Binärdarstellung:

0	1	0	0	1	1	1	0	$= 78_{\text{dez}} = 4E_{\text{hex}}$
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	

8 bit = 1 Byte
16 bit = 1 Wort
1024 Byte = 1 kByte

ganze Zahlen: Integer; Fließkommazahlen: Float.

Dynamik eines Zahlenvorrats: Verhältnis vom größten zum kleinsten Betrag ($\neq 0$).

Auflösung eines Zahlenvorrats: Abstand benachbarter Zahlen.

Floats: $z = (-1)^s\ m\ 2^e$ mit 64 bit:

IEEE-Standard

Signum	s	1bit
Mantisse	m	52 bit
Exponent	e	11 bit

64 bit:
Auflösung $\approx 10^{-16}$
Dynamik: $\approx 10^{-308} \rightarrow 10^{308}$
$1 \leq m \leq 2$!

Diskretisierung: $f(x) \Rightarrow f_i$ über verschiedene Methoden (Wert von $f(x)$ an x_i, Mittelwert im Intervall um x_i, Entwicklungskoeffizienten in Funktionensystem (Fourier, Taylor, ...)).

Diskretisierungsfehler (sofern Operation $O[f(x)]$ bekannt): $E_h = |O[f(x)] - D_h[f_i]|$

Zu fordernde Eigenschaften diskreter Operatoren:
$D_h[f_i]$ konvergiert (für $\lim_{h\to 0}$) und $D_h[f_i]$ ist konsistent (konvergiert gegen den richtigen Wert).

Differentiation: linksseitig: $\frac{1}{h}[-1\ 1\ 0]$; rechtsseitig: $\frac{1}{h}[0\ -1\ 1]$; besser: $\frac{1}{2h}[-1\ 0\ 1]$; noch besser:

Stencil-Notation: $\frac{1}{12h}[1\ -8\ 0\ 8\ -1] := \frac{1}{12h}\left(1f(x-2h) - 8f(x-h) + 0f(x) + 8f(x+h) - 1f(x+2h)\right)$.
(Stencil = Schablone, engl.)

2. Ableitung (schon recht ungenau): $\frac{1}{h^2}[1\ -2\ 1]$

Spektralverhalten: $f(x) = \mathrm{e}^{\mathrm{i}kx}$; Ableitung: $D_h[\mathrm{e}^{\mathrm{i}kx}] = \dfrac{\mathrm{e}^{\mathrm{i}k(x+h)} - \mathrm{e}^{\mathrm{i}k(x-h)}}{2h} = f'(x)\ \mathrm{sinc}(kh)$;
deren Fehler: $E_h = |f'(x)\ (1 - \mathrm{sinc}(kh))| \Rightarrow h$ muss klein gegen Periode sein.

Integration:

Rechteckregel: $\overline{f}(x)_i \approx f(x_i + \frac{h}{2}) := f_{i+\frac{1}{2}}$, $\displaystyle\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{N}\sum_{i=0}^{N-1} f_{i+\frac{1}{2}}$; Konvergenz in 2. Ordnung.

Trapezregel: $\displaystyle\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{N-1} h\frac{f_i + f_{i+1}}{2} = \frac{b-a}{N}\left[\frac{f_0}{2} + f_1 + f_2 + ... + f_{N-1} + \frac{f_N}{2}\right]$; Konvergenz in 2. Ordnung.

Der Fehler der Trapezregel ist bei gleichem h doppelt so groß; diese lässt sich aber rekursiv besser auf $2N$ verfeinern.

Simpson-Regel: $\displaystyle\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=1,3,5,...}^{...,N-3,N-1}\left[2hf_i + \frac{h^3}{3}\left(\frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}\right)\right] = \frac{h}{3}\left[f_0 + 4f_1 + 2f_2 + 4f_3 + ... + 4f_{N-1} + f_n\right]$;

Konvergenz in 4. Ordnung.

Intervallteilungsverfahren:

1. Vorgabe einer Zielgenauigkeit ε
2. N nach Physik wählen, A_N berechnen
3. Anzahl der Teilungen verdoppeln, bis $|A_N - A_{2N}| < \varepsilon$. (dabei muss N entsprechend groß sein, so dass Schwankungen in der Konvergenz schon vorbei.)

