

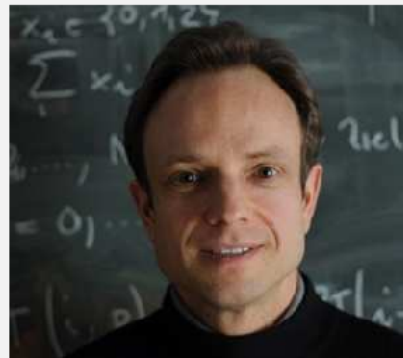
Course Video

Section 8.5.1: The Newton Iteration in $\mathbb{R}^n$ (II)

Prof. R. Hiptmair, SAM, ETH Zurich

Date: December 5, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Dependencies. Section 8.5.1 (Part I)

Duration: 30 minutes

VIII Iterative Methods for Non-Linear Systems of Equations

8.5 Newton's Method in $\mathbb{R}^n$

NLSE: Given $C^1 F: \mathcal{D} \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $n \in \mathbb{N}$,
find $\underline{x}^* \in \mathcal{I} : F(\underline{x}^*) = 0$

8.5.1. The Newton Iteration

► **Newton iteration:** (generalizes (8.4.2.1) to $n > 1$)

$\underline{x}^{(k+1)} := \underline{x}^{(k)} - DF(\underline{x}^{(k)})^{-1}F(\underline{x}^{(k)})$, [if $DF(\underline{x}^{(k)})$ regular] (8.5.1.6)

Terminology: $-DF(\underline{x}^{(k)})^{-1}F(\underline{x}^{(k)})$ is called the **Newton correction** $\underline{s} \in \mathbb{R}^n$

Handwritten notes: Jacobian $\in \mathbb{R}^{n,n}$ (pointing to $DF(\underline{x}^{(k)})$)

Newton correction by solving LSE

$$DF(\underline{x}^{(k)}) \underline{s} = -F(\underline{x}^{(k)}) \quad (\text{NLSE})$$

§ 8.5.1.25 (A "quasi-linear" system of equations)
 $\rightarrow$ § 8.5.1.22

$$A(\underline{x}) \cdot \underline{x} = \underline{b}, \quad \underline{b} \in \mathbb{R}^n, \quad A: \mathbb{R}^n \rightarrow \mathbb{R}^{n,n} \text{ cont. diff.}$$

$$\begin{aligned} &\Downarrow \\ F(\underline{x}) &= 0, \quad F(\underline{x}) := \underline{b} - A(\underline{x})\underline{x}, \quad DF(\underline{x})? \\ &\bullet \hat{=} \text{bilinear product: } \text{product rule} \end{aligned}$$

Product rule: $F: D \subset V \mapsto W, G: D \subset V \mapsto U$ sufficiently smooth, $b: W \times U \mapsto Z$ **bilinear**, ie., linear in each argument:

$$T(\underline{x}) := b(F(\underline{x}), G(\underline{x})) \Rightarrow D T(\underline{x}) \underline{h} = b(D F(\underline{x}) \underline{h}, G(\underline{x})) + b(F(\underline{x}), D G(\underline{x}) \underline{h}), \quad \underline{h} \in V, \underline{x} \in D. \quad (8.5.1.17)$$

Here: $b \hat{=} M \times V, F(\underline{x}) \leftrightarrow A(\underline{x}), G := Id$

$$\triangleright DF(\underline{x}) \underline{h} = \underbrace{(DA(\underline{x}) \underline{h}) \cdot \underline{x}}_{\in \mathbb{R}^{n,n}} + A(\underline{x}) \cdot \underline{h}$$

$$\Downarrow$$

NCLSE

$$(DA(\underline{x}^{(k)}) \underline{\underline{\varepsilon}}) \underline{x}^{(k)} + A(\underline{x}^{(k)}) \underline{\underline{\varepsilon}} = A(\underline{x}^{(k)}) \underline{x}^{(k)} - \underline{b}$$

Ex. 8.5.1.30: (A special quasi-linear system of equations)

$$\underline{b} \in \mathbb{R}^n, \quad \underline{A}(\underline{x}) \underline{x} = \underline{b}, \quad \underline{A}(\underline{x}) := \begin{bmatrix} \gamma(\underline{x}) & 1 & & & \\ 1 & \gamma(\underline{x}) & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & \gamma(\underline{x}) & 1 \\ & & & 1 & \gamma(\underline{x}) \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad (8.5.1.31)$$

$$\Downarrow F(\underline{x}) := A(\underline{x}) \underline{x} - \underline{b} = 0$$

$$y(\underline{x}) = 3 + \|\underline{x}\|_2$$

$$\triangleright \underline{A}(\underline{x}) \underline{x} = \underline{T} \underline{x} + \underline{x} \|\underline{x}\|_2, \quad \underline{T} := \begin{bmatrix} 3 & 1 & & & \\ 1 & 3 & 1 & & \\ & \ddots & 3 & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & 3 & 1 \\ & & & & 1 & 3 \end{bmatrix}$$

