

1x Gruppe PÜ: → E-Mail

A • No lecture this week

• Recap: klassifizierung skalare PDEs

$$u: \mathbb{R}^n \rightarrow \mathbb{R}^3 \quad \begin{cases} = 1 \\ = 2 \\ = 3 \end{cases}$$

Einschub: Hyperbolizität Quasi-Lineares Systeme 1. Ordnung (G+HA)

→ Quasi-Lin. System 1. Ordnung:

$$\partial_t u + \sum_{k=1}^n \partial_x^{(k)} f^{(k)}(u) = \partial_t u + \operatorname{div} \vec{f}(u) = 0$$

$$\text{mit Erhaltungsgröße } u: \Omega \subset \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^N \text{ und } \vec{f}(u) = \begin{bmatrix} f^{(1)}(u) \\ \vdots \\ f^{(N)}(u) \end{bmatrix}$$

heißt System von Erhaltungsgleichungen.

→ Eigenschaften:

- $N \hat{=} \text{Anzahl Erhaltungsgrößen (Masse, Energie, ...)}$
- $n \hat{=} \text{Anzahl Raumdimensionen } (n=3, n=2)$
- $f^{(k)}: \mathbb{R}^N \rightarrow \mathbb{R}^n \hat{=} \text{Flussfunktionen}$

→ Das System heißt hyperbolisch falls \forall Richtungen $\vec{z} \in \mathbb{R}^n$, $\|\vec{z}\|_2 = 1$

die gerichtete Transportmatrix $A(u) = \left(\sum_{k=1}^n z_k \cdot \frac{\partial f^{(k)}}{\partial u_i} \right)_{i=1 \dots n}$ ist.

$$= z_1 \cdot \frac{\partial f^{(1)}}{\partial u_i} + z_2 \cdot \frac{\partial f^{(2)}}{\partial u_i} \quad (n=2)$$

Beispiel: $n=2, N=3$, $u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$, $\vec{f}(u) = \begin{bmatrix} f^{(1)}(u) \\ f^{(2)}(u) \end{bmatrix} = A \cdot u$

$$\text{mit } A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 4 \end{bmatrix}, B = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 4 & 0 \end{bmatrix}$$

$$\Rightarrow \partial_t u + \operatorname{div} \vec{f}(u) = \partial_t \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \partial_x \left(A \cdot u \right) + B \cdot \partial_x u$$

$\stackrel{n=2}{\Rightarrow \{x_1, x_2\}}$

$$\frac{\partial f^{(1)}(u)}{\partial u_i} = A$$

$$f^{(2)}(u)$$

HA:
"Rückwärts"

$A \neq A(x)$

3 Gl.

$$\begin{aligned} I) &\rightarrow \dots \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 4 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{cases} \partial_t u_1 + \partial_x(u_1 + 4u_3) + \partial_y(u_2) = 0 \\ \partial_t u_2 + \partial_x(u_2 + u_3) + \partial_y(u_3) = 0 \\ \partial_t u_3 + \partial_x(4u_1 + 4u_2) + \partial_y(u_2) = 0 \end{cases} \\ II) &= \partial_t \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 4 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + B \cdot \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}}_u = \begin{cases} \partial_t u_1 + \partial_x(u_1 + u_2) + \partial_y(u_2) = 0 \\ \partial_t u_2 + \partial_x(u_2 + u_3) + \partial_y(u_3) = 0 \\ \partial_t u_3 + \partial_x(4u_1 + 4u_2) + \partial_y(u_2) = 0 \end{cases} \\ III) &= \dots \end{aligned}$$

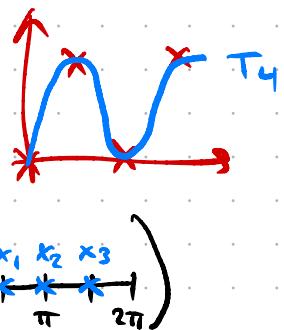
$$A(u) = \sum_{k=1}^2 z_k \left(\frac{\partial f^{(k)}}{\partial u_i} \right) = z_1 \cdot A + z_2 \cdot B = \begin{bmatrix} z_1 & z_1 + z_2 & 0 \\ 0 & z_1 & z_1 + z_2 \\ 0 & 4(z_1 + z_2) & z_1 \end{bmatrix}$$

HA!

