Zusammenfassung vom 22.04.2020

Nullstellensuche

Bisektionsverfahren - Teilen eines Startintervalls, in dem Vorzeichen von *f* wechselt
Newtonverfahren - lineare Näherung an die Funktion (benötigt Ableitung)
Sekantenverfahren - lineare Näherung an die Funktion (numerische Ableitung)

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Bisektionsverfahren erfordert viele Schritte
Newton/Sekantenverfahren konvergieren nicht/schlecht in der Nähe von Extrema

Numerische Integration: Romberg Verfahren

basiert auf Trapezregel
Integrationsfehler wird zu h → 0 **extrapoliert**
höhere Genauigkeit bei vergleichsweise wenigen Stützstellen
Möglichkeit **adaptiven Code** zu schreiben
Extrapolation wurde mit Hilfe eines Polynoms durchgeführt:
Neville Schema zur Bestimmung des Polynoms

Gauß Integration

Ein Nachteil des Romberg Verfahrens ist, dass es keinen Satz Stützpunkte gibt. Die **Gauß Integration** ist eine Methode, die zu **Stützstellen und Gewichten** führt. Für glatte Funktionen erhält man hier eine extrem gute Näherung an das Integral.

Bisher einfache Verfahren (Trapez, Simpson, …):

- äquidistante Stützstellen
- sehr hohe Ordnungen sind nicht brauchbar, da Gewichte negativ werden

Jetzt zeigen wir, dass durch nicht äquidistante Stützstellen auch bei hohen Ordnungen rein positive Gewichte gefunden werden.

Problemstellung: Finde optimale x_i und ω_i, so dass:

$$\int_a^b dx\ w(x) f(x) \approx \sum_{i=0}^{n-1} \omega_i\ f(x_i) \quad \text{für} \quad w(x) \geq 0$$

Falls f ein Polynom vom Höchstgrad $2n-1$ ist : mit x_i als Nullstellen eines Polynoms vom Grad N ist die Approximation exakt!

Definieren zunächst einen Satz **orthogonaler Polynome**:

$$\text{Skalarprodukt:} \quad (f, g) \equiv \int_a^b dx \; w(x) \; f(x)g(x)$$

Betrachten wir den Vektorraum Π_{n-1} der Polynome vom Grad ≤ n-1:

Durch **Gram-Schmidt Orthogonalisierung** kann man eine orthogonale Basis von Π_{n-1} erhalten

$$p_{i+1}(x) = (x - a_i)\, p_i(x) - b_i \; p_{i-1}(x) \quad \text{mit} \quad a_i = \frac{(xp_i, p_i)}{(p_i, p_i)} \qquad b_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}$$

wobei man mit $p_{-1} = 0$, $p_0 = 1$ und $b_0 = 1$ starten kann.

Wichtig!

Da man $\forall \; q(x) \in \Pi_{n-1}$ durch $p_0, \ldots, p_{n-1}$ darstellen kann, folgt $(q, p_n) = 0$.

Das hat aber die wichtige Konsequenz, dass $p_n(x)$ ***n* einfache Nullstellen** in]*a*,*b*[hat.

Beweis

Konstruiere $q(x) = \Pi_{j=0}^{k-1}(x - \xi_j)$ wobei $\xi_0, \ldots, \xi_{k-1}$ die Nullstellen ***ungerader*** Vielfachheit von $p_n(x)$ sind, und k die Anzahl solcher Nullstellen ist.

Wir nehmen $k < n$ an. Dann ist $q(x) \in \Pi_{n-1}$ und somit $(q, p_n) = 0$.

Aber $q(x)p_n(x)$ hat keinen Vorzeichenwechsel in]a,b[und

$$(q, p_n) = \int_a^b dx\ w(x)q(x)p_n(x) \neq 0$$

Widerspruch zu $k < n$!

$\therefore$ $q(x)$ ist vom Grad n und $p_n(x)$ hat ***n* einfache Nullstellen** in]a,b[

Wir wählen jetzt als Punkte x_i die Nullstellen ξ_i von $p_n(x)$.