Gauß-Quadratur: $f(x) \approx \displaystyle\sum_{i=0}^{n} \alpha_i P_i(x)$ mit angepasst gewählten n Stützstellen; zur Bestimmung der α_i Orthogonalität der P_i ($\int_a^b \mathrm{d}x P_i(x)\,P_j(x) = \delta_{ij}\ c$, $c \in \mathbb{R}$) nutzen, dann:

$$\int_a^b \mathrm{d}x f(x) \approx \sum_{i=0}^{n} \alpha_i \underbrace{\int_a^b \mathrm{d}x\ P_i(x)}_{\text{analytisch bekannt}}$$

Nullstellen:

- $\text{sign}(f(a)) \cdot \text{sign}(f(b)) \leq 0$, mit $[a,b]$ Intervall; dieses immer weiter halbieren, bis $|a-b| \leq \varepsilon$ ($\varepsilon \approx$ Maschinengenauigkeit);
- Sekantenverfahren: $x_{i+1} = x_i - f(x_i)\dfrac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$, wobei zusätzlich: $|f(x_i) - f(x_{i-1})| \leq 10\varepsilon|f(x_i)|$ (damit der Nenner nicht zu klein wird, muss nicht 10 sein).
- Regula Falsi: Sekantenverfahren, nur dass das Intervall mit Nullstelle ausgewählt wird: $f(x_{i+1}) \cdot f(x_i) < 0 \Rightarrow (x_{i+1}, x_i)$, $f(x_{i+1}) \cdot f(x_{i-1}) < 0 \Rightarrow (x_{i+1}, x_{i-1})$, $(f(x_{i+1}) = 0 \Rightarrow$ Ende).
- Newton-Raphson: $x_{i+1} = x_i - \dfrac{f(x_i)}{f'(x_i)}$ wenn $f(x)$ im Intervall gut <u>linear</u> näherbar; Fehler: $E_{i+1} \approx E_i^2 \frac{f''(x_i)}{2f'(x_i)}$.

<u>Kombination</u>: erst Regula Falsi, dann Newton-Raphson liefert schnell ein gutes Ergebnis.

Minima: notwenidg $x_a < x_b < x_c$ mit $f(x_a) > f(x_b) \wedge f(x_c) > f(x_b)$:

- Goldener Schnitt: gleichbleibende Intervallreduktion mit $\frac{x_c - x_b}{x_b - x_a} = \frac{1+\sqrt{5}}{2} \approx 1{,}618$.
- Quadratische Interpolation: Parabel in die 3 Punkte, Scheitel als neuen Iterationspunkt:

$$x = b + \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(a)]}$$

Mehrdimensionale Minimierung:
Simplex: minimale Anzahl von durch Linien verbundener Punkte, die den gesamten N-dimensionalen Raum aufspannen.
Intervallschachtelung: Wandern & Kontrahieren.
Verschiedene Methoden: „Downhill-Simplex-Methode“, „Minimierung in alternierende Richtungen“, „Gradientenverfahren“, „Powell's Methode“.

Interpolation: $P_{n-1}(x) = \sum_{j=1}^{n} y_j \prod_{k=1, k\neq j}^{n} \left(\frac{x - x_k}{x_j - x_k}\right)$

1. Ordnet n Punkten ein Polynom $(n-1)$-ten Grades zu.
2. Schlängelt stark, divergiert außerhalb des Testintervalls, stößt schnell ($n \gtrsim 100$) gegen Zahlenformatgrenzen.

Spline-Interpolation: Auf Teilintervallen Interpolation mit Polynomen geringen Grades (s_j), glatt aneinander setzen ($\Rightarrow s_j(x_j) = y_j$, $s_j(x_{j+1}) = y_{j+1}$, ${s_j}'(x_j) = {s_{j-1}}'(x_j)$, ${s_j}''(x_j) = {s_{j-1}}''(x_j)$, ${s_1}''(x_1) = {s_{n-1}}''(x_n) = 0$).
$\Rightarrow 4(n-1)$ Gleichungen zur Bestimmung von $4(n-1)$ Variablen.

