Project: In the octopus' garden

We've done a little one-dimensional interpolation, and we will do more later in the course. We won't be going beyond one dimension, because things can get a little tricky out there. But there is a method known as **radial basis functions** (RBF) that is not hard to get started with. RBFs have applications in machine learning, computer graphics, and differential equations.

Let $\phi(r)$ be a *shape function* defined for $r \ge 0$. Popular choices include $\phi(r) = r^3$ and $\phi(r) = e^{-r^2}$. We'll limit ourselves to two dimensions, so let (x_i, y_i), for $i = 1, \dots, n$, be points in a plane region R. Each RBF has the form

$$u_i(x,y) = \phi\Big(\sqrt{(x-x_i)^2 + (y-y_i)^2}\Big). \tag{1}$$

The level curves of u_i are circles centered at (x_i, y_i), which is the reason for the adjective "radial."

Given a function $f(x,y)$ on the domain R, its RBF interpolant on the point set $\{(x_i, y_i)\}$ is

$$s(x,y) = \sum_{i=1}^{n} c_i u_i(x,y), \tag{2}$$

where the c_i values are chosen so that

$$s(x_i, y_i) = f(x_i, y_i), \qquad i = 1, \dots, n. \tag{3}$$

These conditions take the form of a linear system:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{z}, \qquad A_{ij} = u_j(x_i, y_i), \quad z_i = f(x_i, y_i). \tag{4}$$

Once $\boldsymbol{c}$ has been obtained, the interpolant can be computed anywhere using (2).

One of the earliest applications of RBFs was to draw elevation contour lines on maps. Imagine you are given height data z_i at each location instead of a function $f(x,y)$. You can replace the constraint (3) with $s(x_i, y_i) = z_i$ for all i, and then s is defined throughout the plane as an approximation to the true elevation.

Project assignment

You should submit only M-files. Four of them are scripts required to produce output as described in detail below. **Each of these scripts should start with the lines**

```
close all, clear all, rng(foo)
```

where `foo` is a 4-digit integer made up by your team. (The call to `rng` makes your "random" results reproducible.) You may also use any function built into core MATLAB (not any toolboxes), any of the book functions, and any of your own helper functions. In your submission, **include all other files needed, including those from the book.** You can locate all dependencies for file `p1.m`, for example, with

```
matlab.codetools.requiredFilesAndProducts('p1')
```

If a script does not run successfully, you may receive no credit on that part of the assignment. If any code is found to be plagiarized from the internet or another group, you may receive a zero on the entire assignment.

script `p1.m`: This part is in one dimension, with $x_i = (i-1)/10$ and $y_i = 0$ for $i = 1, \ldots, 11$, $\phi(r) = r^3$, and $f(x, y) = 1/(20x^2 + 1)$. On one graph, plot the basis functions $u_2(x, 0)$ and $u_7(x, 0)$ for $0 \le x \le 1$. Create and solve the linear system (4). Evaluate and print out the values of $s(x, 0)$ and $f(x, 0) - s(x, 0)$ at $x = 0.25$, $x = 0.5$, and $x = 0.75$. On a new graph, plot $f(x, 0)$ and $s(x, 0)$ together.

script `p2.m`: Now you go into two dimensions, for the function

$$f(x, y) = e^{-x^2+2xy+2y-y^2}, \quad 0 \le x \le 1, \, 0 \le y \le 1, \tag{5}$$

and with random points selected by

```
x = rand(n,1);  y = rand(n,1);
```

for the values $n = 100, 200, \ldots, 1000$. Make a table showing the values of n, the error $f - s$ at the points $(0.5, 0.5)$, $(0, 0.6)$, and $(1, 1)$, and $\kappa(\boldsymbol{A})$.

script `p3.m`: Same as for p2, except using $\phi(r) = e^{-100r^2}$.

script `p4.m`: MATLAB ships with a data set of elevation readings of an undersea mountain. You can load and see the data using

```
load seamount
plot3(x,y,z,'o')
```

(You do not need to include this plot in your submitted script.) Using the method described in the introduction, find an interpolating $s(x, y)$ using whatever you like for $\phi(r)$. Then make a contour plot of the elevation as follows:

```
xc = linspace(min(x),max(x),100);
yc = linspace(min(y),max(y),100);
[X,Y] = meshgrid(xc,yc);
% ... you write some code here to define Z ...
contourf(X,Y,Z,32), axis equal
```

You need to compute `Z` such that its (i, j) element is the value of s at `(X(i,j),Y(i,j))`.