Stephan Laage

Grafik-Paket in C

Darstellung von Funktionen und Meßwerten

Mit einem universellen Grafikprogramm ist es kein Problem, die zahlreichen Werte einer Meßreihe ansprechend darzustellen. Als Ausgabegerät kann ein Plotter, ein Matrixdrucker oder ein Monitor dienen. Die Berechnung von Spline-Funktionen ist enthalten.

Nicht nur, wer sich mit wissenschaftlichen Fragestellungen beschäftigt, steht oft vor dem Problem, Meßwerte oder mathematische Funktionen in einem zweidimensionalen Koordinatensystem darzustellen. Während man früher zu Millimeterpapier und Lineal griff, wird heute meist der inzwischen überall vorhandene Mikrocomputer eingesetzt. Doch allzu oft wird dann „auf die Schnelle" ein kleines Programm entworfen, daß das Gewünschte auf den Plotter bringen soll. Dabei zeigt jedoch die Erfahrung, daß aus dem kleinen Programm eine unangenehm lange Sitzung werden kann.
Wie schön, wenn man in solchen Fällen auf fertige Routinen zurückgreifen kann, die die Achsenskalierung, das Zeichnen der Achsen und Funktionswerte übernehmen; so, wie es auf Großrechner schon lange üblich ist.
Hier soll ein solches Paket vorgestellt werden, das es ermöglicht, mit einem wenige Zeilen umfassenden Programm sehr ansprechende Grafiken zu erzeugen. Dabei wird eine automatische Achsenskalierung und die Berechnung einer Spline-Funktion angeboten, die diskrete Werte durch eine glatte Funktion miteinander verbindet.

Die Umgebung

Dieses Paket wurde in der Programmiersprache C unter dem Betriebssystem OS-9 entwickelt. C bietet gegenüber (Turbo-)Pascal den Vorteil, daß ein komplettes Programm aus mehreren Dateien zusammengesetzt werden kann, wovon hier ausgiebig Gebrauch gemacht wird. Da, abgesehen von vier elementaren mathematischen Routinen, keine Funktionen verwendet werden, die über den von Kernighan und Ritchie (1) definierten Standard hinausgehen, sollte (fast) jeder andere C-Compiler auch unter anderen Betriebssystemen die Programme verdauen. Bei den vier mathematischen Routinen handelt es sich um
– double log10(x),
– double floor(x), liefert den größten, ganzzahligen Wert, der kleiner als x ist,
– double pow(x,y), liefert x hoch y.
– double fabs (x), liefert abs (x)
Um den Umfang der Listings möglichst klein zu halten und betriebssystemspezifische Dinge herauszuhalten, wurde auf die Behandlung von Fehlern vollständig verzichtet. Wer intensiver mit diesem Paket arbeiten möchte, sollte sich zu diesem Punkt eigene Gedanken machen – denn beim Arbeiten mit Zeigern ist schnell etwas „lebenswichtiges" im Rechner „abgeschossen".

Die Datei „scale.c"

Bild 1 umfaßt fünf Funktionen, die das komfortable Zeichnen von Daten in zweidimensionalen Koordinatensystemen ermöglichen. Die einzelnen Funktionen sollen kurz vorgestellt werden:
SCALE: Diese Funktion führt eine automatische Skalierung für ein eindimensionales Datenfeld durch. Die X- und die Y-Werte der zu zeichnenden Daten müssen getrennt mit SCALE() behandelt werden. SCALE() übernimmt drei Parameter: einen Zeiger auf das Datenfeld (DATA), die Anzahl der Werte im Feld (N) und die Zahl der gewünschten Achsenmarkierungen (NM). Die Funktion berechnet daraus die drei Skalierungskoeffizienten: den Wert des ersten Markierungsstriches für die entsprechende Achse (START), die Differenz zwischen den Werten zweier benachbarter Markierungsstriche (STEP) und die Anzahl der

```
/* SCALE ***************************************************************/
/* Paket zum Zeichnen von Funktionen und Daten in zweidimensionalen    */
/* Koordinatenkreuzen, enthaelt die Funktionen                         */
/*                                                                     */
/*      SCALE       automatische Skalierung eines Datenfeldes          */
/*      AXIS        Zeichnen von Achsen mit Markierungen               */
/*      RASTER      Zeichnen von Rastern                               */
/*      LINE        Zeichnen von Datenfeldern durch Linien oder Symbole*/
/*      SPLINE      natuerliche, kubische Splineinterpolation          */
/*                                                                     */
/* Stephan Laage                                        12. Oktober '86*/
/*                                    letzte Aenderung  29. Dezember '86*/
/***********************************************************************/


#include <math.h>
#define RND 1.0E-10
#define MAXS 50

/* Externe Deklarationen ***********************************************/
extern init(), move(), draw(), rotate(), print(), color(), symbol();
extern double chwe, chhe;


/* SCALE ***************************************************************/
/* berechnet die Skalierungskoeffizienten fuer ein eindimensionales DOUBLE */
/* Feld, die zum Zeichnen einer Koordinatenachse benoetigt werden: Wert    */
/* der ersten Achsenmarkierung, Differenz zwischen zwei Markierungen und   */
/* Anzahl der Markierungen                                                 */

scale(data,n,nm)
    double *data;       /* zeigt auf das Datenfeld                     */
    int n;              /* Anzahl der Werte im Datenfeld               */
    int nm;             /* gewuenschte Zahl der Achsenmarkierungen     */
{
    double min,max,start,step,fak;
    int i,intstep;
```

