CME292: Advanced MATLAB for Scientific Computing

Homework #1
Background & Graphics

Due: Thursday, April 9, 2014

Instructions

For this problem set, 2 problems out of 5 are required. You are free to choose the 2 problems you complete.

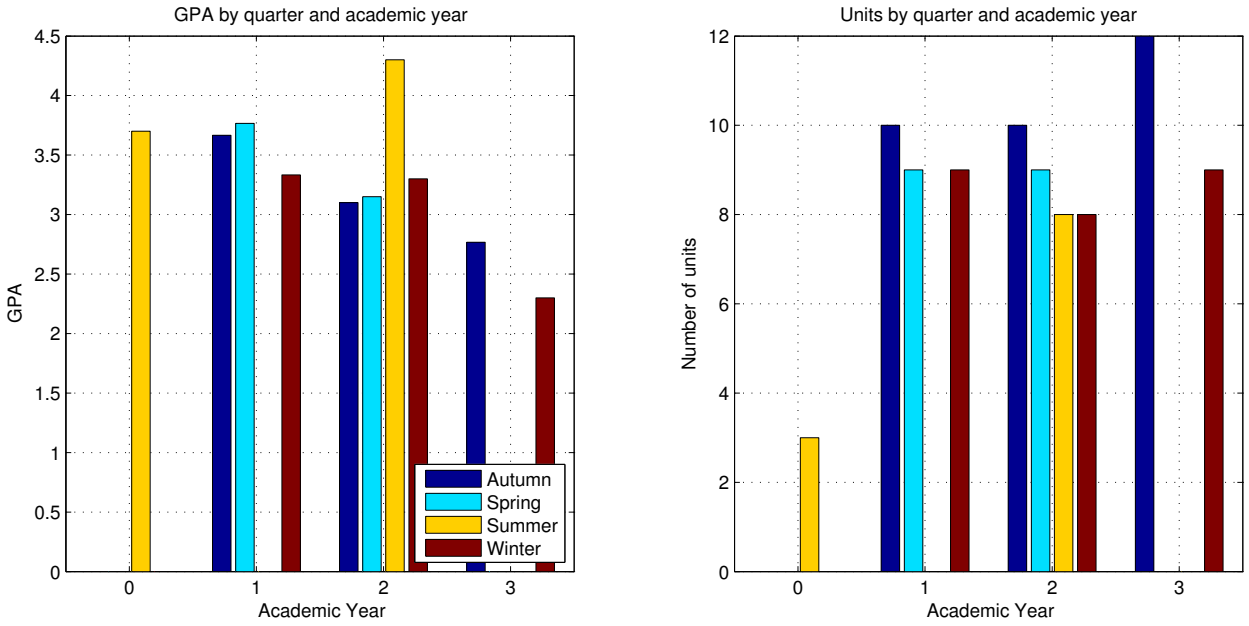
Before completing problem set, please see `HomeworkInstructions` on Coursework for general homework instructions, grading policy, and number of required problems per problem set.

Problem 1

In this problem, you will use either a cell or structure array to store a database of classes taken by student X while at Stanford University. Subsequently, the data structure will be used to compute the Grade Point Average (GPA) and other measures of academic performance for student X.

- You will be given a text file `courses.txt` containing a list of courses/grades and a function `read_courses.m` that loads the contents of `courses.txt` into a cell array.
  - `g = read_courses('courses.txt');`
    - `g{i,1}` - cell array for course number (cross-listings not included)
    - `g{i,2}` - string containing course title
    - `g{i,3}` - string containing term course was taken
    - `g{i,4}` - double containing number of units
    - `g{i,5}` - string specifying whether pass/not pass or letter grade
    - `g{i,6}` - string containing grade
  - The format used in `courses.txt` is exactly the format obtained by copy/pasting one's courses from Axess (under "Course History") into a text file. Therefore, you can use the code from this problem to compute your own GPA and related statistics without having to go through course-by-course and enter the data manually. For this assignment, *submit only the output corresponding to `courses.txt`, not your own grades.*
- Load output of `read_grades` into a convenient structure array with fields of your choosing. *Warning* - This will involve parsing strings such as `'AA210A'` to obtain the department `'AA'` and the course number `'210A'` and `'2012-2013 Winter'` to obtain the quarter (`'Winter'`) and year (`2013`) the course was taken.
  - As I have not discussed string parsing, I have provided you with a function `make_course_struct.m` that takes the output of `read_courses` and parses the text in the cell array to a more useful form in a structure array.
  - Feel free to use `make_course_struct.m` or make your own structures.

