, and below by the plane $z = 1$. All of the digits to the left of the decimal point in your approximation should be correct.

Exercise 3.4 Find the volume of the solid that is bounded above by the cone

$$z = 2 - \sqrt{x^2 + y^2},$$

and below by the plane $z = 0$, with a square hole bounded by the four planes $x = \pm 1/2$ and $y = \pm 1/2$. Your answer should have at least one correct digit to the right of the decimal point.

Exercise 3.5 Find the mass M of the solid occupying the region R that is bounded on the sides by the planes $x = -1$, $x = 1$, $y = -1$, $y = 1$, and is bounded above by and below by the sphere $x^2 + y^2 + z^2 = 4$. Assume that the density is given by $\rho(x, y, z) = 2\sqrt{x^2 + y^2}$. Recall that the relationship between mass and density is

$$M = \int\int\int_R \rho(x, y, z)\, dA.$$

Your answer should have at least one correct digit to the right of the decimal point, i.e., the error should be less than 10^{-1}.

Exercise 3.6 Find the moment of inertia about the z-axis for the solid bounded by the surfaces

$$z = 0, \quad z = x, \quad y^2 = x - 4.$$

Use formula (3.3), and assume that $\rho(x, y, z) = x$. Your approximation should have two correct digits to the right of the decimal point.

Chapter 4

Arc length integrals for plane curves

4.1 Arc length for simple closed plane curves

The standard formula for the length of a plane curve that starts at (x_1, y_1) and ends at (x_2, y_2) is this:

$$\text{arc length} = \int_a^b \sqrt{x'(t)^2 + y'(t)^2}\, dt. \tag{4.1}$$

This formula assumes that you have a pair of parametric equations for the curve:

$$\begin{cases} x = x(t) \\ y = y(t) \end{cases} \tag{4.2}$$

and that

$$(x(a), y(a)) = (x_1, y_1), \qquad (x(b), y(b)) = (x_2, y_2).$$

In this section, we will assume that $(x_1, y_1) = (x_2, y_2)$. This means that the curve ends where it started. This is called a **closed** curve. A circle would be a simple example of a closed curve. In fact, we are going to assume that the curve is the boundary of a simple region R, and that the curve consists of those points in the plane satisfying $u(x, y) = 0$, where u is a level set function. We will call such a curve **simple** closed curve. For our circle example, we could take $u(x, y) = 1 - \sqrt{x^2 + y^2}$, exactly as we did previously.

Recall that if the curve is parameterized by arc length, then

$$\begin{cases} x = x(s) \\ y = y(s) \end{cases} \tag{4.3}$$

with

$$x'(s)^2 + y'(s)^2 = 1$$

and

$$(x(0), y(0)) = (x_1, y_1), \qquad (x(L), y(L)) = (x_2, y_2).$$

In this case, we will have

$$\text{arc length} = \int_0^L ds.$$

From the previous chapters, we know that if B is a bounding box for the region R, then

$$\text{area of } R = \int\int_B H(u(x, y))\, dA. \tag{4.4}$$

There is a very similar formula for the arc length of the boundary of R:

$$\text{arc length of } \partial R = \int\int_B DH(u(x, y))\, dA. \tag{4.5}$$

We could equivalently write this formula as

$$\int_{\partial R} ds = \int\int_B DH(u(x, y))\, dA. \tag{4.6}$$

Notice that the only difference between (4.4) and (4.5) is that the function DH has replaced the Heaviside function H. For now, what you need to know about DH is that the computer program for arc length is based on it, in the same way that the computer program for area was based on the Heaviside function H.

The computer program arc_2d_closed.m (see the listing below) computes the arc length of the circle $x^2 + y^2 = 1$, based on the formula (4.5). If you take a look at the listing for arc_2d_closed.m that appears below, you will see that it looks very similar to the program area_simple.m. The main difference is that instead of invoking the script file area_make_H.m to build a Heaviside

function, it invokes the script file arc_2d_make_DH.m, which builds the object DH.

Figure 4.1 shows the plots produced by the program. The plot on the right shows that DH is zero everywhere except in a narrow region surrounding the level set $u = 0$.

The program arc_2d_closed.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%arc_2d_closed.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 101;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 101; % number of grid points in y direction
y_lo =-1.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       f(j,k) = 1; % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH function for u
arc_2d_make_DH;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*DH))  %this is the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u',[0 0]); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,DH');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 5/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

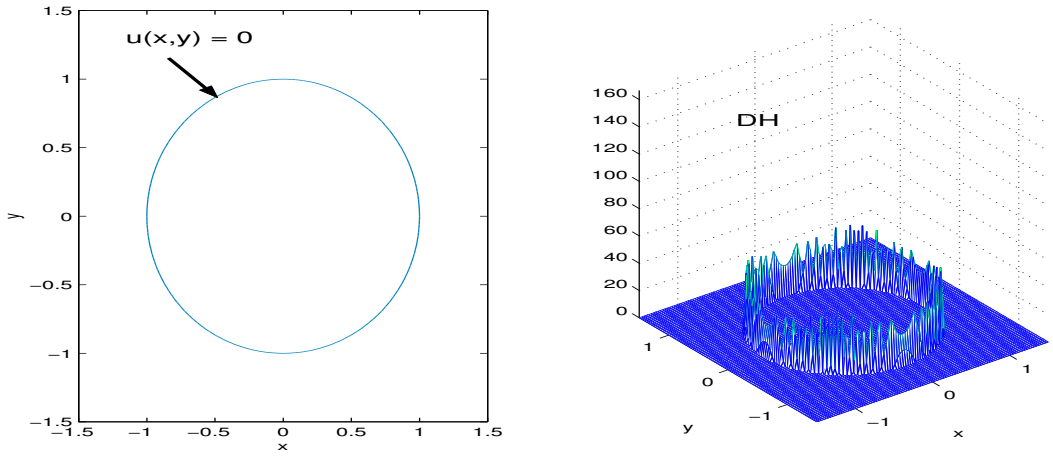

Figure 4.1: Plots for arc_2d_2d_closed.m. Arc length of a cicle.

Exercise 4.1 Find the arc length of the ellipse

$$\frac{x^2}{4} + \frac{y^2}{9} = 1.$$

Do this by modifying the program arc_2d_closed.m. Your answer should have two correct digits to the right of the decimal point.

4.1.1 A technical note

The quantity DH that we have just encountered is really the following:

$$DH(u) = \delta(u)||\nabla u||. \tag{4.7}$$

In this formula δ is an object called the Dirac delta function. It is actually not a function, but in many situations you can operate with it as if it were. You may have already encountered the Dirac delta function (often just called the delta function, or sometimes the unit impulse function), especially if you have taken a course in differential equations. We won't get too involved with the details here. I will just note a couple of things. The first is that formula (4.5) is just expressing the following:

$$\int_{\partial R} ds = \int\int_B \delta(u(x,y))||\nabla u(x,y)||\, dA, \tag{4.8}$$

which is a well known fact. The second is that, from a certain point of view,

$$\frac{d}{dz}H(z) = \delta(z).$$

4.1.2 Paul Dirac (1902-1984)

Wikipedia has a sketch of the life and scientific accomplishments of Dirac, for whom the delta function is named. He had the title of Lucasian Professor of Mathematics at Cambridge, which was also Isaac Newton's title at Cambridge. If you search the web using "wikipedia dirac", you will find the article.

4.1.3 A more general arc length integral

Notice that in the program arc_2d_closed.m there is still an integrand. I am referring to the line of code consisting of the instruction f(j,k)=1. For problems where just the arc length is desired, we can simply leave this line of code as it is. However, it is possible to use the program to compute more general integrals of the form

$$\int_a^b f(x(t), y(t))\sqrt{x'(t)^2 + y'(t)^2}\, dt. \tag{4.9}$$

Of course, the program bases its approximation on the equivalent formulation

$$\begin{aligned}\int_{\partial R} f(x(s), y(s))\, ds &= \int\int_B f(x, y)\, \delta(u(x, y))||\nabla u(x, y)||\, dA \\ &= \int\int_B f(x, y)\, DH(u(x, y))\, dA\end{aligned} \tag{4.10}$$

Example 4.1 A circular wire of radius 2, centered at $(0, 0)$ has a linear density given by

$$\rho(x, y) = \frac{1}{4}(x^2 + y^4).$$

Find the total mass of the wire. The mass M is given by

$$M = \int_C \rho(x, y)\, ds.$$

If we take $u(x,y) = 1 - \sqrt{x^2+y^2}$ we can equivalently express the mass as

$$M = \int_B \rho(x,y) DH(u(x,y))\, dA.$$

The program arc_2d_closed.m, listed below, implements this calculation. The mass is ≈ 1.37.

The program arc_2d_closed.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%arc_2d_wire.m
clear;
%------------------------------
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 401;  % number of grid points in x direction
x_lo =-1.25;  % smallest x value
x_hi = 1.25;  % largest y value
ny = 401; % number of grid points in y direction
y_lo =-1.25; % smallest y value
y_hi = 1.25; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u(j,k) = 1- sqrt(y(k)^2 + x(j)^2);  % level set function
       f(j,k) = (1/4)*(x(j)^2+y(k)^4); % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH function for u_1
arc_2d_make_DH;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*DH))  %this is the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u',[0 0]); xlabel('x'); ylabel('y');
hold on;
subplot(1,2,2);mesh(x,y,DH');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 10/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

4.2 Arc length integrals for closed plane curves that are not simple

In this section we are still dealing with arc length integrals for closed curves in the plane. We will consider the situation where the curve is the boundary of a region R that is not simple. Recall that we dealt with area integrals for regions of this type in Chapter 2. The good news is that we can use much of what we learned there for the arc length problems that we will deal with here.

