Technische Universität Berlin

# Grundlagen wissenschaftlicher Programmierung

Christian Hennig | PC-Pool Physik | Templates, numerische Ableitung und Integration

Technische Universität Berlin

# Inhalt I

# Organisatorisches

## Termine für die Rücksprachen

Die mündlichen Rücksprachen finden vormittags statt und dauern ca. 30 Minuten. Folgende Tage stehen zur Auswahl:

- Dienstag, 25. Juli
- Mittwoch, 26. Juli
- Donnerstag, 27. Juli
- Dienstag, 1. August
- Mittwoch, 2. August
- Donnerstag, 3. August

Schicken Sie mir bitte bis 10. Juli eine E-Mail mit Ihrem Wunschtermin.

Technische Universität Berlin

# Funktions-Templates: Motivation

## Drei Funktionen:

```
float max (float a, float b)   // Erwartet zwei 'float'
{
  return ((a > b) ? a : b);    // Gibt groessere Zahl zurueck
}

int max (int a, int b)         // Erwartet zwei 'int'
{
  return ((a > b) ? a : b);    // Gibt groessere Zahl zurueck
}

char max (char a, char b)      // Erwartet zwei 'char'
{
  return ((a > b) ? a : b);    // Gibt groessere Zahl zurueck
}
```

- Unterschiede?
- Gemeinsamkeiten?

Technische Universität Berlin

# Funktions-Templates

## Definition

Ein Funktions-Template ist eine Vorlage für gleichartige Funktionen.

```
template <typename T> T max (T a, T b)
{
  return ((a > b) ? a : b);
}
```

- Festlegung des Platzhalters in Form eines formalen Datentyps: `<typename T>`
- Austausch des realen Datentyps (`float`, `int`, `char`) durch den formalen Datentyp `T` (➪ Name beliebig)
- Auch gebräuchlich: `<class T>` statt `<typename T>`

Technische Universität Berlin

# Funktionsaufruf

```
template <typename T> T max (T a, T b)
{
  return ((a > b) ? a : b);
}

floatmax = max(float x, float y);
intmax = max(int i, int j);
charmax = max(char s, char t);
```

- Zuerst wird geprüft, ob eine Funktionsdefinition existiert, die zu den angegebenen Parametern passt.
- Wenn keine passende Funktionsdefinition existiert, wird nach einem Funktions-Template gesucht.
- Wird ein Template gefunden, wird der formale Datentyp durch den tatsächlichen Datentyp ersetzt und die Funktion erstellt.

# Überschreiben von Funktions-Templates

Was passiert hier?

```
template <typename T> T max (T a, T b)
{
  return ((a > b)? a : b);
}

char* name1;
char* name2;
... //
pmax = max(name1, name2)
```

Die Zeiger werden miteinander verglichen:

```
// Vom Kompiler generierte Funktion
char* max (char* a, char* b)
{
  return ((a > b) ? a : b);
}
```

Das ist nicht das beabsichtigte Verhalten dieses Templates!

# Überschreiben von Funktions-Templates

Technische Universität Berlin

Funktions-Templates können daher für bestimmte Datentypen spezialisiert werden:

```
template <typename T> T max (T a, T b)
{
  return ((a > b)? a : b);
}

template <> char* max <char*>(char* p1, char* p2)
{
  if (strcmp(p1,p2) > 0)
    return p1;
  else
    return p2;
}
```

# Lokale Daten in Funktions-Templates

Der formale Datentyp T kann auch lokal verwendet werden:

```
// Definition der Template-Funktion
template <class T> T swap (T& a, T& b)
{
  T temp = a;
  a = b;
  b = temp;
}
```

# Mehrere Parameter I

Ein Template kann auch mehrere formale Parameter haben:

```
template <class T1, class T2>
void funktion (T1 par1, T2 par2)
{
  T1 lokal1 = par1;
  T2 lokal2 = par2;
  ... //
}

float zahl;
char* text;
funktion(zahl, text);

// Vom Kompiler generierte Funktionen
void funktion (float par1, char* par2)
{
  float lokal1 = par1;
  char* lokal2 = par2;
  ... //
}
```

# Mehrere Parameter II

Ein Template kann auch »normale« Parameter haben:

```
template <typename T> void funktion (T par1, int par2 = 42)
{
  ... //
}
```

