---

21 Gegeben sei das lineare Gleichungssystem $A\underline{x} = \underline{f}$ mit einer invertierbaren, tridiagonalen Matrix

$$A = \begin{pmatrix} d_1 & c_1 & & & 0 \\ a_1 & d_2 & c_2 & & \\ & a_2 & d_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_{n-1} & d_n \end{pmatrix} \quad \text{und} \quad \underline{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}.$$

Für dieses Gleichungssystem leite man einen effizienten Gauß-Algorithmus her. Dazu zeige man, dass das gegebene Gleichungssystem äquivalent ist zu $\tilde{A}\underline{x} = \underline{\tilde{f}}$ mit

$$\tilde{A} = \begin{pmatrix} 1 & \tilde{c}_1 & & & 0 \\ & 1 & \tilde{c}_2 & & \\ & & 1 & \ddots & \\ & & & \ddots & \tilde{c}_{n-1} \\ 0 & & & & 1 \end{pmatrix} \quad \text{und} \quad \underline{\tilde{f}} = \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \vdots \\ \tilde{f}_n \end{pmatrix}.$$

Danach löse man das äquivalente Gleichungssystem $\tilde{A}\underline{x} = \underline{\tilde{f}}$.

22 Analog zu Lemma 1.56 zeige man für den Interpolationsoperator $I_h : H^1(0,1) \to V_h$ die Fehlerabschätzung

$$|u - I_h(u)|^2_{H^1(0,1)} \leq \sum_{k=1}^{n_h} h_k^2 \int_{T_k} |u''(x)|^2 \, \mathrm{d}x$$

für alle $u \in \mathcal{C}^2[0,1]$.

Programmierteil.
Für den Programmcode ist es hilfreich sich die folgenden Datentypen zu definieren:

```
typedef double Vec2[2];
typedef double Mat22[2][2];
```

Dadurch wird durch `Vec2` ein Vektor im $\mathbb{R}^2$ und durch `Mat22` eine 2×2 Matrix definiert. Weiters ist es hilfreich sich Funktionen mit

```
typedef double (*RealFunction)(double x);
```

vorzugeben. Zum speichern von größeren Vektoren empfiehlt es sich die Vektor Klasse, die auf der Übungs-Webseite zur Verfügung gestellt wurde, zu verwenden.

---

23 Man erstelle einen Datentyp bzw. eine Klasse, die die Informationen über eine eindimensionale Zerlegung speichert. Es sollen allen notwendigen Informationen gespeichert werden, die zur Assemblierung der Steifigkeitsmatrix notwendig sind. Insbesondere sollten folgende Routinen vorhanden sein:

- Initialisierung: Zum Beispiel eine gleichmäßige Zerlegung mit einer vorgegebenen Anzahl von Knoten.

- Die Abfrage über die Anzahl der Knoten bzw. Anzahl der Elemente.
- Die Abfrage über die Koordinaten eines Knotens der Zerlegung.
- Die Abfrage über die Knotennummern von einem beliebigen Element der Zerlegung.

Weiters implementiere man eine Methode `void TestMesh()` die eine Zerlegung des Intervalls $(0, 1)$ speichert und weiters führe man verschiedene Testfälle für die implementierten Funktionen aus.

24 Man erstelle einen Datentyp bzw. eine Klasse `SMatrix`, die effizient Matrizen in Tridiagonalform speichern kann. Insbesondere sollten folgende Operationen möglich sein:

- Initialisierung: Mit einer festen Anzahl an Zeilen bzw. Spalten und Nulleinträgen.
- Der Zugriff auf beliebige Matrixeinträge.
- Das dazuaddieren eines Wertes auf einen beliebigen Eintrag in der Diagonale bzw. den Nebendiagonalen.

Man implementiere eine Methode `void TestSMatrix()` die eine Tridiagonalmatrix erstellt und einige der implementierten Funktionen aufruft.

25 Man erstelle eine Funktion

```
void ElementStiffnessMatrix(double xa, double xb, Mat22& elMat);
```

die für ein gegebenes Element $T_K = (x_{k-1}, x_k) = (\texttt{xa}, \texttt{xb})$ einer Triangulierung die Element-Steifigkeitsmatrix

$$K_h^{(k)} = \int_{T_k} \begin{pmatrix} \varphi'_{k-1}(x) \\ \varphi'_k(x) \end{pmatrix} \begin{pmatrix} \varphi'_{k-1}(x) \\ \varphi'_k(x) \end{pmatrix}^\top \mathrm{d}x = \frac{1}{h_k} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

für stückweise lineare und stetige Funktionen berechnet und in dem Matrixtyp `elMat` speichert. Weiters implementiere man eine Methode `void TestElementMatrix()` welche eine Element-Steifigkeitsmatrix berechnet und die berechnete Werte in der Konsole ausgibt.

26 Man implementiere eine Funktion

```
void ElementLoadVector(RealFunction f, double xa, double xb,
                       Vec2& elVec);
```

welche für eine gegebene Funktion $\texttt{f} = f \in \mathcal{C}[0, 1]$ und für ein gegebenes Element $T_K = (x_{k-1}, x_k) = (\texttt{xa}, \texttt{xb})$ eine Approximation des Element-Lastenvektors

$$\underline{f}_h^{(k)} = \int_{T_k} f(x) \begin{pmatrix} \varphi_{k-1}(x) \\ \varphi_k(x) \end{pmatrix} \mathrm{d}x$$

mit Hilfe der Trapezregel berechnet und im Datentyp `elVec` speichert. Weiters implementiere man eine Methode `void TestElementVector()` welche für ein gegebenes Element den Element-Lastenvektor berechnet und die berechneten Werte in der Konsole ausgibt.