Rationale Approximation: Die Funktion weist Polstellen auf $\Rightarrow$ Ansatz: $y(x) = \dfrac{P_n(x)}{Q_m(x)}$ mit P/Q Polynomen vom Grad n/m. Bei N Stützstellen sei $N = m + n + 1$, dann ist $y(x)$ eindeutig bestimmt. Anzahl der Pole $\leq m$ und für große x: $y \sim x^{n-m}$ ($\Rightarrow$ Problemangepasst n,m wählen!). Dann $P_n(x_k) = y_k \cdot Q_m(x_k) : \forall k = 1, ..., N$ lösen.

Lineare Gleichungssysteme: $\hat{A}\,\vec{x} = \vec{b}$, mit $\hat{A} \in \mathbb{R}^m \times \mathbb{R}^n$, $\vec{x} \in \mathbb{R}^n$, $\vec{b} \in \mathbb{R}^m$.

- $m > n$: überbestimmtes System $\rightarrow$ entweder keine Lösung oder Regression,

- $m < n$: unterbestimmtes Problem $\rightarrow$ Lösungsraum,

- $m = n$: (wahrscheinlich numerisch) lösbar.

Sind die Zeilen bzw. Spalten von $\hat{A}$ nicht linear unabhängig, so heißt $\hat{A}$ „Zeilen- / Spaltendegeneriert“.

Matrixtypen: hermitesche Matrizen ($\overline{\hat{A}^t} = \hat{A}$), positiv definite Matrizen ($\vec{v}\,\hat{A}\,\vec{v} > 0 : \forall \vec{v}$), tridiagonale Matrizen (Diagonale und ersten Nebendiagonalen $\neq 0$), Bandmatrizen (Diagonale und symmetrisch dazu einige $\neq 0$), Blockmatrizen (nur überlappende Blöcke $\neq 0$), Sparsematrizen (sehr viele 0 enthalten).

Gauß-Jordan-Verfahren: $\left(\hat{A}\,\vec{b}\right)$ durch elementare Zeilen-/Spaltenoperationenen (Vertauschen, Multiplizieren mit $\lambda \neq 0$, Addieren verschiedener) strenge Stufenform ($\left(\hat{E}\,\vec{x}\right)$) erhalten (dies ist bei tridiagonalen Matrizen recht einfach).

Dazu **Pivotisierung**:
Möglichst große Werte an die ersten Stellen der Zeilen der Matrix, damit man beim Subtrahieren des Vielfachen der ersten Zeilen von den anderen zum Erreichen der Stufenform jeweils mit möglichst kleinen numerischen Fehlern rechnet.

LU-Zerlegung (Lower- / Upper-):
Man stellt $\hat{A} = \hat{L}\,\hat{U}$ dar und löst dann $\hat{A}\vec{x} = \hat{L}(\hat{U}\,\vec{x}) = \hat{L}\vec{y} = \vec{b}$, mit $\hat{L}$ oberer Dreiecksmatrix, $\hat{U}$ unterer Dreiecksmatrix nach $\vec{y}$ auf: $y_1 = \dfrac{b_1}{L_{11}}$, $y_i = \dfrac{1}{L_{ii}}\left[b_i - \sum_{j=1}^{i-1} L_{ij}y_j\right]$, $i > 1$.

Schließlich löst man $\hat{U}\vec{x} = \vec{y}$ nach $\vec{x}$ auf: $x_n = \dfrac{y_n}{U_{nn}}$, $x_i = \dfrac{1}{U_{ii}}\left[y_i - \sum_{j=i+1}^{n} U_{ij}x_j\right]$.