Dann kann man Gewichte finden, so dass für jedes Polynom $f(x) \in \Pi_{2n-1}$ exakt gilt:

$$\int_a^b dx\ w(x) f(x) = \sum_{i=0}^{n-1} \omega_i f(x_i)$$

Dazu zerlegen wir erst mit Polynomdivision $f(x) = p_n(x)\ q(x) + r(x)$

wobei $q, r \in \Pi_{n-1}$

Dann gilt für das Integral

$$\int_a^b dx\ w(x) f(x) = \underbrace{(p_n, q)}_{=0} + (r, p_0) = (r, p_0)$$

Außerdem kann man $r(x)$ entwickeln als $r(x) = \sum_{j=0}^{n-1} \beta_j \, p_j(x)$

Das Integral vereinfacht sich damit zu $\int_a^b dx \; w(x) f(x) = \sum_{j=0}^{n-1} \beta_j \underbrace{(p_j, p_0)}_{\propto \; \delta_{j0}} = \beta_0 \, (p_0, p_0)$

Für die Nullstellen x_i von $p_n(x)$ findet man $f(x_i) = \underbrace{p_n(x_i)}_{=0} q(x_i) + r(x_i) = r(x_i) = \sum_{j=0}^{n-1} \beta_j \, p_j(x_i)$

oder als Gleichungssystem

$$\begin{pmatrix} f(x_0) \\ \vdots \\ f(x_{n-1}) \end{pmatrix} = \underbrace{\begin{pmatrix} p_0\,(x_0) & \cdots & p_j\,(x_0) & \cdots & p_{n-1}\,(x_0) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_0\,(x_{n-1}) & \cdots & p_j\,(x_{n-1}) & \cdots & p_{n-1}\,(x_{n-1}) \end{pmatrix}}_{A} \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{n-1} \end{pmatrix}$$

woraus man durch Invertieren

$$\int_a^b dx \; w(x) f(x) = \beta_0 \, (p_0, p_0) = \sum_{i=0}^{n-1} \underbrace{(p_0, p_0) \, [A^{-1}]_{0i}}_{\omega_i} f(x_i) = \sum_{i=0}^{n-1} \omega_i \; f(x_i)$$

erhält.

Man kann mit Hilfe der positiven Polynome $f(x) \equiv f_j(x) = \Pi_{k=0\ (k\neq j)}^{n-1}(x - x_k)^2$ zeigen, dass die Gewichte positiv sind.

Beweis

$$\overbrace{\int_a^b dx\ w(x) f_j(x)}^{>\ 0} = \sum_{i=0}^{n-1} \omega_i\ \underbrace{f_j(x_i)}_{\propto\ \delta_{ij}} = \omega_j \overbrace{\prod_{k\neq j}^{n-1} (x_j - x_k)^2}^{>\ 0}$$

$$\Longrightarrow\ \omega_j > 0\ \forall\ j$$

Diese Gewichte eignen sich auch um beliebige glatte Funktionen mit hoher Genauigkeit numerisch zu integrieren:

$$\int_a^b dx\ w(x) f(x) \approx \sum_{i=0}^{n-1} \omega_i\ f(x_i)$$

Der Verlauf der Gaußschen Integration für allgemeine $w(x) \geq 0$ und $\boldsymbol{n}$ Stützstellen:

Schritt 1

Konstruieren Sie $p_0, p_1(x), \ldots, p_n(x)$ mithilfe der Rekursion

Skalarprodukt

$$p_{i+1}(x) = (x - a_i)\, p_i(x) - b_i \; p_{i-1}(x) \quad \text{mit} \quad a_i = \frac{(xp_i, p_i)}{(p_i, p_i)} \qquad b_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}$$

wobei man mit $p_{-1} = 0,\; p_0 = 1$ und $b_0 = 1$ starten kann.

Schritt 2

Finden Sie die $\boldsymbol{n}$ Nullstellen $x_0, x_1, \ldots, x_{n-1}$ vom $p_n(x)$

Nutzen Sie das Newtonverfahren (z.B.)!