Technische Universität Berlin

# Klassen-Templates: Motivation

```
class ZahlenStapel       // Stack fuer Zahlen
{
  short *zahlen;
  public:
  ZahlenStapel(int size)
  {
    zahlen = new short[size];
  }
};

class MitarbeiterStapel  // Stack fuer Zeiger
{
  Mitarbeiter* stapel;
  public:
  MitarbeiterStapel(int size)
  {
    stapel = new Mitarbeiter*[size];
  }
};
```

# Klassen-Templates

- `ZahlenStapel`
  Klassendefinition für einen Stack aus Zahlen
- `MitarbeiterStapel`
  Klassendefinition für einen Stack aus Zeigern
- Unterschiedliche Daten, aber ähnliche Funktionalität:
  - Speicherplatz anfordern, freigeben
  - Elemente zum Stapel hinzufügen
  - Elemente vom Stapel entfernen
  - ...
- »Identische« Memberfunktionen
- Definition eines Klassen-Templates sinnvoll.

# Definition von Klassen-Templates

```
template <typename T> class Stapel
{
  T* daten;
  ... //
  public:
  Stapel(int size)
  {
    ... //
  }

  bool push(const T& wert)
  {
    daten[index++] = wert;
  }

  bool pop(T& wert)
  {
    wert = daten[--index];
  }
};
```

# Verwendung von Klassen-Templates

Technische Universität Berlin

## Template- und Objektdefinition

```
template <typename T> class Stapel
{
  T* data;
  public:
  Stapel(int size);
  ... //
};

// Objektdefinition + Template-Instanziierung
Stapel<double> doubleStapel(13);
Stapel<char*> charStapel(42);
```

- Der Name eines Klassen-Templates gefolgt von einem Datentyp in spitzen Klammern <> ist der Name einer Klasse und kann wie ein Klassenname benutzt werden.

Technische Universität Berlin

# Weitere Eigenschaften

- Überschreiben von Template-Memberfunktionen
- Verwendung mehrerer formaler Datentypen

```
template <typename T1, typename T2> class Zeugs
{
  ... //
};

Zeugs<unsigned int, char*> objekt1;
Zeugs<float, double> objekt2;
```

- Voreingestellter Datentyp

```
template <typename T=int> class Stapel
{
  ... //
};

Stapel<> intStapel;
Stapel<float> floatStapel;
```

Technische Universität Berlin

# Template-Bibliotheken

Es gibt bereits zahlreiche fertige Templates in der STL (Standard Template Library) von C++, zum Beispiel unterschiedliche Container:

| | |
|---|---|
| `vector` | eindimensionales Feld |
| `list` | doppelt verkettete Liste |
| `queue` | Queue |
| `deque` | Queue mit zwei Enden |
| `stack` | Stapel |
| `map` | Assoziatives sortiertes Feld |
| `set` | Menge |
| `bitset` | Feld von boolschen Werten |

Eine umfangreiche Alternative ist die Boost C++-Bibliothek: `www.boost.org`

Technische Universität Berlin

# Numerische Ableitung

## Definition der Ableitung

$$f'(x) = \frac{f(x+h) - f(x)}{h} \qquad \text{für} \qquad h \to 0 \tag{1}$$

1. Wähle kleines $h$.
2. $f(x)$ ausrechnen.
3. $f(x+h)$ ausrechnen.
4. Formel 1 anwenden.
5. Fertig!
6. Probleme?

# Berechnung des Differenzenquotienten

Technische Universität Berlin

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

- Unkritische Verwendung führt mit hoher Wahrscheinlichkeit zu ungenauen Ergebnissen.
- Verwendung in »besonderen« Situationen:
  - $f(x)$ ist sehr aufwendig zu berechnen,
  - $f(x)$ wurde bereits verwendet und soll wiederverwendet werden,
  - die Ableitung soll nur einen weiteren Rechenschritt erfordern.
- Die Herausforderung ist, das richtige $h$ zu finden.
- Fehlerquellen:
  - Abbruchfehler $e_t$ (*truncation error*)
  - Rundungsfehler $e_r$ (*roundoff error*)

Technische Universität Berlin

# Abbruchfehler

Der Abbruchfehler $e_t$ (*truncation error*) entsteht zum Beispiel durch

- die Diskretisierung kontinuierlicher Prozesse
- die »endliche« Auswertung eines »unendlichen« Prozesses