For example, if $R = R\_1 \cap R\_2$, where R_1 and R_2 are simple regions, we know that

$$H\_R = H(u\_1)H(u\_2)$$

is a Heaviside function for R. Recall that we needed this Heaviside function to plug into the formula

$$\text{area of } R = \int\int_B H\_R(x, y)\, dA.$$

Similarly, to compute the arclength of ∂R, we need a DH_R function to plug into the formula

$$\text{arc length of } \partial R = \int\int_B DH\_R(x, y)\, dA.$$

It turns out that there is a simple formula for DH_R:

$$DH\_R = H(u\_1)DH(u\_2) + H(u\_2)DH(u\_1).$$

Similarly, there are formulas for the operations of union and difference. They, along with the formula above, are included in the following expanded list of formulas that we compiled in Section 2.4:

Intersection:

$$\begin{aligned} &R = R\_1 \cap R\_2 \\ &H\_R = H(u\_1) \cdot H(u\_2) \\ &DH\_R = H(u\_1)DH(u\_2) + H(u\_2)DH(u\_1) \end{aligned} \tag{4.11}$$

Union:

$$\begin{aligned} &R = R\_1 \cup R\_2 \\ &H\_R = H(u\_1) + H(u\_2) - H(u\_1) \cdot H(u\_2) \\ &DH\_R = DH(u\_1) + DH(u\_2) - H(u\_1)DH(u\_2) - H(u\_2)DH(u\_1) \end{aligned} \tag{4.12}$$

Difference:

$$
\begin{aligned}
&R = R\_1 - R\_2 \\
&H\_R = H(u\_1) \cdot (1 - H(u\_2)) \\
&DH\_R = DH(u\_1) + H(u\_1)DH(u\_2) - H(u\_2)DH(u\_1)
\end{aligned}
\tag{4.13}
$$

If the curve is the boundary ∂R of a more complicated region R, the basic process is the same:

- Compute H_R using the rules above for intersection, union, and difference as many times as necessary.
- Compute DH_R by applying the rules above to H_R.

Example 4.2 Apply the intersection formula to find the arc length of the curve formed by the intersection of the circles

$$
\begin{aligned}
(x - 1/2)^2 + y^2 &= 1, \\
(x + 1/2)^2 + y^2 &= 1.
\end{aligned}
\tag{4.14}
$$

For level set functions, we can use

$$
\begin{aligned}
u\_1(x, y) &= 1 - \sqrt{(x - 1/2)^2 + y^2}, \\
u\_1(x, y) &= 1 - \sqrt{(x + 1/2)^2 + y^2}.
\end{aligned}
\tag{4.15}
$$

Take a look at the listing for the program arc_2d_2circles_intersection.m. It implements the intersection formula, based on these two level set functions. The arc length of the curve is ≈ 4.18.

The program arc_2d_2circles_intersection.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%arc_2d_2circles_intersection.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 101;  % number of grid points in x direction
x_lo =-2.5;  % smallest x value
x_hi = 2.5;  % largest y value
ny = 101; % number of grid points in y direction
y_lo =-2.5; % smallest y value
y_hi = 2.5; % largest y value
```

```
dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
   for k = 1:ny
       u_1(j,k) = 1 - sqrt((x(j)-1/2)^2+y(k)^2);  % level set function
       u_2(j,k) = 1 - sqrt((x(j)+1/2)^2+y(k)^2);  % level set function
       f(j,k) = 1; % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH functions
u = u_1;
area_make_H;
H_1 = H;
arc_2d_make_DH;
DH_1 = DH;

u = u_2;
area_make_H;
H_2 = H;
arc_2d_make_DH;
DH_2 = DH;

%the combined DH function
DH_R = H_2.*DH_1 + H_1.*DH_2;

%Compute the integral
Intgrl = dx*dy*sum(sum(f.*DH_R)) %the answer, to be output to the screen

subplot(1,2,1);contour(x,y,u_1',[0 0]); hold on;
contour(x,y,u_2',[0 0]); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,(DH_R)');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 2/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 4.2 Find the arc length of the boundary of the region formed by the union of the two circles

$$\begin{aligned}(x-1/2)^2+y^2&=1,\\(x+1/2)^2+y^2&=1.\end{aligned}\tag{4.16}$$

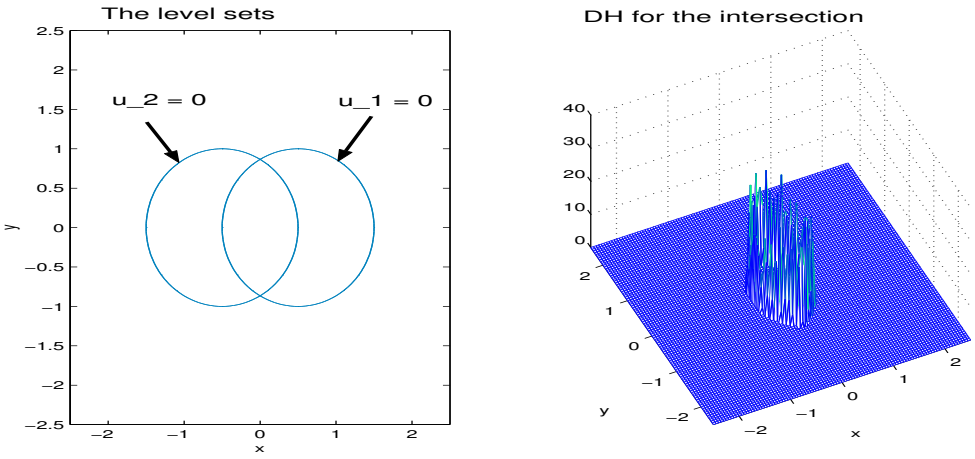

Figure 4.2: Example 4.2. Plots for arc_2d_2circles_intersection.m. Arc length of the intersection of two circles.

More specifically, take R_1 to be the region bounded by the first circle, R_2 to be the region bounded by the second circle, and find the arc length of the boundary of $R\_1 \cup R\_2$. Your answer should have two correct digits to the right of the decimal point.

Exercise 4.3 Find the arc length of the boundary of the region formed by the difference of the two circles

$$\begin{aligned} (x - 1/2)^2 + y^2 &= 1, \\ (x + 1/2)^2 + y^2 &= 1. \end{aligned} \tag{4.17}$$

More specifically, take R_1 to be the region bounded by the first circle, R_2 to be the region bounded by the second circle, and find the arc length of the boundary of $R\_1 - R\_2$. Your answer should have two correct digits to the right of the decimal point.

4.3 Arc length for open plane curves

Up to this point we have concentrated on closed plane curves, i.e., curves that end where they started. In this section we will deal with open plane curves, i.e., plane curves that are not closed.

For example, suppose that we want to find the arc length of the semicircular curve satisfying

$$x^2 + y^2 = 1, \quad y > 0.$$

We will use two level set functions for this problem. The first one,

$$u\_1(x, y) = 1 - \sqrt{x^2 + y^2}$$

is the same one the we used to represent the entire circle $x^2 + y^2 = 1$ in Section 4.1. The second level set function is

$$u\_2(x, y) = y,$$

which is positive only for points (x, y) with $y > 0$. Recall that the arc length of the entire circle was given by the integral

$$\int\int_B DHu\_1(x, y)\, dA.$$

In order to modify this so that we only get the arc length of the semicircle, we can multiply $DHu\_1(x, y)$ by $H(u\_2(x, y))$, which cancels out the contribution that comes from the portion of the curve where $y < 0$. So the integral for arc length of the semicircle is

$$\int\int_B H(u\_2(x, y))\, DHu\_1(x, y)\, dA.$$

If you take a look at the listing of the program arc_2d_semicircle.m that appears below, you will see that this is what is being implemented. Figure 4.3 shows the plots that are output by the program.

The program arc_2d_semicircle.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%arc_2d_semicircle.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 101;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 101; % number of grid points in y direction
y_lo =-1.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
```

```
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       u_2(j,k) = y(k); % level set function
       f(j,k) = 1; % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH function for u_1
u=u_1;
arc_2d_make_DH;
DH_1 = DH;
%make the Heaviside function for u_2
u=u_2;
area_make_H;
H_2 = H;

%the DH function for the open curve;
DH_C = DH_1.*H_2;
%Compute the integral
Intgrl = dx*dy*sum(sum(f.*DH_1.*H_2))  %this is the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u_1',[0 0]); xlabel('x'); ylabel('y');
hold on;
contour(x,y,u_2',[0 0])
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,DH_C');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi 0 10/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 4.4 Find the arc length of the portion of the curve $y = x^2$ between the points $(0, 0)$ and $(1, 1)$. Your answer should have three correct digits to the right of the decimal point.