Schritt 3

Bestimmen Sie die Gewichte $\omega_0, \omega_1, \ldots, \omega_{n-1}$ durch Lösen des Gleichungssystems

$$\underbrace{\begin{pmatrix} p_0(x_0) & p_0(x_1) & \cdots & p_0(x_{n-1}) \\ p_1(x_0) & p_1(x_1) & \cdots & p_1(x_{n-1}) \\ \vdots & \vdots & \ldots & \vdots \\ p_{n-1}(x_0) & p_{n-1}(x_1) & \cdots & p_{n-1}(x_{n-1}) \end{pmatrix}}_{A^{\mathrm{T}}} \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{pmatrix} = \begin{pmatrix} (p_0, p_0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Annäherung des Integrals: $\displaystyle\int_a^b dx\; w(x) f(x) \approx \sum_{i=0}^{n-1} \omega_i \; f(x_i)$

Der Verlauf der Gaußschen Integration für allgemeine $w(x) \geq 0$ und ***n*** Stützstellen:

Schritt 1

Konstruieren Sie $p_0, p_1(x), \ldots, p_n(x)$ mithilfe der Rekursion

Skalarprodukt

$$p_{i+1}(x) = (x - a_i)\, p_i(x) - b_i\ p_{i-1}(x) \quad \text{mit} \quad a_i = \frac{(xp_i, p_i)}{(p_i, p_i)} \qquad b_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}$$

wobei man mit $p_{-1} = 0,\ p_0 = 1$ und $b_0 = 1$ starten kann.

Schritt 2

Finden Sie die ***n*** Nullstellen $x_0, x_1, \ldots, x_{n-1}$ vom $p_n(x)$

Nutzen Sie das Newtonverfahren (z.B.)!

Schritt 3

Bestimmen Sie die Gewichte $\omega_0, \omega_1, \ldots, \omega_{n-1}$ durch Lösen des Gleichungssystems

$$\underbrace{\begin{pmatrix} p_0(x_0) & \cdots & p_0(x_{n-1}) \\ p_1(x_0) & \cdots & p_1(x_{n-1}) \\ \vdots & & \vdots \\ p_{n-1}(x_0) & \cdots & p_{n-1}(x_{n-1}) \end{pmatrix}}_{A^{\mathrm{T}}} \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{pmatrix} = \begin{pmatrix} (p_0, p_0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\boxed{\underbrace{(p_0, p_0)\,[A^{-1}]_{0i}}_{\omega_i}}$$

Annäherung des Integrals: $\displaystyle\int_a^b dx\ w(x) f(x) \approx \sum_{i=0}^{n-1} \omega_i\ f(x_i)$

Der Verlauf der Gaußschen Integration für allgemeine $w(x) \geq 0$ und ***n*** Stützstellen:

Schritt 1

Konstruieren Sie $p_0, p_1(x), \ldots, p_n(x)$ mithilfe der Rekursion

Skalarprodukt

$$p_{i+1}(x) = (x - a_i)\, p_i(x) - b_i\; p_{i-1}(x) \quad \text{mit} \quad a_i = \frac{(xp_i, p_i)}{(p_i, p_i)} \qquad b_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}$$

wobei man mit $p_{-1} = 0,\ p_0 = 1$ und $b_0 = 1$ starten kann.

Schritt 2

Finden Sie die ***n*** Nullstellen $x_0, x_1, \ldots, x_{n-1}$ vom $p_n(x)$

Nutzen Sie das Newtonverfahren (z.B.)!

Schritt 3

Bestimmen Sie die Gewichte $\omega_0, \omega_1, \ldots, \omega_{n-1}$ durch Lösen des Gleichungssystems

$$\underbrace{\begin{pmatrix} p_0(x_0) & \cdots & p_0(x_{n-1}) \\ p_1(x_0) & \cdots & p_1(x_{n-1}) \\ \vdots & & \vdots \\ p_{n-1}(x_0) & p_{n-1}(x_1) \cdots & p_{n-1}(x_{n-1}) \end{pmatrix}}_{A^{\mathrm{T}}} \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{pmatrix} = \begin{pmatrix} (p_0, p_0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\boxed{\underbrace{(p_0, p_0)\, [A^{-1}]_{0i}}_{\omega_i}}$$

Annäherung des Integrals: $\displaystyle\int_a^b dx\; w(x) f(x) \approx \sum_{i=0}^{n-1} \omega_i\; f(x_i)$

Der Verlauf der Gaußschen Integration für allgemeine $w(x) \geq 0$ und *n* Stützstellen:

Schritt 1

Konstruieren Sie $p_0, p_1(x), \ldots, p_n(x)$ mithilfe der Rekursion

Skalarprodukt

$$p_{i+1}(x) = (x - a_i)\, p_i(x) - b_i\ p_{i-1}(x) \quad \text{mit} \quad a_i = \frac{(xp_i, p_i)}{(p_i, p_i)} \qquad b_i = \frac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}$$

wobei man mit $p_{-1} = 0,\ p_0 = 1$ und $b_0 = 1$ starten kann.