Bild 1. Die Datei scale.c enthält alle Funktionen, die zum komfortablen Zeichnen von mathematischen Funktionen oder Meßwerten mit skalierten Koordinatenachsen notwendig sind

```
    min = *data;
    max = *data;
    for (i = 1; i < n; ++i) {
        ++data;
        if (*data < min) min = *data;
        if (*data > max) max = *data;
    };
    if (min > 0.0 && min - (max - min) / 4 < 0.0) min = 0.0;
    step = (max-min)/nm;
    fak = pow(10.0,floor(log10(step)));
    intstep = (int) (step/fak);
    switch (intstep) {
        case 1: intstep = 2;
                break;
        case 2:
        case 3:
        case 4: intstep = 5;
                break;
        case 5:
        case 6:
        case 7: intstep = 8;
                break;
        case 8:
        case 9: intstep = 10;
    };
    step = intstep * fak;
    if (min <= 0.0 && max > 0.0) {
        start = 0.0;
        while (start > min) start -= step;
        } else {
            fak *= 10.0;
            start = floor(min/fak)*fak;
            while (start + step <= min) start += step;
        };
        nm = 1;
        while (start + nm * step < max) ++nm;
        ++data;
        *(data++) = start;
        *(data++) = step;
        *(data) = (double) nm;
    };
    /* AXIS ***************************************************************/
    /* zeichnet eine Koordinaten-Achse mit Markierungen                   */

    axis(xp,yp,len,start,step,nmark,atyp,mtyp,titel)
        double xp,yp;       /* Startpunkt (mm)                            */
        double len;         /* Achsenlaenge (mm)                          */
        double start,step;  /* Achsenmarkierung: 1. Wert und Schrittweite */
        double nmark;       /* Anzahl der Achsenmarkierungen              */
        int atyp;           /* Achsentyp   0: X-Achse, 1: Y-Achse         */
        int mtyp;           /* Markierungstyp   0: unten/rechts, 1: oben/links */
        char *titel;        /* Achsenbeschriftung (wird zentriert)        */
    {
        int i, nks, expo = 0;
        double fak = len/nmark, dis1, dis2, dis3, dis4;
        char str[20], form[5];

        if (fabs(start) + nmark * step < 0.001)
            while (fabs(start) + nmark * step < 1.0) {
                start *= 10.0;
                step *= 10.0;
                --expo;
            }
        else if (fabs(start) + nmark * step > 10000.0)
            while (fabs(start) + nmark * step > 10.0) {
                start /= 10.0;
                step /= 10.0;
                ++expo;
            };
        strcpy(form,"%.xf");
        nks = (int) -floor(log10(step+RND));
        if (nks < 0) nks = 0;
        form[2] = (char) nks+'0';

        if (mtyp == 0) {
            dis1 = -0.5*chhe;
            dis2 = -2.0*chhe;
            dis3 = -3.0*chhe;
            dis4 = -3.5*chhe;
        } else {
            dis1 = 0.5*chhe;
            dis2 = 1.0*chhe;
            dis3 = 3.0*chhe;
            dis4 = 2.5*chhe;
        };
        if (atyp == 0) {
            rotate(0);
            move(xp+len/2-strlen(titel)*chwe/2,yp+dis4);
        } else {
            rotate(1);
            dis1 = -dis1;
            dis2 = -dis2;
            dis3 = -dis3;
            dis4 = -dis4;
            move(xp+dis4,yp+len/2-strlen(titel)*chwe/2);
        };
        print(titel);

        start = start + RND;
        for (i = 0; i < nmark+0.1; ++i) {
            sprintf(str,form,start+i*step);
            if (atyp == 0) {
                move(xp+i*fak,yp);
                draw(xp+i*fak,yp+dis1);
                move(xp+i*fak-chwe*strlen(str)/2,yp+dis2);
                print(str);
                if (i < nmark-0.9) {
                    move(xp+i*fak,yp);
                    draw(xp+(i+1)*fak,yp);
                }
            } else {
                move(xp,yp+i*fak);
                draw(xp+dis1,yp+i*fak);
                move(xp+dis2,yp+i*fak-chwe*strlen(str)/2);
                print(str);
                if (i < nmark-0.9) {
                    move(xp,yp+i*fak);
                    draw(xp,yp+(i+1)*fak);
                }
            };
        };
        if (expo != 0) {
            sprintf(str,"%d",expo);
            if (atyp == 0) {
                move(xp+len-7*chwe,yp+dis4);
                print("*10");
                move(xp+len-4*chwe,yp+dis3);
            } else {
                move(xp+dis4,yp+len-7*chwe);
                print("*10");
                move(xp+dis3,yp+len-4*chwe);
            };
            print(str);
        };
    }
```

```
/* RASTER ************************************************************/
/* zeichnet ein Raster in das durch SCALE definierte Koordinatenkreuz */

raster(xp,yp,xl,yl,xn,yn)
    double xp,yp;       /* untere, linke Eckposition des Kreuzes (mm)  */
    double xl,yl;       /* Laenge der X- und der Y-Achse (mm)          */
    double xn,yn;       /* Anzahl der Markierungen                     */
{
    double xfak = xl / xn,
           yfak = yl / yn;
    int i, sw = 1;

    color(2);
    for (i = 0; i < xn + 0.1; ++i)
        if (sw) {
            move(xp+i*xfak,yp);
            draw(xp+i*xfak,yp+yl);
            sw = 0;
        } else {
            move(xp+i*xfak,yp+yl);
            draw(xp+i*xfak,yp);
            sw = 1;
        };
    for (i = 0; i < yn + 0.1; ++i)
        if (sw) {
            move(xp,yp+i*yfak);
            draw(xp+xl,yp+i*yfak);
            sw = 0;
        } else {
            move(xp+xl,yp+i*yfak);
            draw(xp,yp+i*yfak);
            sw = 1;
        };
    color(0);
}


/* LINE **************************************************************/
/* zeichnet einen Kurvenzug in das durch SCALE definierte Koordinatenkreuz */

line(xdata,ydata,n,xp,yp,xl,yl,symb)
    double *xdata,*ydata;   /* Datenfelder mit X- und Y-Werten         */
    int n;                  /* Anzahl der Werte in xdata/ydata         */
    double xp,yp;           /* unterer, linker Eckpunkt des Kreuzes (mm) */
    double xl,yl;           /* Laenge der Achsen (mm)                  */
    int symb;               /* < 0: nur Datenpunkte, > 0: mit Linienzug */
{
    double xst = xdata[n], yst = ydata[n];
    double xf,yf;
    int i;

    xf = xl/xdata[n+1]/xdata[n+2];
    yf = yl/ydata[n+1]/ydata[n+2];
    if (symb >= 0) {
        move(xp + (*xdata - xst) * xf, yp + (*ydata - yst) * yf);
        for (i = 0; i < n-1; ++i) {
            if (symb > 0) symbol(symb);
            ++xdata;
            ++ydata;
            draw(xp + (*xdata - xst) * xf, yp + (*ydata - yst) * yf);
        };
        if (symb > 0) symbol(symb);
    } else
        for (i = 0; i < n; ++i) {
            move(xp + (*xdata - xst) * xf, yp + (*ydata - yst) * yf);
            symbol(-symb);
            ++xdata;
            ++ydata;
        };
}


/* SPLINE ************************************************************/
/* berechnet aus diskreten Stuetzstellen eine kontinuierliche Spline-Funk- */
/* tion. Die Anzahl der Stuetzpunkte ist begrenzt.                     */

spline(xs,ys,ns,xf,yf,nf)
    double *xs,*ys; /* Felder mit den Koordinaten der Stuetzpunkte     */
    int ns;         /* Anzahl der Werte in xs und ys                   */
    double *xf,*yf; /* Felder, in denen die Funktionswerte abgelegt werden */
    int nf;         /* Anzahl der gewuenschten Spline-Funktionswerte   */
{
    static double a[MAXS],b[MAXS],c[MAXS],d[MAXS],
                  l[MAXS],m[MAXS],p[MAXS],h[MAXS],y2[MAXS];
    double xx,xstep;
    int i,k;

    if (ns > MAXS) ns = MAXS;                   /* maximal MAXS Stuetzpunte */
    for (i = 0; i <= ns - 2; ++i)
        h[i] = xs[i+1] - xs[i];
    for (i = 1; i <= ns - 2; ++i) {
        a[i] = 2 * (h[i-1] + h[i]);
        b[i] = h[i];
        c[i] = h[i];
        d[i] = 6 * ((ys[i] - ys[i-1]) / h[i-1] - (ys[i+1] - ys[i]) / h[i]);
    };
    m[1] = a[1];
    for (i = 1; i <= ns - 3; ++i) {
        l[i] = c[i] / m[i];
        m[i+1] = a[i+1] - l[i] * b[i];
    };
    p[1] = d[1];
    for (i = 2; i <= ns - 2; ++i)
        p[i] = d[i] - l[i-1] * p[i-1];
    y2[ns-2] = -p[ns-2] / m[ns-2];
    for (i = ns - 3; i >= 1; --i)
        y2[i] = -(p[i] + b[i] * y2[i+1]) / m[i];
    y2[0] = 0.0;
    y2[ns-1] = 0.0;
    for (i = 0; i <= ns - 2; ++i) {
        a[i] = (y2[i+1] - y2[i]) / (6 * h[i]);
        b[i] = y2[i] / 2;
        c[i] = (ys[i+1] - ys[i]) / h[i] - h[i] * (y2[i+1] + 2 * y2[i]) / 6;
        d[i] = ys[i];
    };
    xf[0] = xs[0];
    yf[0] = ys[0];
    xstep = (xs[ns-1] - xs[0]) / (nf - 1);
    k = 0;
    for (i = 1; i <= nf - 2; ++i) {
        xf[i] = xs[0] + i * xstep;
        if (xf[i] > xs[k+1]) ++k;
        xx = xf[i] - xs[k];
        yf[i] = a[k]*xx*xx*xx + b[k]*xx*xx + c[k]*xx + d[k];
    };
    xf[nf-1] = xs[ns-1];
    yf[nf-1] = ys[ns-1];
}
```

```
/* GPLOT ***************************************************************/
/* Graphik-Ankopplung C.ITOH-Plotter                                   */
/* Stephan Laage                                        Dezember '86   */
/***********************************************************************/

#include <stdio.h>

double chhe =         2.40,     /* Zeichen-Hoehe in Millimeter          */
       chwe =         1.60;     /* Zeichen-Breite                       */

double xfak = 2950.0/295.0,     /* 2950 Schritte auf 295.0 Millimeter   */
       yfak = 2100.0/210.0;     /* 2100 Schritte auf 210.0 Millimeter   */

FILE *fopen(), *pr;             /* "pr" = Druckerport                   */

init()
{
    pr = fopen("/p","w");       /* Druckerport oeffnen                  */
    fprintf(pr,"G1\n");         /* Schrittweite 0.1 mm                  */
    fprintf(pr,"H\n");          /* Stift in "Home"-Position             */
}

draw(x,y)
double x,y;
{
    fprintf(pr,"D %.0f,%.0f\n", x * xfak, y * yfak);
}                                /* fahren mit gesenktem Stift */

move(x,y)
double x,y;
{
    fprintf(pr,"M %.0f,%.0f\n", x * xfak, y * yfak);
}                                /* fahren mit gehobenem Stift */

print(str)
char *str;
{
    fprintf(pr,"P%s\n", str);                    /* Text ausgeben */
}

rotate(dir)
int dir;
{
  switch (dir) {
    case 0: fprintf(pr,"Q0\n"); break;      /* waagerecht, nach rechts  */
    case 1: fprintf(pr,"Q3\n"); break;      /* senkrecht, aufwaerts     */
    case 2: fprintf(pr,"Q2\n"); break;      /* waagerecht, nach links   */
    case 3: fprintf(pr,"Q1\n");             /* senkrecht, abwaerts      */
  };
}

color(c)
int c;
{
    fprintf(pr,"C%d\n", c);                 /* waehlt die Farbe         */
}

symbol(i)
int i;
{
    switch (i) {
        case 1: fprintf(pr,"N5\n"); break;      /* Symbol 5 = Kreis     */
        case 2: fprintf(pr,"N2\n"); break;      /* Symbol 2 = Quadrat   */
        case 3: fprintf(pr,"N1\n");             /* Symbol 1 = Karo      */
    }
}
```

