MATHS 2104 Numerical Methods
Assignment 3: 2D splines and weather maps
Due: 1:00 pm Tuesday 12 September 2017
30 Marks (+ Cody tests)

CHECKLIST

- ☐: Have you shown all of your working for every mathematical question?
- ☐: Have you included all Matlab code and plots to support your answers in programming questions?
- ☐: Have you made sure that all plots have labelled axes, and legend where appropriate?
- ☐: Have you submitted all code required to Cody Coursework?
- ☐: If before the deadline, have you submitted your assignment via the online submission on Canvas?
- ☐: Is your submission a single pdf file - correctly orientated, easy to read? If not, penalties apply.
- ☐: Penalty for submitting more than one document: 10% of final mark, per extra document. Note: you may resubmit and your final version is marked, but final document must be a single file.
- ☐: If after the deadline, but within 24 hours, have you contacted us via the enquiry page on Canvas (if applying for an extension) and submitted your assignment online via Canvas?
- ☐: Penalties for late submission: within 24 hours, 40% of final mark. After 24 hours, assignment cannot be submitted and you get zero.
- ☐: Assignments emailed instead of submitted via Canvas will not be marked and will receive zero.
- ☐: Have you checked that the assignment submitted is the correct one, as we cannot accept other submissions after the due date?

Preamble

The aims of this assignment are to:

- Develop code to interpolate 2D data using a spline function;
- Explore how interpolation can be used in weather prediction (and what some of its limitations are)

Contents

1 A polyharmonic 2D spline

In this assignment, we will interpolate data points (x_j, y_j, f_j) using the two-dimensional polyharmonic spline interpolant

$$f(x, y) = a + b_1 x + b_2 y + \sum_{j=1}^{N} c_j \phi(r_j), \qquad (1)$$

where $r_j = \sqrt{(x - x_j)^2 + (y - y_j)^2}$, $a, b_1, b_2, c_1, \ldots, c_N$ are coefficients, N is the number of data points, and

$$\phi(r_j) = r_j^2 \ln r_j. \qquad (2)$$

Note: $\ln(x)$ is `log(x)` in Matlab

1.1. What happens when you try to evaluate the radial basis function ϕ at the data points $(x, y) = (x_j, y_j)$? What is causing this problem?

1.2. Find the limit of $\phi(r_j)$ as $(x, y) \to (x_j, y_j)$. Hence, determine an appropriate value to assign $\phi(r_j)$ to when $(x, y) = (x_j, y_j)$.

L'Hopital's rule should help here.

1.3. Similarly to lectures, write out by hand the system of equations which must be satisfied for the interpolant (2) to pass through all N data points and also satisfy the extra three constraints on the coefficients c_j that we used in lectures. Identify the matrices A, $\mathbf{x}$ and $\mathbf{b}$ in the system $A\mathbf{x} = \mathbf{b}$ of linear equations that you need to solve.

1.4. Write a MATLAB function called `polyharm.m` that evaluates the interpolant (2), and which passes through the two-dimensional data points (x_j, y_j, f_j), for $j = 1, 2, \ldots, N$. You should use the same constraints on the coefficients that are used in lectures, ie, your matrix A from the previous question should make an appearance.

Your code from item 2.3(e) of Practical 3 is an excellent starting point.

Your function must use the interface

```
function f = polyharm(x, y, xj, yj, fj)
```

where `x` and `y` are *matrices* of arbitrary size containing the x and y coordinates of the points at which the interpolant is to be evaluated (the interpolation points), `xj` and `yj` are column vectors containing the x and y coordinates of the data points and `fj` is a column vector containing the data to be interpolated. The output argument `f` must be of the same size as `x` and `y` and must contain the values of the interpolant f for the corresponding elements of `x` and `y`. Your function should be fully vectorised, without any `for` loops.

1.5. Write a MATLAB script `testpolyharm.m` that tests your function `polyharm.m`. The script should:

   (a) Use `meshgrid` to create a coarse 20×20 grid of data points in the domain $0 \leq x \leq 2\pi$, $0 \leq y \leq 2\pi$.

   (b) Evaluate the test function

   $$g(x, y) = \cos x \sin y$$

   at each of these data points.

   (c) Use `meshgrid` to create a fine 80×80 grid of points to evaluate the interpolant at, for $0 \leq x \leq 2\pi$, $0 \leq y \leq 2\pi$.

   (d) Use `polyharm` to interpolate the coarse grid data onto the fine grid of interpolation points.

1.6. Create

   (a) a `surf`ace plot of the interpolated spline, showing the data points;

   (b) a `contour` plot of the absolute value of the difference between the test function $g(x, y)$ and the interpolant.

