CS2035 - Assignment 3 - 2018

Animating Warping Surfaces

Out: March 6th, 2018

In: March 18th, 2018 at 11:55 pm via Owl

Introduction

This MATLAB assignment requires you to write one MATLAB script file, a3.m, to compute a linear interpolation of one surface into another and back again, together with advanced annotation of the plots..

You will learn about:

- linear interpolation of surfaces and quantities;
- definite numerical integration of a surface;
- animating variation of surfaces;
- finer control of labeling and coloring of axis elements;
- symbolic integration of functions.

This assignment is worth $11\frac{2}{3}\%$ of the course mark.

Introduction

A surface plot of the equation

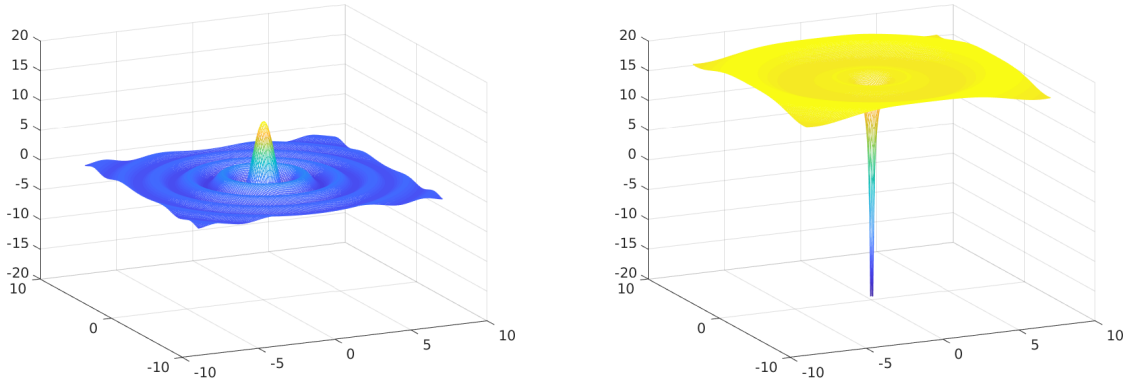
$$f_1(x, y) = 10\,\mathrm{sinc}(\sqrt{x^2 + y^2}), \tag{1}$$

where $\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ is the normalized sinc function, looks a bit like the surface of a pool of water after a stone has been dropped in it, while the following equation

$$f_2(x, y) = 18 - \frac{3}{\sqrt{x^2 + y^2}} + \sin(\sqrt{x^2 + y^2}) + \frac{\sqrt{200 - (x^2 + y^2) + 10\sin(x) + 10\sin(y)}}{1000} \tag{2}$$

produces a surface that looks like a flower.

This assignment requires you to write a MATLAB program that performs an animation of these 2 function surface functions warping the first surface into the second surface, then the second surface into the first surface. Figure 1 shows the mesh plots of these 2 surfaces.



(a) $f_1(x, y)$ (b) $f_2(x, y)$

Figure 1: Surface plots for (a) $f_1(x, y)$ from equation (1) and (b) $f_2(x, y)$ from equation (2), both with numerically and symbolically evaluated integral values printed on them.

Plotting the Functions

The various tasks in this assignment include:

1. First, you have to plot the 2 functions. Use x_{min} and y_{min} values of -8 and x_{max} and y_{max} values of +8, and construct arrays for the x and y values using `linspace` vectors `xs` and `ys` with 201 points. From these, generate `X` and `Y` from `meshgrid(xs,ys)`. You then compute the z-values for functions $f_1(x, y)$ and $f_2(x, y)$ in the arrays `Z1` and `Z2` using a vectorized calculation based on the expressions in equations (1) and (2) using `X` and `Y`. When you plot the surfaces, fix the displyed z-values to be within the range $z_{min} = -20$ and $z_{max} = +20$ using `axis`. Do this for all the figures you display in this assignment. Plot the 2 functions, given by `(X,Y,Z1)` and `(X,Y,Z2)` using `mesh`.

2. Next, perform a numerical integration on these 2 functions and use `text` to print out these values in magenta. Choose an appropriate fontname, fontsize and appropriate 3D coordinates for `text` to positioning the text while printing the numerical integration results on the graphs.

3. To numerically integrate the 2 functions you need to write an anonymous function defined for two surfaces as:

```
fun1 = @(X,Y) (your vectorized expression for f1(x,y));
num_area1=integral2(fun1,xmin,xmax,ymin,ymax);
fun2 = @(X,Y) (your vectorized expression for f2(x,y));
num_area2=integral2(fun2,xmin,xmax,ymin,ymax);
```

   Here `fun1` and `fun2` are the "handles" (or pointers) to **anonymous** functions (functions that have no name). You can pass this handle to a function as a parameter to another function. Effectively, you can have a function as a parameter to another function. In this assignment, you can integrate functions, `fun1` and `fun2`, using `integral2`. The MATLAB function `integral2` evaluates the area under this function using numerical quadrature.

