CS2035 - Assignment 3 - 2017

Animation by Warping Mesh Surfaces

Out: Monday, February 27th 2017
In: Sunday, March 19th, 2017 at 11:55pm via Owl

Introduction

The MatLab version of the `peaks` function is given as:

$$\begin{aligned} f_1(x,y) = z &= 3*(1-x)^2*exp(-(x^2)-(y+1)^2) \\ &- 10*(x/5-x^3-y^5)*exp(-x^2-y^2) \\ &- 1/3*exp(-(x+1)^2-y^2) \end{aligned}$$

while a simple, second version, `-peaks`, could be specified as

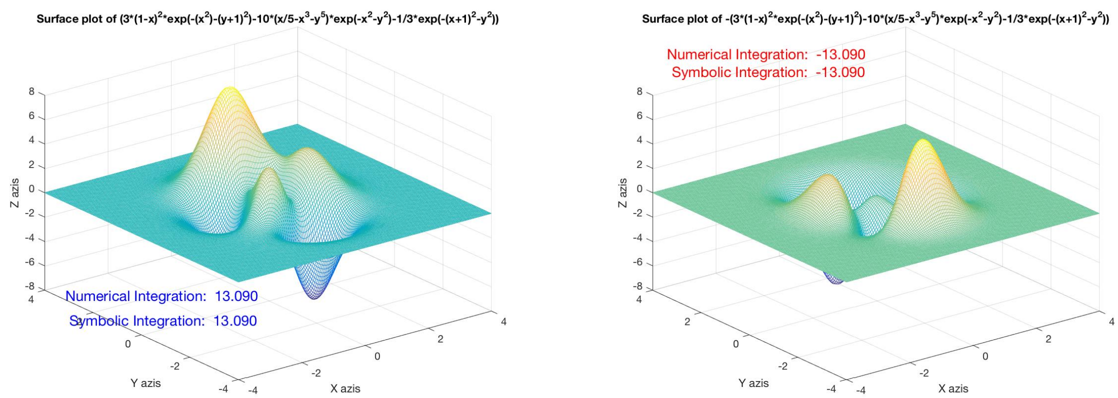
$$f_2(x,y) = -f_1(x,y)$$

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. Figures 1a to 1b shows the mesh plots of these 2 surfaces. Surface (b) is the inverse of surface (a).

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 -4 and x_{max} and y_{max} values of +4, along with $\Delta x = \Delta y = 0.05$, to generate `X` and `Y` from `meshgrid`. The z values are computed by MatLab (but fix them at $z_{min} = -8$ and $z_{max} = +8$ using `axis` for all the figures you display in this assignment). Then the 3D depth values, `Z1` and `Z2`, can be generated by a vectorized calculation for these 2 functions. `Z1` is the vectorized form of $f_1(x,y)$ while `Z2` is the vectorized form of $f_2(x,y)$. Using `meshgrid(x,y)`, where `x` is $\mathbf{x_{min} : \Delta x : x_{max}}$ and `y` is $\mathbf{y_{min} : \Delta y : y_{max}}$, compute `X` and `Y` and plot the 2 functions, given by `(X,Y,Z1)` and `(X,Y,Z2)`



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

Figure 1: Surface plots for (a) $f_1(x, y) = (3 * (1 - x)^2 * exp(-(x^2) - (y + 1)^2)10 * (x/5 - x^3 - y^5) * exp(-x^2 - y^2)1/3 * exp(-(x + 1)^2 - y^2))$ and (b) $f_2(x, y) = -(3 * (1 - x)^2 * exp(-(x^2) - (y + 1)^2)10 * (x/5 - x^3 - y^5) * exp(-x^2 - y^2)1/3 * exp(-(x + 1)^2 - y^2))$, both with numerically and symbolically evaluated integral values printed on them.

2. Next, perform numerical and symbolic integration on these 2 functions and use `text` to print out these values in blue and red respectively. Choose an appropriate fontsize and appropriate 3D coordinates for `text` to positioning the text while print the numerical and symbolic 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 has 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`.

MatLab function, `integral2`, evaluates the area under this function using numerical quadrature.

4. To symbolically integrate the 2 functions you need to declare $f1$, $f2$, x and y to be symbol variables, compute `f1` and `f2` and then evaluate their symbolic integral values.

   ```
   syms f1 f2 x y
   ```

   and then use:

   ```
   f1=(3*(1-x)^2*exp(-(x^2) - (y+1)^2)
   10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) 1/3*exp(-(x+1)^2 - y^2));
   sym_area1=eval(int(int(f1,y,ymin,ymax),x,xmin,xmax));
   f2=-(3*(1-x)^2*exp(-(x^2) - (y+1)^2)
   10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) 1/3*exp(-(x+1)^2 - y^2));
   sym_area2=eval(int(int(f2,y,ymin,ymax),x,xmin,xmax));
   ```

   to integrate the 2 surfaces. `f1` and `f2` are each 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 For this assignment numerical and symbolic integration always yield the same answers.

5. You can use the `title` command to print out the mathematical formula 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 Abe set the font-size to a large value. Use the 3D version of `text` to print out the numerical and symbolic integration results on these graphs (see below). You have to chose the x, y and z values to position these integration values on the plots.

