MECH 222 Computer Lab 6—Optimization by Hill-Climbing

Read and understand the briefing notes and lab instructions before your lab period. Each lab activity corresponds to one of the main ideas in the briefing notes.

Learning Objectives By the end of this lab, with reference to the Briefing Notes, you should be able to:

- Produce surface plots for a given function of two variables on a rectagular region using `meshgrid` and `surf`.
- Articulate and exploit a physical/geometrical interpretation of a function's gradient as the vector that shows the direction of steepest increase for the function at the evaluation point.
- Produce contour plots for a given function of two variables on a rectagular region using `meshgrid` and `contour`.
- Approximate partial derivatives using numerical calculations.
- Plot a gradient field using `quiver`, given a Matlab function to calculate the gradient.
- Find an approximate solution for a 2-variable system of autonomous first-order ordinary differential equations using Matlab's standard ODE solver, `ode45`.
- Illustrate hill-climbing trajectories both on a contour plot (using `plot`) and on a surface plot (using `plot3`).
- Modify a syntactically-correct but mathematically incomplete gradient-finding function so that it returns correct values.
- Write nicely-formatted messages that mix text and calculated values to the main display, using one of `sprintf` or `fprintf`.

PRE-LAB ASSIGNMENT:

Read through all the Briefing Notes and the Activities described below.

PL1: Carry out an efficiency analysis on the exhaustive-search method in Activity 2. Taking the set S and function f for your assigned lab day, find (and justify) approximate values for ...

(i) the number of times the computer must evaluate f in parts (d) and (e) of Activity 2,

(ii) the number of function evaluations that would be needed to get a componentwise absolute error not larger than 1×10^{-4},

(iii) the number of bytes of storage required for each of matrices `X`, `Y`, and `Z` in scenario (ii) above, assuming each matrix entry uses 8 bytes; and

(iv) the number of function evaluations and bytes of storage that would be required by the obvious generalization of this scheme to maximize a function $g(w, x, y, z)$ with four input variables, each chosen from the interval $[0, 1]$, if we insist on a componentwise absolute error of at most 1×10^{-4}.

PL2: In Activity 3(b) you need to generate a two-component vector for each point in a known grid. The Briefing Notes (paragraph 3.2) are deliberately vague on this point: they simply mention "nested `for`-loops," counting on you to know what this means and how to implement it. Respond by planning the commands you are going to write before you arrive at the lab. Prepare a block of code involving nested `for`-loops and bring a copy on paper to hand in at the beginning of the lab. Keep another copy with you, so you can try it and improve it during the lab session. (*Suggestions*: You could just write out some Matlab commands by hand, twice, or type some Matlab commands in an email to yourself and print a copy to bring to the lab, or type the commands into a word processor and print one copy, etc. Any method that will give you one copy to hand in promptly at 2:00 p.m. and another copy to work with is acceptable.)

ACTIVITY 0: *Keep a session transcript*

0.1 Download these files from Vista:

- ☐ `startup.m`
- ☐ `finish.m`
- ☐ `localLOGSAVENOW.m`

Save these files in the default Matlab directory on your lab computer.

0.2 Start Matlab *after* the three files named above are in place. (If Matlab is already running before you do the download, please stop Matlab and restart it once the files are in place.) When Matlab notices a file named `startup.m` in the current directory, it will run that script. You should see a short request to type your name and student number. Enter them and then keep working as usual. If there is no such request, it means the `startup.m` file was not found. Try again to put it into the correct working directory. Ask the Lab TA if your attempts are unsuccessful.

0.3 Reserve a minute or two at the end of the lab to submit the archive file the session logging software will create on your computer. See Activity 4 below for details.

Explanation. Science education researcher Dr. W. J. Code is helping study and improve the computer labs. Dr. Code has built the Matlab functions named above to quietly collect information about your progress as you work. They keep track of what commands have been run in Matlab and when, and what output and error messages you saw. They DO NOT track m-file code that you write in the editor, and they cannot "see" anything outside of Matlab (for example, web browser use). This is intended to be an invisible modification to your normal workflow that offers the possibility of big improvements for future generations of Mech 2 students.

If you would like to know more about how the logging system works, keep reading this section. For further questions, email warcode@math.ubc.ca.

Matlab automatically executes the script `startup.m` (if it exists) whenever it starts. Similarly, it automatically runs `finish.m` just before it quits. Dr Code has prepared files with these names to do the data-collection mentioned above. The system, built exclusively in Matlab, will create three files:

(i) `LOGFILEINFO.dat` – ignore this, but please don't disturb it: this file needed for the system to work properly;

(ii) `labNN-MAR-2011diary.txt` – a transcript of all your output, built using Matlab's `diary` command to capture all your output; browse for interest or ignore as you please;

(iii) `log12345678logNN-MAR-2011.mat`, with 12345678 replaced by your student number, and NN by the date – a detailed log file that stores a summary of your commands and output. This is the file to upload at the end of your session

ACTIVITY 1: *Maximize a 1-variable function by exhaustive search.*

Work on the interval $I = [0, 10]$ with the single-variable function

$$f(x) = (x^4 - x^2 - x)e^{-x}.$$

Produce and hand in the items detailed below.

Hand-in Checklist:

(a) The graph of f on the interval I, with axis labels and your name in the title.