3-Richtungen

$$\sigma(A) = \{z_1, z_1 - 13, z_1 + 3_2, z_1 + (z_1 + z_2)\} \cdot \text{Bei } [z_1 = -3_2] \text{ alle EW gleich aber } \dim(\operatorname{ER}(S)) = 3 \Rightarrow \text{Digbar} \Rightarrow \text{HYP}$$

N Discrete Fourier Transform (DFT)



→ Recap: Trigonometrische Interpolation mit Coeffs $d_j(y)$

$$d_j(y) = \frac{1}{n} \sum_{k=0}^{n-1} y_k e^{-ijk \cdot x_k}, \quad x_k = \frac{2\pi k}{n} \quad \left(x_0, x_1, x_2, x_3 \right)$$

$$\text{E}_4 \stackrel{\text{def}}{=} \text{4-te Einheitswurzel} = (\text{E}_4)^{k \cdot i} = \left(e^{-\frac{(2\pi i)}{4}} \right)^{k \cdot i}$$

$$n\text{-te EH: } E_n = e^{-\frac{2\pi i k}{n}}$$

DFT: $y: \mathbb{C}^n \rightarrow \mathbb{C}^n, y \mapsto \hat{y} = (d_0(y), \dots, d_{n-1}(y))$ easy.

→ Als Linearer Operator: $\hat{y} = \frac{1}{n} \mathcal{B} \cdot y$

$$\text{mit } \mathcal{B} = \begin{bmatrix} b^0 & b^1 & \cdots & b^{n-1} \end{bmatrix} \quad \text{mit } b^j = \begin{bmatrix} e^{-2\pi i j \cdot 0/n} \\ e^{-2\pi i j \cdot 1/n} \\ \vdots \\ e^{-2\pi i j \cdot (n-1)/n} \end{bmatrix}$$

Beispiel: $n=4, y = \begin{bmatrix} 4 \\ 2 \\ 1 \\ 2 \end{bmatrix}, \hat{y} = ?$

$$E_4 = e^{-\frac{2\pi i}{4}} = e^{-\frac{\pi}{2}i} = -i = \cos(\frac{\pi}{2}) + i \cdot \sin(\frac{\pi}{2})$$

$$b^0 = \begin{bmatrix} E_4^{0 \cdot 0} \\ E_4^{0 \cdot 1} \\ E_4^{0 \cdot 2} \\ E_4^{0 \cdot 3} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$b^1 = \begin{bmatrix} E_4^{1 \cdot 0} \\ E_4^{1 \cdot 1} \\ E_4^{1 \cdot 2} \\ E_4^{1 \cdot 3} \end{bmatrix} = \begin{bmatrix} (-i)^0 \\ (-i)^1 \\ (-i)^2 \\ (-i)^3 \end{bmatrix} = \begin{bmatrix} 1 \\ -i \\ -1 \\ i \end{bmatrix}$$

$$b^2 = \begin{bmatrix} E_4^{2 \cdot 0} \\ E_4^{2 \cdot 1} \\ E_4^{2 \cdot 2} \\ E_4^{2 \cdot 3} \end{bmatrix} = \begin{bmatrix} (-i)^0 \\ (-i)^2 \\ (-i)^4 \\ (-i)^6 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, \quad b^3 = \begin{bmatrix} \text{Mind the pattern...} \\ \text{MIND THE GAP} \end{bmatrix} = \begin{bmatrix} (-i)^0 \\ (-i)^3 \\ (-i)^6 \\ (-i)^9 \end{bmatrix} = \begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix}$$

$$\mathcal{B} = \begin{bmatrix} b^0 & \cdots & b^3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

Side
GUE

$$\hat{y} = \frac{1}{4} \mathcal{B} \cdot y = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 9 \\ 3 \\ 1 \\ 3 \end{bmatrix} \quad (\text{vgl. SEL} \neq 1) \quad \checkmark$$

Besser:
 $\mathcal{B}^{-1} = n \cdot \bar{\mathcal{B}}^T$

→ Inverse DFT (IDFT): Naiv über $\hat{y} = \frac{1}{n} \mathcal{B} \cdot y \Leftrightarrow y = n \cdot \mathcal{B}^{-1} \cdot \hat{y}$

Fast Fourier Transform (FFT)

Ziel: Wir wollen $d_j(y) = \frac{1}{n} \sum_{x=0}^{n-1} y_x \cdot e^{-ij \cdot x \cdot \underbrace{\varepsilon_n}_{E_n}} \stackrel{!}{=} \$ [DFT]FT$

Sei nun $n = 2^L$, z.B. $n = 8$, $n = 2^{10} = 1024$ Merkhilfe

Vorgehen: Zerteile $d = (d_0, d_1, \dots, d_{n-2}, d_{n-1})$ in $d_{2k} = (d_0, d_2, \dots, d_{n-2})$

