Ingenieurinformatik 2: Numerik

Sommersemester 2026

David Straub

Ziel dieser Lehrveranstaltung

- **Verständnis** grundlegender **numerischer Methoden** zur Lösung technisch-wissenschaftlicher Probleme
- **Anwendung** und praktische Umsetzung dieser Methoden in einer wissenschaftlichen Entwicklungsumgebung (**Matlab**)

Verhältnis zur Ingenierinformatik 1

Wir bauen auf den in Teil 1 erworbenen Kompetenzen im wissenschaftlichen Programmieren auf, insbesondere:

- Variablen, Schleifen, Funktionen
- Arbeiten mit Datenstrukturen
- Visualisierung von Funktionen

Die Konzepte in beiden Lehrveranstaltungen sind unabhängig von der verwendeten Programmiersprache/Entwicklungsumgebung (Python, Matlab) übertragbar!

Gliederung

1. **Einführung in Matlab**
2. **Arbeiten mit Arrays**
3. **Funktionen und Kontrollstrukturen**
4. **Analysis** (Polynome, Ableitung, Integration, …)

5. **Lineare Algebra** (Gleichungssysteme, Eigenwerte, …)
6. **Differentialgleichungen**
7. **Einführung in Simulink**

Organisatorisches

Ingenieurinformatik, Teilmodul 2: Numerik für Ingenieure (L1172)

- 2 SWS Seminaristischer Unterricht, wöchentlich
- 2 SWS Übung, 14-tägig (2 Gruppen, Einteilung am Ende), Start nächste Woche
- Prüfung: schriftlich, 60 Minuten, 40% der Gesamtnote für Modul Ingenieurinformatik

Prüfungsinhalte

Schriftlich, 60 Minuten – 40 % der Gesamtnote Ingenieurinformatik

Examplarische Inhalte:

- Matlab-Syntax
- Implementierung numerischer Methoden in Matlab (Analysis, lineare Algebra, Diffefrentialgleichungen, …)
- Simulink (z.B. Differentialgleichungen)

  Wer die Methoden **versteht**, kann sie in Matlab **anwenden** – beides gehört zusammen.

Hilfsmittel: Schriftliche Unterlagen

Gemeinschaftsprüfung Straub/Hirschmann/Jäger-Hezel/Krug/Selting

Warum sind die Übungen wichtig?

Sowohl Programmieren als auch die numerischen Methoden lernt man durch **Anwenden**!

Wer das Praktikum ernstnimmt, hat in der Prüfung einen deutlichen Vorteil.

Anwesenheit

Die Anwesenheit ist **freiwillig**, es wird aber eine Anwesenheitsliste per Moodle geführt, damit ich einen Überblick habe.

Einteilung der Übungsgruppen

Jetzt in Moodle: https://moodle.hm.edu/course/view.php?id=24726

Gruppe	Tag	Uhrzeit	Raum	Start
A1	Di	11:45	B355	24.3.
A2	Di	11:45	B355	31.3.
B1	Do	15:15	B350a	26.3.
B2	Do	15:15	B350a	1.3.
C1	Di	13:30	B355	24.3.
C2	Di	13:30	B355	31.3.

Bitte Gruppeneinteilung strikt einhalten.

Materialien & Kommunikation

- Moodle: Links zu Materialien, Gruppeneinteilung, …
  - https://moodle.hm.edu/course/view.php?id=24726
- Matrix: Chatgruppe für Fragen, Diskussionen, Ankündigungen, … **bitte beitreten!**
  - https://matrix.hm.edu → FK03 LRB Numerik

Übersicht Vorlesungsunterlagen

- PDF-Vorlesungsfolien von Prof. Küpper, Prof. Hirschmann u.a. (Moodle)
  - Übersicht über den Vorlesungsstoff (prüfungsrelevant!)
- Meine Folien (HTML/PDF)
  - Kein vollständiges Skript! Nur ergänzend zu den Live-Code-Beispielen in der Vorlesung
- PDF-Übungsunterlagen von Prof. Küpper, Prof. Hirschmann u.a. (Moodle)
  - Wird für die Übungsgruppen verwendet. Falls Sie in 90 Minuten nicht fertig werden,

bitte nacharbeiten – beste Prüfungsvorbereitung!