$\hat{L}$ und $\hat{U}$ kann man über den *Crout's Algorithmus* berechnen:
Da $\hat{L}$ und $\hat{U}$ zusammen $n^2 + n$ unbestimmte Einträge haben, $\hat{A}$ aber nur n^2, kann man n Stück frei wählen. So wählt man die Diagonalelemente von $\hat{L}$ alle zu 1 und hat dann ein einfaches Gleichungssystem zu lösen:

$$L_{ii} = 1\,, \forall i = 1,\dots,n \quad , \quad \forall j = 1,\dots,n: \quad \begin{array}{ll} i = 1,\dots,j & U_{ij} = A_{ij} - \sum_{k=1}^{i-1} L_{ik}U_{kj} \\ i = j+1,\dots,n & L_{ij} = \frac{1}{U_{jj}}(A_{ij} - \sum_{k=1}^{j-1} L_{ik}U_{kj}) \end{array} .$$

Iterative Verbesserung:
$\hat{A}\,\vec{x} = \vec{b}$ liefert ein numerisch verfälschtes Ergebnis $\vec{x}\,'$. Dann $\hat{A}\vec{\delta x} = \hat{A}\vec{x}\,' - \vec{b}$ nach $\vec{\delta x}$ lösen und dieses von $\vec{x}\,'$ abziehen $\Rightarrow$ so oft, bis $\vec{\delta x}$ klein genug!

Zur Lösung von Eigenwertproblemen gibt es zwei Verfahrensarten: - iterative Methode (z.B. Jacobi-Methode) .
- direkte Verfahren (z.B. Householder-Methode)

Diskrete Fourier-Transformation:
$\tilde{f}(k) = a\sum_x \mathrm{e}^{-\mathrm{i}kx}\, f(x)$
$f(x) = \frac{1}{L}\sum_k \mathrm{e}^{\mathrm{i}kx}\, \tilde{f}(k)$
mit $\{x\} = \{0, a, 2a, \dots, L-a\}$,
$\{k\} = \{0, 1\frac{2\pi}{L}, 2\frac{2\pi}{L}, \dots, \overbrace{\frac{L-a}{a}}^{N-1}\frac{2\pi}{L}\}$
auf Periode der Länge L, Schrittweite a.
Diese ist energieerhaltend, benötigt ein periodisches oder periodisch fortsetzbares Problem, ist in der Notation unterschiedlich (problemangepasst) und benötigt so für N Stützstellen in m Dimensionen N^{2m} Multiplikationen!

Fast Fourier Transformation:
Rückführung durch $\log_2(N)$ Intervallteilungen in Stützstellen gerader und ungerader Nummerierung auf N eindimensionale Probleme;
unter Rückgriff auf schon bekannte Ergebnisse durch spezielle Reihenfolge: Nur noch $N\,\log_2(N)$ Operationen.

Sampling- / Abtast- / Nyquisttheorem: $\nu_{\mathrm{abtast}} \geq 2\,(\nu_{\max} - \nu_{\min})$.
Nyquist-Frequenz: $\nu_{\mathrm{Ny}} = \frac{1}{2}\nu_{\mathrm{abtast}}$.

Differentialgleichungen:
Allgemeine Form: $\dfrac{\partial^k f(t)}{\partial t^k} = G(f, f', f'', \dots, f^{(k-1)}, t)$;

a) Anfangswertaufgaben:
Gegeben: $f(t=0), f'(t=0), f''(t=0), \dots, f^{(k-1)}(t=0)$;
Transformation in System gekoppelter DGL erster Ordnung:

$$\begin{aligned} \frac{\partial f_1(t)}{\partial t} &= G_1(f_1, f_2, \dots, f_k, t) \\ \frac{\partial f_2(t)}{\partial t} &= G_2(f_1, f_2, \dots, f_k, t) \\ &\vdots \\ \frac{\partial f_k(t)}{\partial t} &= G_k(f_1, f_2, \dots, f_k, t) \end{aligned} .$$

Diskretisierung der Ableitungen; Ausführen von N Integrationsschritten mit Δt $(f(t) \to f_n$ mit $f_n := f(n \cdot \Delta t))$.

Fehler: lokale Fehler (Fehler eines Einzelschritts), globale Fehler (Summe der lokalen Fehler), Rundungsfehler.

Stabilität:

- harmonischer Oszillator und **Vorwärts-Euler**: Eigenwerte der Iterationsmatrix $> 1 \Rightarrow$ wie Energiegewinn mit der Zeit (physikalisch ungeeignet).
- harmonischer Oszillator und **Rückwärts-Euler**: Iterationsmatrix für jeden Schritt über iterative Lösung zu bestimmen (aufwändig), aber Eigenwerte betragsmäßig $< 1 \Rightarrow$ Energieverlust (eher physikalisch $\to$ Reibung).