Schritt 2

Finden Sie die ***n*** Nullstellen $x_0, x_1, \ldots, x_{n-1}$ vom $p_n(x)$

Nutzen Sie das Newtonverfahren (z.B.)!

Schritt 3

Bestimmen Sie die Gewichte $\omega_0, \omega_1, \ldots, \omega_{n-1}$ durch Lösen des Gleichungssystems

$$\underbrace{\begin{pmatrix} p_0(x_0) & p_0(x_1) & \cdots & p_0(x_{n-1}) \\ p_1(x_0) & p_1(x_1) & \cdots & p_1(x_{n-1}) \\ \vdots & \vdots & \cdots & \vdots \\ p_{n-1}(x_0) & p_{n-1}(x_1) & \cdots & p_{n-1}(x_{n-1}) \end{pmatrix}}_{A^{\mathrm{T}}} \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{pmatrix} = \begin{pmatrix} (p_0, p_0) \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Annäherung des Integrals: $\displaystyle\int_a^b dx\ w(x) f(x) \approx \sum_{i=0}^{n-1} \omega_i\ f(x_i)$

Verschiedene Varianten dieser Methode sind ausgearbeitet:

1. $w(x) = 1$, *a*=-1, *b*=1 → Legendre-Polynome (Gauss-Legendre Int.)
2. $w(x) = e^{-x^2}$, *a*=-∞, *b*=∞ → Hermite-Polynome (Gauss-Hermite Int.)
3. $w(x) = \dfrac{1}{\sqrt{1-x^2}}$, *a*=-1, *b*=1 → Chebyshev-Polynome (Gauss-Chebyshev Int.)

. . . und andere. Sehen Sie, z.B., ***Numerische Recipe*** Kapitel 4.5

Gauss-Legendre aus GSL

```
/* Datei: beispiel-3.5.c   Datum: 21.4.2016 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* GNU Scientific Library (gsl) Routinen erlauben Gauss-Legendre Punkte zu bestimmen */
#include <gsl/gsl_integration.h>

void gausslegendre(double a,double b,double *x,double *w,size_t n)
 { gsl_integration_glfixed_table *xwtable;
   size_t i;
   xwtable=gsl_integration_glfixed_table_alloc(n);
   if(xwtable==NULL)
     {
      printf("Problem with Gauss-Legendre\n");
      abort();
     }
   for(i=0;i<n;i++)
    {
      gsl_integration_glfixed_point (a, b, i, &x[i], &w[i], xwtable);
    }
   gsl_integration_glfixed_table_free(xwtable);
  }

/* Definition der zu integrierenden Funktion */
/* Definiere globale Variabel, um Parameter der Funktion festzulegen */
double aconst=0.5,bconst=1;  /* Werte sind bei Programmstart vorgegeben */

double f(double x)
{
  return aconst/(bconst+x*x);
       /* ... auch hier kann ich globale Variablen benutzen */
}
```

Gauss-Legendre aus GSL

GSL stellt Datentypen zur Verfügung

```
/* Datei: beispiel-3.5.c   Datum: 21.4.2016 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* GNU Scientific Library (gsl) Routinen erlauben Gauss-Legendre Punkte zu bestimmen */
#include <gsl/gsl_integration.h>

void gausslegendre(double a,double b,double *x,double *w,size_t n)
 { gsl_integration_glfixed_table *xwtable;
   size_t i;
   xwtable=gsl_integration_glfixed_table_alloc(n);
   if(xwtable==NULL)
     {
      printf("Problem with Gauss-Legendre\n");
      abort();
     }
   for(i=0;i<n;i++)
    {
      gsl_integration_glfixed_point (a, b, i, &x[i], &w[i], xwtable);
    }
   gsl_integration_glfixed_table_free(xwtable);
  }

/* Definition der zu integrierenden Funktion */
/* Definiere globale Variabel, um Parameter der Funktion festzulegen */
double aconst=0.5,bconst=1;  /* Werte sind bei Programmstart vorgegeben */

double f(double x)
{
  return aconst/(bconst+x*x);
       /* ... auch hier kann ich globale Variablen benutzen */
}
```