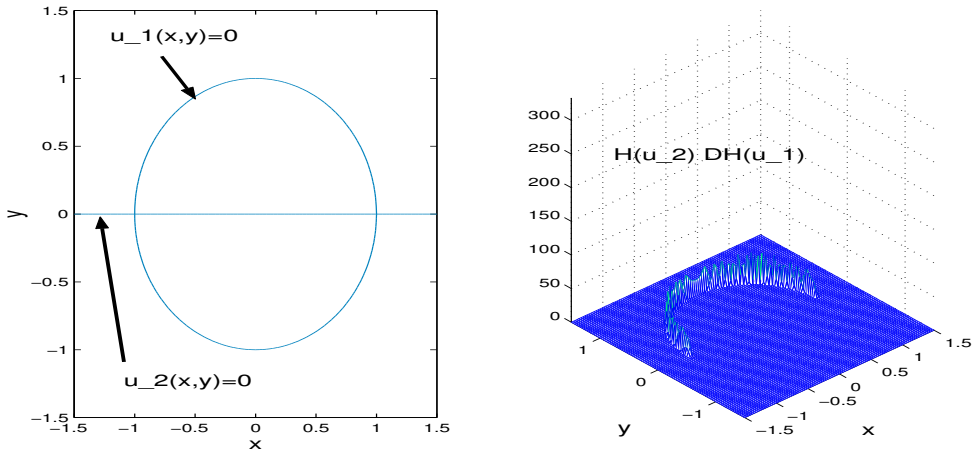

Figure 4.3: Plots for arc_2d_semicircle.m. Arc length of a semicircle.

Chapter 5

Surface integrals

5.1 Simple closed surfaces

Recall that a simple closed curve can be described by a single level set function, $u(x, y)$. It is the set of points in the plane satisfying $u(x, y) = 0$. The boundary of a circle is an example. The analogous type of object in the three dimensional setting is a simple closed surface, which can again be described using a single level set function $u(x, y, z)$, this time using the equation $u(x, y, z) = 0$. An example would be the surface of a sphere. There is a formula for the surface area of a simple closed surface that is analogous to the formula for the arc length of a simple closed curve. The formula is

$$\text{surface area} = \int\int\int_B DH(u(x, y, z))\, dV.$$

Here B is a bounding box that contains (with some extra room) the surface $u(x, y, z) = 0$.

Below is a listing of the program surface_3d_closed.m, which implements this formula (actually an approximation of it) for a sphere of radius one. For a level set, it uses

$$u(x, y, z) = 1 - \sqrt{x^2 + y^2 + z^2}.$$

The program surface_3d_closed.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%surface_3d_closed.m
clear;
%set up the grid
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-1.5; % smallest y value
y_hi = 1.5; % largest z value
nz = 51; % number of grid points in z direction
z_lo =-1.5; % smallest z value
z_hi = 1.5; % largest z value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
dz = (z_hi - z_lo)/(nz-1);
l=1:nz;
z= z_lo + (l-1)*dz;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
   for k = 1:ny
      for l = 1:nz
       u(j,k,l) = 1 - sqrt(x(j)^2 + y(k)^2 + z(l)^2);
       f(j,k,l) = 1;
      end %for l
   end %for k
end %for j

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH function for u
surface_3d_make_DH;

%Compute the integral
Intgrl = dx*dy*dz*sum(sum(sum(f.*DH)))  %this is the answer, which will be output to the screen

%Plot the surface
p=patch(isosurface(x,y,z,H,1/2));
axis([x_lo x_hi y_lo y_hi z_lo z_hi]);
xlabel('x'); ylabel('y'); zlabel('z');
set(p, 'FaceColor', 'red', 'EdgeColor', 'red');
       daspect([1 1 1])
       view(3)
       camlight
       lighting phong
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 5.1 Find the surface area of the ellipsoid

$$2x^2 + 5y^2 + z^2 = 3.$$

Your answer should have one correct digit correct to the left of the decimal point.

Exercise 5.2 Find the surface area of the torus in Section 3.1. Your answer should have all digits correct to the left of the decimal point.

5.2 Open surfaces

The surface of a sphere is an example of a closed surface. It has no boundary. An example of an open surface is the surface of a hemisphere. Take for example the set of points satisfying

$$x^2 + y^2 + z^2 = 1, \qquad x > 0.$$

This describes the surface of a hemisphere. The boundary of this surface is the circle (in the $y - z$ plane) defined by $y^2 + z^2 = 1$. Figure 5.1 will give you an idea of what this surface looks like. We proceed as we did when computing the arc length of an open plane curve in Section 4.3 - the situation is entirely analogous. Defining the two level set functions

$$u\_1(x, y, z) = 1 - \sqrt{x^2 + y^2 + z^2}, \qquad u\_2(x, y, z) = x,$$

the integral that gives the desired surface area is

$$\int\int\int_B H(u\_2(x, y, z))\, DH(u\_1(x, y, z))\, dV.$$

Here B is a bounding box that is large enough to contain the hemisphere. The program surface_3d_hemisphere.m, whose listing appears below implements this.

The program surface_3d_hemisphere.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%surface_3d_hemisphere.m
clear; clf;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-1.5; % smallest y value
```

```
y_hi = 1.5; % largest z value
nz = 51; % number of grid points in z direction
z_lo =-1.5; % smallest z value
z_hi = 1.5; % largest z value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
dz = (z_hi - z_lo)/(nz-1);
l=1:nz;
z= z_lo + (l-1)*dz;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
   for k = 1:ny
      for l = 1:nz
         u_1(j,k,l) = 1 - sqrt(x(j)^2 + y(k)^2 + z(l)^2);
         u_2(j,k,l) = x(j);
       f(j,k,l) = 1;
      end %for l
   end %for k
end %for j

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH function for u_1
u = u_1;
surface_3d_make_DH;
DH_1 = DH;
%make the H function for u_2
u=u_2;
volume_make_H;
H_2 = H;

%Compute the integral
Intgrl = dx*dy*dz*sum(sum(sum(f.*H_2.*DH_1)))  %the answer, which will be output to the screen

%Plot the surface
p=patch(isosurface(x,y,z,H_2.*DH_1,1/2));
axis([x_lo x_hi y_lo y_hi z_lo z_hi]);
xlabel('x'); ylabel('y'); zlabel('z');
set(p, 'FaceColor', 'red', 'EdgeColor', 'red');
       daspect([1 1 1])
       view(3)
       camlight
       lighting phong
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Example 5.1 Here is another example showing how to find the surface area of an open surface. The problem is to find the surface area of the portion of

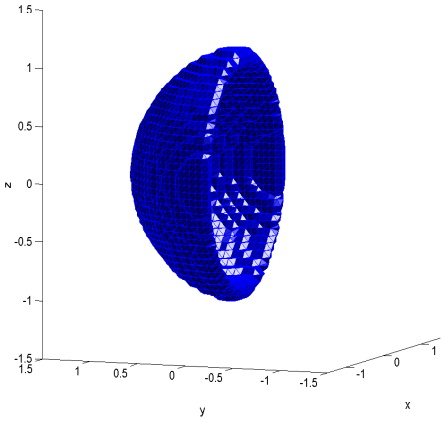

Figure 5.1: Isosurface plot for surface_3d_hemisphere.m. Surface area of a hemisphere.

the surface $z = x^2 + y^2$ that lies above the triangle in the $x - y$ plane whose vertices are the points $(1, 0)$, $(0, 1)$, and $(0, 0)$.

We start by observing that the edges of the triangle define three planes that are parallel to the z-axis. Level set functions for these planes are

$$u\_1(x, y, z) = x, \quad u\_2(x, y, z) = y, \quad u\_3(x, y, z) = 1 - x - y.$$

For the paraboloid, we use the level set function

$$u\_4(x, y, z) = z - (x^2 + y^3).$$

The intersection of the regions $u\_1 > 0$, $u\_2 > 0$, $u\_3 > 0$ is a triangular prism that is parallel to the z-axis. We are interested in the portion of the paraboloid $z = x^2 + y^2$ that lies within this prism region. So, with H_prism denoting a Heaviside function for the prism, we want an approximation to the integral

$$\int\int\int_B H\_prism(x, y, z) \cdot DHu\_4(x, y, z)\, dV.$$

Since the prism is formed by an intersection, we can compute H_prism using the formula

$$H\_prism = H(u\_1) \cdot H(u\_2) \cdot H(u\_3).$$

The program surface_3d_parab_tri.m (see the listing below) implements this. Using nx = ny = nz = 101, the program gives ≈ 0.74 for the integral.

The program surface_3d_parab_tri.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%surface_3d_parab_tri.m
clear; clf;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 51;  % number of grid points in x direction
x_lo =-0.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 51; % number of grid points in y direction
y_lo =-0.5; % smallest y value
y_hi = 1.5; % largest z value
nz = 51; % number of grid points in z direction
z_lo =-0.5; % smallest z value
z_hi = 1.5; % largest z value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
dz = (z_hi - z_lo)/(nz-1);
l=1:nz;
z= z_lo + (l-1)*dz;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
   for k = 1:ny
      for l = 1:nz
         u_1(j,k,l) = x(j);
         u_2(j,k,l) = y(k);
         u_3(j,k,l) = 1 - (x(j) + y(k));
         u_4(j,k,l) = z(l) - (x(j)^2 + y(k)^2);
         f(j,k,l) = 1;
      end %for l
   end %for k
end %for j

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the H function for u_1
u=u_1;
volume_make_H;
H_1 = H;
%make the H function for u_1
u=u_2;
volume_make_H;
H_2 = H;
```

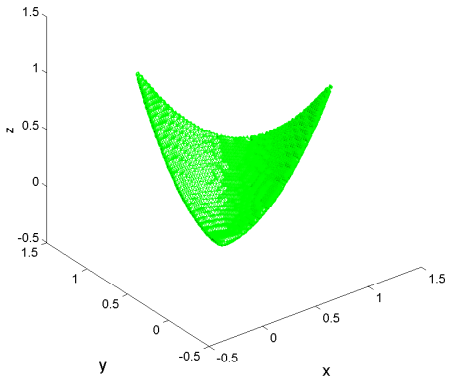

Figure 5.2: Example 5.1. Isosurface plot for surface_3d_parab_tri.m. Surface area of a triangular piece of a paraboloid.