Alpha-Verfahren:
Aus gewichteter Kombination von Vorwärts- und Rückwärts-Euler kann man Energieerhaltung erzeugen; dabei muss man einen Kompromiss zwischen Genauigkeit und Stabilität (Selbstverstärkung gemachter Fehler) finden.
($\alpha = 1$ - Vorwärts-Euler; $\alpha = \frac{1}{2}$ - Crank-Nichelson-Verfahren; $\alpha = 0$ - Rückwärts-Euler)

(harmonischer Oszillator: $f_{n+1} = f_n + \Delta t\,[\alpha\, G(f_n, t) + (1-\alpha)\, G(f_{n+1}, t + \Delta t)]$)

Runge-Kutta-Verfahren:
Durch mehrere Schritte in Δt und gewichteter Kombination der Informationen an den einzelnen Stellen kann man die ersten Terme des Fehlers zu 0 Addieren. Beispielhaft sei hier ein Termschema 4. Ordnung ($\to$ globaler Fehler) und 4. Stufe (4 Schritte) gezeigt:

$$\left.\begin{array}{rcl} k_1 &=& G(t_n, f_n) \\ k_2 &=& G(t_n + \frac{1}{2}\Delta t, f_n + \Delta t\, \frac{1}{2} k_1) \\ k_3 &=& G(t_n + \frac{1}{2}\Delta t, f_n + \Delta t\, \frac{1}{2} k_2) \\ k_4 &=& G(t_n + \Delta t, f_n + \Delta t\, k_3) \end{array}\right\} f_{n+1} = f_n + \Delta t\left[\tfrac{k_1}{6} + \tfrac{k_2}{3} + \tfrac{k_3}{3} + \tfrac{k_4}{6}\right] + \mathcal{O}(\Delta t^5)$$

Der lokale Fehlerterm $\mathcal{O}(\Delta t^5)$ wirkt sich global wie $\mathcal{O}(\Delta t^4)$ aus.

Man kann dies verallgemeinern zu:

$f_{n+1} = f_n + \Delta t \sum_{j=1}^{m} b_j\, k_j$, mit $k_j = G({t_n}^j + {f_n}^j)$, wobei ${t_n}^j = t_n + c_j\, \Delta t$ und ${f_n}^j = f_n + \Delta t \sum_{l=1}^{j} a_{jl} k_l$.

Dies kann man dann im Butcher-Schema darstellen: $\begin{array}{c|c} \vec{c} & \hat{A} \\ \hline & \vec{b}^t \end{array}$,

wobei $\hat{A}$ bei expliziten Verfahren eine untere Dreiecksmatrix ist.

Für implizite Runge-Kutta-Verfahren ergibt sich: $f_{n+1} = f_n + \Delta t\, B(t_n, f_n, \Delta t)$ $\quad n = 0, 1, \ldots N-1$, mit

$$B(t_n, f_n, \Delta t) = \sum_{i=1}^{r} \beta_i\, G_{i\,n}$$

$$G_{i\,n} = G(f_n + \Delta t \sum_{j=1}^{r} \alpha_{ij}\, G_{j\,n}\,,\; t_n + \gamma_i\, \Delta t)$$

entsprechendem Koeffizienten-Tableau: $\begin{array}{c|c} \vec{\gamma} & \hat{\alpha} \\ \hline & \vec{\beta}^t \end{array}$; Genauigkeit (-sobergrenze): $p = 2r$.

Schrittweitensteuerung - $\Delta t(n)$:
Schätzung des lokalen Fehlers $l_n(\Delta t)$:

a) aus physikalischer Nebenbedingung (z.B. Energieerhaltung)

b) aus Vergleich der numerischen Lösung mit Δt mit der Lösung mit $\frac{\Delta t}{2}$:

$$l_{n+1}(\Delta t) \approx \frac{f_{n+1}(\frac{1}{2}\Delta t) - f_{n+1}(\Delta t)}{\Delta t\,[1-2^{-p}]} + \mathcal{O}(\Delta t^{p+1})$$

c) Vergleich zweier numerischer Lösungen aus Verfahren unterschiedlicher Konvergenzordnung („Fehlberg-Methode"):

$$l_{n+1}(\Delta t) \approx \frac{f_{n+1}^{(p+1)}(\Delta t) - f_{n+1}^{(p)}(\Delta t)}{\Delta t} + \mathcal{O}(\Delta t^{p+1})$$