Gauss-Legendre aus GSL

GSL stellt Datentypen zur Verfügung

falls Problem: Programm abbrechen

```
/* Datei: beispiel-3.5.c   Datum: 21.4.2016 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* GNU Scientific Library (gsl) Routinen erlauben Gauss-Legendre Punkte zu bestimmen */
#include <gsl/gsl_integration.h>

void gausslegendre(double a,double b,double *x,double *w,size_t n)
 { gsl_integration_glfixed_table *xwtable;
   size_t i;
   xwtable=gsl_integration_glfixed_table_alloc(n);
   if(xwtable==NULL)
     {
      printf("Problem with Gauss-Legendre\n");
      abort();
     }
   for(i=0;i<n;i++)
    {
      gsl_integration_glfixed_point (a, b, i, &x[i], &w[i], xwtable);
    }
   gsl_integration_glfixed_table_free(xwtable);
  }

/* Definition der zu integrierenden Funktion */
/* Definiere globale Variabel, um Parameter der Funktion festzulegen */
double aconst=0.5,bconst=1;  /* Werte sind bei Programmstart vorgegeben */

double f(double x)
{
  return aconst/(bconst+x*x);
       /* ... auch hier kann ich globale Variablen benutzen */
}
```

Gauss-Legendre aus GSL

```
/* Datei: beispiel-3.5.c   Datum: 21.4.2016 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* GNU Scientific Library (gsl) Routinen erlauben Gauss-Legendre Punkte zu bestimmen */
#include <gsl/gsl_integration.h>

void gausslegendre(double a,double b,double *x,double *w,size_t n)
 { gsl_integration_glfixed_table *xwtable;
   size_t i;
   xwtable=gsl_integration_glfixed_table_alloc(n);
   if(xwtable==NULL)
     {
      printf("Problem with Gauss-Legendre\n");
      abort();
     }
   for(i=0;i<n;i++)
    {
      gsl_integration_glfixed_point (a, b, i, &x[i], &w[i], xwtable);
    }
   gsl_integration_glfixed_table_free(xwtable);
  }

/* Definition der zu integrierenden Funktion */
/* Definiere globale Variabel, um Parameter der Funktion festzulegen */
double aconst=0.5,bconst=1;  /* Werte sind bei Programmstart vorgegeben */

double f(double x)
{
  return aconst/(bconst+x*x);
       /* ... auch hier kann ich globale Variablen benutzen */
}
```

GSL stellt Datentypen zur Verfügung

falls Problem: Programm abbrechen

GSL gibt Stützstellen und Gewichte nur einzeln heraus

```
/* Datei: beispiel-3.5.c   Datum: 21.4.2016 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* GNU Scientific Library (gsl) Routinen erlauben Gauss-Legendre Punkte zu bestimmen */
#include <gsl/gsl_integration.h>

void gausslegendre(double a,double b,double *x,double *w,size_t n)
 { gsl_integration_glfixed_table *xwtable;
   size_t i;
   xwtable=gsl_integration_glfixed_table_alloc(n);
   if(xwtable==NULL)
     {
      printf("Problem with Gauss-Legendre\n");
      abort();
     }
   for(i=0;i<n;i++)
    {
      gsl_integration_glfixed_point (a, b, i, &x[i], &w[i], xwtable);
    }
   gsl_integration_glfixed_table_free(xwtable);
  }

/* Definition der zu integrierenden Funktion */
/* Definiere globale Variabel, um Parameter der Funktion festzulegen */
double aconst=0.5,bconst=1;  /* Werte sind bei Programmstart vorgegeben */

double f(double x)
{
  return aconst/(bconst+x*x);
       /* ... auch hier kann ich globale Variablen benutzen */
}
```

