KV Wissenschaftliches Rechnen, WS 2024/25

Hausübung 2

Es ist ein Header `SCHermiteBandMatrix.h` zu erstellen, in dem eine reell symmetrische/komplex hermitesche Matrix mit Bandstruktur wie unten angegeben implementiert wird. Die in der VL entwickelten Klassen (`SCvector.h` und `SCmatrix.h`) sind zu verwenden, können aber erweitert werden. Die Klasse soll an einem Testproblem getestet werden, und zur numerischen Lösung des 1D-Diffusionsproblems genutzt werden.

Abzugeben sind

- `SCHermiteBandMatrix.h` File, enthält Quellcode,
- `SCmatrix.h` bzw. `SCvector.h` **falls geändert!** In diesem Fall Erweiterungen beschreiben.
- Quellcode mit `main`-Routine, in der die Aufgaben umgesetzt werden (eine gemeinsame oder zwei separate Dateien).
- Protokoll als `.pdf` File.

Aufgabe 1 (Klasse HermiteBandMatrix) – 6 Punkte Eine symmetrische/hermitesche Matrix mit Bandstruktur ist eine quadratische Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}, \mathbb{C}^{n\times n}$ mit $\boldsymbol{A} = \boldsymbol{A}^H$. Die Bandbreite b der Matrix gibt an, wieviele Sub- und Super-Diagonalen der Matrix mit Werten ungleich Null besetzt sein können,

$$\boldsymbol{A} = \begin{bmatrix} * & \dots & * & 0 & 0 \\ \vdots & * & \ddots & \ddots & 0 \\ * & \ddots & * & \ddots & * \\ 0 & \ddots & \ddots & * & \vdots \\ 0 & 0 & * & \dots & * \end{bmatrix} \begin{matrix} \\ n-b \\ \\ \\ b \end{matrix}, \qquad a_{ij} = 0 \text{ für } |i-j| \geq b, \qquad a_{ij} = \overline{a_{ji}}. \tag{1}$$

Wenn die Nullelemente und die Elemente unterhalb oder oberhalb der Diagonalen nicht abgespeichert werden, kann im Vergleich zur Standard-Matrixklasse wesentlicher Speicherplatz gespart werden. Die Anwendung `Apply` kann die Bandstruktur auch ausnutzen.

Die symmetrische Matrix mit Bandstruktur soll von LinearOperator<T> abgeleitet sein:

```
namespace SC {
    template <typename T = double>
    class HermiteBandMatrix : public LinearOperator<T> {
    // ....
    };
}
```

Folgende Spezifikationen müssen erfüllt sein

1. die Matrix ist immer quadratisch,
2. zusätzlich zu LinearOperator<T>::height und LinearOperator<T>::width eine Member-Variable **int** b,
3. **1 Punkt – benötigter Speicher** Es dürfen maximal $n(n+1)/2 - (n-b)(n-b+1)/2$ Speicherplätze vom Typ T für die Daten verwendet werden,

4. **1 Punkt** – Default- und Copy-Construktor, Destruktor und Zuweisungsoperator müssen implementiert oder ausgeschlossen (= **delete**) sein, zusätzlich wird ein Konstruktor für eine $n \times n$ hermitesche Matrix mit Bandbreite b benötigt (HermiteBandMatrix(**int** n, **int** b)),

5. **0.5 Punkte** – **Lesezugriffsoperator** T **operator**()(**int** i, **int** j) **const** liefert *Lese*zugriff auf Eintrag A_{ij}, gibt 0 wenn der Wert abseits der Bandstruktur liegt,

6. **0.5 Punkte** – **Schreibzugriff** **void** Set(**int** i, **int** j, T val) implementiert $A_{ij} = val$, mit Range-Check im Debug-Modus: **throw** bei ungültiger Position abseits der Bandstruktur, im Release-Modus muss nichts überprüft werden,

7. **2 Punkte** die Apply-Funktionalitäten mussen überladen werden,

```
void Apply(const Vector<T> &x, Vector<T> &r, T factor = 1.)
const override
void ApplyT(const Vector<T> &x, Vector<T> &r, T factor = 1.)
const override
void ApplyH(const Vector<T> &x, Vector<T> &r, T factor = 1.)
const override
```

   der Gesamtaufwand muss $O(bn)$ sein,

8. **1 Punkt** nach eigenen Vorstellungen soll eine Ausgabefunktion erstellt werden,

```
void Print(std::ostream& os) const override
```

Dokumentieren Sie im Protokoll jedenfalls, welche Konstruktoren umgesetzt bzw. ausgeschlossen wurden, wie die Matrixeinträge abgespeichert werden (zeilenweise, spaltenweise, anders?), und wie die Apply-Funktionalität umgesetzt ist.

