KfK 4644
November 1989

SINSUR Interpolates Smoothly in a Two-Dimensional Table Using Splines

K. Thurnay
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

KfK 4644

SINSUR INTERPOLATES SMOOTHLY IN A TWO-DIMENSIONAL TABLE USING SPLINES

K. Thurnay

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Abstract

SINSUR interpolates two-dimensional tables using a spline-procedure.

There exist two versions of SINSUR, one - written in FORTRAN77 - to interpolate in a batch-job and one - written in SPEAKEASY - to interpolate in a TSO-session.

The paper describes

- the main features of the procedure,
- what the user should do in order to get access to the batch-version of SINSUR,
- a sample of a SPEAKEASY-session, in which the TSO-version of SINSUR was used.

SINSUR interpoliert glatt zweidimensionale Zahlentafeln mit Hilfe von Splines.

Zusammenfassung :

SINSUR interpoliert zwischen den Werten von zweidimensionalen Zahlentafeln.

SINSUR hat zwei verschiedene Versionen : die eine - in FORTRAN77 geschrieben - soll Interpolationen in Hintergrund-Aufträgen ermöglichen, die zweite - in SPEAKEASY verfasst - dient zum Interpolieren im Vordergrund .

Der vorliegende Arbeit beschreibt

- die mathematische Grundlage des Verfahrens,
- wie ein Benutzer SINSUR in einem Hintergrund-Auftrag einbindet,
- das Protokoll einer SPEAKEASY-Sitzung, in der die Vordergrund-Version benutzt wurde.

Table of Contents

Preface.

Some of the recently created code-systems - written to model complicated physical phenomena - use two-dimensional tables of values to describe different states of the physical system. These tables correspond to three-dimensional surfaces, defined on a two-dimensional, orthogonal (x,y)-mesh. Working with these 2D-tables would be greatly enhanced, if one could also calculate interpolated values to these tables easily.

As de Boor showed in /1/ , there exist a spline-procedure, allowing an unique and smooth interpolation between the values of 2D-tables, if only the partial derivatives on the borders of the table are known. He described also the algorithms, needed to construct the interpolating function.

H. Späth translated these algorithms into the ALGOL (s. /2/) and FORTRAN-IV languages and published corresponding descriptions of the procedure.

SINSUR is a modernized version of the interpolating-procedure, with accommodations to help the user.

Since it is often very cumbersome, to provide the needed partial derivatives on the borders of the table, an algorithm has been constructed which calculates these derivatives from the table-values numerically.

SINSUR has two different versions :

- to interpolate a table in a batch-procedure there exists a version written in FORTRAN77,
- for interpolating tables on-line (in a TSO session) there is a version written in the language SPEAKEASY /3/ .

The procedure.

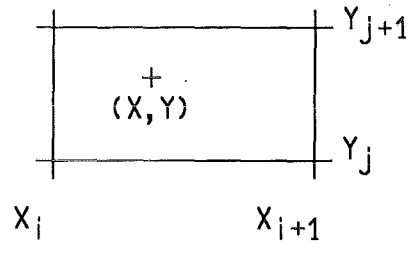
Let be given a 3-D-surface U (s. e.g. the surface in Figure 1) defined at the points of an orthogonal grid in the (x , y) - plane with its first and second derivatives as a set of numbers :

$$u_{i,j} = U(x_i, y_j) \quad , \quad p_{i,j} = \frac{\partial U}{\partial x}(x_i, y_j)$$

$$q_{i,j} = \frac{\partial U}{\partial y}(x_i, y_j) \quad , \quad r_{i,j} = \frac{\partial^2 U}{\partial x\, \partial y}(x_i, y_j) \qquad [B.1]$$

$$\text{for} \quad i = 1, \ \ldots \ , n \quad \text{and} \quad j = 1, \ \ldots \ , m \quad .$$

Inside of a mesh-cell (i,j) (s. fig.)



$$x_i \leq x \leq x_{i+1}$$

$$y_j \leq y \leq y_{j+1}$$

the surface U (x , y) can be smoothly approximated - as de Boor /1/ states - with the following expression :

$$U(x, y) = A^{(i,j)} \times \Delta Y_j \times \Delta X_i \qquad [B.2]$$

ΔY and ΔX in this expression are vectors :

$$\Delta X_i = \begin{bmatrix} 1 \\ x - x_i \\ (x - x_i)^2 \\ (x - x_i)^3 \end{bmatrix} \quad \text{and} \quad \Delta Y_j = \begin{bmatrix} 1 \\ y - y_j \\ (y - y_j)^2 \\ (y - y_j)^3 \end{bmatrix} . \qquad [B.3]$$