Gauss-Legendre aus GSL

GSL stellt Datentypen zur Verfügung

falls Problem: Programm abbrechen

GSL gibt Stützstellen und Gewichte nur einzeln heraus

danach ist GSL Tabelle nicht mehr notwendig

```
/* Datei: beispiel-3.5.c   Datum: 21.4.2016 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* GNU Scientific Library (gsl) Routinen erlauben Gauss-Legendre Punkte zu bestimmen */
#include <gsl/gsl_integration.h>

void gausslegendre(double a,double b,double *x,double *w,size_t n)
 { gsl_integration_glfixed_table *xwtable;
   size_t i;
   xwtable=gsl_integration_glfixed_table_alloc(n);
   if(xwtable==NULL)
     {
      printf("Problem with Gauss-Legendre\n");
      abort();
     }
   for(i=0;i<n;i++)
    {
      gsl_integration_glfixed_point (a, b, i, &x[i], &w[i], xwtable);
    }
   gsl_integration_glfixed_table_free(xwtable);
  }

/* Definition der zu integrierenden Funktion */
/* Definiere globale Variabel, um Parameter der Funktion festzulegen */
double aconst=0.5,bconst=1;  /* Werte sind bei Programmstart vorgegeben */

double f(double x)
{
  return aconst/(bconst+x*x);
       /* ... auch hier kann ich globale Variablen benutzen */
}
```

Gauss-Legendre aus GSL

GSL stellt Datentypen zur Verfügung

falls Problem: Programm abbrechen

GSL gibt Stützstellen und Gewichte nur einzeln heraus

danach ist GSL Tabelle nicht mehr notwendig

zu integrierende Funktion ist mit Hilfe globaler Variablen definiert

erinnert an Trapez

```
int main()
{
  double a,b;                /* Intervallgrenzen */
  int  i,n;                  /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;    /* Variablen, um Ergebnis zu speichern */
  double *x,*w;              /* Zeiger auf Speicherplaetze, die double enthalten */
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe der Parameter */
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der Stuetzstellen bestimmt Laenge des Feldes */
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);  /* uebergibt n = Anzahl der Punkte a,b Intervallgrenzen */
                             /* Adressen der mit new allozierten Speicherbereiche */
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeichert */
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;             /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

Hauptprogramm

erinnert an Trapez

Eingabe für Integralgrenzen etc.

```
int main()
{
  double a,b;                /* Intervallgrenzen */
  int  i,n;                  /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;    /* Variablen, um Ergebnis zu speichern */
  double *x,*w;              /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der Stuetzstellen bestimmt Laenge des Feldes */
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);  /* uebergibt n = Anzahl der Punkte a,b Intervallgrenzen */
                             /* Adressen der mit new allozierten Speicherbereiche */
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeichert */
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;             /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

Hauptprogramm

erinnert an Trapez

Eingabe für Integralgrenzen etc.

Speicher für Stützstellen und Gewichte

```
int main()
{
  double a,b;               /* Intervallgrenzen */
  int  i,n;                 /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;   /* Variablen, um Ergebnis zu speichern */
  double *x,*w;             /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);   /* uebergibt n = Anzahl der Punkte a,b Intervallgrenzen */
                              /* Adressen der mit new allozierten Speicherbereiche */
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeichert */
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;              /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

Hauptprogramm


```
int main()
{
  double a,b;              /* Intervallgrenzen */
  int  i,n;                /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;  /* Variablen, um Ergebnis zu speichern */
  double *x,*w;            /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);   /* uebergibt n = Anzahl der F
                              /* Adressen der mit new alloz
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeic
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;               /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

erinnert an Trapez

Eingabe für Integralgrenzen etc.

