Hardening of Aluminium Alloy AA 7075 T 7351

Univ.-Prof. Dr.-Ing. habil. Josef *BETTEN*
RWTH Aachen University
Mathematical Models in Materials Science and Continuum Mechanics
Augustinerbach 4-20
D-52056 A a c h e n , Germany

<betten@mmw.rwth-aachen.de>

Some tensile tests on *aluminium alloy* AA 7075 T 7351 at room temperature furnish the following experimental data.

```
> restart:
> data:=[0,350],[0.32,472],[1.27,501.2],[2.23,517],
  [3.2,530],[4.17,541];
```

$$data := [0, 350], [0.32, 472], [1.27, 501.2], [2.23, 517], [3.2, 530], [4.17, 541]$$

This hardening behaviour can be expressed by the simple formula

```
> sigma=sigma[F]+k*epsilon[pl]^n;
```

$$\sigma = \sigma_F + k\, \varepsilon_{pl}^{\,n}$$

where the yield stress can be read from the above data:

```
> sigma[F]:=350*MPa;
```

$$\sigma_F := 350\ MPa$$

The hardening factor k and the hardening exponent n have been determined based upon the experimental data by using the *nonlinear MARQUARDT-LEVENBERG-algorithm* [*BETTEN, J.:* Creep Mechanics, Third Edition, 2008, Springer-Verlag, Berlin / Heidelberg]:

```
> k:=146.91*MPa;    n:=0.1758;
```

$$k := 146.91\ MPa$$

$$n := 0.1758$$

Instead of the *MARQUARDT-LEVENBERG-algorithm* one can also effectively apply the *least squares* method in order to determine the hardening parameters, since the yield stress in the above relation is known. Thus, we can transform the nonlinear problem to a *linear regression* by considering the logarithm of the hardening relation:

```
> restart:
> ln(sigma-sigma[F])=ln(k)+n*ln(epsilon[pl]);    Y:=K+n*X;
```

$$\ln(\sigma - \sigma_F) = \ln(k) + n \ln(\varepsilon_{pl})$$

$$Y := K + n\,X$$

The function **leastsquare** of the MAPLE package *stats* [*fit,leastsquare*] fits a curve to the given data, as shown in the following MAPLE program.

```
> restart:
> data:=[0,350],[0.32,472],[1.27,501.2],
  [2.23,517],[3.2,530],[4.17,541];
```

$$data := [0, 350], [0.32, 472], [1.27, 501.2], [2.23, 517], [3.2, 530], [4.17, 541]$$

```
> for i in [472,501.2,517,530,541] do Y[i]:=ln(i-350.) od;
```

$$Y_{472} := 4.804021045$$

$$Y_{501.2} := 5.018603464$$

$$Y_{517} := 5.117993812$$

$$Y_{530} := 5.192956851$$

$$Y_{541} := 5.252273428$$

```
> with(stats):
> fit[leastsquare[[x,y],y=K+n*ln(x),{K,n}]]
  ([[0.32,1.27,2.23,3.2,4.17],[4.804,5.019,5.118,5.193,5.252]]);
```

$$y = 4.991336579 + 0.1722331357\,\ln(x)$$

```
> k:=(exp(4.991336579))*MPa;   n:=0.1722;
```

$$k := 147.1329469\ MPa$$

$$n := 0.1722$$

Compare these parameters with the coresponding values from the *nonlinear algorithm.*

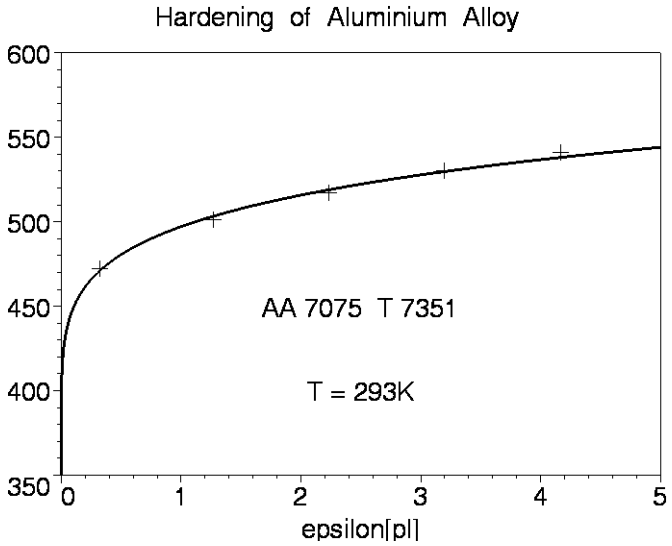
```
> macro(S=sigma, E=epsilon):
> S(E[pl]):=(350+(k/MPa)*E[pl]^n)*MPa;
```

$$\sigma(\varepsilon_{pl}) := (350 + 147.1329469\ \varepsilon_{pl}^{0.1722})\ MPa$$

```
> yield_stress:=S[F]=350*MPa;
```

$$yield\_stress := \sigma_F = 350\ MPa$$

```
> alias(H=Heaviside,th=thickness,co=color):
> p[1]:=plot(S(E[pl])/MPa,E[pl]=0..5,350..600,th=3,co=black):
> p[2]:=plot([data],E[pl]=0..5,style=point,symbol=cross,
  symbolsize=35,co=black):
> p[3]:=plot({600,600*H(E[pl]-5)},
  E[pl]=0..5.001,350..600,co=black,
  title="Hardening  of  Aluminium  Alloy"):
> p[4]:=plots[textplot]({[2.5,450,`AA 7075  T 7351`],
  [2.5,400,`T = 293K`]}):
> plots[display](seq(p[k],k=1..4));
```



