Parameters definition

NP : Number of Parameters should be fitted
β : Vector of independent variable appearing in the function
N_f : Number of parallel functions
F : Vector of functions which must be fitted to experimental data
β_0 : First guess of parameters
N_file : number of file, must be same as Number of parallel functions
Name_files : Vector Names of experimental data files

datafit prints the number of iterations, the *Std Deviation of Residuals* (Goodness of Fit) statistic, the supplementary information required, and returns *f*.

```
restart;
with(plottools) : with(LinearAlgebra) : with(plots) : with(StringTools) : with(stats) :
    with(CurveFitting) : with(VectorCalculus) :
with(linalg) : with(ArrayTools) :

nlf := proc(NP, Betaa, N_f, F, B0, N_file, Name_files)
 description "Levenberg-Marquardt algorithm to calculate material parameters";
 local i, ii, ij, ik, kk, kkk, c_;
local A, X_exp, Y_exp, Jac;
local Total_n_points, Ntot;
local  LSP, Jac_total, Jac_total0;
 local Res, Res0, Res1, Betaa1, Delta1, Betaa0;
 local limit_iteration, Lambda_0, v;
local AA1, BB1, lc, LSP_;

 Betaa0 := B0;

 #=========================
 # read experimental data
 #=========================
 Total_n_points := 0 :
for i from 1 to N_file do

      A := Matrix(readdata(Name_files[i], [float, float])) :
      Ntot[i] := Size(A, 1) :

      for ii from 1 to Ntot[i] do
            X_exp[ii + Total_n_points] := A[ii, 1];
            Y_exp[ii + Total_n_points] := A[ii, 2];
      end do:
       Total_n_points := Total_n_points + Ntot[i] :
```

```
 end do:
unassign('i','ii') :

#=======================
# Caculate Jacobian Vector of functions
 #=======================
for ii from 1 to N_f do
      for ij from 1 to NP do
            Jac[ii, ij] := diff(F[ii](x, Betaa[i] $ i = 1 ..NP), Betaa[ij]);
       end do:
 end do:
unassign('ii','ij') :


#=============================================
#Calculation of  J and Res= y_exp-f(x,beta). x=At each data point.
# J is derived as a vector and can be calculatedusing numerical differentiaition.
#=============================================

 c_, LSP := 0, 0 :
Jac_total := Matrix(Total_n_points, NP);
Jac_total0 := Matrix(Total_n_points, NP);
Res := Matrix(Total_n_points, 1);

 for ii from 1 to N_f do
       for ij from 1 to Ntot[ii] do
             c_ := c_ + 1;
             for ik from 1 to NP do
                   Jac_total[c_, ik] := eval(Jac[ii, ik], x = X_exp[c_]);
             end do;
             Res[c_, 1] := Y_exp[c_]-F[ii](X_exp[c_], Betaa0[i] $ i = 1 ..NP) :
             LSP := LSP + Res[c_, 1]^2;
       end do;
end do;
unassign('ii','ij','ik','i') :

 #================================
#Set initial Levenberg Marquardt Damping parameter
 #================================
 v := 2.10 :
 Lambda_0 := 3/v^6 :
 limit_iteration := 50 :

 Betaa1 := [0 $ i = 1 ..NP] :
Delta1 := Matrix(Np, 1) :
Res1 := Matrix(Total_n_points, 2);
```

```
#==========
# Begin iterations.
#==========
 for kkk from 1 to limit_iteration do
       Jac_total0 := eval(Jac_total, Betaa = Betaa0 ) :

        for kk from 1 to 2 do
                AA1 := transpose(Jac_total0) . Jac_total0 + Lambda_0 / v^(kk-1)
    * IdentityMatrix(NP) :
                BB1 := transpose(Jac_total0) . Res:
                Delta1 := 1 / AA1 . BB1:
                for i from 1 to NP do
                        Betaa1[kk][i] := Betaa0[i] + Delta1[i, 1];
                end do:
                 unassign('i') :
                  c_, LSP_[kk] := 0, 0 :
                  for ii from 1 to N_f do
                          for ij from 1 to Ntot[ii] do
                                  c_ := c_ + 1 :
                                  Res1[c_, kk] := Y_exp[c_]-F[ii](X_exp[c_], Betaa1[kk][i]
    $ i = 1..NP) :
                                  LSP_[kk] := LSP_[kk] + Res1[c_, kk]^2 :
                          end do:
                  end do:
        end do:
         unassign('kk','ij','ik','c_','i') :


    #=============================================================
    ===

    #Choosing between shortening and increasing the steps using lambda1 and
    lambda2

    #=============================================================
    ===
         if min(LSP_[1], LSP_[2]) < LSP then

                  if LSP_[1] ≤ LSP_[2] then
                           for i from 1 to NP do
                                   Betaa0[i] := Betaa1[1][i];
                           end do:
                           Res := convert(Res1[.., 1], Matrix) :
                           LSP := LSP_[1] :
                  else
                          for i from 1 to NP do
                                   Betaa0[i] := Betaa1[2][i];
                           end do:
                          Res := convert(Res1[.., 2], Matrix) :
```

```
                    LSP := LSP_[2]:
                end if:
                unassign('ii','ij','ik','c_','i'):

        else
                if LSP_[1] < LSP_[2] then
                        Lambda_0 := Lambda_0*v^2:
                else
                        Lambda_0 := Lambda_0/v^2:
                end if:
        end if:


end do:
RETURN(Betaa0);

end proc:
```