The matrices

$$A^{(i,j)} = \left[a^{(i,j)}{}_{kl} \right] \quad , \quad k, l = 1, 2, 3, 4 \qquad [B.4]$$

depend on the values of the function U and its derivatives in the corners of the mesh-cell shown above, and are calculated as follows :

$$A^{(i,j)} = B(x_{i+1} - x_i) \times K^{(i,j)} \times B(y_{j+1} - y_j)^T \qquad [B.5]$$

The matrix B depends on the grid-distance $h = x_{i+1} - x_i$ resp. $h = y_{j+1} - y_j$ as

$$B(h) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{h^2} & -\frac{2}{h^1} & \frac{3}{h^2} & -\frac{1}{h^1} \\ \frac{2}{h^3} & \frac{1}{h^2} & -\frac{2}{h^3} & \frac{1}{h^2} \end{vmatrix} . \qquad [B.6]$$

The corner-values of the functions and derivatives are put into the matrix $K^{(i,j)}$:

$$K^{(i,j)} = \begin{vmatrix} u_{i,j} & q_{i,j} & u_{i,j+1} & q_{i,j+1} \\ p_{i,j} & r_{i,j} & p_{i,j+1} & r_{i,j+1} \\ u_{i+1,j} & q_{i+1,j} & u_{i+1,j+1} & q_{i+1,j+1} \\ p_{i+1,j} & r_{i+1,j} & p_{i+1,j+1} & r_{i+1,j+1} \end{vmatrix} . \qquad [B.7]$$

Derivatives of the surface in the internal points.

If the partial derivatives of the surface U are given on the boundaries of the grid

$$(x_1, \ \ldots \ , x_n) \ x \ (y1, \ \ldots \ , y_m)$$

then they can be calculated in the internal mesh-points.

To get the x-derivatives , $p_{i,j} = \frac{\partial U}{\partial x}(x_i, y_j)$ from the mariginal values

$$\{ p_{1,j} \ , \ p_{n,j} \ : \ j = 1, \ \ldots \ , m \}$$

one can use the following system of equations (s. /1/ or /2/) :

$$\delta x_{i-1} \cdot p_{i+1,j} + 2\left[\,\delta x_i + \delta x_{i-1}\,\right] \cdot p_{ij} + \delta x_i \cdot p_{i-1,j} = G_{ij} \qquad [B.8]$$
$$i = 2, \ \ldots \ , n-1$$

with

$$\delta x_i = x_{i+1} - x_i \quad , \quad i = 1, \ \ldots \ , n-1 \qquad [B.9]$$

and

$$G_{ij} = 3\left[\,\delta x_{i-1}\,\frac{u_{i+1,j} - u_{ij}}{\delta x_i} + \delta x_i\,\frac{u_{ij} - u_{i-1,j}}{\delta x_{i-1}}\,\right] \quad , \quad i = 2, \ \ldots \ , n-1 \qquad .[B.10]$$

Using the abbreviations

$$p_{i+0.5,j} = \frac{u_{i+1,j} - u_{ij}}{\delta x_i} \quad , \quad p_{i-0.5,j} = \frac{u_{ij} - u_{i-1,j}}{\delta x_{i-1}} \qquad [B.11]$$
$$i = 2, \ \ldots \ , n-1$$

one can translate the system Eq. [B.8] - Eq. [B.10] into the following one :

$$\frac{\dfrac{p_{i+1,j} + 2 \cdot p_{ij}}{3} - p_{i+0.5,j}}{\delta x_i} = \frac{p_{i-0.5,j} - \dfrac{p_{i-1,j} + 2 \cdot p_{ij}}{3}}{\delta x_{i-1}} \qquad [B.12]$$
$$\text{for} \qquad i = 2, \ \ldots \ , n-1 \qquad .$$

Case n=3 : In this case the Eq. [B.8] turns to

$$\delta x_1 \cdot p_{3,j} + 2\left[\,\delta x_2 + \delta x_1\,\right] \cdot p_{2,j} + \delta x_2 \cdot p_{1,j} = G_{2,j}$$

which calculates

$$p_{2,j} = \frac{\delta x_1 \cdot \left[\,3\,\dfrac{u_{3,j} - u_{2,j}}{\delta x_2} - p_{3,j}\,\right] + \delta x_2 \cdot \left[\,3\,\dfrac{u_{2,j} - u_{1,j}}{\delta x_1} - p_{1,j}\,\right]}{2 \cdot (\delta x_2 + \delta x_1)} \qquad .[B.13]$$