Dimensionless Representation

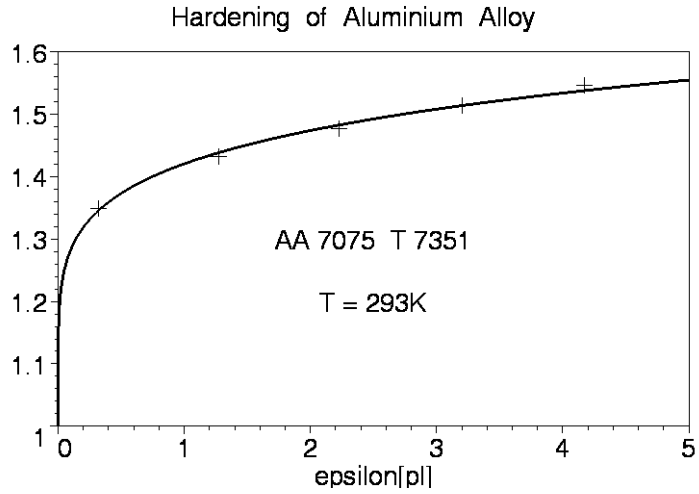
```
> restart:
> macro(S=sigma,E=epsilon):
> s(E[pl]):=S(E[pl])/S[F]=1+0.42038*epsilon[pl]^0.1722;
  # plastic strains in percent
```

$$s(\varepsilon_{pl}) := \frac{\sigma(\varepsilon_{pl})}{\sigma_F} = 1 + 0.42038\ \varepsilon_{pl}^{0.1722}$$

```
> DATA:=[0,1],[0.32,472./350],[1.27,501.2/350],[2.23,517./350],
  [3.2,530./350],[4.17,541./350]:
> Data:=[0,1],[0.32,1.35],[1.27,1.43],[2.23,1.48],[3.2,1.52],
  [4.17,1.55];
```

$$Data := [0, 1], [0.32, 1.35], [1.27, 1.43], [2.23, 1.48], [3.2, 1.52], [4.17, 1.55]$$

```
> alias(H=Heaviside,th=thickness,co=color):
> p[1]:=plot(rhs(s(E[pl])),E[pl]=0..5,1..1.6,th=3,co=black):
> p[2]:=plot([DATA],E[pl]=0..5,style=point,symbol=cross,
  symbolsize=35,co=black):
> p[3]:=plot({1.6,1.6*H(E[pl]-5)},E[pl]=0..5.01,1..1.6,co=black,
  title="Hardening of Aluminium Alloy"):
> p[4]:=plots[textplot]({[2.5,1.3,`AA 7075 T 7351`],
  [2.5,1.2,`T = 293K`]}):
> plots[display](seq(p[k],k=1..4));
```



Distance Vector between the leastaquare approximation and the experimental data:

```
> for i from 1 to 6 do v[i]:=evalf(subs(E[pl]=DATA[i][1],
  rhs(s(E[pl]))-DATA[i][2])) od;
```

$$v_1 := 0.$$

$$v_2 := -0.0030866869$$

$$v_3 := 0.0060433045$$

$$v_4 := 0.0054936214$$

$$v_5 := -0.0006811526$$

$$v_6 := -0.0081511553$$

```
> with(linalg):
> V:=vector([seq(v[i],i=1..6)]);
```

$$V := [0., -0.0030866869, 0.0060433045, 0.0054936214, -0.0006811526, -0.0081511553]$$

L-two and L[p] error norms:

```
> L[2]:=(1/sqrt(number_of_points))*Norm(V,2)=
  evalf((1/sqrt(6))*norm(V,2),4);
```

$$L_2 := \frac{\mathrm{Norm}(V, 2)}{\sqrt{number\_of\_points}} = 0.004881$$

```
> L[infinity]:=
  (1/(number_of_points)^(1/infinity))*Norm(V,infinity)=
  evalf((1/(number_of_points)^(1/infinity))*norm(V,infinity),4);
```

$$L_\infty := \mathrm{Norm}(V, \infty) = 0.008151$$

```
> L[infinity]:=Max(abs(v[1..6]))=evalf(abs(v[6]),4);
```

$$L_\infty := \mathrm{Max}(|v_{1..6}|) = 0.008151$$

```
> L[p]:=(1/(number_of_points)^(1/p))*Norm(V,p);
```

