Solving systems of nonlinear equations
We would like to find a solution (or at least an approximate solution) to a system of nonlinear equations. For example

$$\begin{aligned} x_1^2 - x_2^2 &= 1, \\ x_1 + x_2 - x_1 x_2 &= 1. \end{aligned}$$

This system is equivalent to the system

$$\begin{aligned} x_1^2 - x_2^2 - 1 &= 0, \\ x_1 + x_2 - x_1 x_2 - 1 &= 0. \end{aligned}$$

If we set $\mathbf{F}(x_1, x_2) = [x_1^2 - x_2^2 - 1, x_1 + x_2 - x_1 x_2 - 1]^\mathsf{T}$, we can rewrite this system as

$$\mathbf{F}(\mathbf{x}) = \mathbf{0},$$

where $\mathbf{x} = [x_1, x_2]^\mathsf{T}$. In other words, we are looking for zeros of a vector–valued function of several variables.

Let us formulate a more general problem. Let $U \subseteq \mathbb{R}^n$ be the domain of the function $\mathbf{F}$, $\mathbf{F}\colon U \to \mathbb{R}^n$. The idea is to generalise Newton's method for finding approximations to zeros of a function of a single variable, which suggests that for $f : D \to \mathbb{R}$ we pick an initial guess $x^{(0)} \in D$ and then iteratively improve the accuracy of the solution using the recursive formula

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

For a vector-valued function $\mathbf{F}(\mathbf{x}) = [F_1(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n)]^\mathsf{T}$ we must substitute the derivative f' with the *Jacobi matrix* of the function $\mathbf{F}$:

$$J\mathbf{F} = \left[\frac{\partial F_i}{\partial x_j}\right]_{i,j}.$$

One step of *Newton's iteration* is then written as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (J\mathbf{F})^{-1}\mathbf{F}(\mathbf{x}^{(k)}).$$

1. Find the approximate solution $[x_1, x_2]^\mathsf{T}$ to the system

   $$\begin{aligned} x_1^2 - x_2^2 &= 1, \\ x_1 + x_2 - x_1 x_2 &= 1, \end{aligned}$$

   which is accurate to 10 decimal places.
   Write an octave function `x = newton(F, JF, x0, tol, maxit)` which performs Newton's iteration with the initial approximation `x0` for the function `F` and Jacobi matrix function `JF`. We use `maxit` to limit the maximum number of allowed iterations (in order to avoid a potentially infinite loop), and we use `tol` to prescribe the desired accuracy.

2. Let f be a function of two variables, x and y. We would like to find a sequence of equidistant points (according to the Euclidean distance) on the curve defined by

$$f(x,y) = 0.$$

Denote the given distance between two successive points by δ. Assume that the first point (x_0, y_0) is given. The next point, say (x,y), is determined by the conditions that the distance from (x_0, y_0) equals δ, and that it lies on the curve $f(x,y) = 0$. This means that (x,y) should solve the system of equations

$$\begin{aligned} f(x,y) &= 0, \\ (x-x_0)^2 + (y-y_0)^2 &= \delta^2. \end{aligned}$$

The next point is therefore obtained as a solution to this system, and we denote this solution by (x_1, y_1). We repeat the procedure to obtain the next point (x_2, y_2) and so on.

Write an octave function `K = krivulja(f, gradf, T0, delta, n)` that returns the $2\times n$ matrix `K` containing the coordinates of the sequence of points on $f(x,y) = 0$, with mutual distances δ. (`f` is the given function of two variables and `gradf` is its gradient, `T0` is the initial point).