Computermathematik – Übung Sage 3

- **Abgabeschluss:** Di 12. 01. 2016 um 20:00
- **Präsentation:** Mi 13. 01. 2016 in der Übungsgruppe
- **Abgabeformat:** `.sws` mit Worksheet-Namen `BspX_Nachname_Vorname` (`X` $\in \{$`13`, `B6`, `B7`$\}$)

Aufgabe 13 – Algorithmen implementieren (8 Punkte)

Implementiere selbst *einen* der folgenden Algorithmen aus Lineare Algebra als Funktion.

Vergleiche zur Probe anhand einer Beispiel-Matrix A (z.B. die angegebenen Beispiele aus dem Tutorium NRLA) das selbst-berechnete Ergebnis mit der eingebauten Library-Funktion.

- **LR-Zerlegung** (wie `A.LU()` und Bsp. 11): Berechne eine untere Dreiecksmatrix L, obere Dreiecksmatrix R und Zeilen-Vertauschungs-Matrix P, so dass $P \cdot A = L \cdot R$ (mittels Gauß-Elimination).

  Optional: Demonstriere, wie deine LR-Zerlegung genutzt werden kann, um ein lineares Gleichungssystem $A \cdot x = b$ zu lösen.

- **QR-Zerlegung** (wie `A.QR()` und Bsp. 38): Berechne eine Orthogonalmatrix Q und obere Dreiecksmatrix R, so dass $A = Q \cdot R$ (mittels Gram-Schmidt-Orthogonalisierung).

  Optional: Demonstriere, wie deine QR-Zerlegung genutzt werden kann, um überbestimmte Gleichungssysteme $A \cdot x = b$ annähernd zu lösen.

- **Jacobi** und **Gauß-Seidel** (wie `A.solve_right()` und Bsp. 44): Berechne eine approximierte Lösung x für das Gleichungssystem $A \cdot x = b$. Vergleiche auch die Ergebnisse der beiden eigenen Algorithmen miteinander.

Falls du „Numerisches Rechnen und Lineare Algebra“ dieses Semester (noch) nicht belegst, schlag deinem Tutor oder mir einen anderen Algorithmus vor, z.B. aus Analysis (Taylorpolynome, ...) oder deinem Matura-Stoff (Lösen von Gleichungssystemen, ...).

Hint: Grundlegende Library-Funktionen wie `.swap_rows()`, `.add_multiple_of_row()`, `.norm()`, `sum()`, etc. dürfen natürlich verwendet werden. Gib bei der Erstellung der Beispiel-Matrizen den Datentyp für die Elemente sicherheitshalber explizit an (z.B. `RDF` für $\mathbb{R}$, `CDF` für $\mathbb{C}$).

Bonus – Formatierter Output (2 Bonuspunkte)

Gib relevante Zwischenergebnisse und Statusinformationen deines Algorithmus übersichtlich formatiert aus, um beispielsweise mit einer händischen Lösung vergleichen zu können.

Bonus – Numerische Instabilität (3 Bonuspunkte)

Informiere dich über die „numerische Stabilität“ deines gewählten Algorithmus: Können Matrizen mit ungünstigen Eigenschaften („schlecht konditioniert“) zu sehr großen numerischen Rundungsfehlern führen? Demonstriere anhand eines selbstgewählten Beispiels, bei dem das Ergebnis ungenauer ist als die gewählte Rechengenauigkeit des Datentyps (wie Aufgabe 10).