Bild 2. Ein Beispiel für die Anbindung eines Plotters an die Datei scale.c. Hier handelt es sich um einen Plotter vom Typ C.Itoh, der eine Schrittweite von 0,1 mm aufweist

```
/* GSCREEN *************************************************************/
/* Graphik-Ankopplung fuer NEC7220-Graphikeinheit                      */
/* Aufloesung:  X-Richtung: 768 Punkte      (Spreizung auf DIN A 4)    */
/*              Y-Richtung: 576 Punkte                                 */
/*                                                                     */
/* Stephan Laage                                        Dezember '86   */
/***********************************************************************/

/* Externe Deklaration der Graphik-Systemroutinen **********************/
extern g_init(), g_move(), g_draw(), g_dir(), g_lpat(), g_print();

double chhe =         4.80,     /* Schriftzeichen-Hoehe in Millimeter   */
       chwe =         3.10;     /* Schriftzeichen-Breite                */

double xfak =  768.0/295.0,     /* 768 Pixel auf 295 Millimeter         */
       yfak = -576.0/210.0,     /* 576 Pixel auf 210 Millimeter         */
       yoff =        576.0;     /* Ursprung fuer Y                      */
int    adir =            2,     /* aktuelle Zeichenrichtung             */
       xpi  =            0,     /* aktuelle X-Position (Pixel)          */
       ypi  =            0;     /* aktuelle Y-Position (Pixel)          */

init()
{
    g_init(1);          /* initialisiert die Graphik-Ausgabeeinheit     */
}

move(x,y)
double x,y;
{
    xpi = (int) (x * xfak + 0.5);
    ypi = (int) (y * yfak + yoff + 0.5);
    g_move(xpi,ypi);    /* setzt den Graphik-Cursor                     */
}

draw(x,y)
double x,y;
{
    xpi = (int) (x * xfak + 0.5);
    ypi = (int) (y * yfak + yoff + 0.5);
    g_draw(xpi,ypi);    /* zeichnet eine Gerade von der Cursor-Position */
}

print(str)
char *str;
{
    g_print(str);       /* gibt Text aus                                */
    g_lpat(-1);         /* setzt Linienmuster zurueck (= $ffff)         */
}

rotate(dir)
int dir;
{
    switch (dir) {
        case 0: adir = 2; break;    /* waagerecht, nach rechts          */
        case 1: adir = 4; break;    /* senkrecht, aufwaerts             */
        case 2: adir = 6; break;    /* waagerecht, nach links           */
        case 3: adir = 0;           /* senkrecht, abwaerts              */
    };
    g_dir(adir);        /* setzt Schreib- und Zeichenrichtung           */
}

color(c)
int c;
{
```

