Übung Nr. 7:

Die „gebrauchsfertige“ Programmdatei `Curves.c` ist vorbereitet, Stützstellen für eine Kurve anzunehmen und sie entweder mit Geradenstücken oder mit Kurvensegmenten einer Spline-Kurve zu verbinden. Anstelle von Pixeln werden ASCII-Zeichen verwendet (Abb.1). Unter http://homepages.thm.de/christ/ kann das entsprechende MS-VS-Projekt heruntergeladen werden; dort findet sich auch (im Ordner `Demo`) eine lauffähige Version des in dieser Übung noch fertigzustellenden Programms (Abb.1).

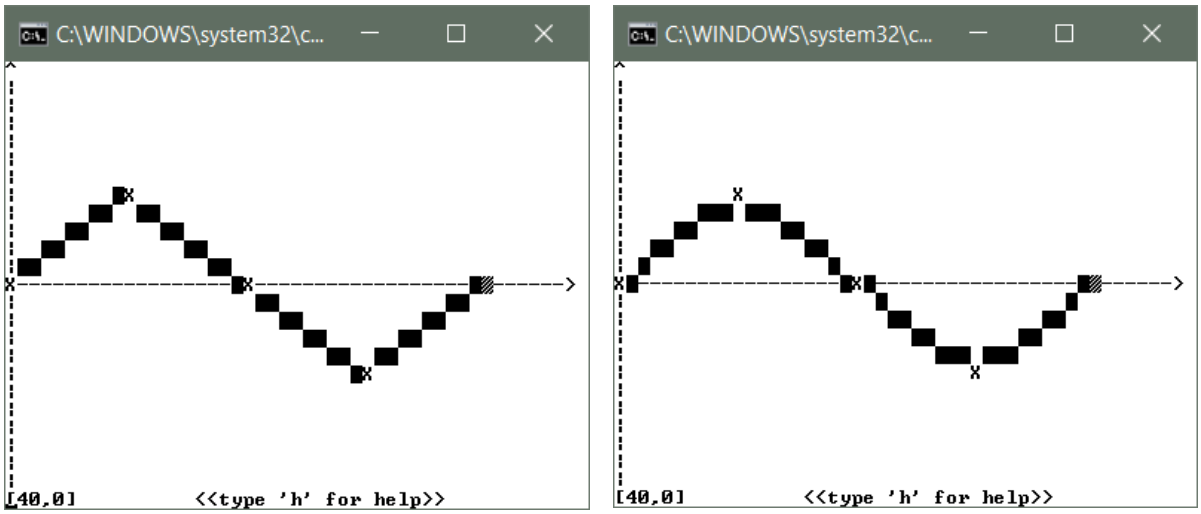

Abb. 1 Kurvenzeichnung als gebrochene Linie (li.) und als Spline (re.)

Die Programm-Bedienung wird im eingebauten Hilfe-Menü (Taste 'h') erläutert; im selben Fenster werden auch die bislang eingegebenen Knoten gelistet (Abb. 2). Ist in `Spline.h` der „Debug-Modus“ eingestellt (der Bezeichner `SPLINE_DBG` definiert), so werden beim Wechsel zwischen Strecken und Splines auch die Zwischenergebnisse mit den Bestimmungsgrößen des berechneten Splines ausgegeben. Diese sind i.d.R. kaum überschaubare Zahlen; bei manchen (insb. symmetrischen) geometrischen Strukturen lassen sich jedoch gewisse Zahlenmuster erkennen (s. Abb. 3 zum Spline nach Abb. 1).