Motivation

Nochmal das Ziel:

- **Verständnis** grundlegender **numerischer Methoden** zur Lösung technisch-wissenschaftlicher Probleme
- **Anwendung** und praktische Umsetzung dieser Methoden in einer wissenschaftlichen Entwicklungsumgebung (**Matlab**)

Was heißt numerische Methoden?

Uns was ist eigentlich Matlab?

Was ist Numerik?

Numerik = Mathematische Methoden zur **Lösung von Problemen mit dem Computer**

- Keine analytische Lösung existiert
- Problem zu komplex für exakte Rechnung
- Nur Näherungen sind praktisch möglich
- **Überall im Ingenieuralltag: FEM, CFD, Regelung, Optimierung, ...**

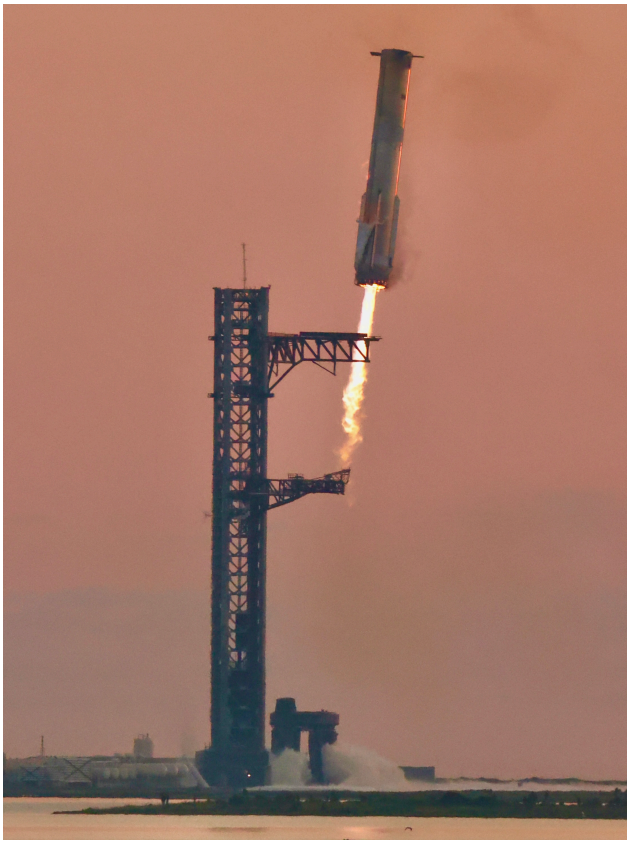
Numerische Methoden in dieser Lehrveranstaltung

- Lineare Algebra
- Interpolation
- Nullstellenbestimmung
- Integration
- Ableitung
- Differentialgleichungen

Diese Methoden sind überall!

Beispiel wiederverwendbarer Launcher

Der Bordrechner berechnet in **jedem Zeitschritt (~10 ms)** den optimalen Schubvektor – durch Lösung eines **linearen Gleichungssystems** in Echtzeit.



Zustandsraumdarstellung

Systemzustand und Dynamik in Matrixform:

$$\dot{\mathbf{q}} = A\,\mathbf{q} + B\,u$$

Größe	Bedeutung	Beispiel (Booster)
$\mathbf{q}$	Zustandsvektor	Position, Geschwindigkeit, Kippwinkel, Winkelrate
A	Systemmatrix (Physik)	Kopplung von Kippwinkel → Horizontalbewegung
B	Eingangsmatrix	Wirkung des Schubvektors auf den Zustand
u	Stellgröße	Gimbalwinkel der Triebwerksdüse

Regelgesetz: $u = -K\,\mathbf{q}$ — K wird mit Methoden der **linearen Algebra** berechnet

Numerische Methoden beim autonomen Landen

Konzept	Einsatz
Lineare Algebra	Berechnung von K; Kalman-Filter (Zustandsschätzung)
Differentialgleichungen	Trajektorienprädiktion im Regler
Integration	Numerische Lösung der Bewegungsgleichungen
Interpolation	Aerodynamik-Kennfelder (Widerstand, Auftrieb)

Drohne – Ladezustandsschätzung

Batteriebetriebene Drohne: Der Ladezustand (SOC) muss jederzeit bekannt sein – für Reichweitenplanung und Sicherheitsabschaltung.