Case n > 3 : In this case p is set to

$$p_{ij} = s_{i-1,j} \cdot p_{i+1,j} + f_{i-1,j} \quad , \quad i = 2, \ \ldots \ , n-1 \qquad [B.14]$$

and one evaluates the functions s and f by inserting Eq. [B.14] into Eq. [B.8] . This procedure results in the following algorithm to calculate the derivatives $p_{ij} = \dfrac{\partial U}{\partial x}(x_i, y_j)$:

$$s_{0,j} = 0 \quad , \quad f_{0,j} = 0$$

$$s_{i-1,j} = -\frac{\delta x_{i-1}}{w_{i-1,j}} \quad , \quad f_{i-1,j} = \frac{G_{i,j} - \delta x_i \cdot f_{i-2,j}}{w_{i-1,j}}$$

$$w_{i-1,j} = 2 \cdot [\delta x_i + \delta x_{i-1}] + \delta x_i \cdot s_{i-2,j} \qquad [B.15]$$

$$p_{i,j} = s_{i-1,j} \cdot p_{i+1,j} + f_{i-1,j}$$

for $i = 2, \ldots, n-1$ and $j = 1, \ldots, m$.

To calculate the y-derivatives , $q_{i,j} = \frac{\partial U}{\partial y}(x_i, y_j)$ from the border values

$$\{ q_{i,1} , q_{i,m} : i = 1, \ldots, n \}$$

the following system of equations can be used :

$$\delta y_{j-1} \cdot q_{i,j+1} + 2[\delta y_j + \delta y_{j-1}] \cdot q_{i,j} + \delta y_j \cdot q_{i,j-1} = H_{i,j} \qquad [B.16]$$

$$j = 2, \ldots, m-1$$

with

$$\delta y_j = y_{j+1} - y_j \quad , \quad j = 1, \ldots, m-1 \qquad [B.17]$$

and

$$H_{i,j} = 3\left[\delta y_{j-1} \frac{u_{i,j+1} - u_{i,j}}{\delta y_j} + \delta y_j \frac{u_{i,j} - u_{i,j-1}}{\delta y_{j-1}}\right] \quad , \quad j = 2, \ldots, m-1 \quad . [B.18]$$

Case n=3 : In this case the Eq. [B.16] - Eq. [B.18] calculate

$$q_{i,2} = \frac{\delta y1 \cdot \left[3\frac{u_{i,3} - u_{i,2}}{\delta y2} - q_{i,3}\right] + \delta y2 \cdot \left[3\frac{u_{i,2} - u_{i,1}}{\delta y1} - q_{i,1}\right]}{2 \cdot (\delta y2 + \delta y1)} \quad . [B.19]$$

Case n > 3 : In this case q is constructed as

$$q_{i,j} = z_{i-1,j} \cdot q_{i,j+1} + g_{i,j-1} \quad , \quad j = 2, \ldots, m-1 \qquad [B.20]$$

and by inserting Eq. [B.20] into Eq. [B.16] the following $q_{i,j}$ - algorithm emerges :

$$z_{i,0} = 0 \quad , \quad g_{i,0} = 0$$

$$z_{i,j-1} = -\frac{\delta y_{j-1}}{v_{i,j-1}} \quad , \quad g_{i,j-1} = \frac{G_{i,j} - \delta y_j \cdot g_{i,j-2}}{v_{i,j-1}}$$

$$v_{i,j-1} = 2 \cdot [\delta y_j + \delta y_{j-1}] + \delta y_j \cdot z_{i,j-2} \qquad [B.21]$$

$$q_{i,j} = z_{i,j-1} \cdot q_{i,j+1} + g_{i,j-1}$$

for $j = 2, \ldots, m-1$ and $i = 1, \ldots, n$.

Caution! The procedures "PREPARE" , "SOLVEX" resp. "SOLVEY" in /2/ - which correspond to the algorithms Eq. [B.15] resp. Eq. [B.21] - contain errors !

All of the second derivatives , $r_{i,j} = \frac{\partial^2 U}{\partial x \, \partial y}(x_i, y_j)$ can be calculated , if only the following corner values

$$\{ r_{1,1} , r_{1,m} , r_{n,1} , r_{n,m} \}$$

are given.

In a first step one can apply the system Eq. [B.8] - Eq. [B.10] to calculate from the y-derivatives the (x,y)-derivatives in the first and last y-row :

$$\delta x_{i-1} \cdot r_{i+1,j} + 2 [\delta x_i + \delta x_{i-1}] \cdot r_{i,j} + \delta x_i \cdot r_{i-1,j} = M_{i,j}$$

$$M_{i,j} = 3 \left[\delta x_{i-1} \frac{q_{i+1,j} - q_{i,j}}{\delta x_i} + \delta x_i \frac{q_{i,j} - q_{i-1,j}}{\delta x_{i-1}} \right] \qquad [B.22]$$

for $j = 1, m$, $i = 2, \ldots, n-1$.

