Two elements, monotonic loading, case: “epe” (elastic-plastic*turn*elastic)

```
SetDirectory[NotebookDirectory[]]
```

```
C:\Users\Admin\Documents\reserach\codes\SPP2256
```

Der Datensatz enthält die Werte ($\tilde{\rho}$ $\tilde{\tau}$ $\tilde{\xi}$ $\tilde{\sigma}$)
$\tilde{\xi}$: “letzter” Wert der vorgegebenen Dehnung (Endpunkt des jeweils vorgegebenen Belastungspfades)
$\tilde{\sigma}$: “letzter” Wert der ausgewerteten Spannung (Endpunkt des jeweils vorgegebenen Belastungspfades)
$\tilde{\tau}$: Extremum entlang des Belastungspfades bzgl. ξ
$\tilde{\rho}$: Index bzgl. $\tilde{\tau}$: +1 Maximum, -1 Minimum

Import des Datensatzes

```
𝒟 = Import["data-set-03.txt", "CSV"];
```

Geometrie - Parameter

```
AA = 1.;
```

Parameter des Optimierungsproblems

```
EE = 1.;
```

Distanzfunktion (für die Suche im Datensatz)

```
Cξ = 500 000;
Cσ = 500 000^-1;
Cρ = 1;
Cτ = 10 000 000;
```

```
fdist[ρ_, τ_, ξ_, σ_] := Cρ ρ^2 + Cτ τ^2 + Cξ ξ^2 + Cσ σ^2
```

Algorithmus

Dirichlet - Randbedingung (rechter Rand)

```
ubarmax = 0.019;
ubar = 0.017;
```

Startwerte

```
init1 = Random[Integer, {1, Length[𝒟]}]; (*Random[Integer,{1,Length[𝒟]}]*)
ξstar[1] = 𝒟[[init1, 3]];
σstar[1] = 𝒟[[init1, 4]];
init2 = init1; (*Random[Integer,{1,Length[𝒟]}]*)
ξstar[2] = 𝒟[[init2, 3]];
σstar[2] = 𝒟[[init2, 4]];
```

Initialisierung(en)

```
posminold = {-1, -1};
posmin = posminold;
ρalgo[1] = ρalgo[2] = 1; (* fuer die vorgegebene
 monotone Belastung kann es sich nur um ein Maximum handeln *)
τalgo[1] = τalgo[2] = 0.;
```

```
Monitor[
 Do[
  (* Calculate solutions *)
  un = ξstar[1] - ξstar[2] + 1/2 ubar;
  ηn = 1/EE (σstar[1] - σstar[2]);
  (* update phsically sound stresses and strains *)
  σℰ[1] = 1/2 (σstar[1] + σstar[2]);
  σℰ[2] = 1/2 (σstar[1] + σstar[2]);
  ξℰ[1] = 1/2 (ξstar[1] - ξstar[2] + 1/2 ubar);
  ξℰ[2] = 1/2 (ξstar[2] - ξstar[1] + 1/2 ubar);
  τalgo[1] = 1/2 (ξstar[1] - ξstar[2] + 1/2 ubarmax);
  τalgo[2] = 1/2 (ξstar[2] - ξstar[1] + 1/2 ubarmax);
  Export["algorithm_sequence_step" <> ToString[iter] <> ".png", Show[
    ListPlot[𝒟[[All, 3 ;; 4]]], Graphics[{Red, PointSize[.015], Point[{ξℰ[1], σℰ[1]}]}],
    Graphics[{Blue, PointSize[.015], Point[{ξℰ[2], σℰ[2]}]}], PlotRange → All]];
  (* determine closest points of the data set and check convergence *)
  Print["current state: ", {0., τalgo[1], ξℰ[1], σℰ[1]}];
  Do[distances[i] =
    Map[fdist[0., τalgo[i] - #[[2]], ξℰ[i] - #[[3]], σℰ[i] - #[[4]]] &, 𝒟], {i, 2}];
  posmin = Table[Flatten[Position[distances[i], Min[distances[i]]]][[1]], {i, 2}];
  If[posmin == posminold, Print["Found solution in step ", iter, "!!!"];
   Break[]];
  posminold = posmin;
  ξstar[1] = 𝒟[[posmin[[1]], 3]];
  σstar[1] = 𝒟[[posmin[[1]], 4]];
  ξstar[2] = 𝒟[[posmin[[2]], 3]];
  σstar[2] = 𝒟[[posmin[[2]], 4]];
  Print[Array[ξstar, 2]];
  Print[Array[σstar, 2]];
  Export["algorithm_sequence1_step" <> ToString[iter] <> ".png", Show[
    ListPlot[𝒟[[All, 3 ;; 4]]], Graphics[{Red, PointSize[.015], Point[{ξℰ[1], σℰ[1]}]}],
    Graphics[{Green, PointSize[.015], Point[{ξstar[1], σstar[1]}]}]]];
  Export["algorithm_sequence2_step" <> ToString[iter] <> ".png", Show[
    ListPlot[𝒟[[All, 3 ;; 4]]], Graphics[{Red, PointSize[.015], Point[{ξℰ[2], σℰ[2]}]}],
    Graphics[{Green, PointSize[.015], Point[{ξstar[2], σstar[2]}]}]]];
  If[iter == maxsteps, Print["No convergence in ", maxsteps, " steps!!!"];
   Quit[]];
  , {iter, maxsteps}]
 , {iter, Array[σℰ, 2], Array[ξℰ, 2]}]
```

```
current state: {0., 0.00475, 0.00425, -55.5556}
{0.0037037, 0.0037037}
{277.778, 277.778}
current state: {0., 0.00475, 0.00425, 277.778}
{0.0042328, 0.0042328}
{388.889, 388.889}
current state: {0., 0.00475, 0.00425, 388.889}
Found solution in step 3!!!
```

Array[ξε, 2]

```
{0.00425, 0.00425}
```

Array[σε, 2]

```
{388.889, 388.889}
```

Analytische Lösung für den Fall “epe”:

$$\frac{\mathbf{ubar}}{\mathbf{4}}$$

0.00425

$$\mathbf{210\,000}\left(\frac{\mathbf{ubar}}{\mathbf{4}} - \frac{\mathbf{ubarmax}}{\mathbf{4}} + \frac{\mathbf{500}}{\mathbf{210\,000}}\right)$$

395.