```
%make the H function for u_3
u=u_3;
volume_make_H;
H_3 = H;
%make the DH function for u_4
u = u_4;
surface_3d_make_DH;
DH_4 = DH;

%make the H function for the triangle
H_prism = H_1.*H_2.*H_3;

%Compute the integral
Intgrl = dx*dy*dz*sum(sum(sum(f.*H_prism.*DH_4)))  %the answer, which will be output to the screen

%Plot the surface
p=patch(isosurface(x,y,z,(H_prism.*DH_4),1/2));
axis([x_lo x_hi y_lo y_hi z_lo z_hi]);
xlabel('x'); ylabel('y'); zlabel('z');
set(p, 'FaceColor', 'red', 'EdgeColor', 'red');
       daspect([1 1 1])
       view(3)
       camlight
       lighting phong
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 5.3 Find the surface area of the portion of the surface $z = (y^2 - x^2)/2$ that lies above the circle $x^2 + y^2 = 1$. This object looks like a potato

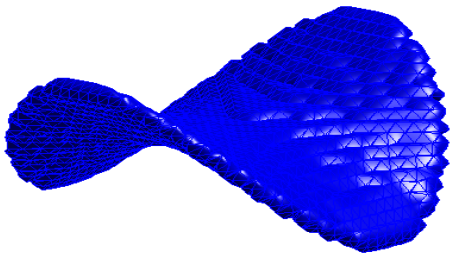

Figure 5.3: Exercise 5.3. Surface that looks like a potato chip.

chip. See Figure 5.3. Your answer should have one correct digit to the right of the decimal point.

Chapter 6

Line integrals over plane curves

In this chapter we focus on line integrals in the plane, which are of the form

$$\int_C f(x,y)\,dx + g(x,y)\,dy. \tag{6.1}$$

Here C is a curve that starts at some point (x_1, y_1) and ends at the point (x_2, y_2).

6.1 Line integrals over simple closed plane curves

In this section we will assume that the curve C is closed (the starting and ending points are the same). In addition, we will assume that C is the boundary of a simple region R in the plane, and that $u(x,y)$ is a level set function for R, so that ∂R consists of the level set $u(x,y) = 0$. We will assume that the curve is traversed in such a way that the region R would be on the left for a (two-dimensional) person traversing the curve. We call this a positive traversal. Of course, in some cases we may want to traverse the curve in the opposite direction. Then the region R would be on the right, and we call this a negative traversal. Notice that for arc length integrals over closed plane curves, the direction of traversal does not matter. For line integrals, it definitely does matter, so we have to be precise about this. In fact, we will use the notation

$$\begin{aligned} \partial R &\text{ means positive traversal of } \partial R \\ -\partial R &\text{ means negative traversal of } \partial R \end{aligned} \tag{6.2}$$

With this notation, the following formula is true:

$$\int_{-\partial R} f(x,y)\,dx + g(x,y)\,dy = -\int_{\partial R} f(x,y)\,dx + g(x,y)\,dy,$$

which just states the familiar fact that reversing the direction of traversal reverses the sign of the integral. This allows us to traverse the curve in either direction.

Recall that for arc length integrals over simple plane curves, we used the formula

$$\int_{\partial R} f(x(s),y(s))ds = \int\int_B DH(u(x,y))\,dA.$$

For line integrals over a curve ∂R, we will use a similar-looking formula:

$$\int_{\pm\partial R} f(x,y)\,dx + g(x,y)\,dy = \int\int_B Q(x,y)\,dH(u(x,y))\,dA.$$

Here the function Q is defined by

$$Q = (f\,u_y - g\,u_x)\,/||\nabla u||,$$

and

$$dH(u) = \begin{cases} DH(u), & \text{if } +\partial R \\ -DH(u), & \text{if } -\partial R \end{cases} \tag{6.3}$$

Example 6.1 Suppose that we want to compute the line integral

$$\int_{\partial R} y\,dx - x\,dy$$

where ∂R is the circle $x^2 + y^2 = 1$ traversed counterclockwise.

If we use the level set function $u(x,y) = 1 - \sqrt{x^2+y^2}$, then counterclockwise traversal means that we will always have the region R where $u > 0$ on our left, so $dH(u) = DH(u)$. Take a look at the listing of the program line_2d_closed.m. You will see that it is very similar to the program arc_2d_closed.m that we encountered in Section 4.1. The main differences are that in the case of line_2d_closed.m, one has to specify two functions, f and g, and multiply Q and DH together. Also, notice that the script file line_2d_make_DH.m is invoked to compute Q and DH. You should verify that you see where in the program these things are being performed.

The program line_2d_closed.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%line_2d_closed.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 101;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 101; % number of grid points in y direction
y_lo =-1.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       f(j,k) = y(k); % integrand
       g(j,k) =-x(j); % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH and Q functionw for u, f, g
line_2d_make_DH;

%Compute the integral
Intgrl = dx*dy*sum(sum(Q.*DH))  %this is the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u',[0 0]); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,(Q.*DH)');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi -5/dx 5/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A useful fact is that

$$dH(-u) = -dH(u).$$

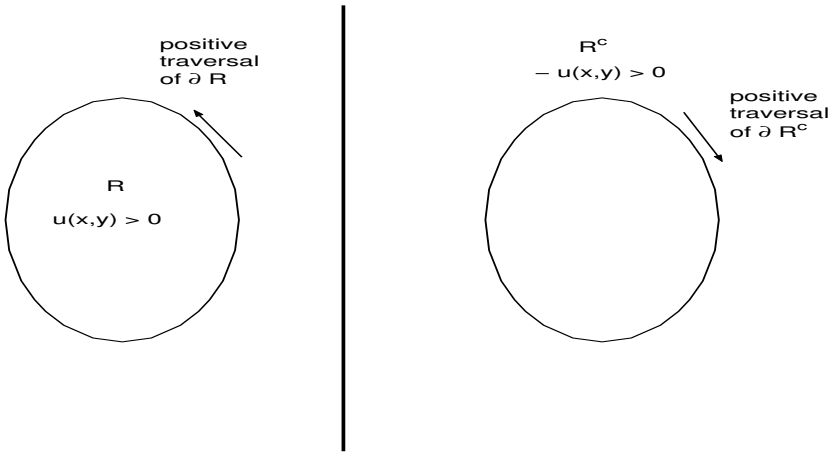

Figure 6.1: Negating the level set function reverses the meaning of positive traversal.

What this means is that negating the level set function for R is equivalent to traversing the curve ∂R in the opposite direction.

Exercise 6.1 In the program line_2d_closed.m, use $u(x,y) = -(1-\sqrt{x^2+y^2})$. Notice that this change negates the value of the integral. See Figure 6.1.

Exercise 6.2 Compute an approximation to the line integral

$$\int_{\partial R} \frac{-y}{x^2+y^2+\epsilon}\,dx + \frac{x}{x^2+y^2+\epsilon}\,dy.$$

Here ϵ is a small positive number that will avoid division by zero in the computer program. Use $\epsilon = 10^{-17}$. The curve ∂R is the ellipse

$$\frac{x^2}{9} + \frac{y^2}{16} = 1,$$

traversed counterclockwise. Your approximation should have four correct digits to the right of the decimal point.

6.1.1 Technical note

If we parameterize the curve ∂R by arclength, the line integral becomes

$$\begin{aligned}\int_{\partial R} f(x,y)\,dx + g(x,y)\,dy &= \int_{\partial R} f(x(s),y(s))\frac{dx}{ds}\,ds + g(x(s),y(s))\frac{dy}{ds}\,ds \\ &= \int_{\partial R} (f(x(s),y(s)), g(x(s),y(s))) \cdot \left(\frac{dx}{ds}, \frac{dy}{ds}\right)\,ds\end{aligned} \tag{6.4}$$

Recall that the vector $(dx/ds, dy/ds)$ is the unit tangent vector to the curve. The vector $(u_y, -u_x)/||\nabla u||$ is also a unit tangent to ∂R, so we can write the line integral in the form

$$\begin{aligned}\int_{\partial R} f(x,y)\,dx + g(x,y)\,dy &= \int_{\partial R} \frac{f\,u_y - g\,u_x}{||\nabla u||}\,ds \\ &= \int\int_B Q(x,y)\delta(u(x,y))||\nabla u||\,dA \\ &= \int\int_B Q(x,y)DH(u(x,y))\,dA.\end{aligned} \tag{6.5}$$

This calculation shows that line integrals can be viewed as arclength integrals of the type that we dealt with in Chapter 4. The main thing that is different for line integrals is that we must pay attention to the direction of traversal. For arclength integrals, the answer is the same with either direction of traversal.