Speicher für Stützstellen und Gewichte

Aufruf der Routine die Gauss-Legendre Stützstellen und Gewichte festlegt

Hauptprogramm

```
int main()
{
  double a,b;               /* Intervallgrenzen */
  int  i,n;                 /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;   /* Variablen, um Ergebnis zu speichern */
  double *x,*w;             /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);   /* uebergibt n = Anzahl der P
                              /* Adressen der mit new alloz
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeic
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;               /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

erinnert an Trapez

Eingabe für Integralgrenzen etc.

Speicher für Stützstellen und Gewichte

Aufruf der Routine die Gauss-Legendre Stützstellen und Gewichte festlegt

bestimme Parameter der Funktion f

Hauptprogramm

```
int main()
{
  double a,b;                /* Intervallgrenzen */
  int  i,n;                  /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;    /* Variablen, um Ergebnis zu speichern */
  double *x,*w;              /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);   /* uebergibt n = Anzahl der F
                              /* Adressen der mit new alloz
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeic
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;               /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

erinnert an Trapez

Eingabe für Integralgrenzen etc.

Speicher für Stützstellen und Gewichte

Aufruf der Routine die Gauss-Legendre Stützstellen und Gewichte festlegt

bestimme Parameter der Funktion f

Summe über Stützstellen: wie bei trapez

Hauptprogramm

```
int main()
{
  double a,b;                /* Intervallgrenzen */
  int  i,n;                  /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;    /* Variablen, um Ergebnis zu speichern */
  double *x,*w;              /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);   /* uebergibt n = Anzahl der F
                             /* Adressen der mit new alloz
  /* Gitterpunkte und Gewichte sind nun bei x und w gespei
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;             /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}
```

erinnert an Trapez

Eingabe für Integralgrenzen etc.

Speicher für Stützstellen und Gewichte

Aufruf der Routine die Gauss-Legendre Stützstellen und Gewichte festlegt

bestimme Parameter der Funktion f

Summe über Stützstellen: wie bei trapez

Ausgabe und Vergleich

Speicherplatz freigeben

Hauptprogramm

```
int main()
{
  double a,b;               /* Intervallgrenzen */
  int  i,n;                 /* Schleifenvariable,Anzahl der Stuetzstellen */
  double  exact,diff,sum;   /* Variablen, um Ergebnis zu speichern */
  double *x,*w;             /* Zeiger auf Speicherplaetze
  printf("Bitte geben Sie a,b und n ein: ");  /* Eingabe de
  scanf("%lf %lf %d",&a,&b,&n);
  x=(double *) malloc(n*sizeof(double));     /* Anzahl der
  w=(double *) malloc(n*sizeof(double));     /* waehrend des Programmlaufes ! */
  gausslegendre(a,b,x,w,n);   /* uebergibt n = Anzahl der F
                              /* Adressen der mit new alloz
  /* Gitterpunkte und Gewichte sind nun bei x und w gespeic
  /* Diese kann man fuer beliebige Funktionen benutzen */
  /* Lege jetzt die Parameter der Funktion durch  Definition der globalen Variabeln fest */
  aconst=1.0;
  bconst=1.0;
  sum=0.0;            /* Bestimmung eines Integrals mit den Gitter und Gewichten */
  for(i=0;i<n;i++)
   {
    sum+=f(x[i])*w[i];   /* "+=" Operator summiert f(xi)*w(xi) auf Summe auf */
   }
  exact=atan(b)-atan(a)
  diff=fabs(sum-exact);
  printf("N      gauleg      exact       diff \n\n");
  printf("%d      %15.6e      %15.6e       %15.6e \n",n,sum,exact,diff);
  free(x);   /* Speicherbereich wieder freigeben */
  free(w);   /*  ("deallozieren") */
  return 0;
}

/* Compilation auf cip Rechnern:    gcc beispiel-3.5.c -lgsl -lblas -o beispiel-3.5
Ergebnis:  fuer a b n = 0.0 1.0 5
Bitte geben Sie a,b und n ein: 0 1 5
N      gauleg      exact       diff

5          7.853982e-01         7.853982e-01          3.426266e-09
*/
```

erinnert an Trapez

Eingabe für Integralgrenzen etc.