$$L_p := \frac{\mathrm{Norm}(V, p)}{number\_of\_points^{\left(\frac{1}{p}\right)}}$$

```
> for i from 1 by 5 to 31 do N[i]:=evalf(norm(V,i),4) od;
```

$$N_1 := 0.02345$$

$$N_6 := 0.008474$$

$$N_{11} := 0.008187$$

$$N_{16} := 0.008156$$

$$N_{21} := 0.008152$$

$$N_{26} := 0.008151$$

$$N_{31} := 0.008151$$

```
> for i from 1 by 1000 to 5001 do
  L[i]:=evalf((1/(6.)^(1/i))*norm(V,i),4) od;
```

$$L_1 := 0.003909$$

$$L_{1001} := 0.008135$$

$$L_{2001} := 0.008143$$

$$L_{3001} := 0.008143$$

$$L_{4001} := 0.008151$$

$$L_{5001} := 0.008151$$

The error norms show that we have determined a suitable approximation to the given experimental data.

In the following part of this MAPLE-program the above *leastsquare approximation* should be compared with the hardening curve based upon the *nonlinear MARQUARDT-LEVENBERG-algorithm:*

The ralative difference between the two approaches is defined as:

```
> restart:
  macro(l=LEASTSQUARE,M_L=MARQUARDT_LEVENBERG,S=sigma,E=epsilon):
> rel_difference:=1-S[l]/S[M_L];
```

$$rel\_difference := 1 - \frac{\sigma_{LEASTSQUARE}}{\sigma_{MARQUARDT\_LEVENBERG}}$$

```
> Delta[rel](E[pl]):=
  1-147.1329469*E[pl]^0.1722/(146.9071*E[pl]^0.1758);
```

$$\Delta_{rel}(\varepsilon_{pl}) := 1 - \frac{1.001537345}{\varepsilon_{pl}^{0.0036}}$$

```
> Delta[rel][mean_value]:=(1/5)*Int(abs(Delta[rel]),E[pl]=0..5)=
  (1/5)*int(abs(Delta[rel](E[pl])),E[pl]=0.000001..5);
```

$$\Delta_{rel_{mean\_value}} := \frac{1}{5}\int_0^5 \left|\Delta_{rel}\right| d\varepsilon_{pl} = 0.002865449834$$

```
> zero_of_Delta:=evalf(fsolve(1.001537345=E[pl]^0.0036,E[pl]),4);
```

$$zero\_of\_Delta := 1.532$$

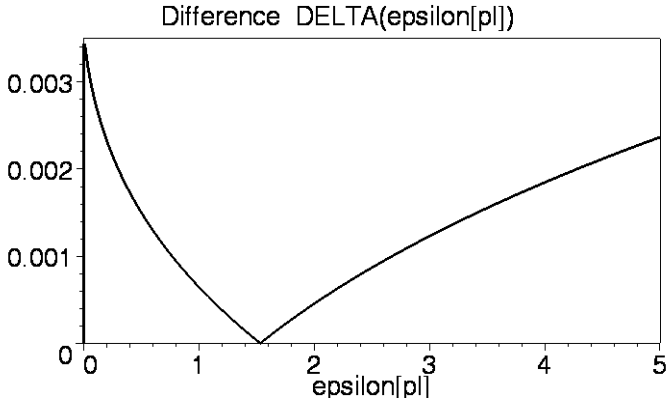
```
> difference:=(S[M_L]-S[l])/S[yield];
```

$$difference := \frac{\sigma_{MARQUARDT\_LEVENBERG} - \sigma_{LEASTSQUARE}}{\sigma_{yield}}$$

```
> DELTA(E[pl]):=
  0.4197345714*E[pl]^0.1758-0.4203798483*E[pl]^0.1722;
```

$$DELTA(\varepsilon_{pl}) := 0.4197345714\ \varepsilon_{pl}^{0.1758} - 0.4203798483\ \varepsilon_{pl}^{0.1722}$$

```
> alias(th=thickness,co=color):
> plot(abs(DELTA(E[pl])),E[pl]=0..5.001,0..0.0035,axes=boxed,
  ytickmarks=4,th=3,co=black,title="Difference
  DELTA(epsilon[pl])");
```



error norms:

```
L[p]:=((1/5)*Int((abs(DELTA))^p,E[pl]=0..5))^(1/p);
```

$$L_p := \left(\frac{1}{5}\int_0^5 |DELTA|^p\, d\varepsilon_{pl}\right)^{\left(\frac{1}{p}\right)}$$

```
> for i from 1 to 5 do
  L[i]:=evalf(((1/5)*(int((-DELTA(E[pl]))^i,E[pl]=0..1.532)+
  int((DELTA(E[pl]))^i,E[pl]=1.532..5)))^(1/i),4) od;
```

$$L_1 := 0.001288$$

$$L_2 := 0.001487$$

$$L_3 := 0.001629$$

$$L_4 := 0.001741$$

$$L_5 := 0.001836$$

These error norms illustrate that in the present case the *least squares method* furnishes comparable approximations with the *nonlinear MARQUARDT-LEVENBERG-algorithm.* However, other examples show significant differences between the two approaches.

```
>
```