Bild 3. Mit dieser Datei wird gezeigt, wie die Ausgaben von scale.c auf einem Grafik-Monitor ausgegeben werden. Dabei werden elementare Zeichenroutinen und Routinen zur Ausgabe von Schriftzeichen eingesetzt

```
    switch (c) {
        case 0: g_lpat(-1); break;          /* durchgezogene Linie        */
        case 1: g_lpat(0xaaaa); break;      /* gestrichelte Linie         */
        case 2: g_lpat(0x8888); break;      /* gepunktelte Linie          */
        case 3: g_lpat(0x8080);             /* stark gepunktelte Linie    */
    };
}

symbol(i)
int i;
{
    switch (i) {
        case 1: g_circle(6);        /* zeichnet einen Kreis           */
                break;
        case 2: g_move(xpi-5,ypi+5);
                g_dir(2);
                g_box(10,10);       /* zeichnet ein Rechteck          */
                g_move(xpi,ypi);
                g_dir(adir);
                break;
        case 3: g_move(xpi-7,ypi);
                g_dir(1);
                g_box(7,7);         /* zeichnet ein Karo              */
                g_move(xpi,ypi);
                g_dir(adir);
                break;
    };
}
```

```
/* BEISP1 ************************************************************/
/* Fuer das Paket SCALE.C                              Stephan Laage */
/*********************************************************************/


#define MAXS 11
#define MAXF 100


/* Externe Deklarationen                                             */

extern scale(), axis(), raster(), line(), spline();
extern init(), move(), draw(), rotate(), print();


main()
{
    static double xf[MAXF+3], yf[MAXF+3];
    static double xs[MAXS+3] =
        {2.5, 3.3, 4.0, 4.5, 5.1, 5.4, 6.0, 6.45, 7.0, 7.8, 8.6};
    static double ys[MAXS+3] =
        {1.02E-5, 1.55E-5, 2.73E-5, 4.2E-5, 5.76E-5, 7.2E-5, 6.38E-5,
         2.8E-5, 8.9E-6, 1.4E-6, 5.0E-7};
    double xp = 30.0, yp = 35.0, xl = 240.0, yl = 150.0;
    int i;

    init();
    scale(xs,MAXS,12);
    scale(ys,MAXS,8);
    axis(xp,yp,xl,xs[MAXS],xs[MAXS+1],xs[MAXS+2],0,0,"pH");
    axis(xp,yp,yl,ys[MAXS],ys[MAXS+1],ys[MAXS+2],1,1,
                              "Reaktionsgeschwindigkeit [mol/min]");
    raster(xp,yp,xl,yl,xs[MAXS+2],ys[MAXS+2]);
    line(xs,ys,MAXS,xp,yp,xl,yl,-2);
    spline(xs,ys,MAXS,xf,yf,MAXF);
    for (i = 0; i < 3; ++i) {
        xf[MAXF+i] = xs[MAXS+i];
        yf[MAXF+i] = ys[MAXS+i];
    };
    line(xf,yf,MAXF,xp,yp,xl,yl,0);
    move(xp-20.0,yp-25.0);
    rotate(0);
    print("pH-Abhaengigkeit der sauren Phosphatase");
    move(xp-25.0,yp-30.0);
    draw(xp-25.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp-30.0);
    draw(xp-25.0,yp-30.0);
}
```