6.2 Line integrals over closed plane curves that are not simple

Recall that in Section 4.2 we had list of rules for computing DH_R when R was formed by various set operations of simple regions. Here is the corresponding list for dH_R:

Intersection:

$$\begin{aligned}&R = R\_1 \cap R\_2 \\ &H\_R = H(u\_1) \cdot H(u\_2) \\ &Q\,dH\_R = Q\_2\,H(u\_1)dH(u\_2) + Q\_1\,H(u\_2)dH(u\_1)\end{aligned} \tag{6.6}$$

Union:

$$\begin{aligned} &R = R\_1 \cup R\_2 \\ &H\_R = H(u\_1) + H(u\_2) - H(u\_1) \cdot H(u\_2) \\ &Q\, dH\_R = Q\_1\, dH(u\_1) + Q\_2\, dH(u\_2) - Q\_2\, H(u\_1) dH(u\_2) \\ &\qquad\qquad - Q\_1\, H(u\_2) dH(u\_1) \end{aligned} \tag{6.7}$$

Difference:

$$\begin{aligned} &R = R\_1 - R\_2 \\ &H\_R = H(u\_1) \cdot (1 - H(u\_2)) \\ &Q\, dH\_R = Q\_1\, dH(u\_1) - Q\_2\, H(u\_1) dH(u\_2) - Q\_1\, H(u\_2) dH(u\_1) \end{aligned} \tag{6.8}$$

In these formulas

$$\begin{aligned} Q\_1 &= \left(f \frac{\partial}{\partial y} u\_1 - g \frac{\partial}{\partial x} u\_1 \right) / ||\nabla u\_1||, \\ Q\_2 &= \left(f \frac{\partial}{\partial y} u\_2 - g \frac{\partial}{\partial x} u\_2 \right) / ||\nabla u\_2||. \end{aligned} \tag{6.9}$$

Example 6.2 Approximate the integral

$$\int_C y^2\, dx + x\, dy,$$

where C connects the point $(0, 0)$ to $(1, 1)$ by a portion of the curve $x = y^2$, and then connects the point $(1, 1)$ back to $(0, 0)$ by a portion of the curve $y = x^2$. See Figure 6.2.

For level set functions, we use

$$u\_1(x, y) = x - y^2, \qquad u\_2(x, y) = y - x^2.$$

Clearly, $f(x, y) = y^2$, $g(x, y) = x$, and the curve C is the boundary of the region R defined by the inequalities

$$u\_1(x, y) > 0, \qquad u\_2(x, y) > 0,$$

except with a negative traversal. So, the integral that we want to approximate is

$$\int \int_B -\Big(Q\_1 H(u\_2) dH(u\_1) + Q\_2 H(u\_1) dH(u\_2) \Big)\, dA.$$

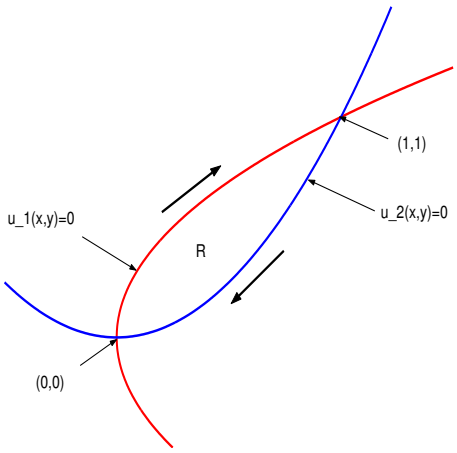

Figure 6.2: The curve for Example 6.2

Notice the minus sign, which accounts for the negative traversal of ∂R. This integral is approximated in the program line_2d_2parabolas.m. A listing appears below. The exact value of the integral is $-1/30$. Using fdm=2 and nx = ny = 101, the program gives a value of ≈ -0.03332.

The program line_2d_2parabolas.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%line_2d_2parabolas.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 201;  % number of grid points in x direction
x_lo =-0.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 201; % number of grid points in y direction
y_lo =-0.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
```

```
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = x(j) - y(k)^2;     % level set function
       u_2(j,k) = y(k) - x(j)^2;     % level set function
       f(j,k) = y(k)^2; % integrand
       g(j,k) = x(j);   % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%compute H, Q, and dH function for u_1
u = u_1;
area_make_H;
H_1 = H;
line_2d_make_DH;
dH_1 = DH;
Q_1 = Q;
%compute H, Q, and dH function for u_2
u = u_2;
area_make_H;
H_2 = H;
line_2d_make_DH;
dH_2 = DH;
Q_2 = Q;

summand = -(Q_2.*H_1.*dH_2 + Q_1.*H_2.*dH_1); %note minus sign -
                                               %negative traversal
%Compute the integral
Intgrl = dx*dy*sum(sum(summand)) %the answer - output to screen

mesh(x,y,(summand)');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi -3/dx 3/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Example 6.3 Approximate the line integral

$$\int_C (x + 2y)\,dx + (x^2 - y^2)\,dy,$$

where C is the triangle with vertices $(0, 0)$, $(1, 0)$, $(1, 1)$. Assume that the traversal of the curve is such that the vertices are visited in the order that they are listed above. See Figure 6.3.

We start by observing that if R is the interior of the triangle, then ∂R, traversed in the positive sense, represents the curve C. Using the three level set functions

$$\begin{aligned} u\_1(x, y) &= y, \\ u\_2(x, y) &= 1 - x, \\ u\_3(x, y) &= x - y, \end{aligned} \tag{6.10}$$

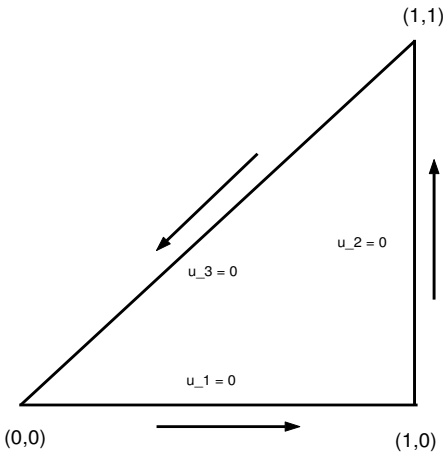

Figure 6.3: The curve for Example 6.3

a Heaviside function for the region R is

$$H\_R = H(u\_1)H(u\_2)H(u\_3).$$

To find $Q\,dH\_R$, we use the rule for intersections twice. The result is

$$\begin{aligned} Q\,dH\_R = {} & Q\_3\,H(u\_1)H(u\_2)dH(u\_3) \\ & + Q\_2\,H(u\_1)H(u\_3)dH(u\_2) \\ & + Q\_1\,H(u\_3)H(u\_2)dH(u\_1). \end{aligned} \tag{6.11}$$

Take a look at the listing of the program line_2d_triangle.m. You will see where the various quantities in this equation are computed and assembled. The exact value of this integral is $-1/3$. Using fdm=2, and $nx = ny = 101$, the program gives 14 correct digits to the right of the decimal point.

The program line_2d_triangle.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%line_2d_triangle.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 201;  % number of grid points in x direction
x_lo =-0.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 201; % number of grid points in y direction
```

```
y_lo =-0.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = y(k);         % level set function
       u_2(j,k) = 1 - x(j);     % level set function
       u_3(j,k) = x(j) - y(k);  % level set function
       f(j,k) = x(j) + 2*y(k);   % integrand
       g(j,k) = x(j)^2 - y(k)^2; % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%compute H, Q, and dH function for u_1
u = u_1;
area_make_H;
H_1 = H;
line_2d_make_DH;
dH_1 = DH;
Q_1 = Q;
%compute H, Q, and dH function for u_2
u = u_2;
area_make_H;
H_2 = H;
line_2d_make_DH;
dH_2 = DH;
Q_2 = Q;
%compute H, Q, and dH function for u_3
u = u_3;
area_make_H;
H_3 = H;
line_2d_make_DH;
dH_3 = DH;
Q_3 = Q;

summand = Q_3.*H_1.*H_2.*dH_3 + Q_2.*H_1.*H_3.*dH_2 ...
                              + Q_1.*H_2.*H_3.*dH_1;

%Compute the integral
Intgrl = dx*dy*sum(sum(summand)) %the answer - output to screen

mesh(x,y,(summand)');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi -3/dx 3/dx]);
```

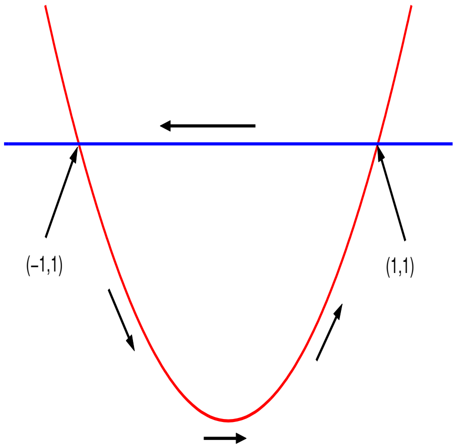

Figure 6.4: The curve for Exercise 6.3

```
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Exercise 6.3 Approximate the integral

$$\int_C (x^3 + y)\,dx + (y^2 - x)\,dy,$$

where C is the curve that connects $(-1, 1)$ to $(1, 1)$ by the curve $y = x^2$, and then $(1, 1)$ back to $(-1, 1)$ by a straight line segment. Your answer should have two correct digits to the right of the decimal point.

6.3 Line integrals over open plane curves

A line integral over an open curve in the plane can be handled in a manner that is similar to the way that we dealt with closed curves. The idea is to start with a closed curve, and proceed as we did in Section 6.1, but then multiply the integrand by an appropriate Heaviside function that cuts off the part of the curve that we don't want. We will see that in fact the starting curve doesn't even have to be closed.