Problem: SOC ist nicht direkt messbar → Schätzung aus Strom $I(t)$, Spannung $U(t)$, Temperatur T

Coulomb-Counting: Integration des Stroms über die Zeit:

$$\mathrm{SOC}(t) = \mathrm{SOC}(t_0) + \frac{1}{C_\mathrm{nenn}} \int_{t_0}^{t} I(\tau)\, d\tau$$

Da $I(t)$ nur als diskrete Messwerte vorliegt → **numerische Integration** erforderlich.

SOC-Schätzung: Genauere Methoden

Problem Coulomb-Counting: Messfehler akkumulieren sich → Drift

Lösung – Kalman-Filter: Kombiniert Integration mit einem Batteriemodell (elektrisches Ersatzschaltbild)

Numerische Methode	Einsatz
Integration	Coulomb-Counting (SOC-Basisschätzung)
Differentialgleichungen	Dynamik des RC-Batteriemodells

Numerische Methode	Einsatz
Interpolation	Kennfeld: Leerlaufspannung $U_{\mathrm{OCV}}(\mathrm{SOC}, T)$
Lineare Algebra	Kalman-Filter-Gleichungen

Was ist Matlab?

> Matlab ist eine proprietäre Programmiersprache und Entwicklungsumgebung des Unternehmens MathWorks zur Lösung mathematischer Probleme und zur grafischen Darstellung der Ergebnisse.

Wikipedia

Matlab: grobe Analogie zu den Python-Tools aus Teil 1

Funktionalität	Matlab	Python
Numerische Berechnungen	Matlab (Sprache)	Python + NumPy
Interaktive Eingabe	Matlab Command Window	Python-Terminal
Plotten	Matlab Plot	Matplotlib
Entwicklungsumgebung	Matlab Desktop	z.B. VS Code
Interaktive Notebooks	Matlab Live Editor	Jupyter Notebooks
Erweiterungen	Toolboxes	externe Pakete
Skript-Dateien	.m-Dateien	.py-Dateien

Matlab in dieser Lehrveranstaltung

Wir nutzen Matlab, um fürs Ingenieurswesen relevante numerische Probleme zu lösen und die zugrundeliegenden Methoden zu verstehen: Wann wende ich was an? Warum funktioniert es? Welche Fallstricke gibt es?

Die Details der Software erarbeiten Sie sich im Praktikum und zu Hause unter Verwendung der Dokumentation!

Empfohlene Ressourcen: - Matlab für Python-Nutzer (Cheat Sheet) - Matlab PDF-Dokumentation - z.B. Matlab Elementarbuch - `help <Funktion>` im Command Window

Installation von Matlab

- Anleitung: https://collab.dvb.bayern/display/HMUT/MATLAB
- Registrierung bei MathWorks mit Hochschul-E-Mail-Adresse
- Alternative zur Installation: Matlab Online https://matlab.mathworks.com/

1. Einführung in Matlab

Matlab – Entwicklungsumgebung

- **Editor**: Skripte und Funktionen schreiben
- **Command Window**: Befehle eingeben, Ergebnisse und Fehlermeldungen anzeigen, Skripte ausführen
- **Current Folder**: Arbeitsverzeichnis – Zugriff auf `.m`-Dateien
- **Workspace**: Liste der aktuellen Variablen und ihrer Eigenschaften
- **Code Analyzer**: Hinweise auf Fehler und Warnungen im Editor

Matlab – Interaktiver Modus (Taschenrechner)

- Das **Command Window** kann wie ein Taschenrechner verwendet werden
- Ausdruck eingeben → Matlab führt ihn aus und zeigt das Ergebnis an
- Ergebnis wird in Standardvariable `ans` (answer) gespeichert, oder in einer benannten Variablen

Grundrechenarten: `+ - * / ^` (Potenzierung)

Elementare Funktionen: `sin cos tan exp log log10 sqrt abs mod sign`

> Winkel in trigonometrischen Funktionen werden im **Bogenmaß** (Radiant) angegeben.
> Alle Funktionen können auf Skalare, Vektoren und Matrizen angewendet werden.

Matlab – Variablen

- Variable wird durch Zuweisung mit `=` erzeugt – keine explizite Deklaration nötig
- Standard-Datentyp: **`double`** (64-Bit Gleitkomma)

```
>> a = 5.7
>> b = 99
```

Im Unterschied zu Python sind `a` und `b` beide `double` (vgl. `float` in Python) – auch `b = 99` ist **kein** Ganzzahl-Typ!

