A. jacobnnf

Calculates the Jacobian matrix of an n-dimensional function of n variables using forward differences. *jacobnnf* computes first order difference quotient approximations

$$J_{i,j} = (f_i(x_1,...,x_{j-1},x_j+\delta_i,x_{j+1},...,x_n)-f_i(x_1,...,x_{j-1},x_j,x_{j+1},...,x_n))/\delta_i \quad (i,j=1,...,n)$$

to the partial derivatives $J_{i,j} = \partial f_i(x)/\partial x_j$ of the components of the function $f(x)$ $(f,x\epsilon R^n)$.

Function Parameters:

void jacobnnf (*n,x,f,jac,di,funct*)

n: int;
entry: the number of independent variables and the dimension of the function;

x: float *x[1:n]*;
entry: the point at which the Jacobian has to be calculated;

f: float *f[1:n]*;
entry: the values of the function components at the point given in array *x*;

jac: float *jac[1:n,1:n]*;
exit: the Jacobian matrix in such a way that the partial derivative of *f[i]* with respect to *x[j]* is given in *jac[i,j]*, *i,j=1,...,n*;

di: float *(*di)(i), int i*;
entry: the partial derivatives to *x[i]* are approximated by forward differences, using an increment in the *i*-th variable equal to the value of *di*, *i=1,...,n*;

funct: void *(*funct)(n,x,f)*;
entry: the meaning of the parameters of the function *funct* is as follows:
n: the number of independent variables of the function *f*;
x: the independent variables are given in *x[1:n]*;
f: after a call of *funct* the function components should be given in *f[1:n]*.

```
void jacobnnf(int n, float x[], float f[], float **jac,
              float (*di)(int), void (*funct)(int, float[], float[]))
{
    float *allocate_real_vector(int, int);
    void free_real_vector(float *, int);
    int i,j;
    float step,aid,*f1;

    f1=allocate_real_vector(1,n);
    for (i=1; i<=n; i++) {
        step=(*di)(i);
        aid=x[i];
        x[i]=aid+step;
        step=1.0/step;
        (*funct)(n,x,f1);
        for (j=1; j<=n; j++) jac[j][i]=(f1[j]-f[j])*step;
```