3. Übung: Aufgabe 7

Computernumerik SS 2016

[redacted]

2. Juni 2016

1 Mehrdimensionales Newton-Verfahren

Betrachten Sie wieder das Polynom aus der vorigen Aufgabe. Eine alternative Methode zur Berechnung der Nullstellen ist folgende: Für ein Polynom mit Nullstellen x_1, x_2, x_3 der Form $p(x) = x^3 - c_1x^2 + c_2x - c_3$ gelten nach einer Verallgemeinerung des Satzes von Vieta die folgenden Gleichungen:

$c_1 = x_1 + x_2 + x_3$
$c_2 = x_1x_2 + x_1x_3 + x_2x_3$
$c_3 = x_1x_2x_3$

Benutzen Sie das Newton-Verfahren zur Lösung dieses nichtlinearen Gleichungssystems.

1.1 Theorie

Um die Nullstellen des Polynoms zu finden, muss das nichtlineare Problem

$$\begin{pmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

mit nichtlinearen Funktionen f_1, f_2, f_3 gelöst werden. Zur Lösung kommt das mehrdimensionale Newton-Verfahren zum Einsatz [1]. Die Iterationsvorschrift wird dabei analog zum Newton-Verfahren im eindimensionalen Fall festgelegt.

Interationsvorschrift im eindimensionalen Fall:

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

Iterationsvorschrift im mehrdimensionalen Fall:

$$\vec{x}_{n+1} = \vec{x}_n - (J(\vec{x}_n))^{-1} f(\vec{x}_n)$$

Wobei $J(\vec{x})$ die Jakobimatrix, also die Matrix der partiellen Ableitungen von $f(\vec{x})$, bezeichnet.

Da die Berechnung der Inversen Matrix $(J(\vec{x}_n))^{-1}$ und die anschließende Multiplikation mit $f(\vec{x}_n)$ aufwendig ist, wird stattdessen das folgende lineare Gleichungssystem gelöst:

$$J(\vec{x}_n)\Delta\vec{x}_n = -f(\vec{x}_n)$$

Danach berechnet sich $\vec{x}_{n+1}$ aus: $\vec{x}_{n+1} = \vec{x}_n + \Delta\vec{x}_n$.

1.2 Lösungsansatz

Das Polynom aus Aufgabe 6 lautet: $p(x) = x^3 - 3x^2 + (3 - \varepsilon^2)x + \varepsilon - 1$

Daraus ergeben sich die folgenden Gleichungen

$$f_1(x_1, x_2, x_3) = x_1 + x_2 + x_3 - 3$$
$$f_2(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3 - (3 - \varepsilon^2)$$
$$f_2(x_1, x_2, x_3) = x_1x_2x_3 - (1 - \varepsilon^2)$$

und die dazugehörige Jakobimatrix:

$$J(\vec{x}) := \frac{\partial f_i}{\partial x_j} = \begin{pmatrix} 1 & 1 & 1 \\ x_2 + x_3 & x_1 + x_3 & x_1 + x_2 \\ x_2 * x_3 & x_1 * x_3 & x_1 * x_2 \end{pmatrix}$$

Abbildung 1 zeigt einen Plot des Polynoms mit unterschiedlichen Werten für ε und markierten Nullstellen, die durch das mehrdimensionale Newton-Verfahren gefunden wurden.

