Université de Picardie Jules Verne
Cours de l'Ecole Doctorale/Doctoral School's Lectures
Calculs Numériques/Numerical Compuations
2019-2020

Solution of nonlinear systems of equations with Scilab

1 The scalar case

1.1 Polynomials

Let us begin with the problem of finding roots of a polynomial (this is a very difficult problem for deep reasons when the degree is larger or equal to 5).
The command `roots` allows to compute numerically the roots of a polynomial, as described in the following simple example

```
// a real polynomial defined by its roots
p = poly([1 2 3],"x")
roots(p)
// We give directly the polynomial's coefficients
p = [3 2 1] roots(p) // Here a polynomial with complex roots
p=poly([0,10,1+%i,1-%i],'x');
roots(p)
// roots of the characteristic polynomial of a matrix A=rand(3,3);
p = poly(A,'x')
roots(p)
spec(A)
```

To be convinced of the difficulty of finding the roots of a polynomial, we consider the famous Wilkinson example:

```
n=10;
p=poly(1:n,'x');
r=roots(p)
// Now represent the roots in the complex plane
plot(real(r),imag(r),'*')
```

up to $n = 20$ the roots computed are very closed numerically to the ones expected. The difficulty appears for larger values of n. We can illustrate this purpose by generating an animation

```
for n=5:5:40
p=poly(1:n,'x');
r=roots(p)
// Now represent the roots in the complex plane
plot(real(r),imag(r),'*')
drawnow
//pause in microsec
xpause(1000000)
end
```

A numerical method is used by Scilab (by default, the RPOLY method of Jenkins-Traub we will not discuss here).

Another way to compute the roots is to build a matrix M whose the characteristic polynomial is p and then to compute the eigenvalues of M, say the roots of p. M is said to be the companion matrix of p and be built b in Scilab as `M=companion(p)`.

1.2 Fixed point

We recall that a fixed point of a function $g(x)$ is a real x that is unchanged by g so that

$$g(x) = x$$

A classical method to compute a fixed point of $g$is the so-called Picard method:

$$x^{(k+1)} = g(x^{(k)})$$

And if $x^{(k)}$ converges, the limit must be a fixed point of g. Of course the convergence is not always guaranted, assumption to g have to be made.

Finding a root of f is equivalent to find a fixed point of $g(x) = x + \alpha f(x)$, where $\alpha \neq 0$.

Exercice: write a code in Scilab to compute the fixed point of $g(x) = exp(-x)$ by Picard iterates, for $x \in [0, 1]$.

1.3 Newton-Like methods

Let f be a regular function and x^* an isolated root of f. By Taylor's formula we can write

$$f(x^*) = 0 = f(x + (x^* - x)) = f(x) + f'(x)(x^* - x) + \mathcal{O}((x^* - x)^2)$$

hence,

$$x^* \simeq x - f(x)/f'(x).$$

So, in a neigborhood of x^*, $x - f(x)/f'(x)$ provides a better approximation of x^* than x. The idea of Newton's method is then to loop the approximation as

$$x^{(k+1)} = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$$

of course $f'(x^{(k)}$ must be different to 0. A way to ensure this condition is to assume that $f'(x^*) \neq 0$, so by continuity of f' this holds in a sufficiently neigborhood of x^*.

Exercice: write a code in Scilab to compute the root of $x - exp(-x)$ by Newton's method for $x \in [0, 1]$.

2 Systems of equations

2.1 Fixed point

Here we have $F(x) = (F_1(x), \cdots, F_n(x))^T$ and $x = (x_1, x_2, \cdots, x_n)^T$. The Picard iterates write exactly as in the scalar case, namely,

$$x^{(k+1)} = F(x^{(k)})$$

Exercice: write a code in Scilab to compute the root of

$$Ax + x * Dx = b$$

with

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \text{ and } D = \frac{1}{2h}\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{pmatrix}.$$

where we have set $h = \frac{1}{n+1}$. We set $b = (\sin(\pi h), \sin(2\pi h), \cdots, \sin(n\pi h))^T$.

2.2 The Newton-Raphson method

We derive the Newton method for finding a root of a function F with values in $\mathbb{R}^n$ ina similar way to the scalar case ($n = 1$).
By Taylor's formula we can write

$$F(x^*) = 0 = F(x + (x^* - x)) = f(x) + DF(x)(x^* - x) + \mathcal{O}((x^* - x)^2)$$

so

$$x^* \simeq x - (DF(x))^{-1}F(x)$$

Here $DF(x)$ is the Jacobian matrix of F at x and is defined by

$$DF(x)_{ij} = \frac{\partial F_i(x)}{\partial x_j}$$

So the Newton-Raphson method is obtained by cycling the approximation as

$$x^{(k+1)} = x^{(k)} - (DF(x^{(k)}))^{-1}F(x^{(k)})$$

Exercice: write a code in Scilab to compute the root of $F(x) = Ax + x * Dx - b$, with A, D and b as above.