In a second step one can use the y-system Eq. [B.16] - Eq. [B.18] to calculate from the x-derivatives and from the border values

$$\{ r_{i,1} , r_{i,m} : i = 1, \ldots, n \}$$

the the remaining values of the (x,y)-derivatives :

$$\delta y_{j-1} \cdot r_{i,j+1} + 2 [\delta y_j + \delta y_{j-1}] \cdot r_{i,j} + \delta y_j \cdot r_{i,j-1} = N_{i,j}$$

$$N_{i,j} = 3 \left[\delta y_{j-1} \frac{p_{i,j+1} - p_{i,j}}{\delta y_j} + \delta y_j \frac{p_{i,j} - p_{i,j-1}}{\delta y_{j-1}} \right] \qquad [B.23]$$

for $i = 1, \ldots, n$ and $j = 2, \ldots, m-1$.

Derivatives of the surface at the borders.

If the user cannot supply all the needed partial derivatives of U on the borders of the (x,y)-region, one can use the algoritm - described below - to define approximated partial derivatives at the borders with the help of the following sets of functions :

$$\{\ U(x_i, y_j)\ :\ i = 1\ ,\ \ldots\ ,\ n\ \}\quad ,\quad j = 1\ ,\ \ldots\ ,\ m \qquad \text{and}$$

$$\{\ U(x_i, y_j)\ :\ j = 1\ ,\ \ldots\ ,\ m\ \}\quad ,\quad i = 1\ ,\ \ldots\ ,\ n \qquad .$$

Calculating the derivative of a numerically defined function

Let a function F(x) be defined numerically, i.e. by its ordinates and abscissae :

$$F(x) \equiv \{\ F(i)\ ,\ x(i)\ ;\ i = 1\ ,\ \ldots\ ,\ n\ \} \quad . \qquad [B.24]$$

As a derivative of this function one needs a set of approximated values for $F1 \equiv dF/dx$ at the points x(i) .

In the median points of the ordinates

$$x_1(i) \equiv \frac{x(i+1) + x(i)}{2}\ ,\ i = 1,\ldots,n-1 \qquad [B.25]$$

one can calculate the following differences

$$\begin{aligned} \delta x(i) &\equiv x(i+1) - x(i) \quad , \\ \delta F(i) &\equiv F(i+1) - F(i) \quad , \\ i &= 1,\ldots,n-1 \quad . \end{aligned} \qquad [B.26]$$

and the following approximated derivatives :

$$F1^o(i) \equiv \frac{\delta F(i)}{\delta x(i)}\quad ,\ i = 1,\ldots,n-1 \quad . \qquad [B.27]$$

Since these derivatives are not located properly at x(i) where they are needed, but at the places

$$x_1(i) = x(i) + \frac{\delta x(i)}{2} \quad ,$$

one has to evaluate $F1^o$ into a series of Taylor :

$$\begin{aligned} &F1^o[x(1)] \simeq F1^o[x_1(1)] + \Delta x(1)\ \{\ F2[x_1(1)] + \frac{\Delta x(1)}{2} F3[x_1(1)]\ \} \\ &\text{with} \qquad \Delta x(1) \equiv x(1) - x_1(1) \qquad \text{and} \\ &F1^o[x(i+1)] \simeq F1^o[x_1(i)] + \Delta x(i)\ \{\ F2[x_1(i)] + \frac{\Delta x(i)}{2} F3[x_1(i)]\ \} \\ &\text{with} \qquad \Delta x(i) \equiv x(i+1) - x_1(i)\ ,\ i = 1\ ,\ \ldots\ ,\ n-1 \quad . \end{aligned} \qquad [B.28]$$

F2 and F3 in this formula denote the 2. and 3. derivatives of the function Eq. [B.24] . For approximating these derivatives of higher order one needs some additional differences.

At the median points of second order

$$x_2(i) \equiv \frac{x_1(i+1) + x_1(i)}{2}\ ,\ i = 1,\ldots,n-2 \qquad [B.29]$$

one has the following secondary differences :

$$\delta x_2(i) \equiv x_1(i+1) - x_1(i) \quad ,$$
$$\delta F2(i) \equiv F1(i+1) - F1(i) \quad , \qquad [B.30]$$
$$i = 1,...,n-2 \quad .$$