4. You can use the `title` command to print out the mathematical formulas for each of the 2 meshes. [An expression is interpreted as latex in MATLAB, with `_` indicating a subscript and a `^` indicating a superscript. Thus `x_2` is x_2 while `x^2` is x^2.] Also use `xlabel`, `ylabel` and `zlabel` to label the x, y and z axes in your figures. Color these using blue, green and red. Also set the font-size to a larger value. Use the 3D version of `text` to print out the numerical integration results on these graphs (see below). You

have to choose the x, y and z values to position these integration values appropriately on the plots.

5. You create the animation by warping `Z1` into `Z2`, pausing briefly, and then `Z2` back into `Z1`. So the initial and final surfaces are the same. To code the warping for `Z1` to `Z2` you can use something like:

```
for t=0:delta_t:1
    Z=Z1*(1-t)+Z2*(t);
    mesh(X,Y,Z)
    axis([xmin xmax ymin ymax zmin zmax]);
    text(xpos1,ypos1,zpos1,['\fontsize{14}\bf \color{magenta} ' ...
        'Integral Value: ' ...
        sprintf('%8.3f',num_area1*(1-t)+num_area2*t)]);
    xlabel('\it\bf\color{blue} x');
    ylabel('\it\bf\color{green} y');
    zlabel('\it\bf\color{red} z');
    shading interp
    pause(0.001);
    drawnow;
end % for t
```

The statement `Z=Z1*(1-t)+Z2*(t)` does the surface warping, i.e., this is the linear interpolation of the two surfaces `Z1` and `Z2`. For $t = 0$, `Z=Z1` while for $t = 1$, `Z=Z2`. Intermediate values of t give you the various combined surfaces of `Z1` and `Z2`. δt is a small number, say 0.01. Thus, when this runs you will see a total of 101 surfaces displayed rapidly as `Z1` warps into `Z2`. If the display is too rapid, you can pause a small amount of time between adjacent displays to slow things down. Use the `text` command to print out the areas computed by numerical integration. Set variables `xpos1`, `ypos1` and `zpos1` appropriately. Trial and error is required here. You could use `grid on` and `box on` to get some good initial values. Note that the text is printed in magenta for the numerical integration areas, which are linearly interpolated to correspond with the current surface being displayed. After warping `Z1` into `Z2`, you will need to pause or insert additional values of t=1 to delay the image of `Z2`, and then you need to warp `Z2` into `Z1`. At this point you will

have your animation working. Figures 2 and 3 show the surfaces $f_1(x, y)$ and $f_2(x, y)$, with the axes labelled, a title in orange and black and the numerically and symbolically evaluates integral values in magenta. Note that you may want to use `shading interp` and to set the view to a nicer angle than the default (recall that you can use `rotate3d on` to manually change the view and find good values to set with `view`).

6. The last task for the animation is to make the animation figure bigger than normal. The handle for the screen is 0. `get(0,'screensize')` gets the lower x and y coordinates of the screen plus its width and height. Set the height of the figure to be 85% of the height of the screen. Given the height, compute the width that satisfies an aspect ratio of 16/11. Finally, compute the lower x and y coordinates of the figure to be small percentages of the screen width and height. These will offset your figure from the lower left corner of the screen. When you generate the animation figure you must save its handle and use this handle to set the figure's position properties using `set`. The following MATLAB code outlines how all this might be done:

```
set(0,'units','pixels');
screenSizePixels=get(0,'screensize');
screenWidth=screenSizePixels(3);
screenHeight=screenSizePixels(4);
figureAspectRatio=16/11; % height to width

figureHeight=screenHeight*0.85;
figureWidth=screenHeight*1.0/figureAspectRatio;

% shift left 5% of the screen width
leftx=screenWidth*0.05;
% shift up 15% of the screen height
lefty=screenHeight*0.15;

ha=figure;
set(ha,'Position',[leftx lefty figureWidth figureHeight]);
```

By setting all position and height/width figure information in terms of the computer's

