Newton Iteration

Das nach Isaak Newton benannte Verfahren, ist in der Mathematik ein häufig verwendeter Approximations-Algorithmus zur numerischen Lösung von nichtlinearen Gleichungen.

Die grundlegende Idee dieses Verfahrens ist, die Funktion in einem Ausgangspunkt zu linearisieren, d. h. ihre Tangente zu bestimmen, und die Nullstelle der Tangente als verbesserte Näherung der Nullstelle der Funktion zu verwenden.

Die erhaltene Näherung dient als Ausgangspunkt für einen weiteren Verbesserungsschritt. Diese Iteration erfolgt solange, bis die Änderung in der Näherungslösung eine festgesetzte Schranke unterschritten hat.

Formal ausgedrückt, wird ausgehend von einem Startwert x_0 die Iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

wiederholt, bis die gewünschte Genauigkeit erzielt wird.

Das Verfahren soll am Beispiel der Funktion $f(x) = x - \cos(x)$ veranschaulicht werden.

In[43]:=
```
ClearAll["`*"];
f[x_] := x - Cos[x];
next[x_, fun_] := N[x - (fun[x] / fun'[x])];
a = 1;
Print["Iteration nach Newton"];
Print["Nullstelle von f(x): ", f[x]];
Print["Ableitung von f(x) : ", f'[x]];
Print["Startwert : ", a];
Print["Näherung         X-Wert       f(x) "];
Print["-----------------------------------"];
Do[a = next[a, f];
 Print["Näherung x", i, ":   ", a, "    ", N[f[a], 4] ], {i, 1, 6}]
```

```
Iteration nach Newton
Nullstelle von f(x): x - Cos[x]
Ableitung von f(x) : 1 + Sin[x]
Startwert : 1
Näherung          X-Wert        f(x)
-------------------------------------
Näherung x1:   0.750364    0.0189231
Näherung x2:   0.739113    0.0000464559
Näherung x3:   0.739085    2.8472 × 10^-10
Näherung x4:   0.739085    -1.11022 × 10^-16
Näherung x5:   0.739085    0.
Näherung x6:   0.739085    0.
```

Diese Implementierung lehnt sich an das prozedurale Programmier-Paradigma an. Mathematica bietet allerdings noch weitere Möglichkeiten.

In[63]:=
```
ClearAll["`*"]
g[x_] = x - ((x^2 - 2) / (2 x))
```

Out[64]=
$$x - \frac{-2 + x^2}{2\,x}$$

In[65]:=
```
g[1.0]
```

Out[65]=
```
1.5
```

In[66]:=
```
g[%]
```

Out[66]=
```
1.41667
```

So könnte man nun die Iteration von Hand weiter führen. Zum Glück gibt es aber eine Funktion *NestList* die das übernehmen kann:

In[76]:=
```
TableForm[NestList[g, N[1, 20], 10]]
```

Out[76]//TableForm=
```
1.0000000000000000000
0.7503638678402438930
0.7391128909113616704
0.7390851333852839698
0.739085133215160642
0.739085133215160642
0.739085133215160642
0.73908513321516064
0.73908513321516064
0.73908513321516064
0.7390851332151606
```

Man erkennt sehr leicht die Arbeitsweise von *NestList.* Der Funktionswert von *g* startet bei 1 und wird 20 mal mit einer Genauigkeit von 20-Stellen immer wieder eingesetzt.

Am Ergebnis sieht man sehr schön, wie die Folge konvergiert.
Zusammenfassend benötigt man nur die Definition der Folge und die NetList-Funktion.

Angewendet auf das Eingangsbeispiel würde das Newton-Iterations-Verfahren dann programmtechnisch mit den Funktionen
f(x):= x - cos[x], f'(x) = 1 - sin[x]
so ausschauen:

In[87]:=
```
ClearAll["`*"];
g[x_] = x - ((x - Cos[x]) / (1 + Sin[x]));
TableForm[NestList[g, N[1, 20], 15]]
```

Out[89]//TableForm=
```
1.0000000000000000000
0.7503638678402438930
0.7391128909113616704
0.7390851333852839698
0.739085133215160642
0.739085133215160642
0.739085133215160642
0.73908513321516064
0.73908513321516064
0.73908513321516064
0.7390851332151606
0.7390851332151606
0.7390851332151606
0.7390851332151606
0.739085133215161
0.739085133215161
```

In[105]:=
```
Plot[x - Cos[x], {x, -1, Pi / 2}]
```

Out[105]=