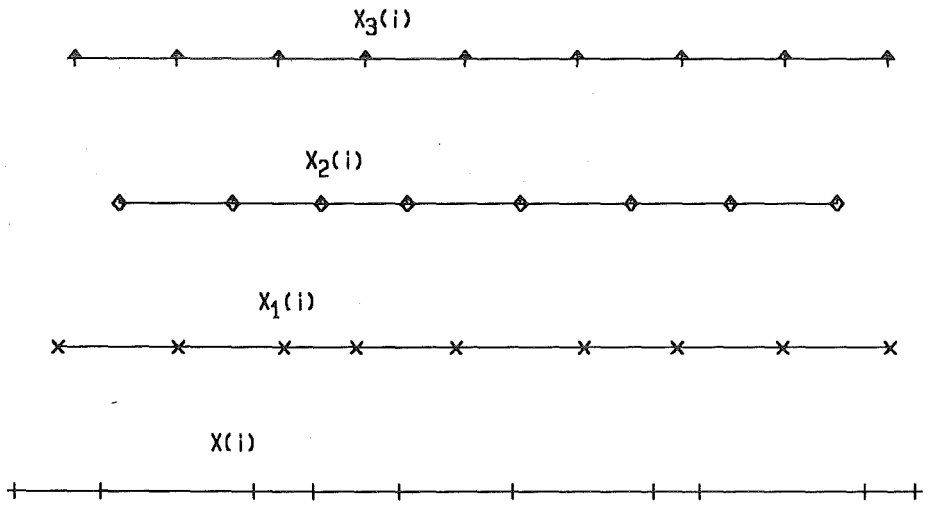
So at the secondary medians the following approximated secondary derivatives can be formed :

$$F2^o(i) \equiv \frac{\delta F2(i)}{\delta x_2(i)} \quad , \quad i = 1,...,n-2 \quad . \qquad [B.31]$$

Correspondingly at the medians of third order

$$x_3(i) \equiv \frac{x_2(i+1) + x_2(i)}{2} \quad , \quad i = 1,...,n-3 \qquad [B.32]$$

(s. Fig.)



there are the differences

$$\delta x_3(i) \equiv x_2(i+1) - x_2(i) \quad ,$$
$$\delta F3(i) \equiv F2(i+1) - F2(i) \quad , \qquad [B.33]$$
$$i = 1,...,n-3 \quad ,$$

and the approximated derivatives of the third order :

$$F3^o(i) \equiv \frac{\delta F3(i)}{\delta x_3(i)} \quad , \quad i = 1,...,n-3 \quad . \qquad [B.34]$$

Evaluating the approximated derivatives of the third order, F3 . In the Eq. [B.28] n-1 values of the derivatives $F3^o$ are needed, instead of the n-3 values given by the Eq. [B.34] . The missing two derivatives can be implemented by extrapolating x_3 and $F3^o$ at the boundaries linearly :

$$
\begin{aligned}
x_3(i) &\equiv \frac{x_2(i) + x_2(i-1)}{2} \quad , \; i = 2,\ldots,n-2 \quad , \\
x_3(1) &\equiv 2 \cdot x_3(2) - x_3(3) \\
x_3(n-1) &\equiv 2 \cdot x_3(n-2) - x_3(n-3)
\end{aligned}
\qquad [B.35]
$$

respectively

$$
\begin{aligned}
F3^{\times}(i) &\equiv F3^{o}(i-1) \, , \; i = 2,\ldots,n-2 \quad , \\
F3^{\times}(1) &\equiv 2 \cdot F3^{\times}(2) - F3^{\times}(3) \\
F3^{\times}(n-1) &\equiv 2 \cdot F3^{\times}(n-2) - F3^{\times}(n-3) \quad .
\end{aligned}
\qquad [B.36]
$$

$F3^{\times}$ is then used in the Eq. [B.28] as an approximation for the third derivative :

$$
F3[x_1(i)] \simeq F3^{\times}[x_3(i)] \qquad , \qquad i = 1,\ldots,n-1 \quad .
$$

The approximated derivatives of the second order, F2 . Since $F2^{o}$ - needed in the Eq. [B.28] - is also given at the wrong places $x_2(i)$ it has to be evaluated for the locations $x_1(i)$:

$$
\begin{aligned}
& F2[x_1(1)] \simeq F2^{o}[x_1(1)] \simeq F2^{o}[x_2(1)] + \Delta x_1(1) \cdot F3[x_2(1)] \\
& \text{with} \quad \Delta x_1(1) \equiv x_1(1) - x_2(1) \quad \text{and} \\
& F2[x_1(i+1)] \simeq F2^{o}[x_1(i+1)] \simeq F2^{o}[x_2(i)] + \Delta x_1(i) \cdot F3[x_2(i)] \\
& \text{with} \quad \Delta x_1(i) \equiv x_1(i+1) - x_2(i) \quad \text{for} \quad i = 1,\ldots,n-2 \quad .
\end{aligned}
\qquad [B.37]
$$

The third order derivatives in this eq. are approximated by median values :

$$
F3[x_2(i)] \simeq \frac{F3^{\times}[x_3(i+1)] + F3^{\times}[x_3(i)]}{2} \qquad , \quad i = 1,\ldots,n-2 \qquad [B.38]
$$