Example

```
> NP := 3: Betaa := [a[i] $ i = 1..NP];
  F[1] := unapply(sum(Betaa[i]*x^(i-1), i = 1..NP), x, Betaa[i] $ i = 1
      ..NP);
  F[1] := unapply(Betaa[1]·x + Betaa[2]·x^2 + Betaa[3]·sin(x), x, Betaa[i]$ i
      = 1..NP);
  N_f := 1;
  #Betaa is the array that stores the values of the material parameters
  #F[1] is an example function for three material parameters
  Betaa0 := [0 $ i = 1..NP];
  N_file := 1;
  Name_files := ["exp-1st-unload-diani.txt"];
  Aa := nlf(NP, Betaa, N_f, F, Betaa0, N_file, Name_files);
  A := Matrix(readdata(Name_files[1], [float, float])):
  pl[1] := plot(A, color = BLACK, caption = "E fitted vs experiment", title
      = "Stress-Strain Curve for NpPEG", labels
      = ["Strain (Length/Original Length)", "Stress (MPa)"]):
  pl[2] := plot(F[1](x, Aa[i] $ i = 1..NP), x = 1..2, color = BLUE, caption
      = "fitted"):
  display(pl[1], pl[2]);

      #the blue line is the curve generated by the algorithm, and the black line
      is the curve from the experimental data
```

$$Betaa := [a_1, a_2, a_3]$$

$$F_1 := (x, a\_1, a\_2, a\_3) \rightarrow a\_3\, x^2 + a\_2\, x + a\_1$$

$$F_1 := (x, a\_1, a\_2, a\_3) \rightarrow a\_2\, x^2 + a\_1\, x + a\_3 \sin(x)$$

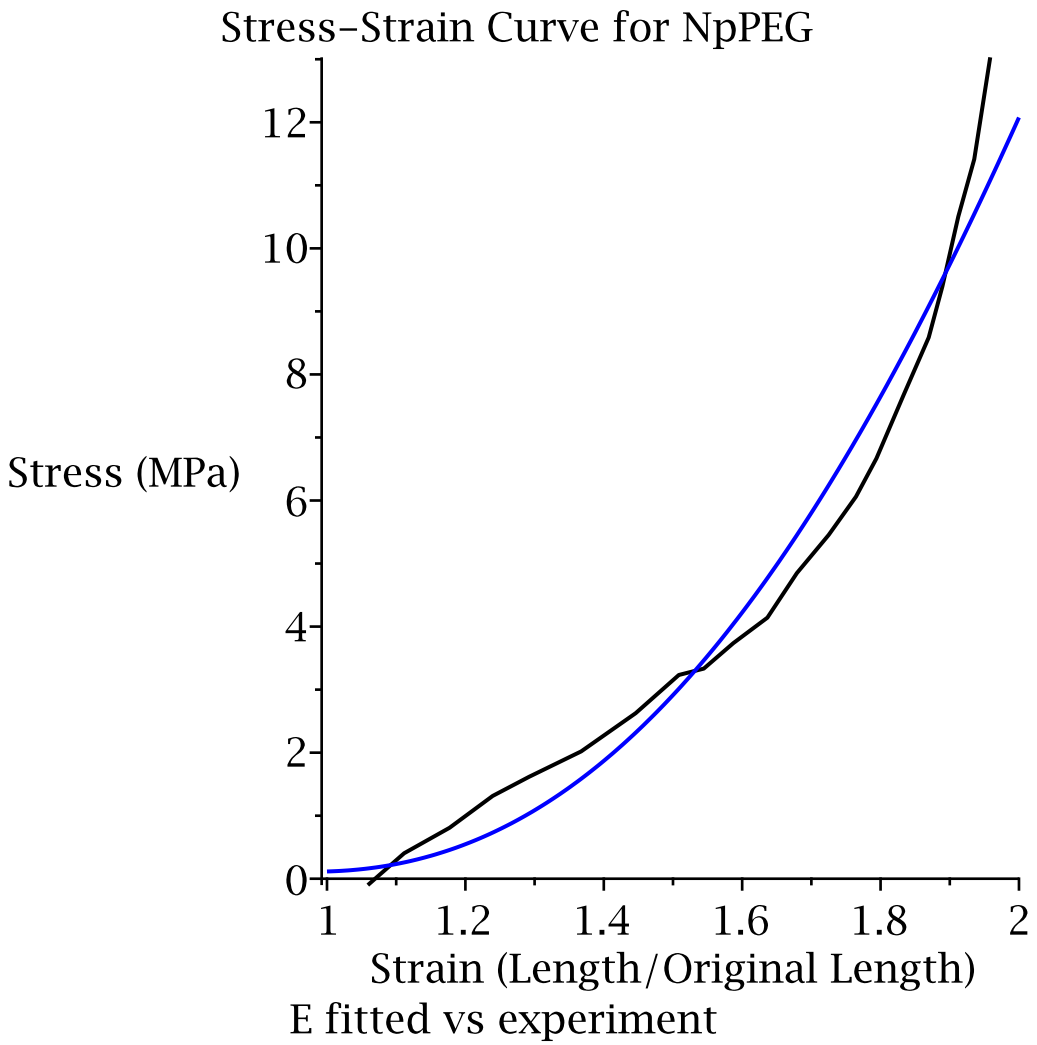
$$N\_f := 1$$

$$Betaa0 := [0, 0, 0]$$

$$N\_file := 1$$

$$Name\_files := ["exp-1st-unload-diani.txt"]$$

$$Aa := [76.4863003235468, -19.9937431231326, -66.9985457439815]$$



>