3.5: The Jacobian

Newton's method requires first derivatives.

We recall the definition (see calculus in several variables)

Definition. [4.3] *Let $f : D \subset \mathbb{R}^n \to \mathbb{R}^n$, $x \in D$.*
The $n \times n$ matrix

$$J_f(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ & & \dots & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

is called the Jacobian or functional matrix of g at x.

3.6: Jacobian: Example

$$F(x) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 1 \\ 5x_1^2 + 21x_2^2 - 9 \end{pmatrix} = 0$$

Then

$$J_F(x) = \begin{pmatrix} 2x_1 & 2x_2 \\ 10x_1 & 42x_2 \end{pmatrix}$$

3.7: Jacobian: Numerical Computation

Often the Jacobian is not analytically available and it has to be computed numerically.
It can be computed column wise by finite differences

```
function [J]=jacobian(func,x)
% computes the Jacobian of a function
n=length(x);
fx=feval(func,x);
eps=1.e-8;  % could be made better
xperturb=x;
for i=1:n
   xperturb(i)=xperturb(i)+eps;
   J(:,i)=(feval(func,xperturb)-fx)/eps;
   xperturb(i)=x(i);
end;
```

3.8: Newton's method in $\mathbb{R}^n$

Newton's method for systems of equations is a direct generalization of the scalar case:

Definition. [4.5] *The recursion*

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

with $J_F(x)$ being the Jacobian of F is called Newton's method.

Note, in order to avoid confusion with the i-th component of a vector, we set now the iteration counter as a superscript $x^{(i)}$ and no longer as a subscript x_i.

3.9: Newton's method: Implementation remarks

For implementing Newton's method no matrix inverse is computed (this is to expansive), we solve instead linear equation systems:

$$J_F(x^{(k)})\Delta x = -F(x^{(k)})$$

and compute the next iterate by adding the Newton increment δx:

$$x^{(k+1)} = x^{(k)} + \Delta x$$

Solving linear systems is done in MATLAB with the \-command - not with the command `inv` !!!