Und schließlich muss bei vorgegebener Fehlerobergrenze ε dann $\frac{\varepsilon}{10} \leq |l_{n+1}(\Delta t)| \leq \varepsilon$ erfüllt sein.

steife Probleme:
Entweder entsprechende Skalierung der Schrittweiten, so dass Δt für alle Probleme ca. gleich; oder ein implizites Lösungsverfahren, wobei dies nur so lange, wie nötig.

b) Randwertaufgaben:
Randwerte vorgegeben; Existenz einer Lösung und deren Eindeutigkeit hängt von den Randwerten ab! Lösungsmethoden:

- „Methode der finiten Differenzen":
  Approximation auf Diskretisierungsgitter; Diskretisierung; Umschreiben in Matrix-Formalismus; Lösbar!
  (Bei nicht-linearen Differentialgleichungen können durch die Diskretisierung Fehler auftreten, so dass sich die Lösung (bei linearen $\Leftrightarrow$ Nullstellensuche) in ein Minimierungsproblem verändert.)
- „Schießverfahren":
  Umwandlung von Randwert- in Anfangswertaufgabe mit Paramterabhängigkeit; dann Suche nach Parametern, so dass die Lösung der AWA die RWA automatisch erfüllt.
- „Verbessertes Schießverfahren":
  Zerlegunge eines Integrationsintervalls in mehrere kleinere; Beachtung der Übergangsbedingungen (Stetigkeit); dies kann man dann entweder schrittweise lösen oder über das „Schießen aus zwei Richtungen".

Partielle Differentialgleichungen:
Allgemeine Form:

$$p\frac{\partial^2 f}{\partial x^2} + q\frac{\partial^2 f}{\partial x \partial y} + r\frac{\partial^2 f}{\partial y^2} + s\frac{\partial f}{\partial x} + t\frac{\partial f}{\partial y} + uf + v = 0$$

Nomenklatur:
$q^2 < 4pr$ - elliptische PDE; $q^2 = 4pr$ - parabolische PDE; $q^2 > 4pr$ - hyperbolische PDE.

Für Randwertprobleme sucht man meist statische Lösungen, die numerisch dann durch den Speicher begrenzt sind; meist elliptische PDE.

Für Anfangswertprobleme sucht man meist dynamische Lösungen, die numerisch dann durch Speicher und Zahlenformat in Zeit und Stabilität begrenzt sind; meist para- oder hyperbolische PDE.

Nach Diskretisierung mit der Methode der finiten Differenzen kann das Problem auf unterschiedliche Arten in einen Matrixformalismus überführt werden; beispielhaft für die Diffusionsgleichung $\frac{\partial f(x,t)}{\partial t} = \lambda \frac{\partial^2 f(x,t)}{\partial x^2}$ mit $f(x,0) = f_0(x)$ und der Raumzeitdiskretisierung $x_j = j \cdot \Delta x$, $t_n = n \cdot \Delta t$:

- Vorwärts-Euler: $f_j^{n+1} = f_j^n + \frac{\lambda\Delta t}{[\Delta x]^2}[f_{j+1}^n - 2f_j^n + f_{j-1}^n] \quad \Rightarrow \quad \vec{f}^n = \hat{D}^n \vec{f}^0$ mit

$$\hat{D} = \begin{bmatrix} 1-2\frac{\lambda\Delta t}{[\Delta x]^2} & \frac{\lambda\Delta t}{[\Delta x]^2} & & & 0 \\ \frac{\lambda\Delta t}{[\Delta x]^2} & 1-2\frac{\lambda\Delta t}{[\Delta x]^2} & \frac{\lambda\Delta t}{[\Delta x]^2} & & \\ & & \ddots & & \\ & & \frac{\lambda\Delta t}{[\Delta x]^2} & 1-2\frac{\lambda\Delta t}{[\Delta x]^2} & \frac{\lambda\Delta t}{[\Delta x]^2} \\ 0 & & & \frac{\lambda\Delta t}{[\Delta x]^2} & 1-2\frac{\lambda\Delta t}{[\Delta x]^2} \end{bmatrix}.$$

  Aus dem Vergleich mit der hier bestimmbaren analytischen Lösung ergibt sich die Courant-Friedrichs-Leavy-Bedingung: $\Delta t \leq \frac{[\Delta x]^2}{2\lambda}$ für Stabilität der Lösung.