Aufgabe 2 (Unit-Test) – 1 Punkt Testen Sie Ihre hermitesche Band-Matrix an folgender Matrix mit $n = 6$ und $b = 3$,

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2+3i & 1+i & 0 & 0 & 0 \\ 2-3i & 2 & 1.4 & 3i & 0 & 0 \\ 1-i & 1.4 & 3 & 1-i & 2i & 0 \\ 0 & -3i & 1+i & 4 & 1 & 1 \\ 0 & 0 & -2i & 1 & 5 & 1-2i \\ 0 & 0 & 0 & 1 & 1+2i & 6 \end{bmatrix}, \tag{2}$$

indem Sie umsetzen

- Setzen der Matrix mit Set-Routine,
- Ausgabe in der Kommandozeile,
- Multiplikationen $\boldsymbol{Ax}$, $\boldsymbol{A}^T\boldsymbol{x}$ und $\boldsymbol{A}^H\boldsymbol{x}$ für $\boldsymbol{x} = [1, i, -1, -i, 1, i]^T$

Der Unit-Test braucht im Protokoll nicht dokumentiert werden!

Aufgabe 3 (Anwendungsbeispiel) – 3 Punkte Gesucht ist eine Näherungslösung der Differentialgleichung

$$-ku'' = f, \qquad u(0) = u(1) = 0. \tag{3}$$

Es handelt sich um ein eindimensionales Diffusions-Modellproblem. Der Parameter $k > 0$ beschreibt die Diffusion (z.B. Wärmeleitfähigkeit), $f(x)$ die Quellterme (z.B. Wärmequellen). Als Parameter wählen wir

$$k = 10, \qquad f(x) = \begin{cases} 0 & \text{falls } x < 0.5, \\ 1 & \text{falls } x \geq 0.5 \end{cases}. \tag{4}$$

Unterteilen Sie das Rechengebiet $[0,1]$ gleichmäßig in n_e Teilstücke, indem Sie $n = n_e - 1$ Punkte definieren, $x_i = \frac{i+1}{n_e}$ für $i = 0 \dots n-1$. Die Netzfeinheit ist entsprechend $h = 1/n_e$. In jedem inneren Punkt werden die Ableitungen von u durch Differenzenquotienten ersetzt,

$$-u'' \simeq \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2}. \tag{5}$$

Dies führt auf ein lineares Gleichungssystem in Tridiagonalstruktur, für jedes $i = 0 \dots n-1$

$$k\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f(x_i), \qquad \text{mit Randwerten } u_{-1} = 0 \text{ und } u_n = 0. \tag{6}$$

Legen Sie eine entsprechende Bandmatrix $\boldsymbol{A}$ mit n und Bandbreite $b = 2$ an. Legen Sie den rechte-Seite-Vektor $\boldsymbol{b}$ an,

$$a_{ii} = 2k/h^2, \qquad a_{i\,i-1} = a_{i-1\,i} = -k/h^2, \qquad b_i = f(x_i). \tag{7}$$

Legen Sie auch einen Lösungsvektor $\boldsymbol{u}$ der Länge n an, den Sie mit $u_i = 0$ initialisieren. Legen Sie noch einen Hilfsvektor $\boldsymbol{r}$ der Länge n an.

Das Gleichungssystem kann prinzipiell in optimaler Weise in $O(n)$ Schritten exakt gelöst werden (u.a. Inhalt von Kapitel 2). In diesem Übungszettel wird es ganz allgemein mit dem Gradientenverfahren gelöst, in jedem Iterationsschritt

$$\boldsymbol{r} \;:= \boldsymbol{b} - \mathbf{A}\boldsymbol{u}, \tag{8}$$

$$\boldsymbol{u} \;+= \theta\, \boldsymbol{r}, \tag{9}$$

mit Dämpfungsfaktor $\theta = h^2/(2k)$. Setzen Sie diese Iterationsvorschrift um! Je größer die Anzahl an Freiheitsgraden n umso (deutlich) schlechter konvergiert die Methode. Allerdings sollte, wenn θ wie in der Angabe gewählt ist, das Verfahren nicht divergieren.

Versuchen Sie ihre Lösung zu visualisieren, indem Sie die Punkte x_i und die Lösungsnäherung u_i exportieren und $u(x)$ z.B. mit python plotten. Für die Visualisierung ist es gut, wenn Sie am Anfang und Ende jeweils den Randwert $x_{-1} = 0, u_{-1} = 0$ und $x_n = 1, u_n = 0$ hinzufügen. Verwenden Sie dazu $n = 5, 10, 50, 100$, machen Sie maximal 20000 Iterationsschritte. Brechen Sie die Iteration ab, sobald die Norm des Residuums $|\boldsymbol{r}| < 10^{-6}$, falls dies vor Erreichen der maximalen Iterationszahl passiert.

Punkteverteilung Aufgabe 3:

- Aufstellen Gleichungssystem in main-Routine **1 Punkt**
- Implementierung der Iterationsvorschrift **1 Punkt** (in der Iterationsschleife dürfen keine Vektoren erzeugt werden, kein **new**/**delete**. Außerhalb dieser Schleife dürfen beliebig zusätzliche Variablen und Vektoren definiert werden.)
- Dokumentation der Ergebnisse (etwa Plot $u(x)$ für verschiedene n, Anzahl der benötigten Iterationen oder verbleibendes Residuum $|\boldsymbol{r}|$ nach 20000 Iterationen) **1 Punkt**