Using the described approximations Eq. [B.38] Eq. [B.37] Eq. [B.36] with the Eq. [B.28] gives the following formula for callculating the derivative , F1 :

$$F1(1) \simeq F1^{o}[x(1)] \simeq F1^{o}(1) - \frac{\delta x(1)}{2} \{ F2^{o}(1) +$$

$$- \frac{3\,\delta x(1) + \delta x(2)}{8} F3^{x}(1) - \frac{\delta x(1) + \delta x(2)}{8} F3^{x}(2) \} \quad ,$$

$$F1(2) \simeq F1^{o}[x(2)] \simeq F1^{o}(1) + \frac{\delta x(1)}{2} \{ F2^{o}(1) +$$

$$+ \frac{\delta x(1) - \delta x(2)}{8} F3^{x}(1) - \frac{\delta x(1) + \delta x(2)}{8} F3^{x}(2) \} \qquad [B.39]$$

and

$$F1(i) \simeq F1^{o}[x(i)] \simeq F1^{o}(i-1) + \frac{\delta x(i-1)}{2} \{ F2^{o}(i-2) +$$

$$+ \frac{3\,\delta x(i-1) + \delta x(i-2)}{8} F3^{x}(i-1) + \frac{\delta x(i-1) + \delta x(i-2)}{8} F3^{x}(i-2) \}$$

for $i = 3, \ldots, n$.

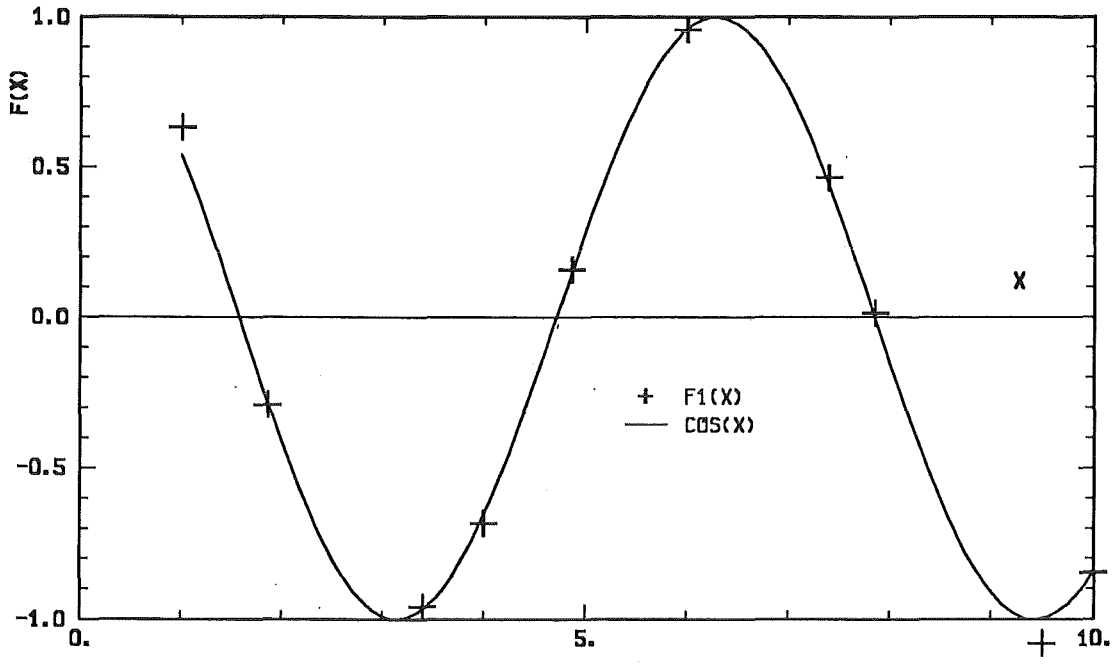
Example. For testing this algorithm the approximated derivative of the function **sin** (x) was calculated. As a set of ordinates the following grid was chosen :

The grid of the abscissae				
i	**x**	x_1	x_2	x_3
1	1.			
		1.43		1.58
2	1.86		2.03	
		2.63		2.5975
3	3.4		3.165	
		3.7		3.615
4	4.0		4.065	
		4.43		4.4975
5	4.86		4.93	
		5.43		5.4975
6	6.0		6.065	
		6.7		6.615
7	7.4		7.165	
		7.63		7.66
8	7.86		8.155	
		8.68		8.685
9	9.5		9.215	
		9.75		9.71
10	10.			