Für den Differenzenquotient folgt $e_t$ aus den höheren Termen der Taylor-Reihe:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \ldots$$

so dass

$$\frac{f(x+h) - f(x)}{h} = f' + \frac{1}{2}hf'' + \ldots$$

# Rundungsfehler

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

## Fehler in $h$

- Beispiel: Ableitung bei $x = 10.3$ mit $h = 0.0001$
- $x$, $h$ und $x + h$ können binär nicht exakt dargestellt werden.
- Jeder Wert hat einen von $\epsilon_m$ (Maschinengenauigkeit) abhängigen Fehler (`float`: $\epsilon_m \sim 10^{-7}$)
- Fehler effektiver Wert von $h$: Größenordnung $\epsilon_m x$
- Fehler von $h$: $\sim \frac{\epsilon_m x}{h} \sim 10^{-2}$

# Rundungsfehler

**Wahl von $h$**

Die Werte für $h$ und $x$ sollten so gewählt werden, dass die Differenz zwischen $x$ und $x+h$ binär dargestellt werden kann.

- Lösung:

```
temp = x + h;
   h = temp - x;
```

- Könnte vom Kompiler »wegoptimiert« werden.
- Funktioniert nicht, wenn intern mit höherer Genauigkeit gerechnet wird.
  - `temp` als `volatile` deklarieren oder
  - leere Funktion `donothing(temp)` definieren und zwischendurch aufrufen.

Technische Universität Berlin

# Rundungsfehler

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

- Rundungsfehler bei »exaktem« $h$: $e_r \sim \epsilon_f \left|\frac{f(x)}{h}\right|$
- Für einfach Funktionen gilt: $\epsilon_f \approx \epsilon_m$.
- Mit $e_t \sim |hf''(x)|$ ergibt sich für das optimale $h$:

$$h \sim \sqrt{\frac{\epsilon_f f}{f''}} \approx \sqrt{\epsilon_f} x_c$$

mit $x_c$ als Krümmungsmaßstab $(f/f'')^{\frac{1}{2}}$ von $f$.

- Maximale Genauigkeit Differenzenquotient: $\sqrt{\epsilon_m}$

## Symmetrische Form

Technische Universität Berlin

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

- Abbruchfehler $e_t \sim h^2 f'''$
- Rundungsfehler unverändert
- Für das optimale $h$ ergibt sich

$$h \sim \left(\frac{\epsilon_f f}{f'''}\right) \sim (\epsilon_f)^{1/3} x_c$$

- Bis zu zwei Größenordnungen besser als im nicht-symmetrischen Fall (`double`).
- Ableitung bleibt trotzdem deutlich schlechter als $\epsilon_m$ oder $\epsilon_f$.

# Alternativen

Technische Universität Berlin

## Differentiationsmethoden

- Differentiation eines Interpolationspolynoms
- Differentiation einer interpolierenden Splinefunktion
- Romberg-Verfahren (Richardson-Extrapolation)
- Adaptive numerische Differentiation

# Numerische Integration

## Motivation

Die Berechnung von

$$\int_a^b f(x)\mathrm{d}x$$

einer gegebenen reellen Funktion ist für einige $f(x)$ manchmal kein Problem, nämlich wenn das unbestimmte Integral

$$\int f(x)\mathrm{d}x = F(x), \quad F'(x) = f(x)$$

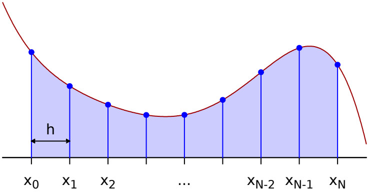
durch algebraische und bekannte transzendente Funktionen von $x$ ausgedrückt werden kann.

- Und wenn das nicht funktioniert?

