Inhomogenes Randwertproblem mit Green's Funktion

Wir betrachten immer noch die linearen inhomogenen DGL 2. Ordnung

$$u''(x) + g(x)\ u(x) = s(x)$$

mit Randbedingungen $u(x_1) = 0$ und $u(x_2) = 0$

Wir nehmen an, dass wir zwei homogenen Lösungen $v(x)$ und $w(x)$ bereits kennen. Die beiden Lösungen erfüllen die Randbedingungen

$$v(x_1) = w(x_2) = 0$$

Mit der **Wronski Determinante** (konstant, da $\frac{dW[v,w](x)}{dx} = 0$)

$$W_0 \equiv W[v,w](x) \equiv v(x)w'(x) - v'(x)w(x)$$

definiert man die **Green'sche Funktion**

$$G(x,y) = \frac{1}{W_0} \begin{cases} v(x)w(y) & \text{für} \quad x \le y \\ v(y)w(x) & \text{für} \quad x > y \end{cases}$$

mit der man die Lösung mit den korrekten Randbedingung durch

$$u(x) \equiv \int_{x_1}^{x_2} dy\, G(x,y)s(y) = \frac{1}{W_0} \left[w(x) \int_{x_1}^{x} dy\ v(y)s(y) + v(x) \int_{x}^{x_2} dy\ w(y)s(y) \right]$$

erhält.

Das kann man durch Ableitung überprüfen

$$
\begin{aligned}
u'(x) &= \frac{1}{W_0}\left[w'(x)\int_{x_1}^{x} dy\; v(y)s(y) + w(x)v(x)s(x) + v'(x)\int_{x}^{x_2} dy\; w(y)s(y) - v(x)w(x)s(x)\right] \\
u''(x) &= \frac{1}{W_0}\left[w''(x)\int_{x_1}^{x} dy\; v(y)s(y) + w'(x)v(x)s(x) + v''(x)\int_{x}^{x_2} dy\; w(y)s(y) - v'(x)w(x)s(x)\right] \\
&= \frac{1}{W_0}\left[W_0 s(x) - g(x)w(x)\int_{x_1}^{x} dy\; v(y)s(y) - g(x)v(x)\int_{x}^{x_2} dy\; w(y)s(y)\right] \\
&= s(x) - g(x)\; u(x)
\end{aligned}
$$

Wegen der Integralgrenzen und Randbedingungen der homogenen Lösungen gilt auch

$$u(x_1) = 0 \quad \text{und} \quad u(x_2) = 0$$

Damit findet man die Lösung durch einfache Integration.

(*Beachte:* hom. Lösungen müssen bekannt o. einfach zu finden sein, Inhomogenität *s(x)* sollte einfach zu integrieren sein. Ansonsten iterative Methode, z.B. Numerov, meist besser)

Beispiel:

$$u''(x) + g(x)\ u(x) = s(x) \quad \text{mit} \quad s(x) = -\frac{1}{2}\ x\ \exp(-x) \quad \text{und} \quad g(x) = -\omega^2$$

Randbedingungen: $u(0) = 0$ und $\lim_{r\to\infty} u(r) = 0$

Erster Schritt: identifiziere Lösungen der homogenen Gleichung
(hier analytisch möglich, auch numerische Lösung kann sinnvoll sein)

$$v(x) = \sinh(\omega x) \quad \text{und} \quad w(x) = \exp(-\omega x)$$

Analytische Lösung des Problems ist bekannt
(numerische Lösung mit Schießverfahren evtl. in Übungen)

$$\rightarrow \quad u(x) = \frac{1}{8}\ x(1+x)\exp(-x) \quad \text{bzw.} \quad u(x) = \mathrm{e}^{-\omega\, x} - \frac{\left(1 + \frac{1}{2}\ \left(-\omega^2+1\right) x\right) \mathrm{e}^{-x}}{\left(-\omega^2+1\right)^2}$$

