Aufgabe1

October 22, 2017

1 Aufgabe 1

1.1 Ziel

Lösen Sie das Randwertproblem für die gewöhnliche Differentialgleichung

$$\begin{aligned} -u''(x) &= f(x) \quad \text{für } x \in (0,1), \\ u(x) &= g(x) \quad \text{für } x \in \{0,1\}. \end{aligned}$$

mit Hilfe eines Finite-Differenzen Verfahrens.

Zu gegebenem $N \in \mathbb{N}$ definieren wir $h = \frac{1}{N}$ und die Stützstellen $x_i = i\,h$, $i = 0, \dots, N$ und approximieren

$$\begin{aligned} u(x_i) &\approx u_i && i = 0, \dots, N, \\ f(x_i) &\approx f_i && i = 0, \dots, N, \\ u''(x_i) &\approx \frac{1}{h^2}(u_{i-1} - 2u_i + u_{i+1}) && i = 1, \dots, N-1. \end{aligned}$$

Dies führt auf ein lineares Gleichungssystem mit einer tridiagonalen Matrix. Assemblieren Sie diese dünnbesetzte Matrix, lösen Sie das lineare Gleichungssystem mit Hilfe des CG Verfahrens und visualisieren Sie das Ergebnis für verschiedene Werte von N.

Ferner werde mit u_h die stückweise lineare Interpolierende der u_i, d.h. $u_h(x_i) = u_i$ bezeichnet und sei e_h der L^2-Fehler

$$e_h^2 := \int_0^1 \|u - u_h\|^2.$$

Berechnen Sie für die angegebenen Werte von N den Fehler e_h die experimentelle Konvergenzordnung

$$EOC_i = \log\Big(\frac{e_{h_i}}{e_{h_{i-1}}}\Big) \Big/ \log\Big(\frac{h_i}{h_{i-1}}\Big)$$

1.2 Beispiel: Dünnbesetzte Matrizen

Exemplarisch werden wie die 3-dimensionale Einheitsmatrix assemblieren.

```
In [1]: from dune.istl import BCRSMatrix

        shape = (3, 3)
        avg_row_entries = 1
        overflow_factor = 0.1
```

```
        matrix = BCRSMatrix(shape, avg_row_entries, overflow_factor, BCRSMatrix.implicit)

        for i in range(3):
            matrix[i, i] = [[1]]
        _ = matrix.compress()
```

Nun Lösen wir das Gleichungssystem

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

mittels des CG Verfahrens und eines Jacobi Vorkondidionierers.

```
In [2]: from dune.istl import BlockVector

        b = BlockVector(3)
        for i in range(3):
            b[i] = [i+1]
        x = BlockVector(3)

        print("b = (" + ", ".join(str(b[i]) for i in range(3)) + ")")
        print("x = (" + ", ".join(str(x[i]) for i in range(3)) + ")")
```

```
b = ((1.000000), (2.000000), (3.000000))
x = ((0.000000), (0.000000), (0.000000))
```

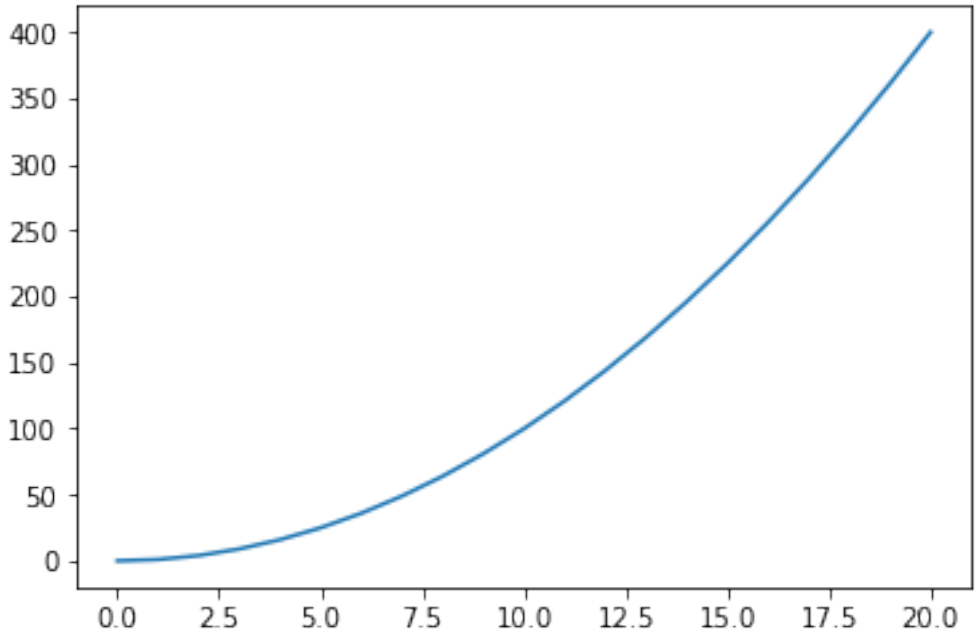
```
In [3]: from dune.istl import SeqJacobi, CGSolver

        solver = CGSolver(matrix.asLinearOperator(), SeqJacobi(matrix), 1e-10)
        _ = solver(x, b)

        print("b = (" + ", ".join(str(b[i]) for i in range(3)) + ")")
        print("x = (" + ", ".join(str(x[i]) for i in range(3)) + ")")
```

```
b = ((0.000000), (0.000000), (0.000000))
x = ((1.000000), (2.000000), (3.000000))
```

1.3 Beispiel: Visualisierung von Knotenwerten

Visualisieren wir Werte der Funktion $f(x) = x^2$ an den Knoten $x = 0, ..., 20$:

```
In [4]: from matplotlib import pyplot
        xs = list(range(21))
        ys = [x*x for x in xs]
        pyplot.figure()
        pyplot.plot(xs, ys)
        pyplot.show()
```



1.4 Beispiel: Darstellung von Tabellen

Wir werden eine kleine Tabelle mit den Werten x_i, x_i^2 und x_i^3 erstellen, etwa

x	y	z
2	4	8

```
In [5]: xs = list(range(5))
        ys = [x**2 for x in xs]
        zs = [x**3 for x in xs]

        from IPython.display import display,Markdown
        table = "| x | y | z |\n|:---:|:---:|:---:|\n"
        for i in range(5):
            table += "|" + str(xs[i]) + "|" + str(ys[i]) + "|" + str(zs[i]) + "|\n"
        display(Markdown(table))
```

x	y	z
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64

2 Lösung