Bild 4. Beispiel für den Einsatz der Datei scale.c. 10 Meßwerte wurden eingetragen und durch die Spline-Funktion verbunden

Markierungsstriche, die die Achse tragen wird. Der letzte Wert kann von der gewünschten Markierungszahl etwas abweichen, je nach dem, wie es sich als günstig erweist. Diese drei Werte werden in dem Datenfeld selbst abgelegt, und zwar in den Zellen, die auf die eigentlichen Daten folgen. Entsprechend muß das Feld DATA zusätzlichen Platz für mindestens drei weitere Werte haben. Dafür hat der Aufrufer Sorge zu tragen, andernfalls ist der Crash vorprogrammiert.
Die Autoskalierung ermittelt für STEP Werte vom Typ 1, 2, 5, 8 oder 10, jeweils multipliziert mit einer geeigneten Zehnerpotenz. Anschließend wird START auf einen möglichst „glatten" Wert gesetzt, der kleiner als der kleinste im Datenfeld vorhandene Wert ist. Schließlich wird NM so gewählt, daß die Achse mindestens bis zum größten Datenwert reicht. In der Regel liefert SCALE() Achsen, die „vernünftig" skaliert sind.

AXIS: Diese Funktion zeichnet die Achse, wobei erst jetzt die Position (XP und YP) und die Achsenlänge (LEN) festgelegt werden. AXIS() übernimmt außerdem die drei durch SCALE() ermittelten Werte. Schließlich kann man noch wählen, ob es eine senkrechte oder eine waagerechte Achse werden soll (ATYP) und ob die Achsenbeschriftung unter bzw. rechts von der Achse oder über bzw. links von der Achse angebracht werden soll (MTYP). Dann ist noch ein Text anzugeben (TITEL), der zentriert an die Achse geschrieben wird.
Die Funktion schaltet bei Markierungswerten, die größer als 10 000 oder kleiner als 0,001 sind, auf eine Beschriftung im wissenschaftlichen Format mit einem „10 hoch X" am Ende der Achse um. Die funktionsinternen Variablen DIS1 bis DIS4 dienen der richtigen Positionierung der Beschriftungen. Im Feld FORM wird ein Formatstring abgelegt, der die geeignete Zahl von Nachkommastellen für die Achsenmarkierungen wählt und beim Aufruf der Funktion SPRINTF() gebraucht wird.

RASTER: Diese Funktion zeichnet ein Raster, nach Möglichkeit in einer anderen Farbe. Um bei Verwendung eines Plotters die abzufahrende Wegstrecke klein zu halten, werden die aufeinanderfolgenden Rasterstriche in wechselnden Richtungen abgefahren.

LINE: Mit dieser Funktion werden die Datenwerte eingezeichnet. LINE() erwartet zwei eindimensionale DOUBLE-Datenfelder, in denen die X- und Y-Koordi-

```
#include <math.h>
#define MAXP 200

/* Externe Deklarationen                                          */

extern init(), scale(), axis(), raster(), spline();
extern move(), draw(), rotate(), print();
main()
{
    static double x[MAXP+3], y[MAXP+3];
    double x1 = -5.0, x2 = 10.0, xstep;
    double xp = 20.0, yp = 20.0, xl = 240.0, yl = 150.0;
    int i;

    xstep = (x2 - x1) / (MAXP - 1);
    for (i = 0; i < MAXP; ++i) {
        x[i] = x1 + i * xstep;
        y[i] = 2 * sin(x[i]) + sin(2*x[i] + PI/2);
    };
    init();
    scale(x,MAXP,15);
    scale(y,MAXP,10);
    axis(xp,yp-y[MAXP]*yl/y[MAXP+1]/y[MAXP+2],xl,
                                x[MAXP],x[MAXP+1],x[MAXP+2],0,0,"X-Achse");
    axis(xp-x[MAXP]*xl/x[MAXP+1]/x[MAXP+2],yp,yl,
                                y[MAXP],y[MAXP+1],y[MAXP+2],1,1,"Y-Achse");
    raster(xp,yp,xl,yl,x[MAXP+2],y[MAXP+2]);
    line(x,y,MAXP,xp,yp,xl,yl,0);
    move(xp,yp+yl+10);
    rotate(0);
    print("f(x) = y = 2 sin x + sin (2 x + pi/2)");
    move(xp-10.0,yp-10.0);
    draw(xp-10.0,yp+yl+25.0);
    draw(xp+xl+10.0,yp+yl+25.0);
    draw(xp+xl+10.0,yp-10.0);
    draw(xp-10.0,yp-10.0);
```

Bild 5. Mit scale.c können beliebige mathematische Funktionen gezeichnet werden. Die Achsenlage ist frei wählbar

naten abgelegt sind. In beiden Feldern müssen auf die eigentlichen Daten die drei Skalierungskoeffizienten folgen, wie sie durch SCALE dort abgelegt werden. Natürlich können diese Werte auch „von Hand" dort eingetragen werden.