   As always, make sure you properly label your plots.

1.7. Based on your error plot from the previous question, suggest a reason why the errors might be worst in the region of the plot where they are.

2 Interpolation in meteorology

I'm very keen on numerical weather prediction, and playing with data from the Bureau of Meterology (BoM). (In fact, weather forecasting is a promising career for a young numerical methodologist, and a well-trodden path by numerous University of Adelaide mathematics graduates!) For the remainder of the assignment we'll use your `polyharm` function to interpolate some real weather data from the BoM.

You may remember that 23 August 2017 was a cold, rainy day in Adelaide. I know it was, because I went and got the data from the BoM website, from most of these observing stations:

I preprocessed the data for you into the file `BOM_20170823_1300_1500.zip`. It contains half-hourly data from 15 stations between 1 and 3pm on 23 August.

Each file is a matrix containing the following columns (in order): station latitude, station longitude, air gust speed (kmh), air temperature (C), dew point, barometric pressure (HPa), rainfall (mm), relative humidity, wind speed (kmh)

Each row in the matrix represents measurements from each of the following stations (in order): Adelaide Airport, Adelaide (Kent Town), Adelaide (West Terrace), Edinburgh, Hindmarsh Island, Kuitpo, Mount Crawford, Mount Lofty, Noarlunga, Nuriootpa, Pallamanna, Parafield, Parawa West, Roseworthy, Strathalbyn.

I've also included a file `sa_coast_lonlat.dat` containing the longitude (column 1) and latitude (column 2) coordinates of the SA coastline.

2.1. Write a script called `plotweather.m` to use `polyharm` to make a contour plot of rainfall at 1pm. Add the station locations and SA coastline, and set the `xlim` and `ylim` of your plot to show a similar region as the map above. Make sure you include a `colorbar` for your contours.

Look at the naming code of the files to figure out which is which – think about dates and times.

2.2. What is the minimum rainfall suggested by your interpolation? Comment on the limitations of this interpolation scheme in the context of this value.

2.3. Make another 3 plots of rainfall at 1:30pm, 2pm and 2:30pm. Can you determine which direction the storm is travelling in?

2.4. Think about the contents of the matrix `A` in your `polyharm` function, which you currently recalculate each time you interpolate a new dataset. Suggest one way in which you might improve the efficiency of your code, if you were going to create a longer animation of rainfall interpolated from a static observing network such as this one.

Bonus material (for your interest only)

For fun I've also uploaded `BOM_bonus_all_data.zip`, containing roughly 72 hours worth of these data files, in case you want to play with the data further. Try plotting some of the other weather variables! What goes wrong when you try to map barometric pressure? What do you suppose this means?

If you've gotten to this point you'll hopefully have some sense that weather prediction is hard, and that interpolation has some issues! It's difficult to tell what's happening in your animations, and you need to be a bit skeptical about some of the results your method produces.

One way that forecasters improve on both of these issues is to use the *dynamical physical information* that is encoded in a numerical weather prediction model. These models ensure that information from a map at one time is propagated to the next time, and that values that are interpolated far from data points are physically reasonable. This process of blending dynamical models with observational data is called data assimilation. It's a fascinating research field, which you should study if you ever get the chance!

3 Assignment requirements

3.1. You must submit the final version of your function `simp` to Cody Coursework, as follows:

(a) Login to Cody Coursework.

(b) Click on "Assignment 3" and then on "Polyharmonic spline".

(c) To submit your code, click on "Solve" and copy and paste your function `polyharm` into the text box provided.

(d) Test your function by clicking on "Test". When you are happy, click "Submit" to save your function. You will receive one mark for each test that is passed.

(e) You can revise and resubmit your function as often as you like, up until the deadline. Your best solution will be used to assign your execution mark.

(f) For general help, consult the Mathworks Cody Coursework documentation.

(g) You do not have to submit your scripts `testpolyharm` or `plotweather` to Cody Coursework.

3.2. In addition, you must also submit a written report answering each item in Sections 1 and 2 as a **single PDF file** via Canvas.

Where relevant, this will include:

(a) printouts of relevant scripts and functions (including code submitted to Cody Coursework),

(b) printouts of numeric output and/or graphs,

(c) any supporting analysis, and

(d) answers to any questions.

These answers must be professionally presented in a readable and informative manner. Graphs and tables must be properly labelled. All scripts and functions must be properly documented and and set out using the conventions established for this course. Haphazard and illegible work will not be marked.

3.3. You may discuss problems and thrash out the answers together but the work you submit must be your own and not copied from someone else.