A second table lists the values of the function x(i) , sin(i) , the values of the derivative cos(i) and the values of the approximated derivative F1(i) :

Results				
x	**sin(x)**	**F1(x)**	**cos(x)**	**F1(x)-cos(x)**
1.00	.8414710	.6318560	.54030200	-.09155370
1.86	.9584710	-.2912690	-.28518900	.00608020
3.40	-.2555410	-.9583770	-.96679800	-.00842167
4.00	-.7568020	-.6834000	-.65364400	.02975610
4.86	-.9891250	.1575790	.14707600	-.01050350
6.00	-.2794150	.9573570	.96017000	.00281366
7.40	.8987080	.4641260	.43854700	-.02557890
7.86	.9999820	.0122363	-.00601833	-.01825460
9.50	-.0751511	-1.078840	-.99717200	.08167180
10.00	-.5440210	-.8456540	-.83907200	.00658228

The expected and calculated derivatives are also shown in the following figure :



Interpolating with SINSUR in a batch-job.

A would be user, who wants to calculate interpolated values to a given surface U(x,y) can do this in a batch-job by linking to his program the FORTRAN-moduls of SINSUR and a block-data description of his surface. He must also supply the coordinates for the points of the interpolation.

The Routines of the FORTRAN-Version of SINSUR.

The FORTRAN-SINSUR is a package of FORTRAN77 routines, all of them using only REAL*8 variables and constants. The package consists of the following members :

SINSUR
CHECK
TORAND
DERAN
SOLVEX
SOLVEY
MAPROD
FINTER

The code is loaded in the dataset 'INR105.KATHER.LOAD' .

SINSUR
is the main routine, it organizes the flow of the calculations. At the first call

```
CALL SINSUR(V,W,FI,IQ)
```

SINSUR uses

CHECK
to controll the datasets $\{ x_i, i = 1, \ldots, n \}$, $\{ y_j, j = 1, \ldots, m \}$ and $\{ U(x_i, y_j) \}$ for formal errors, then it calls

TORAND
to calculates all the needed derivatives of this surface, namely

$$P = \frac{\partial U}{\partial x} \quad , \quad Q = \frac{\partial U}{\partial y} \quad , \quad R = \frac{\partial^2 U}{\partial x \, \partial y}$$

at the borders with the help of

DERAN (N,Z,F,F1)
which returns the derivative F1 of a function (F,Z) supplied as a set of (N,N)-numbers (s. algorithm Eq. [B.39]) . As a last step of the initiation SINSUR calls

SOLVEX
(algorithm Eq. [B.15]) and

SOLVEY
(algorithm Eq. [B.21]) to calculate the derivatives P , Q and R in the remaining grid-points of the surface.
The routines SINSUR, CHECK, TORAND, SOLVEX and SOLVEY communicate with each other via the named common

```
/SURFAC/ U,P,Q,R,X,Y,DX,DY,N,M,N1,M1,INDA
```

For the actual calculation of the interpolated value SINSUR uses

MAPROD (N,R,S,H,IQ)
which multiplies the (N,N) matrices R and S to the matrix H :

$$H = R \cdot S \quad \text{if} \quad IQ = 0 \quad ,$$

$$H = R^T \cdot S \quad \text{if} \quad IQ = 1 \quad ,$$

$$H = R \cdot S^T \quad \text{if} \quad IQ = 2 \quad .$$

as needed in the Eq. [B.5] to calculate the coefficient-matrices $A^{(i,j)}$, valid in this grid-point.
Whith this matrix given

FINTER (DV,DW,A)
calculates the approximated value FINTER $= \tilde{U}(x, y)$ in a distance

$$DV = x - x_i \quad , \quad DW = y - y_j \quad ,$$

to the nearest grid-point (x_i, y_j) using $A^{(i,j)}$.

Supplying the 3D-surface.