Example 6.4 Approximate the line integral

$$\int_C -y\,dx + x\,dy$$

where C is the portion of the circle $x^2+y^2 = 1$ in the upper half-plane ($y > 0$) that starts at $(1, 0)$ and ends at $(-1, 0)$.

Let $u\_1(x, y) = 1 - \sqrt{x^2 + y^2}$. If we were traversing the entire circle, we would approximate the integral

$$\int\int_B Q(x, y)dH(u\_1(x, y))\,dA$$

as we did in Section 6.1. This would give the line integral over the entire circle. But we only want the portion of the curve that lies in the upper half plane. This suggests that we simply multiply the integrand by a Heaviside function for the region $y > 0$. Letting $u\_2(x, y) = y$, the integral that we want to approximate is

$$\int\int_B H(u\_2(x, y))Q(x, y)dH(u\_1(x, y))\,dA.$$

Below is a listing of the program line_2d_open.m, which approximates this integral. The exact value of the desired line integral is π.

The program line_2d_open.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%line_2d_open.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 101;  % number of grid points in x direction
x_lo =-1.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 101; % number of grid points in y direction
y_lo =-1.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

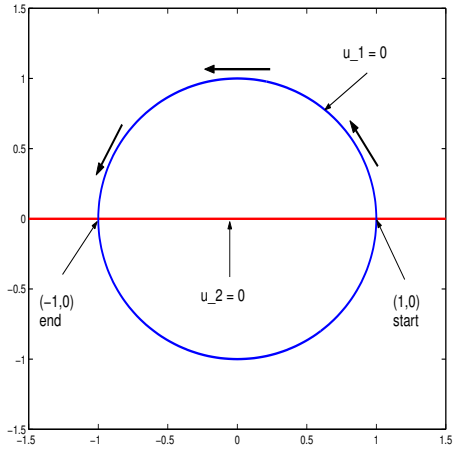

Figure 6.5: The setup for Example 6.4.

```
fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
    for k = 1:ny
       u_1(j,k) = 1 - sqrt(x(j)^2 + y(k)^2);  % level set function
       u_2(j,k) = y(k);
       f(j,k) = -y(k); % integrand
       g(j,k) =  x(j); % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH and Q functions for u_1, f, g
u=u_1;
line_2d_make_DH;
%make the H function for u_2
u=u_2;
area_make_H;

%Compute the integral
Intgrl = dx*dy*sum(sum(H.*Q.*DH))  %the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u_1',[0 0]);hold on;
contour(x,y,u_2',[0 0]); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,(H.*Q.*DH)');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi -5/dx 5/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Example 6.5 Compute the line integral

$$\int_C x^2\,dx + y\,dy$$

where C is the line segment that starts at $(0,0)$ and ends at $(1,1)$.

The line segment of interest is a subset of the line $y = x$. If we use the level set function $u\_1(x,y) = y - x$, then u_1 will be positive on our left as we traverse the line segment from the starting point to the ending point.

Next, we must cut off the part of the curve not on the desired segment by multiplying by a convenient Heaviside function. For this we can use a circular region that has our line segment as a diameter. With

$$u\_2(x,y) = \sqrt{2}/2 - \sqrt{(x-1/2)^2 + (y-1/2)^2},$$

you can check that the region $u\_2(x,y) > 0$ is a circular region that exactly contains the segment of interest. With u_1 and u_2 defined in this way, we can proceed as we did in the previous example. Thus, we use

$$\int\int_B H(u\_2(x,y))Q(x,y)dH(u\_1(x,y))\,dA$$

to compute the line integral. The program line_2d_segment.m approximates this integral. The exact value of the line integral is 5/6.

The program line_2d_segment.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%line_2d_segment.m
clear;
%set up the grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nx = 101;  % number of grid points in x direction
x_lo =-0.5;  % smallest x value
x_hi = 1.5;  % largest y value
ny = 101; % number of grid points in y direction
y_lo =-0.5; % smallest y value
y_hi = 1.5; % largest y value

dx = (x_hi - x_lo)/(nx-1);
j=1:nx;
x= x_lo + (j-1)*dx;
dy = (y_hi - y_lo)/(ny-1);
k=1:ny;
y= y_lo + (k-1)*dy;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

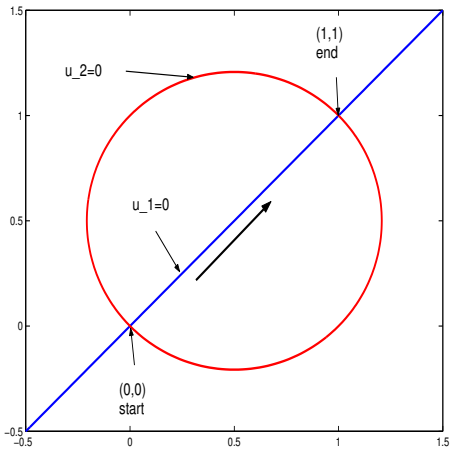

Figure 6.6: The setup for Example 6.5.

```
fdm=2; %to choose which algorithm - choices are 1 or 2

%initialize the level set function u,
%and the intgrand f
for j = 1:nx
   for k = 1:ny
       u_1(j,k) = y(k) - x(j);  % level set function
       u_2(j,k) = sqrt(2)/2 - sqrt((x(j)-1/2)^2 + (y(k)-1/2)^2);
       f(j,k) = x(j)^2; % integrand
       g(j,k) = y(k); % integrand
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%make the DH and Q functionw for u_1, f, g
u=u_1;
line_2d_make_DH;
%make the H function for u_2
u=u_2;
area_make_H;

%Compute the integral
Intgrl = dx*dy*sum(sum(H.*Q.*DH))  %this is the answer, which will be output to the screen

subplot(1,2,1);contour(x,y,u',[0 0]); xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi]);
subplot(1,2,2);mesh(x,y,(H.*Q.*DH)');
xlabel('x'); ylabel('y');
axis([x_lo x_hi y_lo y_hi -5/dx 5/dx]);
colormap(winter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

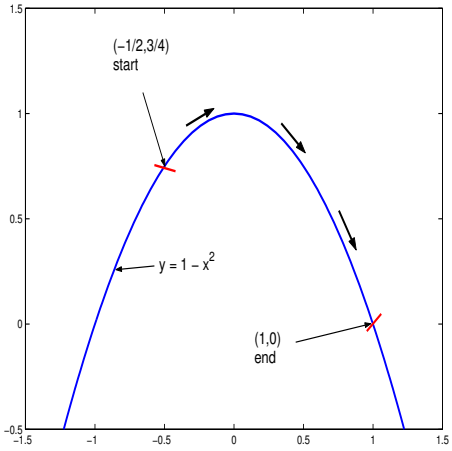

Figure 6.7: The setup for Exercise 6.4.

Exercise 6.4 Approximate the line integral

$$\int_C x\,dx + y\,dy$$

where C is the portion of the parabola $y = 1 - x^2$ that starts at $(-1/2, 3/4)$ and ends at $(1, 0)$. See Figure 6.7. Your answer should have three correct digits to the right of the decimal point.

Chapter 7

Program listings for the algorithms

The program area_make_H.m

```
%area_make_H.m
%%%%%%%%%%%%%

%John D. Towers
%This script file builds a single discretized Heaviside function.
%This is the two-dimensional version. It uses one of two methods.
%The algorithms are described in the paper "Finite difference methods for approximating
%Heaviside functions", to appear in Journal of Computational Physics.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j = 1:nx
    for k = 1:ny
       I(j,k) = max(u(j,k),0);
       J(j,k) = .5*max(u(j,k),0)^2;
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==1;
H=0*u;
for j = 2:nx-1
   for k = 2:ny-1
      if u(j,k)>0
         H(j,k)=1;
      end
      if u(j,k)<0
         H(j,k)=0;
      end
      dux_p = u(j+1,k)*u(j,k);
      dux_m = u(j,k)*u(j-1,k);
      duy_p = u(j,k+1)*u(j,k);
```

```
      duy_m = u(j,k)*u(j,k-1);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0
         u_x(j,k) = (u(j+1,k)-u(j-1,k))/(2*dx);
         u_y(j,k) = (u(j,k+1)-u(j,k-1))/(2*dy);
         I_x(j,k) = (I(j+1,k)-I(j-1,k))/(2*dx);
         I_y(j,k) = (I(j,k+1)-I(j,k-1))/(2*dy);
         norm_u(j,k) = sqrt(u_x(j,k)^2+u_y(j,k)^2);
         H(j,k) = (I_x(j,k)*u_x(j,k) + I_y(j,k)*u_y(j,k))/norm_u(j,k)^2;
      end
  end %for k
end %for j

end %if fdm=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==2
I=0*u;
flag=0*u;
for j = 7:nx-6
   for k = 7:ny-6
      djk_max = 4;
      for djk=1:djk_max
        dux_p = u(j+djk,k)*u(j,k);
        dux_m = u(j,k)*u(j-djk,k);
        duy_p = u(j,k+djk)*u(j,k);
        duy_m = u(j,k)*u(j,k-djk);
        if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0
           flag(j,k) = 1;
           break;
        end
      end
      if flag(j,k) == 1
         u_x(j,k) = (4/3)*(u(j+1,k)-u(j-1,k))/(2*dx) - (1/3)*(u(j+2,k)-u(j-2,k))/(4*dx);
         u_y(j,k) = (4/3)*(u(j,k+1)-u(j,k-1))/(2*dy) - (1/3)*(u(j,k+2)-u(j,k-2))/(4*dy);
         J_x(j,k) = (4/3)*(J(j+1,k)-J(j-1,k))/(2*dx) - (1/3)*(J(j+2,k)-J(j-2,k))/(4*dx);
         J_y(j,k) = (4/3)*(J(j,k+1)-J(j,k-1))/(2*dy) - (1/3)*(J(j,k+2)-J(j,k-2))/(4*dy);
         norm_u(j,k) = sqrt(u_x(j,k)^2+u_y(j,k)^2);
         I(j,k) = (J_x(j,k)*u_x(j,k) + J_y(j,k)*u_y(j,k))/norm_u(j,k)^2;
      else
         I(j,k) = max(0,u(j,k));
      end
  end %for k
end %for j

H=0*u;
for j = 7:nx-6
   for k = 7:ny-6
      if u(j,k)>0
         H(j,k)=1;
      end
      if u(j,k)<0
         H(j,k)=0;
      end
      if flag(j,k)==1
        I_x(j,k) = (4/3)*(I(j+1,k)-I(j-1,k))/(2*dx) - (1/3)*(I(j+2,k)-I(j-2,k))/(4*dx);
        I_y(j,k) = (4/3)*(I(j,k+1)-I(j,k-1))/(2*dy) - (1/3)*(I(j,k+2)-I(j,k-2))/(4*dy);
        H(j,k) = (I_x(j,k)*u_x(j,k) + I_y(j,k)*u_y(j,k))/norm_u(j,k)^2;
      end
```

```
   end %for ny
end %for nx


end %if fdm==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The program volume_make_H.m