Figure 1: GPA/Number of units per quarter per year



- Grade point average of class set $\mathcal{S}$ is defined as

$$\text{GPA} = \frac{1}{\sum_{c \in \mathcal{S}_L} u(c)} \sum_{s \in \mathcal{S}_L} g(s) * u(s)$$

  where $\mathcal{S}_L \subseteq \mathcal{S}$ is the subset of classes in $\mathcal{S}$ taken for a *letter* grade, $g(c)$ and $u(c)$ are the grade letter value and number of units of class $c \in \mathcal{S}$, respectively. The letter grade mapping from letter grade to letter grade value can be found here: http://studentaffairs.stanford.edu/registrar/students/gpa-how. The letter grade mapping is also taken care of in `make_course_struct.m` if you choose to use it.

- Use the above data structure to:
  - Compute GPA for set of all classes
  - Create a bar graph of GPA vs. department.
  - Create a bar graph of GPA vs. quarter taken (Autumn, Winter, Spring, Summer)
  - Create a bar graph of GPA vs. quater *and* year in Stanford career that class was taken (1st, 2nd, ...). Output should look similar to Figure 1.
    - Define Year 1 at Stanford as the academic year starting the *Autumn* quarter you arrived at Stanford. Any class taken the summer before your first Autumn quarter would be considered a 0th year.
    - This was taken care of in `make_course_struct.m` for you
  - Compute total number of classes taken
  - Compute total number of graduation units (includes those taken for letter grade and P/NP)
  - Create/compute one additional plot or statistic you find interesting. I chose number of units per year per quarter as in Figure 1.

Problem 2

Optimize the code in

- `optimize_me.m` and `optimize_me_too.m`

Things to look for when optimizing: (1) numeric arrays have less memory overhead than structure or cell arrays, (2) use `clear` to delete variables no longer needed, (3) vectorize loops whenever possible, (4) logical indexing much faster than using `find` to extract index arrays, (5) delayed copy can make a huge difference (when large matrix passed to function, don't unnecessarily modify it), (6) sparse matrices can be *much* more efficient than full matrices, in terms of both storage and operations. Some, but not necessarily all of these optimizations will be significant in this problem. MATLAB's profiler is a useful tool in determining bottlenecks.

- Profile the code using MATLAB's profiler before and after optimization. What optimization was most significant? *Be sure to set `animate = false`, otherwise plotting will dominate the CPU time.*

