Mech 222 Computer Lab Learning Goals
Based on labs used in 2011 and 2012 written by Philip Loewen.

Higher-Level Matlab Learning Goals

The seven computer labs during Mech 222 each include preparatory reading, a pre-lab assignment and a two-hour in-lab component. The goals are that, by the end of the term and with appropriate reference material at hand, students should be able to use Matlab to:

1. write and debug for-loops that iterate over vectors or matrices (nested loops).
2. write and run their own Matlab functions and scripts after choosing which is more appropriate for a situation.
3. load data into the workspace and manipulate it into a usable form.
4. plot 2D graphs for data analysis.
5. plot and manipulate 3D surfaces.
6. perform numerical integration in 2D and 3D.
7. work with differential equation solvers (especially `ode45`) and solutions: plotting, producing trajectories for different parameter values, producing related solutions using a loop.
8. fix short pieces of given code (involving any of the above items) that is known to be faulty.

Lab-by-lab Matlab Learning Goals

This section provides more detail on the goals for each of the labs. The order here represents the one used in Jan-Apr 2012.

Lab 1: Point Clouds

Computing aggregate properties of given 3D point data (e.g. centre of mass of points or wire connecting points). In terms of Matlab skills gained, with reference to the supplied briefing notes and their pre-lab work, students should be able to:

1. write and use a Matlab function.
2. input and store 3D point mass data as four equal-length vectors, one for each of the space coordinates and one more for mass.
3. work with this point data in order to calculate aggregate properties like mass and centre of mass for a group of points.
4. use this data to plot in 3D using `plot3`.
5. use the `load` function to import pre-formatted data in a provided `.mat` data file.

Lab 2: Iterated Integrals

Further work in 3D visualization; setting up and accumulating sums from many tiny patches; first steps in numerical integration involving discretizing of regions.

1. build a rectangular mesh of points in the plane using `linspace` and `meshgrid`.
2. perform numerical integration in 2D using an iterated trapezoidal method.

Lab 3: Triangulated Surfaces

Grok, orient, and measure triangulated surfaces.

1. input and store 3D triangular surface data in a matrix data structure: vectors that list the vertices involved (as in the point data of Lab 1), and an n-by-3 `FACELIST` matrix that encodes the list of triangles as indices to trios of point vectors.
2. write a Matlab function that calculates the normal vector to such a triangle with magnitude determined by the triangle area.
3. plot triangles in 3D using `trimesh` (unfilled) and `trisurf` (filled).
4. use for loops to iterate over the triangle data structure in order to calculate aggregate properties like surface area and centroid for a group of triangles.
5. view/modify basic plot features using the get/set functions on graphics handles.

Lab 4: Double Integrals

Generating meshes and triangulations for surfaces given algebraically; idea of parametric surfaces.

1. use the normal vector function developed in Lab 3 to write a general surface integral routine for a 3D surface defined by triangles (Lab 3 only asked for surface area and centroid).
2. create a triangular mesh approximation for domains determined by simple expressions (rectangles or inequalities involving quadratic or linear functions) in terms of the `FACELIST` data structure using `delaunay`.
3. create a triangulated approximation of a single-valued function defined on such a domain, again in the `FACELIST` data structure.
4. implement a more general double integral routine using the above routines for surfaces.

Lab 5: Freighter Parameter Estimation (Foam Boat Lab)

Students load and manipulate real data collected in their "foam boat" physical lab; work with global variables; learn about subplots; find approximate solutions for inconsistent linear systems by least squares. In terms of Matlab skills gained, with reference to the supplied briefing notes and their pre-lab work, students should be able to:

1. draw multiple plots on the same figure using the `subplot` function and set their axes to line up for comparison purposes.
2. prepare a collection of data in a data file appropriate for direct Matlab importing.

3. import physical lab data stored in an appropriate text format using the `load` function.
4. use comparison plots to estimate physical constants from the boat experiment.
5. declare physical constants as global variables for use by Matlab functions relevant to the experiment.
6. use information from a least squares calculation along with graphical evidence from plotting lines to estimate an appropriate point to act as an "intersection" of three lines (where all three intersect according to physical theory, but probably not in actual data).
7. define a function using the `@f` syntax, assigning it to a function handle suitable for passing to `ode45`.

Lab 6: Optimization by Hill-Climbing

Exhaustive search versus gradient-ascent approaches to numerical optimization of a differentiable two-variable function.

1. produce surface plots for a given function of two variables on a rectangular region using `meshgrid` and `surf`.
2. produce contour plots for a given function of two variables on a rectangular region using `meshgrid` and `contour`.
3. plot a gradient field using quiver when given an appropriate Matlab function to calculate an approximate gradient.
4. compute (using `ode45`) and plot (using `plot3`) a 2-variable, first-order ODE approximate trajectory.
5. update an m-file that defines a function which is known to be buggy even if it is syntactically correct (no Matlab errors, only mathematical errors).
6. output messages to the main display that mix text and formatted numbers using `sprintf`.

Lab 7: Engine Optimization

Multi-pane plots with subplot; back-door variable sharing with `global`; yet more practice with `ode45` and 2D plotting; line integrals and an ordinary differential equation. In terms of Matlab skills gained, with reference to the supplied briefing notes and their pre-lab work, students should be able to:

1. include more than one plot in the same Matlab figure using `subplot`.
2. use a combination of a setup file and global variables to pass often-used physical constants to a suite of related functions.
3. plot in Matlab using a base-10 log scale on either or both axes (in 2D).
4. incorporate given Matlab functions with multiple outputs into high-level scripts (e.g. for plotting) by reading their source code.
5. use an explicit example to change the settings in `ode45` for trajectory plotting.
6. build an existing script of commands into a large for-loop where one of the parameters in the script is changed for each iteration.