```
%volume_make_H.m
%%%%%%%%%%%%%%%

%John D. Towers
%This script file builds a single discretized Heaviside function.
%This is the three-dimensional version. It uses one of two methods.
%The algorithms are described in the paper "Finite difference methods for approximating
%Heaviside functions", to appear in Journal of Computational Physics.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j = 1:nx
   for k = 1:ny
      for l = 1:nz
       I(j,k,l) = max(u(j,k,l),0);
       J(j,k,l) = .5*max(u(j,k,l),0)^2;
      end
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==1
H=0*u;
for j = 2:nx-1
   for k = 2:nx-1
     for l = 2:nz-1
      if u(j,k,l)>0
         H(j,k,l)=1;
      end
      if u(j,k,l)<0
         H(j,k,l)=0;
      end
      dux_p = u(j+1,k,l)*u(j,k,l);
      dux_m = u(j,k,l)*u(j-1,k,l);
      duy_p = u(j,k+1,l)*u(j,k,l);
      duy_m = u(j,k,l)*u(j,k-1,l);
      duz_p = u(j,k,l+1)*u(j,k,l);
      duz_m = u(j,k,l-1)*u(j,k,l);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0 | duz_p <=0 | duz_m<=0
         u_x(j,k,l) = (u(j+1,k,l)-u(j-1,k,l))/(2*dx);
         u_y(j,k,l) = (u(j,k+1,l)-u(j,k-1,l))/(2*dy);
         u_z(j,k,l) = (u(j,k,l+1)-u(j,k,l-1))/(2*dz);

         I_x(j,k,l) = (I(j+1,k,l)-I(j-1,k,l))/(2*dx);
         I_y(j,k,l) = (I(j,k+1,l)-I(j,k-1,l))/(2*dy);
         I_z(j,k,l) = (I(j,k,l+1)-I(j,k,l-1))/(2*dz);

         norm_u(j,k,l) = sqrt(u_x(j,k,l)^2+u_y(j,k,l)^2+u_z(j,k,l)^2);
         H(j,k,l) = (I_x(j,k,l)*u_x(j,k,l) + I_y(j,k,l)*u_y(j,k,l) ...
                                          +I_z(j,k,l)*u_z(j,k,l))/norm_u(j,k,l)^2;
      end
     end %for l
  end %for k
end %for j
end %if fdm==1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if fdm == 2
I=0*u;
flag =0*u;
for j = 7:nx-6
   for k = 7:nx-6
    for l = 7:nx-6
      djk_max = 4;
      flag(j,k,l) = 0;
      for djk=1:djk_max
        dux_p = u(j+djk,k,l)*u(j,k,l);
        dux_m = u(j,k,l)*u(j-djk,k,l);
        duy_p = u(j,k+djk,l)*u(j,k,l);
        duy_m = u(j,k,l)*u(j,k-djk,l);
        duz_p = u(j,k,l+djk)*u(j,k,l);
        duz_m = u(j,k,l)*u(j,k,l-djk);
        if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0  | duz_p <=0 | duz_m<=0
           flag(j,k,l) = 1;
           break;
        end
      end
      if flag(j,k,l) == 1
         u_x(j,k,l) = (4/3)*(u(j+1,k,l)-u(j-1,k,l))/(2*dx) - (1/3)*(u(j+2,k,l)-u(j-2,k,l))/(4*dx);
         u_y(j,k,l) = (4/3)*(u(j,k+1,l)-u(j,k-1,l))/(2*dy) - (1/3)*(u(j,k+2,l)-u(j,k-2,l))/(4*dy);
         u_z(j,k,l) = (4/3)*(u(j,k,l+1)-u(j,k,l-1))/(2*dz) - (1/3)*(u(j,k,l+2)-u(j,k,l-2))/(4*dz);
         J_x(j,k,l) = (4/3)*(J(j+1,k,l)-J(j-1,k,l))/(2*dx) - (1/3)*(J(j+2,k,l)-J(j-2,k,l))/(4*dx);
         J_y(j,k,l) = (4/3)*(J(j,k+1,l)-J(j,k-1,l))/(2*dy) - (1/3)*(J(j,k+2,l)-J(j,k-2,l))/(4*dy);
         J_z(j,k,l) = (4/3)*(J(j,k,l+1)-J(j,k,l-1))/(2*dz) - (1/3)*(J(j,k,l+2)-J(j,k,l-2))/(4*dz);
         norm_u(j,k,l) = sqrt(u_x(j,k,l)^2 + u_y(j,k,l)^2 + u_z(j,k,l)^2);
         I(j,k,l) = (J_x(j,k,l)*u_x(j,k,l) + J_y(j,k,l)*u_y(j,k,l) ...
                                          + J_z(j,k,l)*u_z(j,k,l))/norm_u(j,k,l)^2;
      else
         I(j,k,l) = max(0,u(j,k,l));
      end
    end %for l
  end %for k
end %for j

H=0*u;
for j = 7:nx-6
   for k = 7:ny-6
    for l = 7:nz-6
      if u(j,k,l)>0
         H(j,k,l)=1;
      end
      if u(j,k,l)<0
         H(j,k,l)=0;
      end
      if flag(j,k,l) == 1
        I_x(j,k,l) = (4/3)*(I(j+1,k,l)-I(j-1,k,l))/(2*dx) - (1/3)*(I(j+2,k,l)-I(j-2,k,l))/(4*dx);
        I_y(j,k,l) = (4/3)*(I(j,k+1,l)-I(j,k-1,l))/(2*dy) - (1/3)*(I(j,k+2,l)-I(j,k-2,l))/(4*dy);
        I_z(j,k,l) = (4/3)*(I(j,k,l+1)-I(j,k,l-1))/(2*dz) - (1/3)*(I(j,k,l+2)-I(j,k,l-2))/(4*dz);
        H(j,k,l) = (I_x(j,k,l)*u_x(j,k,l) + I_y(j,k,l)*u_y(j,k,l) ...
                                         + I_z(j,k,l)*u_z(j,k,l))/norm_u(j,k,l)^2;
      end
    end %for l
   end %for k
```

```
end %for j
end %if fdm==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The program arc_2d_make_DH.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%arc_2d_make_DH.m
%%%%%%%%%%%%%%

%John D. Towers
%This script file builds a single discretized delta function, which is multiplied
%by the norm of the gradient. This product is called DH. The quantity DH instead of
%delta is output because this is what is needed for quadrature.
%This is the two-dimensional version. It uses one of two methods.
%The algorithms are described in the paper "Two methods for discretizing
%a delta function supported on a level set", Journal of Computational Physics
%and "A convergence rate theorem for finite difference approximations to
%delta functions", Journal of Computational Physics.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DH = 0*u;
for j = 1:nx
   for k = 1:ny
      if u(j,k) >= 0
         H(j,k) = 1;
      else
         H(j,k) = 0;
      end
      I(j,k) = max(u(j,k),0);
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==1;
for j = 2:nx-1
   for k = 2:ny-1
      dux_p = u(j+1,k)*u(j,k);
      dux_m = u(j,k)*u(j-1,k);
      duy_p = u(j,k+1)*u(j,k);
      duy_m = u(j,k)*u(j,k-1);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0
         u_x = (u(j+1,k)-u(j-1,k))/(2*dx);
         u_y = (u(j,k+1)-u(j,k-1))/(2*dy);
         H_x = (H(j+1,k)-H(j-1,k))/(2*dx);
         H_y = (H(j,k+1)-H(j,k-1))/(2*dy);
         norm_u = sqrt(u_x^2+u_y^2);
         DH(j,k) = (H_x*u_x + H_y*u_y)/norm_u;
      end
  end %for k
end %for j
end %if fdm=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==2
for j = 2:nx-1
   for k = 2:ny-1
      dux_p = u(j+1,k)*u(j,k);
      dux_m = u(j,k)*u(j-1,k);
      duy_p = u(j,k+1)*u(j,k);
      duy_m = u(j,k)*u(j,k-1);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0
         u_x = (u(j+1,k)-u(j-1,k))/(2*dx);
         u_y = (u(j,k+1)-u(j,k-1))/(2*dy);
```