Matlab – Eingebaute Funktionen

Matlab bietet zahlreiche eingebaute Funktionen:

Kategorie	Beispiele
Trigonometrie	`sin`, `cos`, `tan`, `asin`, `atan2`
Exponential	`exp`, `log`, `log10`, `log2`, `sqrt`
Rundung	`floor`, `ceil`, `round`, `abs`

Gefahr: Funktionen überschreiben

```
sin = 5;          % sin ist jetzt eine Variable – die Funktion sin() ist weg!
sin(pi/2)         % gibt 5, nicht 1 zurück
```

In Python schützt der Namensraum (`import numpy as np`) vor versehentlichem Überschreiben. In Matlab teilen Variablen und Funktionen denselben Namensraum.

Matlab – Workspace bereinigen

Befehl	Wirkung
`clear`	alle Variablen aus dem Workspace löschen
`clear x y`	nur Variable `x` und `y` löschen
`clc`	Command Window leeren (Variablen bleiben erhalten)
`close all`	alle Abbildungsfenster schließen (Variablen bleiben erhalten)

Hinweis: `clear; clc; close all;` am Skriptanfang ist verbreitet, aber zerstört den Workspace des Aufrufers – sinnvoll für eigenständige Skripte, problematisch wenn das Skript von anderen aufgerufen wird.

Matlab – Datentypen

Matlab-Typ	Beschreibung	Python (NumPy)
`double` **(Standard)**	64-Bit Gleitkomma	`np.float64`
`single`	32-Bit Gleitkomma	`np.float32`
`int8/16/32/64`	vorzeichenbehaftete Ganzzahlen	`np.int8/16/32/64`
`uint8/16/32/64`	vorzeichenlose Ganzzahlen	`np.uint8/16/32/64`

Matlab-Typ	Beschreibung	Python (NumPy)
`complex`	komplexe Zahlen (double-Basis)	`np.complex128`

Der fundamentale Datentyp ist das **n-dimensionale Array**, z.B. eine 2-dimensionale $m \times n$-Matrix. Selbst ein Skalar ist technisch eine Matrix der Größe 1×1. Es gibt keine echten Skalare oder 1D-Arrays. Höherdimensionale Arrays (Rang 3, 4, …) sind ebenfalls möglich.

```
size(A)     % Größe aller Dimensionen  →  A.shape
ndims(A)    % Anzahl der Dimensionen   →  A.ndim
```

Matlab – Arrays: Definition

```
>> A = [2,3; 4,5]     % 2×2-Matrix
A =
     2     3
     4     5
>> x = [6; 7]          % Spaltenvektor (2×1)
>> y = A * x           % Matrix-Vektor-Multiplikation
y =
    33
    59
```

- `[ ]` – Array-Konstruktor (zum Erstellen von Arrays)
- `;` – trennt Zeilen, `,` oder Leerzeichen trennen Elemente innerhalb einer Zeile
- `*` führt Matrizenmultiplikation durch

Matlab – Arrays: Zugriff auf Elemente

```
>> A = [2,3; 4,5]
>> A(1,2)         % Zeile 1, Spalte 2
ans =
     3
>> A(2,3)         % Fehler: außerhalb der Matrix
Index exceeds matrix dimension
```

- Zugriff: `A(zeile, spalte)`
- Bei Vektoren: `x(zeile)` bzw. `y(spalte)` (linear indexing)
- Bei Skalaren: `z(1,1)` `z(1)` `z`
- Indizes beginnen bei **1** (nicht 0 wie in Python)

Matlab – Arrays: Änderung der Dimension

```
>> x = 2          % 1×1-Matrix
>> x(2) = 12      % automatisch zu 1×2 erweitert: [2, 12]
>> x(2,3) = 17    % automatisch zu 2×3 erweitert
x =
     2    12     0
     0     0    17
```

- **Lesender Zugriff** auf nicht-existierendes Element → Fehler
- **Schreibender Zugriff** → Matrix wird automatisch erweitert, neue Elemente = 0
- **Achtung:** Kann leicht zu schwer findbaren Fehlern führen!

Zusammenfassung & Ausblick

Numerik – Mathematische Methoden zur Lösung von Problemen mit dem Computer **Matlab** – Programmiersprache und Entwicklungsumgebung für numerische Berechnungen

To Do

- Matlab installieren
- Hausaufgaben Kapitel 1 (PDF auf Moodle)

Nächste Woche

- Kapitel 2: Arbeiten mit Arrays
- Beginn Übungsbetrieb