Code erfordert Implementierung des Ausdrucks (numerische Integration)

$$u(x) \equiv \int_{x_1}^{x_2} dy\, G(x,y)s(y) = \frac{1}{W_0}\ \left[w(x)\int_{x_1}^{x} dy\ v(y)s(y) + v(x)\int_{x}^{x_2} dy\ w(y)s(y)\right]$$

```c
/* Datei:  beispiel-4.5.c  Author: BCM / C version AN  Datum:  08.05.2012 */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<float.h>
#include<limits.h>
#include<string.h>
#include<ctype.h>

/* Vorwaertsdeklaration einer Funktione

/* Eingabe ueber die Kommandozeile auswerten: Routine wird gleich beschrieben
   Ziel ist Eingabeparameter der Kommandozeile in globalen Variabeln zu speichern */
void GetUserParam( int, char *[] );

/* Globale Variablen: */

long   n=100;      /* # Schritten fuer I
double omega=1.0; /* Parameter der Funktion g(x) */
double b=10;       /* Obere Grenze fuer die Integration */

/* Definition der DGL durch Inhomogenit
double sf ( double r ) {
    /* Inhomogenitaet der Differentialg
    return -0.5 * r * exp(-r);
}
double v ( double r ) {
    /* bei r=0 regulaere Loesung der homogenen Gleichung */
    return sinh( omega * r );
}
double w ( double r ) {
    /* bei lim_r->\infty regulaere Loesung der homogenen Gleichung */
    return exp( -omega * r );
}
double sol (double r)  /* analytische Loesung
 ...
double Wronski () {  /* Wronski-Determinante v
    return -omega;
}
```

Code implementiert Green's Funktions Methode

Benutzt Routine zur Auswertung von Kommandozeilen Parameter: GetUserParam(…) Vorwärtsdeklaration (z.B. in Header-Datei)

globale Variablen werden in GetUserParam verändert (je nach Kommandozeile)

Definiert DGL und Randbedingungen: homogene Lösungen und Inhomogenität

Code nutzt analytisch bekannte Lösung zur Ausgabe

Wronski Determinante ist konstant

Hauptprogramm implementiert Green's Methode auf Basis der Trapezregel

```
int main( int argc, char *argv[] ) {
/* argc = # Argumente, argv enthaelt die Argumente, argv[0] enthaelt Programmname */
/* Deklaration der Variablen */
  ...
    GetUserParam( argc, argv ); /* Parameter
  ...
    h = b / (double) n; /* Schrittweite */
    c = 0.5 * h / Wronski();
  ...
    int1 = 0; r = 0;
    sm1 = sf(0); vm1 = v(0);

    phi[0] = 0;
    for (i=1; i<n; i++) {
    r += h;
    v0 = v(r);
    s0 = sf(r);
    int1 += c * ( sm1 * vm1 + s0 * v0 );
    phi[i] = int1 * w(r);
    sm1 = s0;
    vm1 = v0;
    }

    int2 = 0; r = n * h;
    sp1 = sf(r); wp1 = w(r);

    phi[n] = 0;
    for (i=n-1; i>0; i--) {
    r -= h;
    w0 = w(r);
    s0 = sf(r);
    int2 += c * ( wp1 * sp1 + s0 * w0 );
    phi[i] += int2 * v(r);
    sp1 = s0;
    wp1 = w0;
    }
    ...

    return 0;
}
```

main hat Parameter argc und argv! Argumente von GetUserParam

Schrittweite und Gewicht innerhalb eines Intervalls

phi[i] nutzt Integral von phi[i-1]

$$u_1(x) = \frac{1}{W_0}\, w(x) \int_{x_1}^{x} dy\; v(y)s(y)$$

phi[i] nutzt Integral von phi[i+1]