- Rückwärts-Euler: $f_j^{n+1} = f_j^n + \frac{\lambda\Delta t}{[\Delta x]^2}[f_{j+1}^{n+1} - 2f_j^{n+1} + f_{j-1}^{n+1}] \quad \Rightarrow \quad \vec{f}^n = [\hat{E}^{-1}]^n \vec{f}^0$ mit

$$\hat{E} = \begin{bmatrix} 1+2\frac{\lambda\Delta t}{[\Delta x]^2} & -\frac{\lambda\Delta t}{[\Delta x]^2} & & & 0 \\ -\frac{\lambda\Delta t}{[\Delta x]^2} & 1+2\frac{\lambda\Delta t}{[\Delta x]^2} & -\frac{\lambda\Delta t}{[\Delta x]^2} & & \\ & & \ddots & & \\ & & -\frac{\lambda\Delta t}{[\Delta x]^2} & 1+2\frac{\lambda\Delta t}{[\Delta x]^2} & -\frac{\lambda\Delta t}{[\Delta x]^2} \\ 0 & & & -\frac{\lambda\Delta t}{[\Delta x]^2} & 1+2\frac{\lambda\Delta t}{[\Delta x]^2} \end{bmatrix}.$$

  Aus dem Vergleich mit der hier bestimmbaren analytischen Lösung ergibt sich, dass diese Verfahren immer eine stabile Lösung liefert; diese ist aber nicht immer korrekt.

- Leapfrog: $f_j^{n+1} = f_j^{n-1} + \frac{2\lambda\Delta t}{[\Delta x]^2}[f_{j+1}^n - 2f_j^n + f_{j-1}^n]$.

Split-Step-Verfahren:
Wirken physikalisch gleichzeitig 2 Differentialoperatoren (z.B. $\frac{\partial E}{\partial z} = [\hat{A} + \hat{B}]E$), so ergibt sich (bei nicht kommutativen Operatoren) eine Näherung, indem man schrittweise erst den einen und dann noch den anderen auf das Ergebnis des ersten anwendet.

MATLAB (7.12)

- Rechenoperationen: $+,-,*,/$
- Zuweisungsoperator: $=$
- komplex: $z = x + iy$
- Vektor: $a = [1, 2, 3]$
- Matrix: $A = [4, 5, 6; 7, 8, 9]$
- Komponenten: $a(3)$, $A(1, 2)$
- leerer Vektor: []
- transponierter Vektor: A'
- Von:Scrittweite:bis: $1 : 1 : 10$
- Ausgabeunterdrückung: ;
- Dimension: $size(A)$
- Operationen komponentenweise: $a. * b$
- $[X, Y] = meshgrid(-8 : 0.5 : 8, -8 : 0.5 : 8)$
- $mesh(X, Y, Z)$, $plot(x, y, '.')$, $surf(X, Y, Z)$
- Strings: $text = '...'$
- Struct: $s.1 = ...; s.2 = ...;$
- **function** ausgabe = Name (eingabe)
  %Kommentar
  if a > b
  ...
  elseif b == a
  ...
  else
  ...
  end
- *help* Name (erster zusammenhängender Kommentar)
- **for** ... **end** , **while** ... **end**
- continue;, **break**;
- **return**;
- display('text');, display(var);
- **cumsum**()
- **plot**(x,y,'−−k'); **legend**('name');
- **pause**(.01); in Sekunden
- **subplot**(m,n,p); $m \times n$-Aufteilung der figure, Zugriff auf p-tes
- **axis**([xmin xmax ymin ymax]);
- **clf**('reset'); löscht aktuelle figure
- func = @(x,t) name(x,t,param1, param2,...);
- **title**('...','FontSize',14);
- **xlabel**('...');, **ylabel**('...');