# Numerische Integration

Technische Universität Berlin

- In der Regel: Berechnung bestimmter Integrale mittels Diskretisierung
- Aufteilung des Integrationsintervalls $[a, b]$ (Partitionierung)
- Approximation durch endliche Summe
- Auch: »Numerische Quadratur«



# Übersicht

## Eindimensionale Integration

- Basiert auf einer Multiplikation $f(x)\Delta x$
  - ➪ Effizienz?
- Verwende möglichst wenig Stützstellen
  - ➪ Genauigkeit vorgeben

## Näherung

- Inter-/Extrapolationsformeln verwenden
  - zum Beispiel Chebyshev-Polynome
  - ➪ Teilweise direkt integrierbar

# Übersicht

Technische Universität Berlin

## Uneigentliche Integrale

Wie erfolgt die numerische Berechnung von

$$\int_0^\infty f(x)\mathrm{d}x \quad ?$$

- Aufteilen des Intervalls $[0, \infty)$ in gleichmäßige Abschnitte?
- Besser: Vorgehen in zwei Schritten.
  1. Aufteilen des Integrationsbereichs in $[0, 1]$ und $[1, \infty)$.
  2. Geeignete Substitution im Intervall $[1, \infty)$ mit $y = 1/x$:

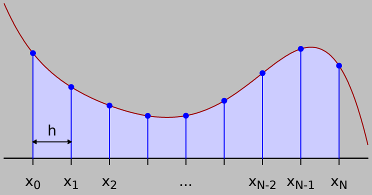
$$\int_1^\infty f(x)\mathrm{d}x = -\int_1^0 f\left(\frac{1}{y}\right)\frac{\mathrm{d}y}{y^2}$$

Technische Universität Berlin

# Konstante Schrittweite

## Klassische Formeln

- Auswertung von $f(x_i)$ mit $x_i = x_0 + ih$ und $i \in \{0, N\}$.




- Geschlossene Formeln verwenden $x_0, x_1, \ldots, x_{N-1}, x_N$
- Offene Formeln verwenden $x_1, \ldots, x_{N-1}$

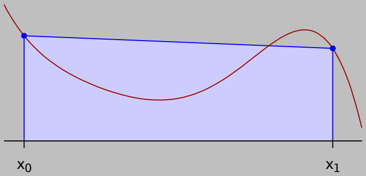
# Trapezregel

Technische Universität Berlin

## Allgemein

- Exakt für Polynome der Ordnung 1, also für $f(x) = ax + b$

$$\int_{x_0}^{x_1} f(x)\mathrm{d}x = h\left(\frac{1}{2}f_0 + \frac{1}{2}f_1\right) \qquad + \mathcal{O}(h^3 f'')$$



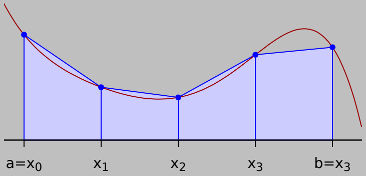

- Fehler proportional zu $h^3 = (b-a)^3$

# Trapezregel

## Beispiel

- Fortgesetzte Trapezregel

$$\int_{x_0}^{x_N} f(x)\mathrm{d}x = h\left(\frac{1}{2}f_0 + f_1 + \cdots + f_{N-1} + \frac{1}{2}f_N\right)$$



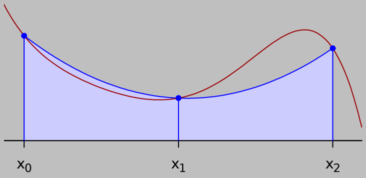

- Verallgemeinerung für alle Arten von Funktionen

# Simpson-Regel

## Allgemein

- Exakt für Polynome der Ordnung 3

$$\int_{x_0}^{x_2} f(x)\mathrm{d}x = h\left(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{1}{3}f_2\right) \qquad + \mathcal{O}(h^4 f^{(4)})$$




- Nur ein Schritt mehr als für die Trapezregel (Symmetrie)

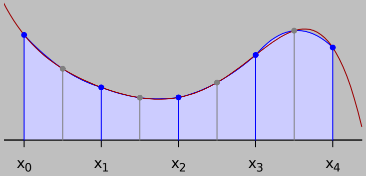
# Simpson-Regel

Technische Universität Berlin

## Beispiel

- Fortgesetzte Simpson-Regel für gerades $N$

$$\int_{x_0}^{x_N} f(x)\mathrm{d}x = \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{N-2} + 4f_{N-1} + f_N\right)$$



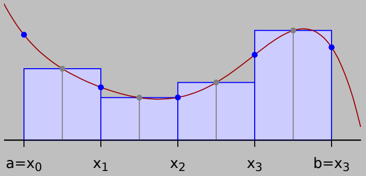
# Offene Integrationsregeln

## Beispiel

- Integral wird durch $x_1, \ldots, x_4$ abgeschätzt

$$\int_{x_0}^{x_5} f(x)\mathrm{d}x = h\left(\frac{55}{24}f_1 + \frac{5}{24}f_2 + \ldots\right) \qquad + \mathcal{O}(h^5 f^{(4)})$$