(b) A point in I that gives the maximum value of f, with an absolute error not larger than 0.001.

(c) The approximate maximum f-value.

(d) The Matlab commands used to produce the items above.

ACTIVITY 2: *Maximize a 2-variable function by exhaustive search.*

Look up the function f and the region S for your lab day on the last page of this document. Use them with the ideas explained in the briefing notes to produce and hand in the items listed below. Notice the strong similarity between elements (d) and (e): consider writing a script for (d) so that a few quick edits will give you a result for (e). (Students who have understood paragraph 2.4 of the Briefing Notes will have a strong advantage in (d)–(e).)

Hand-in Checklist:

(a) A vectorized Matlab function named `f` that evaluates f. It's essential to use the name `f`; see Activity 3.

(b) A nice contour plot of f on S, like the one in the briefing notes. That is,
- evaluation points are shown as dots,
- there are between 20 and 50 dots along each coordinate direction,
- there are between 20 and 50 contours,
- no two contours land nearly on top of one another, and
- every point of S lies reasonably near to some contour.

As usual, label the axes and put your name in the title.

(c) A surface plot showing the graph of f. Include axis labels and choose an orientation where it's possible to estimate the location and the value of the maximizer.

(d) The point in S that gives the largest value of f, with an absolute error in each component not larger than 0.01.

(e) An improved estimate of the maximizing point, with an absolute error not larger than 0.001 in each component. [This requires quite a lot of computational work, and you will have to wait some time to get back the answer. Imagine the wait you would have if you needed another digit of accuracy, or if you had a function depending on four variables instead of two. Your work in the PreLab, combined with your experience here, is supposed to convince you that the method of exhaustive search is too primitive for all but the simplest problems.]

Note: To interrupt a Matlab computation in progress, you can hold down the control key and then press "c" (CTRL-C).

ACTIVITY 3: *Use a hill-climbing dynamical system to maximize a function.*

Continue with f and S from Activity 2.

Hand-in Checklist:

(a) A Matlab function `gradf` suitable for use as a right-hand side in `ode45`. That is, for each given (x, y)-point in the form of a 2-element vector `XY`, the command `gradf(t,XY)` should return a 2-element column vector containing the components of $\nabla f(x, y)$. (The value of t gets ignored.) Internally, the function `gradf` should implement the formulas shown in the briefing notes, assuming that it can get values for f by referring to another function named `f`. This way the same function `gradf` can used with a variety of different functions `f`. A template[1] for the desired function is available for download from Vista. *You must make some changes to the function as supplied, or it will quietly return useless values that spoil your results.*

(b) A contour map just like the one in Activity 2(b), but improved by adding an overlay of arrows showing the gradient vector at various points. (Compare paragraph 3.2 in the briefing notes.)

The Briefing Notes suggest the command `quiver(Xmesh,Ymesh,U,V)` to draw these arrows. To control the length of the arrows, you may wish to experiment with the modified command

```
quiver(Xmesh,Ymesh,m*U,m*V)
```

where `m` is some scalar stretch-factor. Taking `m` $= 1$ will leave the picture unchanged; choosing a specific `m` < 1 will shorten the arrows, while choosing `m` > 1 will lengthen them. Experiment to produce a sketch where the arrows communicate most effectively.

(c) An improvement on the contour map in item 3(b) above that overlays several steepest-uphill trajectories computed using `ode45`. Also, listings of the Matlab scripts or command-line input used to produce the graphic. (*Suggestion*: Write a script to find these trajectories. Use a loop to work through a list of initial points you choose yourself. Initial points near the outer edges of S make the nicest illustration.)

(d) Your best estimates for the maximizing input point (x, y) and the maximum value $f(x, y)$, based on the results illustrated in part 3(c) above.

(e) A 3D plot showing how the plane trajectories displayed in 3(c) correspond to hill-climbing paths to the highest point on the graph of f.

ACTIVITY 4: *Submit your Log File*

At the end of the lab, upload to Vista the log file produced by the scripts described in Activity 0. The name should have the general form `log12345678logNN-MAR-2011.mat`, with 12345678 replaced by your student number, and NN by the date. You almost certainly have to do this in the lab room, because the file has been created on the lab computer.

[1] TEMPLATE (Computer Science): A document or file having a preset format, used as a starting point for a particular application. – the Free Online Dictionary.

Daily Function Choices

Important: In some cases, the absolute maximum of f in the specified rectangular domain S is only a local maximum for f in the plane. Working with the wrong domain spoils everything. Students should make an extra effort to get the domain right.

MONDAY

$$f(x,y) = \frac{xy}{2+2x^4+y^4}; \qquad S: \quad 0 \le x \le 3,\ 0 \le y \le 4.$$

TUESDAY

$$f(x,y) = 30xy - 10x^3 - 5y^3; \qquad S: \quad -0.5 \le x \le 2.5,\ -0.5 \le y \le 3.0.$$

WEDNESDAY

$$f(x,y) = x^2 y e^{-x^2-y^2}; \qquad S: \quad 0 \le x \le 2,\ 0 \le y \le 2.$$

FRIDAY

$$f(x,y) = 4xy - x^4 - y^3; \qquad S: \quad -3 \le x \le 3,\ -\tfrac{1}{2} \le y \le 3.$$