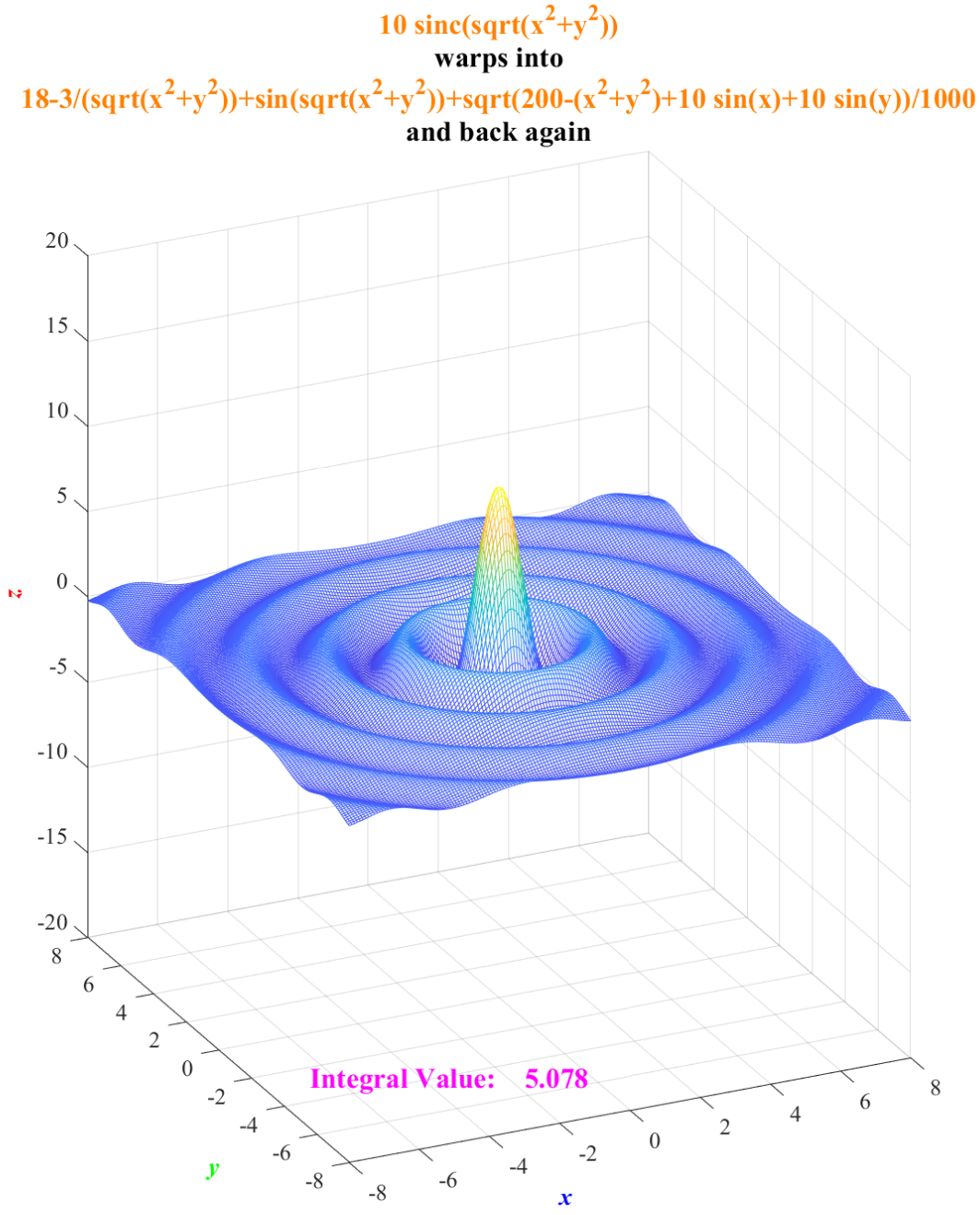


Figure 2: The function, $f_1(x, y)$, in the animation.