$d_{2k+1} = (d_1, \dots, d_{n-1})$

$n = 2m$

$$\text{Beobachtung: } d_{2k}(y) = \frac{1}{2m} \sum_{j=0}^{2m-1} y_j \cdot \underbrace{\varepsilon_{2m}}_{E_n}^{j \cdot 2k}$$



$$= \frac{1}{2m} \left[\sum_{j=0}^{m-1} y_j \cdot \varepsilon_{2m}^{j \cdot 2k} + \sum_{j=m}^{2m-1} y_j \cdot \varepsilon_{2m}^{j \cdot 2k} \right] \quad (\text{legit})$$

Indexshift: $j \mapsto j+m$
(Alle j ersetzen)

$$= \frac{1}{2m} \left[\sum_{j=0}^{m-1} y_j \cdot \varepsilon_{2m}^{j \cdot 2k} + \sum_{j=0}^{m-1} y_{j+m} \cdot \varepsilon_{2m}^{(j+m) \cdot 2k} \right]$$

Spare Zeit & nutze Symmetrien:

$$\varepsilon_{2m}^{j \cdot 2k} \stackrel{\text{def.}}{=} e^{\left(-\frac{i2\pi}{2m}\right) \cdot 2k \cdot j} = e^{-\frac{(2\pi i)}{m} \cdot kj} = \varepsilon_m^{jk}$$

$$\varepsilon_{2m}^{(j+m) \cdot 2k} = \varepsilon_{2m}^{j \cdot 2k} \cdot \varepsilon_{2m}^{m \cdot 2k} = \varepsilon_m^{jk} \cdot \varepsilon_m^k = \varepsilon_m^{jk} \cdot e^{-2\pi i \cdot k} = 1$$

Daten:

$$d_{2k}(y) = \frac{1}{2m} \left[\sum_{j=0}^{m-1} y_j \cdot \varepsilon_m^{jk} + \sum_{j=0}^{m-1} y_{j+m} \cdot \varepsilon_m^{jk} \right]$$

$$= \frac{1}{m} \left[\sum_{j=0}^{m-1} \left[\frac{1}{2} (y_j + y_{j+m}) \right] \cdot \varepsilon_m^{jk} \right]$$

$$= FFT \left(\frac{1}{2} (y_j + y_{j+m}) \right)$$

$$= \varepsilon_2^j \cdot \varepsilon_{2m}^j = (-1) \varepsilon_{2m}^j$$

Analog

$$\varepsilon_{2m}^{(2k+l)(m+s)} = \varepsilon_{2m}^{2k(m+s)} \cdot \varepsilon_{2m}^{m+s} = \underbrace{\varepsilon_{2m}^{2km}}_{=1 \text{ (s.o.)}} \cdot \underbrace{\varepsilon_{2m}^{2kj}}_{\varepsilon_m^{kj} \text{ (s.o.)}} \cdot \varepsilon_{2m}^{m+j} = \varepsilon_m^{kj} \cdot \varepsilon_{2m}^j$$

Daher analog: $d_{2k+n}(y) = \dots = \frac{1}{2^m} \left(\sum_{j=0}^{m-1} y_j e_{2m}^{j \cdot (2k+1)} + \sum_{j=0}^{m-1} y_{j+m} e_{2m}^{(j+m)(2k+1)} \right)$

$$= \frac{1}{m} \left(\sum_{j=0}^{m-1} \left[\frac{1}{2} (y_j - y_{j+m}) \cdot e_{2m}^{jk} \right] \cdot e_m^{jk} \right)$$

$$= \text{FFT}\left(\frac{1}{2}(y_j - y_{j+m}) \cdot e_{2m}^j\right)$$

Algorithmen:

```

def FFT([y]):
    if n < threshold:
        return DFT([y])
    else:
        a = 1/2 FFT([y_k + y_{k+n/2}])
        b = 1/2 FFT([e^{-2\pi i k/n} \cdot (y_k - y_{k+n/2})])
        return [a[0], b[0], ..., a[n/2-1], b[n/2-1]]

```

For Speedup Sometimes

Explore Jupyter Notebook

- Wikipedia Def. vs. Lecture ("=> know the Def.")
- Implement DFT & IDFT using Linear Operator
 - ↳ Compare with Numpy & last weeks example
- Implement FFT & compare with DFT (\Rightarrow No Error)
 - ↳ Test speed DFT vs. FFT (Do Live)
- Use FFT to extract dominant frequencies (+ Noise)
- Show Convolution (Faltung) Multiplication
- Motivate optimization by comparing with np.fft.FFT