$$u_2(x) = \frac{1}{W_0}\, v(x) \int_{x}^{x_2} dy\; w(y)s(y)$$

argc und argv sind Standard-Argumente von main
argc: Anzahl der Parameter
argv: Vektor von Zeigern auf Strings mit argc Elementen

```
void GetUserParam( int argc, char *argv[] )
    int i;              /* Variablen: */
    char* endptr;
    const char usage[] = "# beispiel-4.5 [-n <Schrittzahl>] [-w <Frequenz>] [-b <Obere Grenze>]";
    const char error_message[] = "# FEHLER(GetuserParam): falsche Option: “;

    if (argc>1) /* falls es ueberhaupt
      {
        for (i=1; i<argc; i++)
          {
            /* parameter 2 Charakter lang und sollte mit '-' anfaengen ... */
            if ( (strlen(argv[i])==2) && (argv[i][0] == '-') )
             {
              switch (argv[i][1])
               {
                case 'w':
                        omega = strtod( argv[++i], &endptr);
                        if ( (!isspace(*endptr) && (*endptr) != 0) ) {
                           printf(" %s \n %s \n”,error_message,usage); exit(1); }
                        break;
               case 'b':
                        b = strtod( argv[++i], &endptr);
                        if ( (!isspace(*endptr) && (*endptr) != 0) ) {
                           printf(" %s \n %s \n”,error_message,usage); exit(1); }
                        break;
                case 'n':
                        n = strtol( argv[++i], &endptr, 10);
                        if ( (!isspace(*endptr) && (*endptr) != 0) ) {
                           printf(" %s \n %s \n",error_message,usage); exit(1); }
                        break;
                default:
                  printf(" %s \n %s \n",error_message,usage);
                  exit(1);
               }
             } else
             {   printf(" %s \n %s \n",error_message,usage);
                 exit(1);
             } /* end of: if-then-else */
        } /* end-of: for */
      } /* end-of: if */
}
```

gehe durch alle Argumente
1. Teil hat Form “-x” 2. Teil ist meist Zahl (bei uns)

für jedes mögliche Argument (-w -b -n)
wir der nächste Parameter ausgewertet

switch wählt einen von mehreren Fällen

strtod (für double) und strtol (für long)

strto… gibt Wert zurück und in endptr einen Zeiger auf nicht mehr ausgewertete Zeichen (sollte 0 sein)

Werte werden hier in globalen Variabeln gespeichert

Durch die Routine **`GetUserParam`** ist es möglich im Terminal Parameter der numerischen Lösung zu testen:

```
> gcc -lm beispiel-4.5-green.c -o beispiel-4.5-green
> ./beispiel-4.5-green -w 1.0 -n 100 -b 10.0
# green: # =====  # omega =      1.00000e+00, n =                 100, b =      1.00000e+01,
h =      1.00000e-01
#
#                       r                       phi(r)
#_____________________________ _____________________________
          0.0000000000e+00          0.0000000000e+00          0.0000000000e+00
          1.0000000000e-01          1.2479190403e-02          1.2441514498e-02
          2.0000000000e-01          2.4630103621e-02          2.4561922592e-02
          3.0000000000e-01          3.6207427207e-02          3.6114888258e-02
```

. . .

Für schnelle Prüfung der Konvergenz ist auch ein direkter Aufruf innerhalb gnuplots oder andere Plot-Programmen möglich, z.B.

```
> gnuplot
   . . .
gnuplot> f(x)=x*(1+x)*exp(-x)/8
   . . .
gnuplot> plot „<./beispiel-4.5-green -w 1.0 -n 10 -b 10.0" with lines,“<./beispiel-4.5-
green -w 1.0 -n 20 -b 10.0" with lines,“<./beispiel-4.5-green -w 1.0 -n 30 -b 10.0"
with lines, f(x) with lines
```

Man kann so schnell die Konvergenzeigenschaften erkennen.

—> Green’sche Funktion mit etwa 30 Stützstellen liefert akzeptable Lösung