Mit dem Wert SYMB kann man wählen, ob ein Linienzug gezeichnet werden soll (SYMB = 0), ob die Datenpunkte durch Symbole markiert werden sollen (SYMB < 0) oder ob beides gezeichnet werden soll (SYMB > 0). Ist SYMB ungleich Null, dann wird durch den absoluten Wert von SYMB ein Symbol zur Markierung der Datenpunkte ausgesucht, etwa 1 = Kreis, 2 = Karo und 3 = Quadrat.

SPLINE: Diese Funktion zeichnet nichts. Stattdessen erwartet sie zwei Felder mit diskreten Stützpunkten (XS, YS), z. B. Meßwerten, und berechnet daraus eine natürliche, kubische Spline-Funktion, deren Funktionswerte in den Feldern XF und YF abgelegt werden. Mit NS wird die Anzahl der Stützpunkte und mit NF die gewünschte Anzahl der Spline-Funktionswerte übergeben. Dabei muß NF unbedingt größer als NS sein.

Der hier verwendete Algorithmus ist in der numerischen Mathematik verbreitet und unter anderem bei (2) nachzulesen. Um den Algorithmus einigermaßen überschaubar zu halten, wurde auf die Verwendung von Pointern vollständig verzichtet, was vielleicht ein bißchen Rechenzeit gespart hätte. Stattdessen wird nur die indizierte Feld-Adressierung eingesetzt.

Die Peripherie

Nun bleibt die Frage, auf welche Art und Weise die erstellte Grafik dem Betrachter zugänglich gemacht werden soll.

Prinzipiell sind als Ausgabegeräte (Grafik-)Monitor, Matrixdrucker oder Plotter denkbar. Um das Paket „scale.c" universell verwenden zu können, muß es mit einer Datei zusammengebunden werden, die die Ankopplung an das spezifische Ausgabegerät übernimmt. Zuerst einmal muß diese Datei zwei DOUBLE-Werte (CHHE und CHWE) enthalten, in denen die Breite und die Höhe der Schriftzeichen in Millimetern enthalten ist. Diese Werte werden von der Funk-

```
#include <math.h>

#define MAXN 150
#define LN2 0.6931471806

/* Externe Deklarationen                                          */

extern scale(), axis(), raster(), line();
extern init(), move(), draw(), print(), rotate();
main()
{
    static double time[MAXN+3],
                  ca45[MAXN+3], s35[MAXN+3], i125[MAXN+3], p32[MAXN+3];
    double lca45 = LN2/165.0, ls35 = LN2/87.2,
           li125 = LN2/60.0,  lp32 = LN2/14.2;
    double t = 0.0;
    double xp = 30.0, yp = 40.0, xl = 240.0, yl = 150.0;
    int i;

    for (i = 0; i < MAXN; ++i, t += 1.0) {
        time[i] = t;
        ca45[i] = exp(-lca45 * t);
        s35[i]  = exp(-ls35  * t);
        i125[i] = exp(-li125 * t);
        p32[i]  = exp(-lp32  * t);
    };

    init();
    scale(time,MAXN,12);
    ca45[MAXN] = 0.0;           /* Startwert                      */
    ca45[MAXN+1] = 0.1;         /* Schrittweite                   */
    ca45[MAXN+2] = 10.0;        /* Anzahl Achsenmarkierungen      */
    for (i = MAXN; i < MAXN+3; ++i) {
        s35[i] = ca45[i];
        i125[i] = ca45[i];
        p32[i] = ca45[i];
    };
    axis(xp,yp,xl,time[MAXN],time[MAXN+1],time[MAXN+2],0,0,"Zeit [Tage]");
    axis(xp,yp,yl,ca45[MAXN],ca45[MAXN+1],ca45[MAXN+2],1,1,
                                                    "rel. Aktivitaet");
    raster(xp,yp,xl,yl,time[MAXN+2],ca45[MAXN+2]);
    line(time,ca45,MAXN,xp,yp,xl,yl,0);
    print(" 45Ca");
    line(time,s35,MAXN,xp,yp,xl,yl,0);
    print(" 35S");
    line(time,i125,MAXN,xp,yp,xl,yl,0);
    print(" 125I");
    line(time,p32,MAXN,xp,yp,xl,yl,0);
    print(" 32P");
    move(xp-10.0,yp-25.0);
    rotate(0);
    print("Abfall der Radioaktivitaet einiger Isotope");
    move(xp-20.0,yp-35.0);
    draw(xp-20.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp-35.0);
    draw(xp-20.0,yp-35.0);
}
```

Bild. 6. Bei diesem Beispiel wurde die automatische Skalierung durch eine manuelle Skalierung ersetzt