6. You create the animation by warping `Z1` into `Z2`, 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{18} \color{blue} ' ...
             'Numerical Integration: ' ...
             sprintf('%8.3f',num_area1*(1-t)+num_area2*t)]);
 % {1.0 0.6 0.0} is orange
 text(xpos2,ypos2,zpos2,['\fontsize{18} \color{blue}
             'Symbolic Integration: ' ...
             sprintf('%8.3f',sym_area1*(1-t)+sym_area2*t)]);
 xlabel('\bf \color{red} X azis');
 ylabel('\bf \color{green} Y azis');
 zlabel('\bf \color{blue} Z azis');
 pause(display_pause);
 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 0.05. Thus, when this runs you will see a total of 21 surfaces displayed rapidly as `Z1` warps into `Z2`. If the display is too rapid, you can pause a small amount of time, `display_pause`, between adjacent displays to slow things down. Use the `text` command to print out the numerical and symbolic integration areas. Set variables `xpos1`, `ypos1` and `zpos1` and `xpos2`, `ypos2` and `zpos2` appropriately, where `zpos1` and `zpos2` are the vertical dimension for this figure. 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 the numerical and symbolic integration areas are linearly interpolated to correspond with the current surface being displayed. After warping `Z1` into `Z2`, you need to warp `Z2` into `Z1`. At this point you will have your animation working. Figures 2 and 3 show the **peaks** surface and the inverse **peaks** function, with the axes labelled, a title in orange and black and the numerically and symbolically evaluates integral values in magenta.

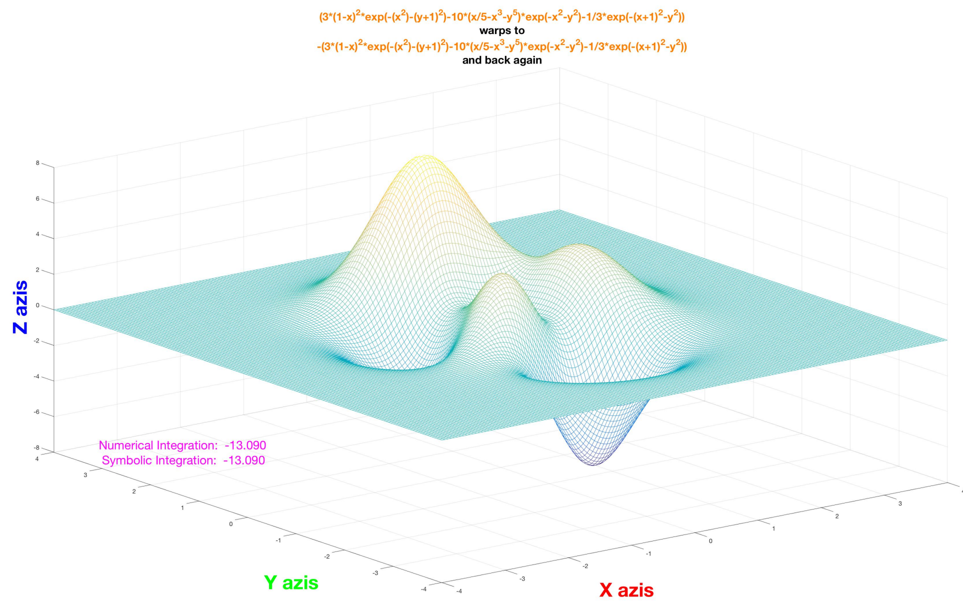


Figure 2: The peaks function, $f_1(x, y)$, in the animation.

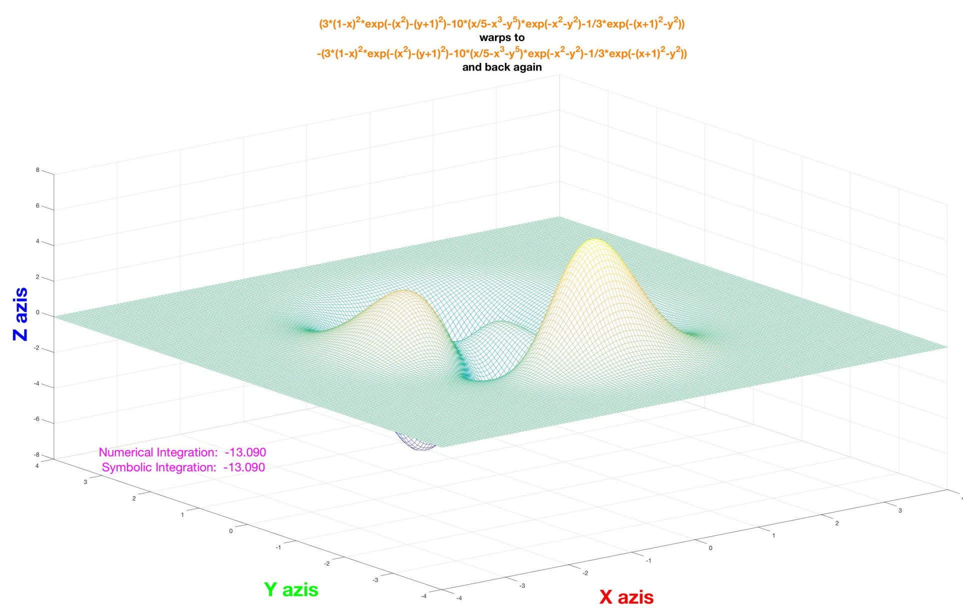


Figure 3: The peaks function, $f_2(x, y)$, in the animation.

7. Finally, the last task 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 75% of the height of the screen. Given the height, compute the width that satisfies an aspect ratio of 3/4. 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=3/4; % height to width

figureHeight=screenHeight*0.75;
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 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).

Lastly, write 3 MatLab functions, `ass3_2017` and 2 functions for your `peaks` functions. Do note use MatLab's `peaks` function.