$$F(\underline{x}) = \underline{T} \underline{x} + \underline{x} \|\underline{x}\|_2 - \underline{b}$$

$$DF(\underline{x}) \underline{h} = D \underbrace{\{\underline{x} \mapsto \underline{T} \underline{x}\}}_{\text{linear}}(\underline{x}) \underline{h} + D \underbrace{\{\underline{x} \mapsto \underline{x} \|\underline{x}\|_2\}}_{\text{product rule}}(\underline{x}) \underline{h}$$

$$\text{Use } D \{\underline{x} \mapsto \|\underline{x}\|_2\}(\underline{x}) \underline{h} = \frac{\underline{x}^T}{\|\underline{x}\|_2} \cdot \underline{h}$$

$$\begin{aligned} \triangleright DF(\underline{x}) \underline{h} &= \underline{T} \underline{h} + \underline{h} \cdot \|\underline{x}\|_2 + \underline{x} \cdot \frac{\underline{x}^T}{\|\underline{x}\|_2} \cdot \underline{h} \\ &= \left(\underline{A}(\underline{x}) + \frac{\underline{x} \underline{x}^T}{\|\underline{x}\|_2} \right) \underline{h} \end{aligned}$$

③

▷ NCLSE:

$$\left(A(x^{(k)}) + \frac{x^{(k)} x^{(k)T}}{\|x^{(k)}\|_2} \right) \underline{s} = \underline{b} - A(x^{(k)}) x^{(k)}$$

rank-1-modification of tridiagonal matrix;

Use SMW formula + QR for $D(n)$ -solve

Ex. 8.5.1.38 (Matrix inversion by means of Newton's method)

$$A \in \mathbb{R}^{n,n} \text{ regular} \\ F(A^{-1}) = 0 \quad \text{for} \quad \begin{cases} F: \mathbb{R}_{*}^{n,n} \longrightarrow \mathbb{R}^{n,n} \\ F(X) := A - X^{-1} \end{cases}$$

NCLSE?

Ex. 8.5.1.35 (Derivative of matrix inversion)

$$\text{inv}: \begin{cases} \mathbb{R}_{*}^{n,n} \longrightarrow \mathbb{R}^{n,n} \\ X \longrightarrow X^{-1} \end{cases}, \quad \mathbb{R}_{*}^{n,n} \hat{=} \text{regular matrices}$$

 $D \text{ inv} = ?$

Idea: implicit differentiation

$$D: \quad X \cdot \text{inv}(X) = I$$

$$\text{product rule: } H \cdot \text{inv}(X) + X \cdot D \text{inv}(X) H = 0, \quad H \in \mathbb{R}^{n,n}$$

$$\triangleright \quad D \text{inv}(X) H = -X^{-1} H X^{-1}$$

Ex. 8.5.1.38 cont'd:

$$DF(X) H = X^{-1} H X^{-1}, \quad X \in \mathbb{R}_{*}^{n,n}$$

NCLSE:

$$DF(X^{(k)}) S = (X^{(k)})^{-1} S (X^{(k)})^{-1} = -F(X^{(k)}) = (X^{(k)})^{-1} - A$$

$$\triangleright \quad S = X^{(k)} ((X^{(k)})^{-1} - A) X^{(k)} = X^{(k)} - X^{(k)} A X^{(k)}$$

$$\text{N.I.: } X^{(k+1)} = X^{(k)} + S = X^{(k)} (2I - A X^{(k)})$$

(4)

Rem. 8.5.1.44 (Simplified Newton method)

$\hat{=}$ Newton method with "frozen Jacobian"

For $F(x) = 0$:

$$x^{(k+1)} = x^{(k)} - DF(x^{(0)})^{-1} F(x^{(k)}), \quad k \in \mathbb{N}$$

Benefit: Reuse of LU-dec. of $DF(x^{(0)})$

C++ code 8.5.1.44: Efficient implementation of simplified Newton method

```

2 // C++ template for simplified Newton method
3 template<typename Func, typename Jac, typename Vec>
4 void simpnewton(Vec& x, Func F, Jac DF, double rtol, double atol)
5 {
6     auto lu = DF(x).lu(); // do LU decomposition once!
7     Vec s;                // Newton correction
8     double ns, nx;        // auxiliary variables for termination control
9     do {
10        s = lu.solve(F(x)); //  $\rightarrow O(n^2)$ 
11        x = x - s;          // new iterate
12        ns = s.norm(); nx = x.norm();
13    }
14    // termination based on relative and absolute tolerance
15    while((ns > rtol*nx) && (ns > atol));
16 }

```

Drawback: Slower conv.

▷ Review questions 8.5.1.46

5

7