```
        I_x = (I(j+1,k)-I(j-1,k))/(2*dx);
        I_y = (I(j,k+1)-I(j,k-1))/(2*dy);
        dot = u_x*I_x + u_y*I_y;
        norm_u = sqrt(u_x^2+u_y^2);
        del_u = (u(j+1,k)-2*u(j,k)+u(j-1,k))/dx^2 ...
              + (u(j,k+1)-2*u(j,k)+u(j,k-1))/dy^2;
        del_I = (I(j+1,k)-2*I(j,k)+I(j-1,k))/dx^2 ...
              + (I(j,k+1)-2*I(j,k)+I(j,k-1))/dy^2;
        DH(j,k) = del_I/norm_u - dot*del_u/norm_u^3;
     end
  end %for k
end %for j
end %if fdm==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The program surface_3d_make_DH.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%surface_3d_make_DH.m
%%%%%%%%%%%%%%%%%%%%%

%John D. Towers
%This script file builds a single discretized delta function, which is multiplied
%by the norm of the gradient. This product is called DH. The quantity DH instead of
%delta is output because this is what is needed for quadrature.
%This is the three-dimensional version. It uses one of two methods.
%The algorithms are described in the paper "Two methods for discretizing
%a delta function supported on a level set", Journal of Computational Physics
%and "A convergence rate theorem for finite difference approximations to
%delta functions", Journal of Computational Physics.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DH = 0*u;
for j = 1:nx
   for k = 1:ny
      for l = 1:nz
         if u(j,k,l) >= 0
            H(j,k,l) = 1;
         else
            H(j,k,l) = 0;
         end
         I(j,k,l) = max(u(j,k,l),0);
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==1
delta = 0*u;
for j = 2:nx-1
   for k = 2:nx-1
     for l = 2:nz-1
      dux_p = u(j+1,k,l)*u(j,k,l);
      dux_m = u(j,k,l)*u(j-1,k,l);
      duy_p = u(j,k+1,l)*u(j,k,l);
      duy_m = u(j,k,l)*u(j,k-1,l);
      duz_p = u(j,k,l+1)*u(j,k,l);
      duz_m = u(j,k,l)*u(j,k,l-1);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0 | duz_p <=0 | duz_m<=0
         u_x = (u(j+1,k,l)-u(j-1,k,l))/(2*dx);
         u_y = (u(j,k+1,l)-u(j,k-1,l))/(2*dy);
         u_z = (u(j,k,l+1)-u(j,k,l-1))/(2*dz);
         H_x = (H(j+1,k,l)-H(j-1,k,l))/(2*dx);
         H_y = (H(j,k+1,l)-H(j,k-1,l))/(2*dy);
         H_z = (H(j,k,l+1)-H(j,k,l-1))/(2*dz);
         norm_u = sqrt(u_x^2+u_y^2+u_z^2);
         DH(j,k,l) = (H_x*u_x + H_y*u_y + H_z*u_z)/norm_u;
      end
     end %for l
  end %for k
end %for j
end %if fdm==1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm == 2
```

```
for j = 2:nx-1
   for k = 2:ny-1
      for l = 2:nz-1
         dux_p = u(j+1,k,l)*u(j,k,l);
         dux_m = u(j,k,l)*u(j-1,k,l);
         duy_p = u(j,k+1,l)*u(j,k,l);
         duy_m = u(j,k,l)*u(j,k-1,l);
         duz_p = u(j,k,l+1)*u(j,k,l);
         duz_m = u(j,k,l)*u(j,k-1,l-1);
         if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0 | duz_p <=0 | duz_m<=0
            u_x = (u(j+1,k,l)-u(j-1,k,l))/(2*dx);
            u_y = (u(j,k+1,l)-u(j,k-1,l))/(2*dy);
            u_z = (u(j,k,l+1)-u(j,k,l-1))/(2*dz);
            I_x = (I(j+1,k,l)-I(j-1,k,l))/(2*dx);
            I_y = (I(j,k+1,l)-I(j,k-1,l))/(2*dy);
            I_z = (I(j,k,l+1)-I(j,k,l-1))/(2*dz);
    dot = u_x*I_x + u_y*I_y + u_z*I_z;
            norm_u = sqrt(u_x^2+u_y^2+u_z^2);
            del_u = (u(j+1,k,l)-2*u(j,k,l)+u(j-1,k,l))/dx^2 ...
                  + (u(j,k+1,l)-2*u(j,k,l)+u(j,k-1,l))/dy^2 ...
                  + (u(j,k,l+1)-2*u(j,k,l)+u(j,k,l-1))/dz^2;
            del_I = (I(j+1,k,l)-2*I(j,k,l)+I(j-1,k,l))/dx^2 ...
                  + (I(j,k+1,l)-2*I(j,k,l)+I(j,k-1,l))/dy^2 ...
                  + (I(j,k,l+1)-2*I(j,k,l)+I(j,k,l-1))/dz^2;
            DH(j,k,l) = del_I/norm_u - dot*del_u/norm_u^3;
         end
      end %for l
  end %for k
end %for j
end %if fdm==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The program line_2d_make_DH.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%line_2d_make_DH.m
%%%%%%%%%%%%%%

%John D. Towers
%This script file computes quantities DH and Q that are used to compute
%line integrals for curves in the plane.
%The code for DH is based on algorithms in the paper "Two methods for discretizing
%a delta function supported on a level set", Journal of Computational Physics
%and "A convergence rate theorem for finite difference approximations to
%delta functions", Journal of Computational Physics.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DH = 0*u;
Q = 0*u;
for j = 1:nx
   for k = 1:ny
      if u(j,k) >= 0
         H(j,k) = 1;
      else
         H(j,k) = 0;
      end
      I(j,k) = max(u(j,k),0);
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==1;
for j = 2:nx-1
   for k = 2:ny-1
      dux_p = u(j+1,k)*u(j,k);
      dux_m = u(j,k)*u(j-1,k);
      duy_p = u(j,k+1)*u(j,k);
      duy_m = u(j,k)*u(j,k-1);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0
         u_x = (u(j+1,k)-u(j-1,k))/(2*dx);
         u_y = (u(j,k+1)-u(j,k-1))/(2*dy);
         H_x = (H(j+1,k)-H(j-1,k))/(2*dx);
         H_y = (H(j,k+1)-H(j,k-1))/(2*dy);
         norm_u = sqrt(u_x^2+u_y^2);
         norm_save(j,k) = norm_u;
         DH(j,k) = (H_x*u_x + H_y*u_y)/norm_u;
         Q(j,k) = (f(j,k)*u_y - g(j,k)*u_x)/norm_u;
      end
  end %for k
end %for j
end %if fdm=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if fdm==2
for j = 2:nx-1
   for k = 2:ny-1
      dux_p = u(j+1,k)*u(j,k);
      dux_m = u(j,k)*u(j-1,k);
      duy_p = u(j,k+1)*u(j,k);
      duy_m = u(j,k)*u(j,k-1);
      if dux_p <= 0 | dux_m <=0 | duy_p <=0 | duy_m<=0
```

```
          u_x = (u(j+1,k)-u(j-1,k))/(2*dx);
          u_y = (u(j,k+1)-u(j,k-1))/(2*dy);
          I_x = (I(j+1,k)-I(j-1,k))/(2*dx);
          I_y = (I(j,k+1)-I(j,k-1))/(2*dy);
          dot = u_x*I_x + u_y*I_y;
          norm_u = sqrt(u_x^2+u_y^2);
          del_u = (u(j+1,k)-2*u(j,k)+u(j-1,k))/dx^2 ...
                + (u(j,k+1)-2*u(j,k)+u(j,k-1))/dy^2;
          del_I = (I(j+1,k)-2*I(j,k)+I(j-1,k))/dx^2 ...
                + (I(j,k+1)-2*I(j,k)+I(j,k-1))/dy^2;
          DH(j,k) = (del_I/norm_u - dot*del_u/norm_u^3);
          Q(j,k) = (f(j,k)*u_y - g(j,k)*u_x)/norm_u;
       end
  end %for k
end %for j
end %if fdm==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

[1] G. Dahlquist, A. Björk, Numerical Methods, Prentice-Hall, Inglewood Cliffs, New Jersey, 1974.

[2] C. Min, F. Gibou, Geometric integration over irregular domains with application to level-set methods, J. Comput. Phys. **226** (2007) 1432–1443.

[3] S. Osher, R. Fedkiw, Level set methods and dynamic implicit surfaces, Springer-Verlag, New York, 2003.

[4] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys. **79** (1988) 12-49.

[5] J.A. Sethian, Level set methods and fast marching methods, Cambridge University Press, Cambridge, 1999.

[6] J.D. Towers. Two methods for discretizing a delta function supported on a level set, J. Comput. Phys. **220** (2007) 915-931.

[7] J.D. Towers. A convergence rate theorem for finite difference approximations to delta functions, J. Comput. Phys. 227 (2008) 6591–6597.

[8] J.D. Towers. Finite difference methods for approximating Heaviside functions, J. Comput. Phys. 228 (2009) 3478–3489.