Bild 7. Die Darstellung von Meßwerten und Funktionen ist mit scale.c universell

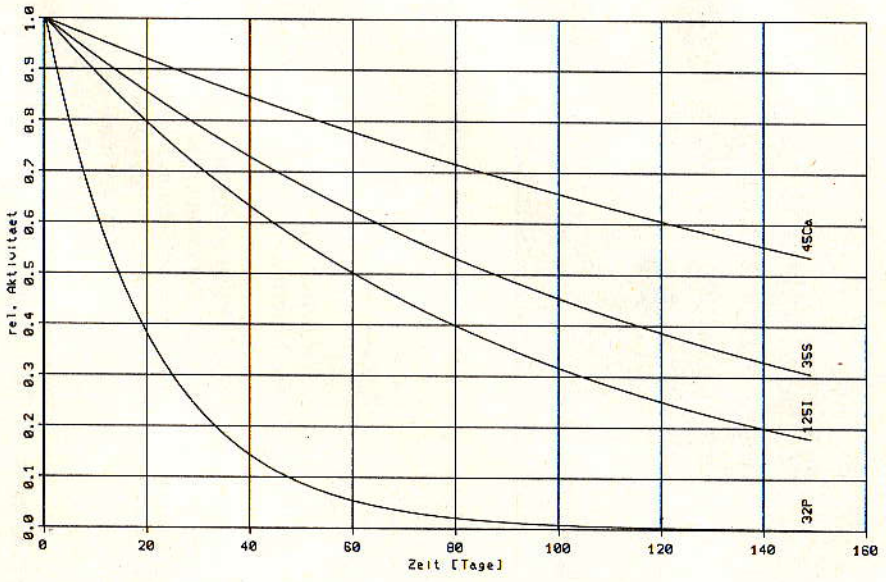

Bild 8. Die Ausgabe des Beispiels aus Bild 6 auf dem Plotter

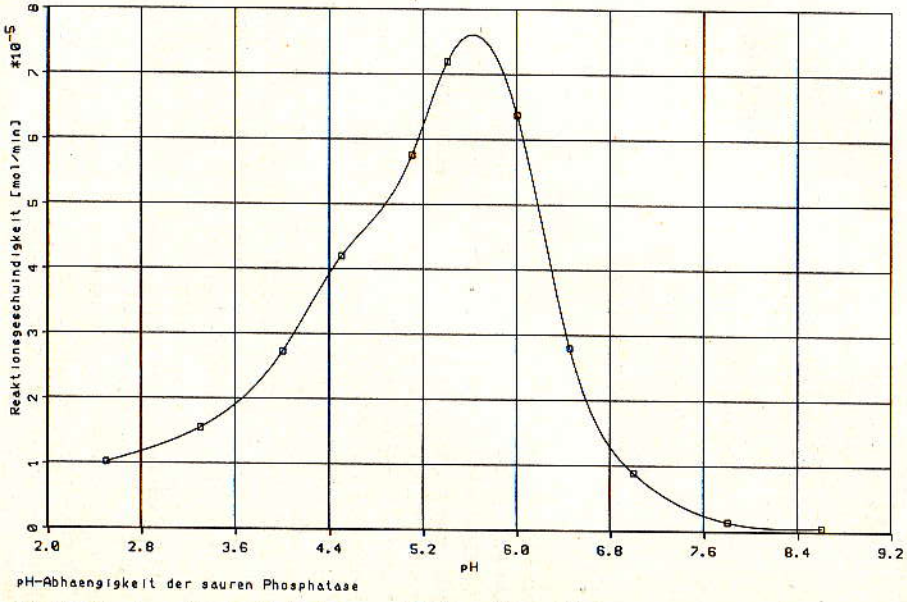

Bild 9. Die Ausgabe des Beispiels aus Bild 4 auf dem Plotter

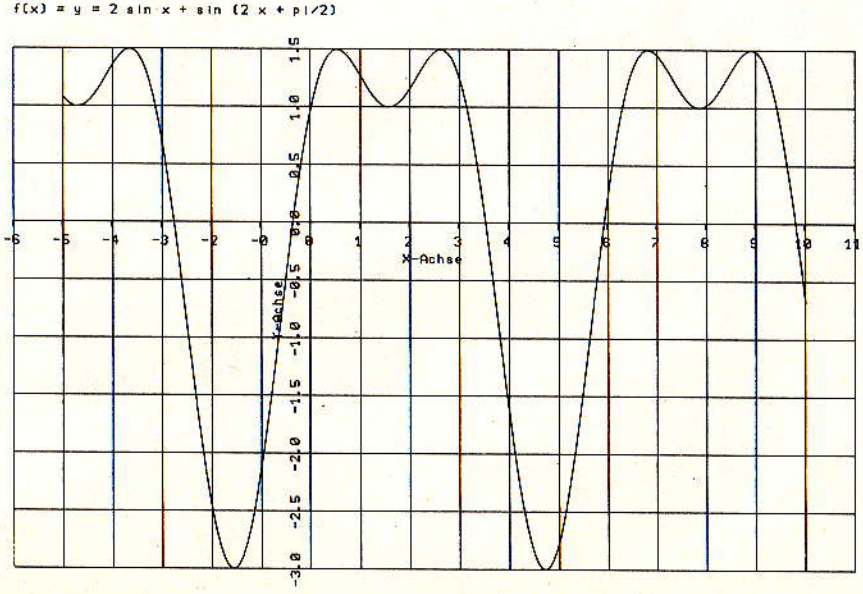

Bild 10. Die Ausgabe des Beispiels aus Bild 5 auf dem Plotter

tion AXIS() zur richtigen Plazierung der Achsenbeschriftung gebraucht. Außerdem werden sechs weitere Funktionen benötigt:

init():	Initialisiert das entsprechende Ausgabegerät.
move():	Fahren mit gehobenem Stift.
draw():	Fahren mit gesenktem Stift.
print():	Gibt einen Text aus.
rotate():	Legt die Schreibrichtung fest. Dabei gilt 0 = waagerecht, rechts 1 = senkrecht, aufwärts
color():	Legt eine Farbe fest. Dabei gilt 0 = Grundfarbe 2 = Farbe für das Raster
symbol():	Zeichnet ein Symbol an die aktuelle Stiftposition.

Die Koordinaten werden grundsätzlich als DOUBLE-Werte übergeben und enthalten Angaben in Millimetern.

Die Auflistung der Funktionen zeigt, was ein Grafik-Ausgabegerät mindestens leisten muß, wenn es zusammen mit „scale.c" eingesetzt werden soll.