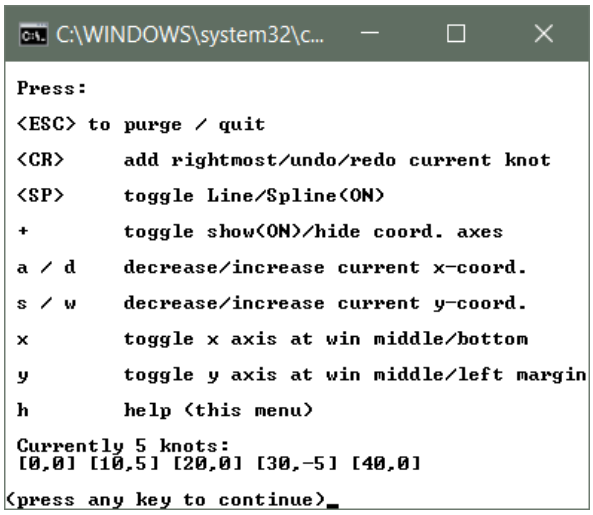

Abb. 2 Hilfe-Menü mit Koordianten-Liste

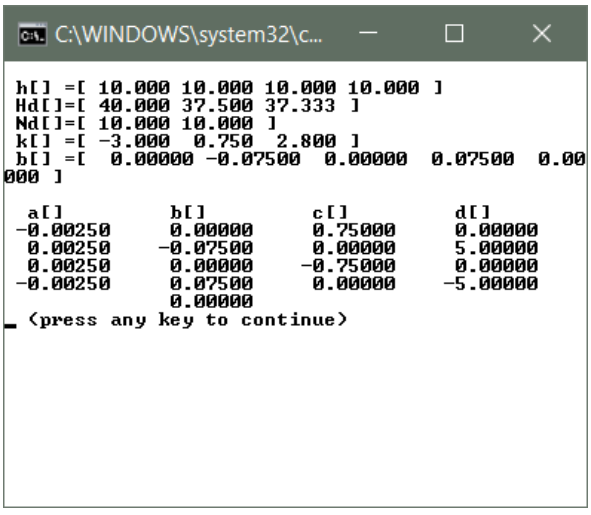

Abb. 3 Bestimmungsgrößen zu Abb. 1

Bis zu seiner Ergänzung kann das Programm zwar die Stützpunkte mit geraden Liniensegmenten verbinden (durch Aufruf des Linienalgorithmus `MidpointLine()` in `Curves.c`); aber beim Wechsel auf Spline-Darstellung (Leertaste ' ') geschieht nichts Sinnvolles. Die Hauptaufgabe besteht nun darin, in `Spline.c` die Elemente der globalen indizierten Variablen `xKnot[]`, `yKnot[]` mit den Koordinaten der eingegebenen Knoten (übernommen aus `Curves.c`) so in der Funktion `makespline()` zu verwenden, daß das in der Vorlesung vorgestellte Gleichungssystem $\underline{H} \bullet \underline{b} = \underline{k}$ aufgestellt wird. Seine Lösung $\underline{b}$ erfolgt dann durch Aufruf von `triDoubDiag()`, einer stark optimierten (Gauß'schen Eliminations-) Methode für Systeme mit tridiagonalen Matrizen, deren Nebendiagonalen zudem untereinander gleich sind (vgl. Robert Sedgewick: "Algorithms in C", S. 542 ff).

Die Stellen in `makespline()`, an denen Code zur Bestimmung der Spline-Koeffizienten (erst b_i, anschließend a_i, c_i, d_i) fehlt, sind mit dem Bezeichner `MORE_SPLINE` markiert (insg. ca. 15 Zeilen). Alles dazu benötigte Wissen ist in den Vorlesungsfolien enthalten. Die beispielhafte Anwendung nach Abbn. 1-3 kann zur Überprüfung der eigenen Codierung herangezogen werden. Beim Abgleich sollte beachtet werden, daß bei jeder Verlegung der x- bzw. der y-Achse (Menüpunkte 'x' bzw. 'y') auch die Knoten-Koordinaten umgerechnet werden.

Auch nach Vervollständigung der Spline-Interpolation in `makespline()` kann es zu unerwünschten Darstellungen kommen, falls die gewählten Knoten zu besonders steilen Verläufen führen (Abb. 4). Hier hilft nur noch eine Linienziehung, um Lücken in der Darstellung zu vermeiden. Die dazu geeignete Stelle in der Zeichenroutine `drawspline()` ist ebenfalls mit dem Bezeichner `MORE_SPLINE` gekennzeichnet, damit eine geschlossene Kurve erzeugt wird (Abb. 5 – hier zu den Knoten [5,15] [10,5] [20,5] [25,15]).

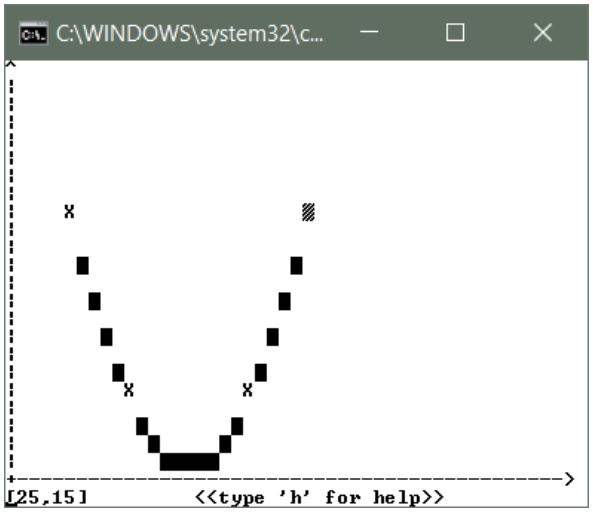

Abb. 4 Visualisiete Spline-Interpolation

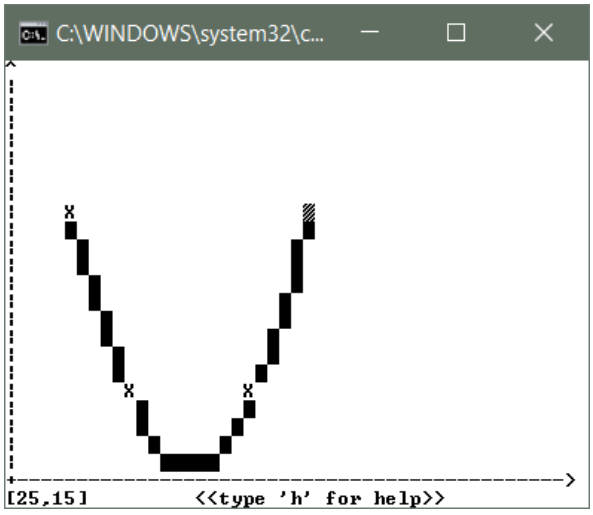

Abb. 5 Spline-Ergänzung durch Linien

Tip: Durch Aktivierung der Direktive `#define GROSSBILD` in der Datei `conDraw.h` wird dem Programm ein Fenster mit dreifacher Kantenlänge zur Verfügung gestellt. Das erlaubt ausgedehntere Versuche mit der erstellten Software.