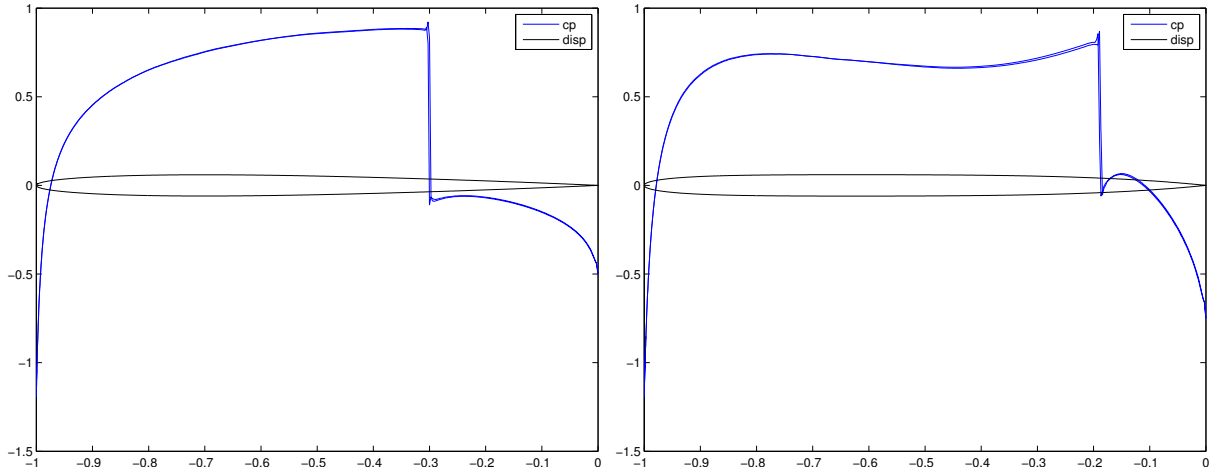
Problem 3

In this problem, you will make a publication-quality plot commonly found in the transonic aerodynamic shape optimization literature. You will plot pressure coefficient variation (C_P) along the surface of the airfoil as well as the airfoil profile on the same plot (`plotyy`) for both the initial and optimized airfoil configurations. Your plot should look similar to Figure 3.
The file `pressure_displacement_profiles.mat` contains the airfoil profile and C_p distribution over the airfoil. This file contains two variabes, `initial` and `optimal`, which each have three fields: `x`, `cp`, `disp` which are vectors defining the x coordinates, C_p profile, and shape profile, respectively. The code below shows how to load and plot the pressure coefficient and shape. The output is in Figure 2.

```
% Plot profiles
load pressure_displacement_profiles

% Plot initial -Cp and shape
fig1=figure;
plot(initial.x,-initial.cp,'b-'); hold on;
plot(initial.x,initial.disp,'k-');
legend('cp','disp');

% Plot optimal -Cp and shape
fig2=figure;
plot(optimal.x,-optimal.cp,'b-'); hold on;
plot(optimal.x,optimal.disp,'k-');
legend('cp','disp');

% Save to eps
print(fig1,'-depsc2','Hwk1Prob3_starter1');
print(fig2,'-depsc2','Hwk1Prob3_starter2');
```

- Modify the starter code to generate plots similar to those in Figure 3 (I recommend using `plotyy`; use MATLAB `help`).
- Plot the *negative* of C_p instead of C_p to generate the plots in Figure 2 and 3.
- The following criteria should be met

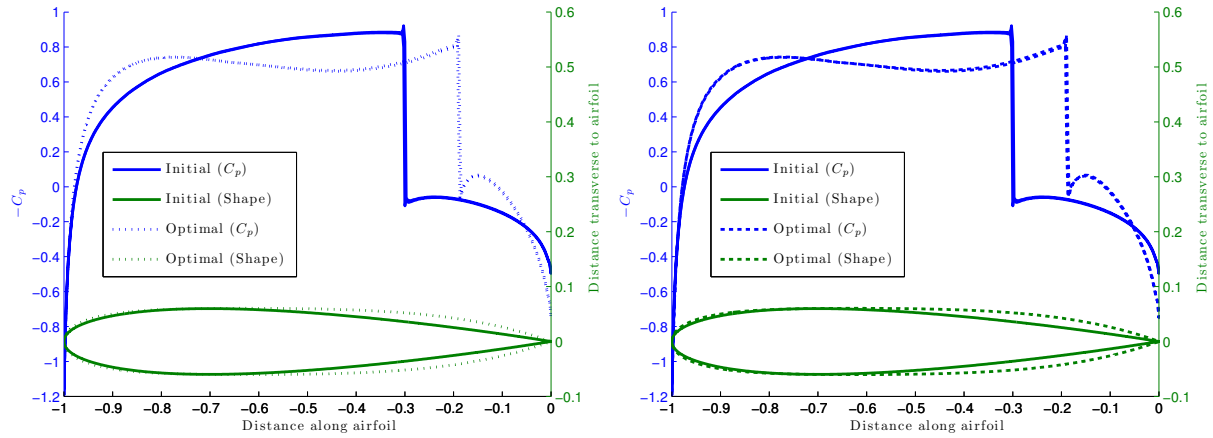
Problem 3 continued on next page...



