Tipps zu Serie 2

Aufgabe 2

Bei der Aufgabe habt ihr eine Blockmatrix gegeben, deren Struktur ihr geschickt anwenden sollt, um ein Gleichungssystem mit dieser zu lösen.

a) Eine Matrix $A \in \mathbb{R}^{n\times n}$ ist invertierbar, falls

$$\det(A) \neq 0 \, .$$

Wann ist diese Bedingung für eine obere Dreiecksmatrix erfüllt?

b) Beachtet den Hinweis und lest euch zur Berechnung das Kapitel (1.6.91) aus dem Skript zum Thema *Block elemination* durch. Am Ende solltet ihr eine Darstellung für $\mathbf{v}$ und ξ haben, welche nur von $\mathbf{R}$, $\mathbf{b}$, $\mathbf{v}$, $\mathbf{u}^T$ und β abhängt.

c) Betrachtet die zuvor gefundenen Ausdrücke für $\mathbf{z}$ und ξ. Wann hat das Gleichungssystem eine eindeutige Lösung?

d) Hier sollt ihr nun eine C++ Funktion schreiben, die das vorherige Gleichungssystem effizient löst. Benutzt dazu die in **b)** gefundene Lösung. Falls euch die Hinweise mehr verwirren als helfen, ignoriert diese einfach. Die Grundstruktur des Codes findet ihr in Listing 1

Listing 1: solvelse

```
1
2  //! \brief Use efficient implementation A*bb = x
3  //! \param[in] R Matrix is nxn and upper triangular
4  //! \param[in] v Vector is nx1
5  //! \param[in] u Vector is nx1
6  //! \param[in] bb vector is (n+1)x1 and is stacked (b, \beta)^T =: b
7  //! \param[out] x solution A*bb = x
8  template <class Matrix, class Vector>
9  void solvelse(const Matrix & R, const Vector & v, const Vector & u,
10                const Vector & bb, Vector & x) {
11 // Size of R, which is wize of u, v, and size of bb is n+1
12
13 // Size check, error if do not match
14
15
16 // Hier kommt dann eure Berechnung des Ergebnisses hin
17 // Speichert das Ergebnis im Vektor x
18 }
```

e) Die in Eigen vorimplementierte LU-Zerlegung zur Lösung des Gleichungssystems erhaltet ihr mit:

```
1 x = A.partialPivLu().solve(bb);
```

A ist dabei die zu Beginn definierte Matrix. Führt nun wie in den vorherigen Serien einen Vergleich der Laufzeiten beider Methoden durch.

f) Betrachtet die Operationen, die ihr zur Berechnung eurer Lösung anwendet, und leitet daraus die Gesamtkomplexität eures Algorithmus' her.