Die Datei „gplot.c" (*Bild 2*) zeigt beispielhaft, wie ein Plotter, hier vom Typ ITOH, angekoppelt wird. Der Plotter ist an den Druckerport angeschlossen. Mit XFAK und YFAK werden die Koordinatenangaben so umgerechnet, daß die Ausgaben in Millimeter herauskommen.

Die Schrittweite des hier verwendeten Plotters beträgt 0,1 mm. Die Funktion SYMBOL lenkt die ankommenden Werte so um, daß durch 1 ein Kreis, durch 2 ein Quadrat und durch 3 ein Karo gezeichnet wird.

Der mir zur Verfügung stehende Rechner besitzt den Grafikprozessor NEC 7220. Der Prozessor wird über Systemroutinen (g...) angesprochen. Die Datei „gscreen.c" (*Bild 3*) zeigt, wie man das Paket „scale.c" an diese Bildschirmgrafik ankoppeln kann. Wieder dienen XFAK und YFAK der Koordinatenanpassung, wobei der abgebildete Bereich von 768 × 576 Punkten so behandelt wird, als ob er einem DIN-A4-Blatt entspräche. Da der Ursprung in der linken, oberen Ecke liegt, ist YFAK negativ. Durch YOFF wird er in die untere, linke Ecke verlegt. Die Grafik-Ausgabe ist nur monochrom, deshalb werden verschiedene Farben durch unterschiedliche Linientypen ersetzt; das Raster wird dann gepunktet. Dem Zeichnen von Symbolen wird etwas nachgeholfen, da der NEC 7220 nur Kreise und Rechtecke kennt. Wichtig ist, daß nach dem Zeichnen eines Symbols der Grafik-Cursor wieder an der alten Stelle steht. Wieder entspricht die 1 dem Kreis, die 2 dem Quadrat und die 3 dem Karo.

Spruch des Monats

„Ich bin immer noch gegen Apfelmännchen in mc; ein Apfelweibchen wäre mir lieber."

Aus einem Gespräch unter mc-Redakteuren

Durch die beiden Dateien „gplot.c" und „gscreen.c" können auf zwei völlig verschiedenen Ausgabegeräten fast identische Bilder erzeugt werden. Selbst die Tatsache, daß die Schriftgröße unterschiedlich ist, wird von dem Paket „scale.c" gut verdaut.

Mit entsprechenden Treiber-Dateien sollte es möglich sein, auch vollkommen andere Grafik-Peripheriegeräte zu betreiben. Schließlich ist es auch möglich, vollständig selbst geschriebene Routinen einzusetzen.

Einige Beispiele

Nun zum Lohn der ganzen Mühe. Die Datei „beisp1.c" (*Bild 4*) enthält ein Anwendungsbeispiel, an dem die typische Vorgehensweise gezeigt ist, die eingehalten werden muß, wenn man mit dem Paket „scale.c" arbeiten möchte. Es handelt sich um 11 Meßwerte, die gezeichnet und durch die Spline-Funktion verbunden werden sollen. Die Meßwerte selbst werden in den Feldern XS und YS der Einfachheit halber gleich bei der Initialisierung abgelegt. Wichtig ist, daß diese Felder Platz für mindestens drei weitere Werte enthalten, wo durch den Aufruf von SCALE() die Skalierungskoeffizienten abgelegt werden. Durch AXIS() werden die beiden Achsen gezeichnet, wobei diese Funktion die Skalierungskoeffizienten aus den Datenfeldern übermittelt bekommt. Außerdem bestimmen XP und YP den unteren, linken Eckpunkt und XL und YL die jeweilige Achsenlänge. Durch die Funktion RASTER() wird das Millimeterpapier simuliert. Die Funktion LINE() markiert bei ihrem ersten Aufruf nur die Meßpunkte durch Quadrate (SYMB = −2). Mit dem Aufruf von SPLINE() wird die Spline-Funktion berechnet und in den Feldern XF und YF abgelegt. Auch diese beiden Felder müssen noch Platz für drei weitere Werte enthalten, die aus den Feldern XS und YS kopiert werden.

Diese Voraussetzung muß erfüllt sein, bevor die Funktion LINE() erneut aufgerufen wird und jetzt nur die Spline-Funktion zeichnet (SYMB = 0). Schließlich erhält die Grafik noch einen Titel und, wenn es gefällt, einen Rahmen. Die Ausgabe dieses Programms auf den Plotter und auf den Grafik-Bildschirm (als Kopie mit dem Matrixdrucker) ist in *Bild 9* zu sehen.

Die Datei „beisp2.c" (*Bild 5*) zeichnet eine mathematische Funktion und demonstriert, wie man die Achsen so verschieben kann, daß sie sich im Nullpunkt schneiden (*Bild 10*).

Das dritte Beispiel in der Datei „beisp3.c" (*Bild 6*) ist nicht nur aktuell, sondern zeigt, wie man die Autoskalierung durch manuelle Skalierung ersetzen kann. Da die Y-Achse definitionsgemäß von 0 bis 1 reicht, werden diese Werte „von Hand" eingesetzt.

(Die Funktion SCALE() erzeugt hier übrigens eine Achse, die von 0 bis 1.2 reicht.) Außerdem ist hier (*Bild 8*) wie schon bei dem ersten Beispiel zu sehen, daß, wenn man mehrere Kurven in eine Grafik setzen möchte, die Skalierungskoeffizienten in die jeweiligen Datenfelder umkopiert werden müssen.

Literatur

[1] *Kernighan, B. W. & Ritchie, D. M.:* Programmieren in C. Carl Hanser Verlag, 1983.

[2] *Schwarz, H. R.:* Numerische Mathematik. B. G. Teubner Verlag, 1986.