(a) Initial Configuration (b) Optimal Configuration

Figure 2: Problem 3 - starter code output

- Use solid lines for the initial C_p and shape profiles and *dotted* (not dashed) for the optimal C_p and shape profiles. All lines should be thick (at least `linewidth` of 2).
- Generate a legend with the entries from Figure 3.
- Use the y-tick labels and spacing from Figure 3.
- Use same x- and y- labels from Figure 3 *with* LaTeX interpreter.
- (extra credit) Use `fixPSlinestyle` (MATLAB File Exchange) to fix the dotted lines in your EPS file.



(a) Without `fixPSlinestyle` (b) With `fixPSlinestyle`

Figure 3: Problem 3

Problem 4

In this problem, you will gain practice using `fill`, `streamline`, `quiver` and `contour`-type plotting functions. You will be provided text files (topo.txt, vel.txt, dens.txt, disp.txt, pres.txt) containing the flow around an

Problem 4 continued on next page...

airfoil on an unstructured grid. A function, `get_all_quantities`, has been provided that loads the flow quantities into MATLAB and interpolates them on a *structured* grid (the interpolation is very crude near the airfoil; as a result, there may be some unphysical artifacts in the flow near the airfoil. Ignore this as it is not important for this problem). See the tips at the end of this problem description for a brief exposition of the outputs of `get_all_quantities.m`.

Generate four plots *similar* to those in Figure 4 (the formatting and color scheme are up to you, the plots in Figure 4 are purely for reference).

- First, plot the flow field (x- and y-velocities) using `quiver`. Plot an outline of the optimal airfoil to highlight the edge of the physical flow (see `starter_code.m` and `pressure_displacement_profiles.mat` from Problem 3). You will notice there are quiver arrows *inside* the airfoil, another artifact of the crude interpolation. Cover these interior arrows by plotting the shape of the airfoil as a *filled, white* polygon using `fill`. Finally, plot streamlines emanating from the points $(-2,-0.6)$, $(-2,-0.5)$, $(-2,-0.4)$, $(-2,-0.3)$, $(-2,-0.2)$, $(-2,-0.1)$, $(-2,0.1)$, $(-2,0.2)$, $(-2,0.3)$, $(-2,0.4)$, $(-2,0.5)$, $(-2,0.6)$. The streamlines should be highlighted in some way to distinguish them from the quiver arrows.
- In the final three plots, generate a contour plot (you can use any contour function you want, I chose `contourf`) of the pressure, density, and velocity *magnitude* throughout the domain. Include a colorbar with each of these plots. Similar to the first part, plot the outline of the airfoil and cover the interior of the airfoil with `fill`. Finally, plot the flow field on top of each contour plot (as streamlines or a quiver plot; I chose quiver).
- Save all plots to colored eps files.

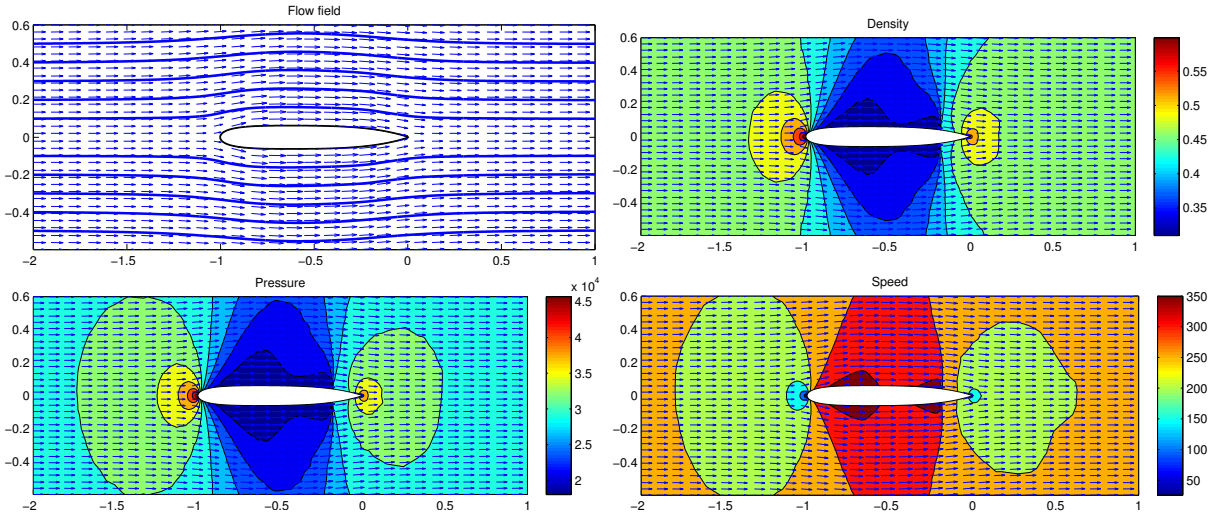


Figure 4: Problem 4

Hints:

- `get_all_quantities` may be given argument (see comments in file) defining the limits of the mesh and number of points in the structured grid. This function was equipped with defaults, so arguments are not required. The defaults were used to create the plots in this document. Feel free to choose the mesh size you wish, provided the quality of the resulting plot is comparable to those shown here.

Problem 4 continued on next page...

```
function  [X,Y,R,P,V1,V2] = get_all_quantities(nx,ny,xlim,ylim)
% X, Y         - nx x ny matrices defining grid (output of meshgrid)
% R, P, V1, V2 - nx x ny matrices defining results (density, pressure,
%                velocity_x, velocity_y) over grid X, Y
```

  - As the comments in `get_all_quantities.m` indicate, the outputs are `X, Y`, the coordinates of the structured grid (`meshgrid` output); `R`, the density over the structured grid; `P`, the pressure over the structured grid; `V1, V2`, the velocity of the flow in the x, y directions.

- `quiver` will plot one arrow per grid point. To make the arrows in the quiver plot visible, I recommend sparsifying the grid.

Problem 5

The function `[X,Y,Z,V] = flow()` is a simple way to generate an interesting scalar-valued function of three variables. In this problem, we use `flow` to reproduce some of the plots seen in lecture and some additional ones. In this problem, generate the plots in Figure 5 using the following steps. Use `set(gca,'view',[149.5 ... 44])` to reproduce the view in Figure 5.

- Use `contourslice` to generate contours in the planes $x = 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0$ and $z = 0.0$ (there will be a total of 10 planes; 9 of them are $y - z$ planes and one an $x - y$ plane). This should look similar to Figure 5a.

- Use `slice` to plot the flow value in the planes $x = 2.0, 8.0$, $y = -2.0$, and $z = 0.0$. This should look similar to Figure 5b.

- Use `surfnorm` to plot the following surfaces, $\mathbf{F}(x, y)$ and $\mathbf{G}(x, y)$, with *surface normals*.

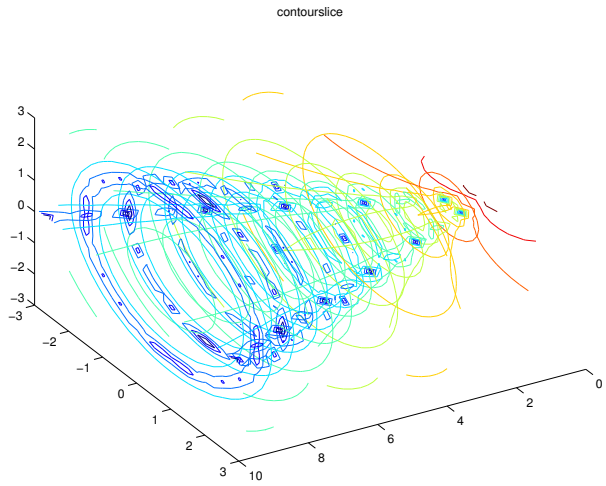
$$\mathbf{F}(x, y) = \left(\frac{1}{10}x\right)^2 + \left(\frac{1}{3}y\right)^2 - 1 \tag{1}$$

$$\mathbf{G}(x, y) = \left(\frac{1}{10}x\right)^2 - \left(\frac{1}{3}y\right)^2 - 1 \tag{2}$$