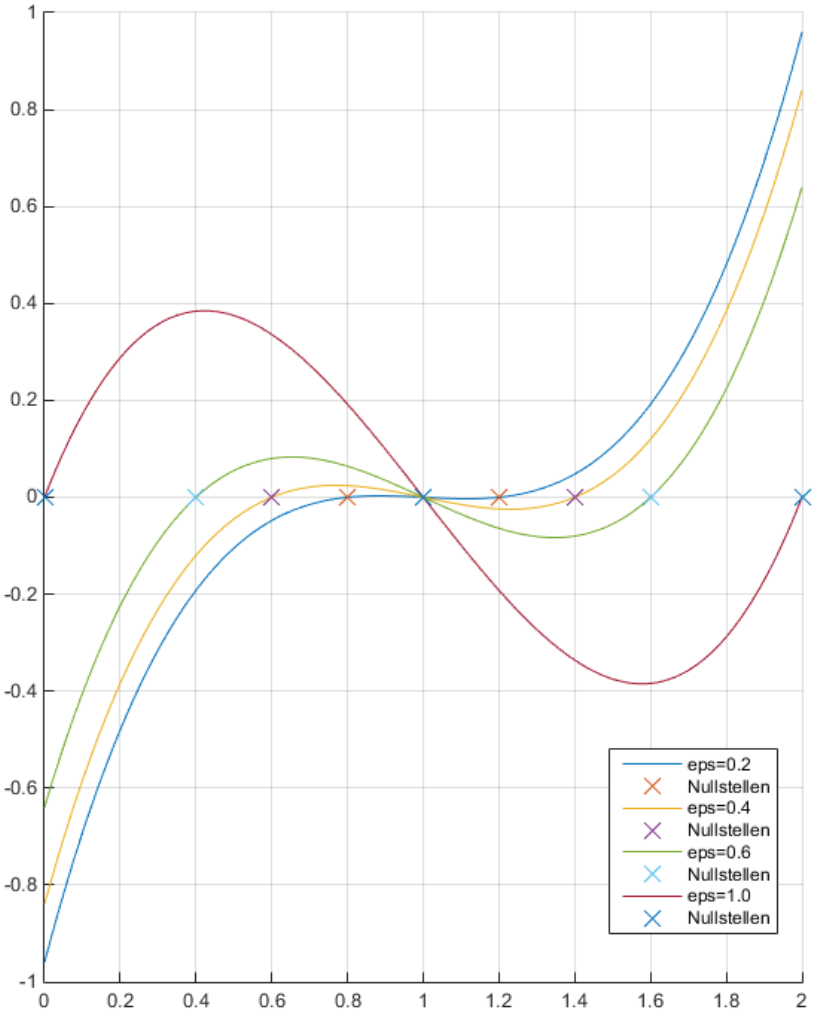

Abbildung 1: Plot des Polynomes mit Nullstellen für verschiedene ε.

1.3 Matlab Code

```
% Toleranz
tol = 1e-9;

% Maximale Anzahl an Iterationen
nMax = 100;

% x Start
x0 = [0.1; 0.15; 0.2];

% Epsilon
eps = 0.2;

% Gleichungen
F = @(x1,x2,x3)[x1 + x2 + x3 - 3;
                x1 * x2 + x1 * x3 + x2 * x3 - (3 - eps^2);
                x1 * x2 * x3 - (1 - eps^2)];

% Jakobi Matrix
dF = @(x1,x2,x3)[      1,       1,       1;
                 x2 + x3, x1 + x3, x1 + x2;
                 x2 * x3, x1 * x3, x1 * x2];

% Newton Verfahren
xi = x0;

% Laeuft bis die Maximale Anzahl an Iterationen erreicht wurde oder ...
    die Aenderung unter den Toleranz-Wert faellt
for n = 1:nMax
    % x = A\-B loest das System Ax = -B => x = -B/A
    deltaX = dF(xi(1), xi(2), xi(3)) \ -(F(xi(1), xi(2), xi(3)));

    xi = xi + deltaX;
    if max(deltaX(:)) < tol
        % Abbruch des Newton Verfahrens wenn Toleranz unterschritten
        break;
    end
end

% Ausgabe der Nullstellen
fprintf('\nNullstellen: \n');
fprintf('x1 = %0.10f \n', xi(1));
fprintf('x2 = %0.10f \n', xi(2));
fprintf('x3 = %0.10f \n\n', xi(3));
```

Literatur

[1] Pascal Sebah and Xavier Gourdon. Newton's method and high order iterations. Technical report, Technical report, 2001. http://numbers. computation. free. fr/-Constants/Algorithms/newton. html, 2001.