Serie 2, Teil 1

Zielstellung dieser Serie

- Erstellen von Block- oder Mehrfach-Diagonal-Matrizen als sparse und vollbesetzte Matrizen
- Erarbeiten des linearen Gleichungssystems um das Poisson-Problem mittels finiter Differenzen zu lösen

Theoretischer Hintergrund — Definition der Block-Diagonal-Matrizen

Wir fixieren Zahlen $d, n \in \mathbb{N}$ mit $n \geq 2$ und es sei $N := (n-1)^d$. Betrachtet werden Matrizen $A^{(d)} \in \mathbb{R}^{N \times N}$, die bei der Finiten-Differenzen-Diskretisierung des Poisson-Problems in d Raumdimensionen mit n Punkten in jeder Dimension auftreten. Diese Matrizen sind rekursiv definiert. Dafür sei

$$A_1^{(d)} := \begin{bmatrix} 2d & -1 & 0 & \dots & 0 \\ -1 & 2d & -1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & -1 & 2d & -1 \\ 0 & 0 & \dots & -1 & 2d \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}$$

Für $\ell = 1, \dots, d-1$ sei $\mathcal{I}_\ell$ die Einheitsmatrix in $\mathbb{R}^{(n-1)^\ell \times (n-1)^\ell}$ und für $\ell = 2, \dots, d$ sei

$$A_\ell^{(d)} := \begin{bmatrix} A_{\ell-1}^d & -\mathcal{I}_{\ell-1} & 0 & 0 & \cdots & 0 \\ -\mathcal{I}_{\ell-1} & A_{\ell-1}^d & -\mathcal{I}_{\ell-1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\mathcal{I}_{\ell-1} & A_{\ell-1}^d & -\mathcal{I}_{\ell-1} \\ 0 & \cdots & 0 & 0 & -\mathcal{I}_{\ell-1} & A_{\ell-1}^d \end{bmatrix} \in \mathbb{R}^{(n-1)^\ell \times (n-1)^\ell}.$$

Dann sind $A^{(d)} := A_d^{(d)}$ die gesuchten Matrizen (vgl. dazu auch Teil 2 dieser Serie).

Die Schrittweite h der Finiten-Differenzen-Diskretisierung ergibt sich als $h = 1/n$. Das heißt für gute Approximationen werden große Werte für n und folglich auch für N nötig. Das Abspeichern von $(N \times N)$-Matrizen, wie z.B. $A^{(d)}$, erfordert im Allgemeinen einen Speicherbedarf von $\mathcal{O}(N^2)$.

Andererseits sind die konkreten Matrizen $A^{(d)}$ dünn besetzt, d.h. sehr viele der N^2 Einträge sind gleich Null. Der Speicherbedarf lässt sich in diesen Fällen reduzieren, indem nur die nicht-Null-Einträge und deren Position in der Matrix abgespeichert werden. Solche Formen der Repräsentation vom Matrizen werden als *sparse* bezeichnet.

Aufgaben Teil 1 — Block- oder Mehrfach-Diagonal-Matrizen

Aufgabe 2.1: Implementierung

Implementieren Sie in einem Modul `block_matrix.py`:

- eine Klasse `BlockMatrix(d, n)`, deren Objekte je eine der Matrizen $(A^{(d)}, d = 1, 2, 3)$ in Abhängigkeit von d und n darstellen.
- eine Methode `get_sparse()`, die die Koeffizientenmatrix $A^{(d)}$ als *sparse*-Matrix zurück gibt.
- eine Methode `eval_zeros()` welche die absolute und relative Anzahl der *nicht-Null-Einträge* bzw. *Null-Einträge* der Matrix zurück gibt.
- eine Funktion `main()` demonstriert die vollständige Funktionalität ihrer Implementierung gemäß Aufgabenstellung.

Die genauen Schnittstellen für die vorgegebenen Programmelemente finden Sie in der Datei `block_matrix.py` vgl. Moodle bzw. nachfolgend.

```python
class BlockMatrix:
    """ Represents block matrices arising from finite difference approximations
    of the Laplace operator.

    Parameters
    ----------
    d : int
        Dimension of the space
    n : int
        Number of intervals in each dimension

    Attributes
    ----------
    d : int
        Dimension of the space
    n : int
        Number of intervals in each dimension
    """

    def __init__(self, d, n):
        pass

    def get_sparse(self):
        """ Returns the block matrix as sparse matrix.

        Returns
        -------
        scipy.sparse.csr_matrix
            block_matrix in a sparse data format
        """

    def eval_zeros(self):
        """ Returns the (absolute and relative) numbers of (non-)zero elements
        of the matrix. The relative number of the (non-)zero elements are with
        respect to the total number of elements of the matrix.

        Returns
        -------
        int
            number of non-zeros
        int
            number of zeros
        float
            relative number of non-zeros
        float
            relative number of zeros
        """
```