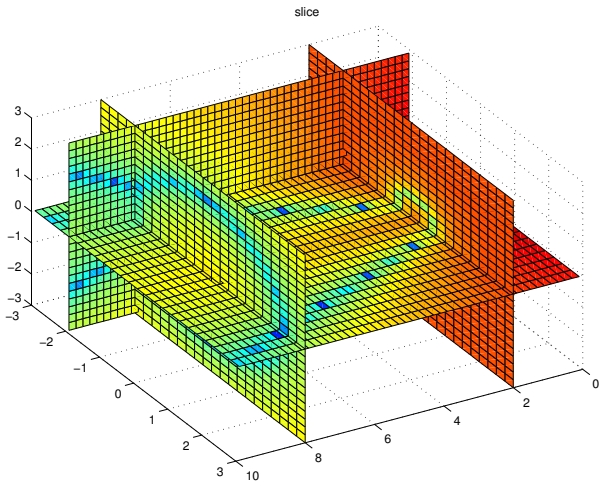
  This should look similar to Figure 5c (it is a little crowded).

- Use `slice` to plot the flow value in the surfaces $\mathbf{F}(x, y)$ and $\mathbf{G}(x, y)$. This should look similar to Figure 5d.

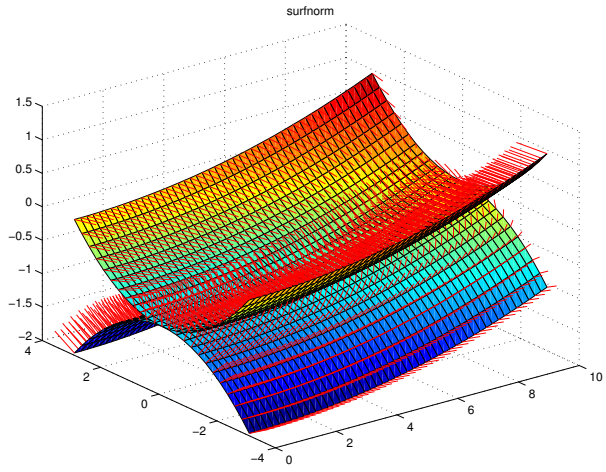
- Use `isosurface` to plot surfaces of the flow at the isovalues $v = -5, -4, -3, -2, -1, 0$. This should look similar to Figure 5e.



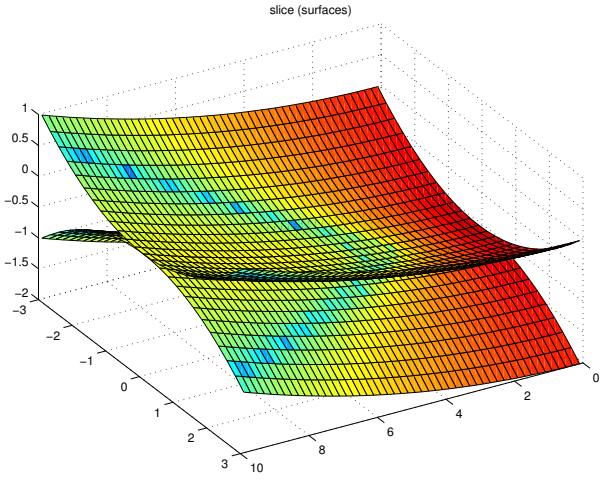
(a) `contourslice`



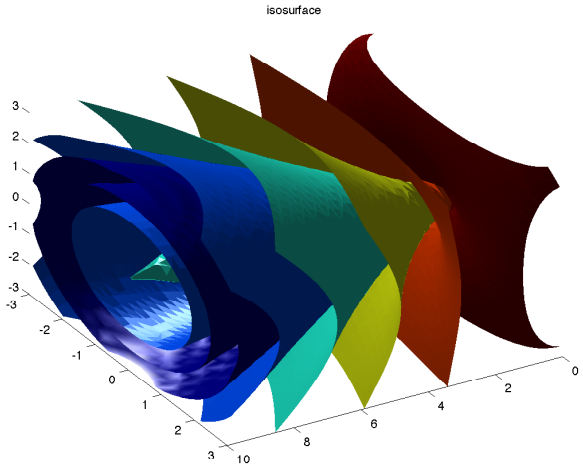
(b) `contourslice`



(c) `contourslice`



(d) `contourslice`



(e) `contourslice`

Figure 5: Problem 5