- Nützlich für Integrale mit Singularitäten an den Enden
- Mittelpunktsregel für offene Integrationsintervalle:



# Approximationsfehler

- Fehler proportional zu $h$: Genauigkeit steigt für kleinere $h$
  - ➪ Genauer Faktor ist aber unbekannt
- Wie klein muss $h$ für die gewünschte Genauigkeit sein?
  - ➪ Lässt sich erst hinterher sagen. . .
- Wichtig: Bereits berechnete Schätzwerte behalten
  - ➪ Klasse erstellen, um frühere Ergebnisse zu speichern

# Adaptive Schrittweite

Technische Universität Berlin

- Einfaches Prinzip: Seien $\mathcal{I}_1$ und $\mathcal{I}_2$ zwei Abschätzungen, dann kann die Differenz $|\mathcal{I}_1 - \mathcal{I}_2|$ zur Fehlerabschätzung verwendet werden.
- Ist der Fehler kleiner als $\varepsilon$, akzeptiere $\mathcal{I}_1$ (vorausgesetzt, $\mathcal{I}_1$ ist genauer als $\mathcal{I}_2$)
- Ist der Fehler zu groß, teile den Bereich gemäß

$$\mathcal{I} = \int_a^b f(x)\mathrm{d}x = \int_a^m f(x)\mathrm{d}x + \int_m^b f(x)\mathrm{d}x$$

  mit $m = (a+b)/2$ als Mittelpunkt des Intervalls $[a, b]$.
- Für beide Teilintegrale $\mathcal{I}_1$ und $\mathcal{I}_2$ bestimmen und separate Fehler abschätzen
- Anschließend rekursiv weiter teilen, sofern nötig.

# Adaptive Schrittweite – Abbruchkriterium

- Beenden, wenn der Beitrag zum tolerierten Fehler $\varepsilon$ klein genug ist.
  - ➪ Integral über $(b-a)/n$ darf höchstens $\varepsilon/n$ beitragen.
  - ➪ Es wird nur dort verfeinert, wo es nötig ist.
- Kein Schema funktioniert mit **allen** Funktionen
  - ➪ »Gutartige« Funktionen (differenzierbar) sind unproblematisch

Technische Universität Berlin

# Mehrdimensonale Integration

- Betrachte

$$\mathcal{I} = \iiint_V f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = \int_{x_0}^{x_1} \mathrm{d}x \int_{y_o}^{y_1} \mathrm{d}y \int_{z_0}^{z_1} \mathrm{d}z\, f(x,y,z)$$

- Bestimme zunächst

$$G(x,y) = \int_{z_0(x,y)}^{z_1(x,y)} f(x,y,z)\,\mathrm{d}z,$$

- und dann

$$H(x) = \int_{y_0(x)}^{y_1(x)} G(x,y)\,\mathrm{d}y.$$

- Das verbleibende eindimensionale Integral liefert das Ergebnis.

# Monte-Carlo-Integration

Technische Universität Berlin

- Integration entspricht einer Flächenberechnung
- Fläche unter/über der Funktion mit zufälligen Punkten abtasten
- Wähle $N$ Punkte im Intervall $\mathcal{I} = [a, b] \times [f(c_-), f(c_+)]$ mit $f(c\mp) = \min, \max(f(a), f(b))$
- Notiere für jeden zufälligen Punkt $(x_i, y_i) \in \mathcal{I}$, ob $y_i \leq f(x_i)$ gilt
- Daraus folgt das Integral

$$\boxed{\frac{F}{(b-a)(f(c_+) - f(c_-))} \approx \frac{\text{Anzahl } y_i \leq f(x_i)}{N}}$$

- Essentielle Bedingung: Zufallsgenerator besteht den Spektraltest
- Mehr Punkte liefern ein besseres Ergebnis
- Fehler ist die Standardabweichung $\sqrt{N}$

# Monte-Carlo-Integration

- Mehrdimensionale Integration
- Hier: Approximation von $2\pi$
- Keine sinnvolle Methode für »wenige« Dimensionen