Numerical Methods 35006
Computer Lab 4: multi-dimensional minimisation

The aim of this lab is to guide you through the steps of creating a Powell Direction-Set multi-dimensional search procedure. It relies on successful implementation of the bracketing and golden searched from last week.

1. Surfaces in 2D can be imaged using both color and contour plots. To enable this it is helpful to use *np.meshgrid*. This takes two arrays generated by `linspace` and creates an array of 2D coordinates.

   a) Use `linspace` to generate an array of 5 `x` values and an array of 10 `y` values in the ranges $x \in [0, 2]$ and $y \in [0, 5]$. Generate a grid by typing

   ```
   X,Y = np.meshgrid(x,y)
   ```

   b) Now generate a 100×100 grid of points (`X,Y`) in the range $x \in [-2, 2]$ and $y \in [-2, 2]$.

   c) Use the `pyplot` plotting command `pcolor`

   ```
   plt.pcolor(X,Y,np.exp(-X**2 - Y**2))
   ```

   to make sure your plotting is working properly.

2. Use the pyplot command `plt.contour` to plot a contour map of the function

   $$f(x, y) = (x - 1)^2 + y^2 \qquad (1)$$

   in the range $x \in [-2, 2], \quad y \in [-1, 1]$.

3. Define the function `f(x,y)` as in Q2 above. Import the 3D Axes module `Axes3D` to plot this function as a surface plot:

   ```
   from mpl_toolkits.mplot3d import Axes3D
   fig = plt.figure()
   ax = fig.add_subplot(111, projection='3d')
   surf = ax.plot_surface(X, Y, f(X,Y))
   ```

4. Building on your code in Q3, create code that, given a vector starting point $\mathbf{xn} = [-2, -1]$ and a direction $\mathbf{pn} = [1, 0]$, plots 20 points along a line starting at $\mathbf{xn}$ and going in the direction $\mathbf{pn}$, with a stepsize of $h = 0.2$ between each point, according to

   $$\mathbf{x} = \mathbf{x}_n + t\mathbf{p}_n \qquad (2)$$

5. Building on your code in Q4, create a function `fline(t)`, that returns the value of f for a particular value of t, according to Eq. (2). Bracket the minima along this curve using your search routine from last week, then find the minimum to within a tolerance of 10^{-5} using a golden section search.

6. Add two loops to your code: one to search over a set of orthogonal directions, which are stored in a matrix `pset`. The other loop should iterate these two searches until a minimum is found (or the minimum is lost).

7. Finally, update the set of coordinate directions at the end of each iteration to create a powell search algorithm, as given in the lecture slides. Test this with a few different starting points on the function given in Eq. (1).

8. Create a function called `powellsearch(f,x0)`, which finds the minimum of a 2D function `f` from a starting point `x0`. Incorporate this routine into your `mysearch.py` module.