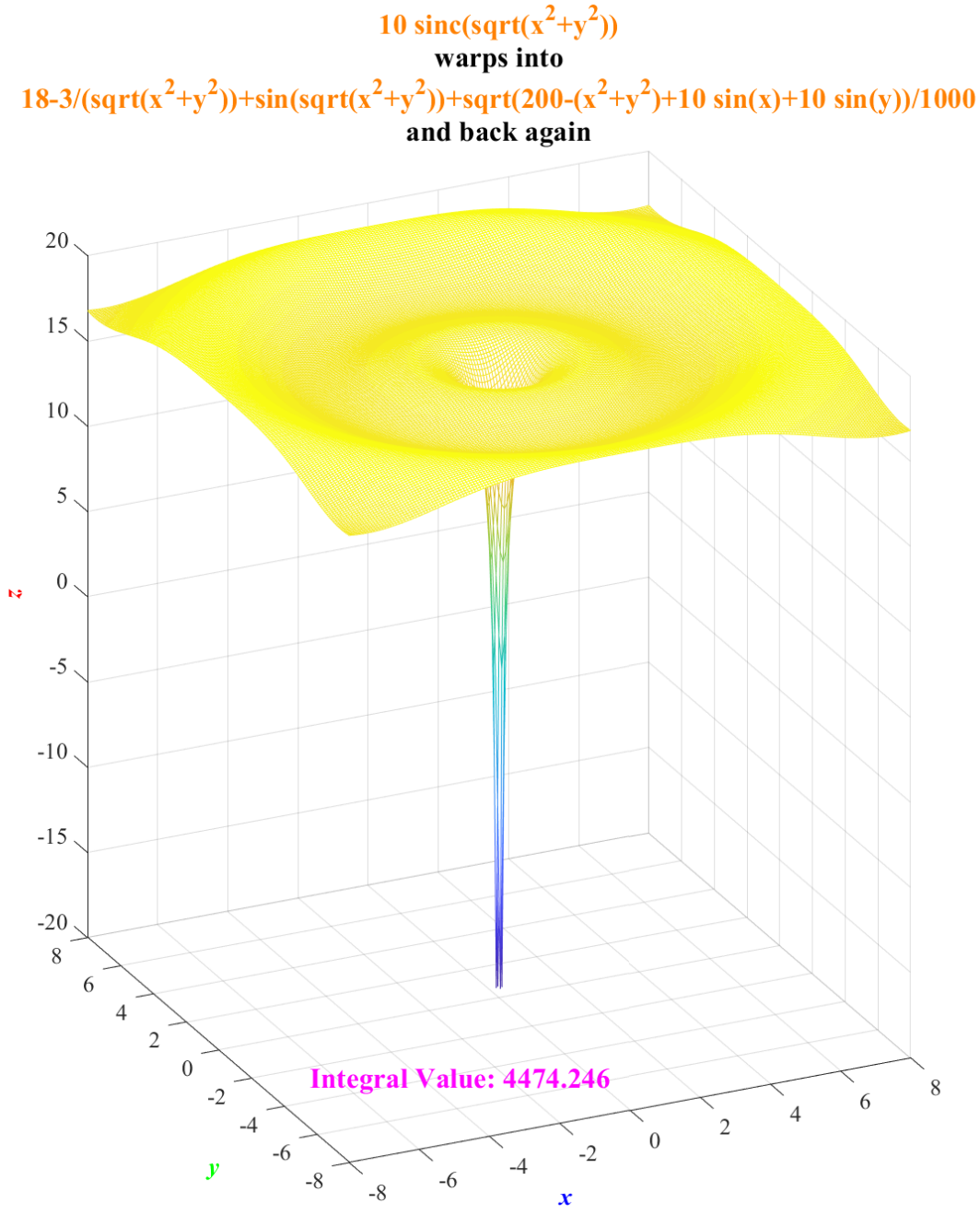


Figure 3: The function, $f_2(x, y)$, in the animation.

screen height and width, you will make the animation figure have the same relative dimensions on all computers (full size desktops or small screen laptops) that run your program (regardless of their screen sizes).

7. In some cases, particularly for functions that have singularities (blow-ups to ∞), it can be desirable to compute symbolic integrals, not only numerical ones. Can you integrate $f_1(x, y)$ and $f_2(x, y)$ symbolically in MATLAB? To answer this question, attempt to symbolically integrate the 2 functions. To do this you need to declare x and y to be symbol variables, compute `f1` and `f2` and then evaluate their symbolic integral values.

   ```
   syms  x y
   ```

   and then use:

   ```
   f1=10*sinc(sqrt(x^2+y^2));
   sym_area1=eval(int(int(f1,y,ymin,ymax),x,xmin,xmax));
   f2=18-3/sqrt(x^2+y^2)+sin(sqrt(x^2+y^2))+
   (sqrt(200-(x^2+y^2)+10\sin(x)+10\sin(y)))/1000;
   sym_area2=eval(int(int(f2,y,ymin,ymax),x,xmin,xmax));
   ```

   to integrate the 2 surfaces. `f2` is specified over 2 lines here for printing purposes, you should specify them on a single line in your program. Note the use of `eval` to evaluate the symbolic integration result. If you are able to integrate either of the functions, verify that the result is the same as the numerical integration result using `integral2`. If either or both functions are not integrable, state this in a comment and how you know that they are not integrable.

Submission

Finally, write all of this code in a script file a3.m, and publish the result of running your code to produce a3.pdf. Make sure that four images equivalent to the four that appear in this assignment sheet appear in your published output (this requires you to plot some of them in separate figures). Submit your source file and published pdf output to OWL when you are finished. **Make sure to start your file a3.m with an appropriate title and comment**

with your name and student number as you have for previous assignments! Also insert section headings (starting with %%) and comments at appropriate places in your code both to make your code easy to understand and to produce a nice sectioning of your code when you run `publish`.