Speicher für Stützstellen und Gewichte

Aufruf der Routine die Gauss-Legendre Stützstellen und Gewichte festlegt

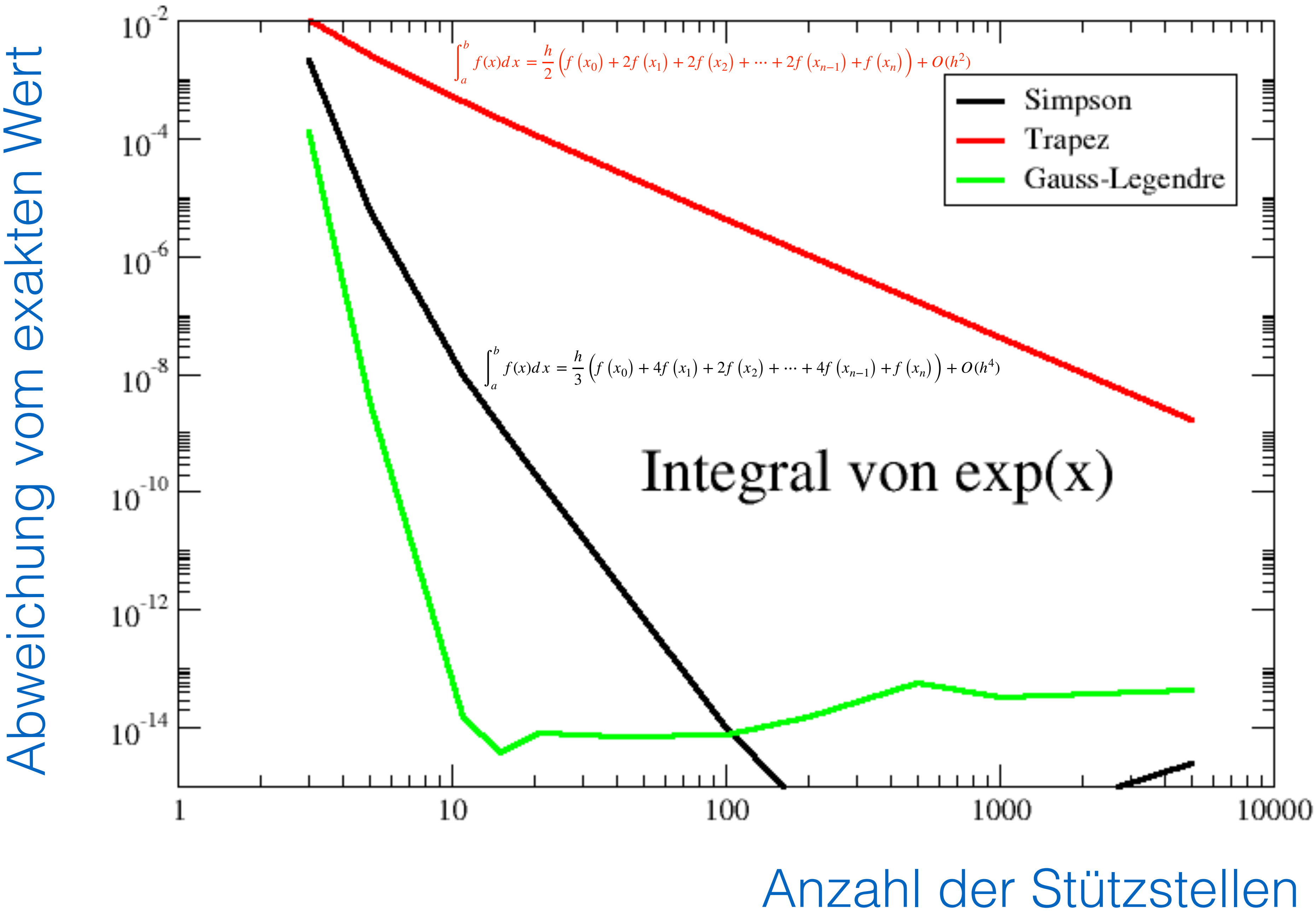
bestimme Parameter der Funktion f

Summe über Stützstellen: wie bei trapez

Ausgabe und Vergleich

Speicherplatz freigeben

Compilation erfordert zusätzliche Bibliotheken



$\int_a^b f(x)dx = \frac{h}{2}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right) + O(h^2)$
Simpson
Trapez
Gauss-Legendre
$\int_a^b f(x)dx = \frac{h}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 4f(x_{n-1}) + f(x_n)\right) + O(h^4)$
Integral von exp(x)
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
1
10
100
1000
10000
Abweichung vom exakten Wert
Anzahl der Stützstellen