Besides the routines of SINSUR a supplementary routine - named DATUXY - is needed to supply the code with the description of the surface to be interpolated. DATUXY contains the ordinates $\{ x_i, i = 1, \ldots, n \}$ and the abscissae $\{ y_j, j = 1, \ldots, m \}$ of the grid, as well as the data values $\{ U(x_i, y_j) \}$ of the surface.
The x,y-grid may not exceed 128 × 128 mesh-points !
A sample of a DATUXY-routine :

```
      BLOCK DATA DATUXY
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION U(128,128),P(128,128),Q(128,128),R(128,128)
     =,X(128),Y(128),DX(128),DY(128)
      COMMON /SURFAC/ U,P,Q,R,X,Y,DX,DY,N,M,N1,M1,INDA
C
      DATA N/48/,M/20/,INDA/0/
      DATA X/370.000000D+0,450.000000D+0,500.000000D+0,550.000000D+0
     =,      600.000000D+0,650.000000D+0,700.000000D+0,750.000000D+0
     =,      800.000000D+0,850.000000D+0,900.000000D+0,950.000000D+0
     =,      1000.00000D+0,1100.00000D+0,1200.00000D+0,1300.00000D+0
     =,      1350.00000D+0,1380.00000D+0,1400.00000D+0,1430.00000D+0
     =,      1450.00000D+0,1470.00000D+0,1500.00000D+0,1530.00000D+0
     =,      1550.00000D+0,1600.00000D+0,1650.00000D+0,1700.00000D+0
     =,      1750.00000D+0,1800.00000D+0,1850.00000D+0,1900.00000D+0
     =,      1950.00000D+0,2000.00000D+0,2100.00000D+0,2200.00000D+0
     =,      2300.00000D+0,2350.00000D+0,2400.00000D+0,2430.00000D+0
     =,      2450.00000D+0,2470.00000D+0,2490.00000D+0,2500.00000D+0
     =,      2502.00000D+0,2504.00000D+0,2506.00000D+0,2507.00000D+0
     =,  80*0.00000D+0    /
```

```
      DATA Y/0.00000000D+0,0.02000000D+0,0.04000000D+0,0.07000000D+0
     =,      0.10000000D+0,0.15000000D+0,0.22000000D+0,0.30000000D+0
     =,      0.38000000D+0,0.45000000D+0,0.52000000D+0,0.59000000D+0
     =,      0.66000000D+0,0.74000000D+0,0.82000000D+0,0.90000000D+0
     =,      0.95000000D+0,0.98000000D+0,0.99000000D+0,1.00000000D+0
     =, 108*0.00000D+0    /
      DATA ( U(1,J),J=1,128) /           1.13939200D+0,1.14453840D+0
     =,      1.14337520D+0,1.11976420D+0,1.08523460D+0,1.02199780D+0
     =,      0.94077050D+0,0.87739300D+0,0.85444148D+0,0.86074543D+0
     =,      0.88092019D+0,0.90474383D+0,0.92264243D+0,0.94269215D+0
     =,      0.96208716D+0,0.97877703D+0,0.98807533D+0,0.99348199D+0
     =,      0.99555696D+0,1.00000000D+0,108*0.00000D+0 /
      DATA ( U(2,J),J=1,128) /           1.15199500D+0,1.15534200D+0
     =,      1.15032230D+0,1.11296780D+0,1.06445420D+0,0.98844373D+0
     =,      0.90711709D+0,0.85534127D+0,0.84517983D+0,0.85848585D+0
     =,      0.88223876D+0,0.90604695D+0,0.92309414D+0,0.94369788D+0
     =,      0.96273704D+0,0.97904157D+0,0.98811988D+0,0.99341784D+0
     =,      0.99548460D+0,1.00000000D+0,108*0.00000D+0 /
      DATA ( U(3,J),J=1,128) /           1.15574200D+0,1.15972460D+0
     =,      1.15335320D+0,1.10753810D+0,1.05146180D+0,0.97013286D+0
     =,      0.89082513D+0,0.84706679D+0,0.84382015D+0,0.86045229D+0
     =,      0.88541168D+0,0.90839914D+0,0.92475414D+0,0.94524295D+0
     =,      0.96374516D+0,0.97955702D+0,0.98836103D+0,0.99351324D+0
     =,      0.99554363D+0,1.00000000D+0,108*0.00000D+0 /
      DATA ( U(4,J),J=1,128) /           1.15921070D+0,1.16401490D+0
     =,      1.15627310D+0,1.10333610D+0,1.04121560D+0,0.95623902D+0
     =,      0.87991067D+0,0.84392370D+0,0.84629641D+0,0.86528552D+0
     =,      0.89046581D+0,0.91195417D+0,0.92753108D+0,0.94743745D+0
     =,      0.96511489D+0,0.98020431D+0,0.98861021D+0,0.99355173D+0
     =,      0.99553350D+0,1.00000000D+0,108*0.00000D+0 /
.....      .....      .....      .....      .....      .....      .....
.....      .....      .....      .....      .....      .....      .....
.....      .....      .....      .....      .....      .....      .....
      DATA (U(46,J),J=1,128) /           0.84135738D+0,0.84272810D+0
     =,      0.84414851D+0,0.84631950D+0,0.84854624D+0,0.85238985D+0
     =,      0.85807462D+0,0.86506352D+0,0.87266642D+0,0.87991761D+0
     =,      0.88784096D+0,0.89658510D+0,0.90636141D+0,0.91921474D+0
     =,      0.93465919D+0,0.95445696D+0,0.97114311D+0,0.98491151D+0
     =,      0.99103114D+0,1.00000000D+0,108*0.00000D+0 /
      DATA (U(47,J),J=1,128) /           0.84680346D+0,0.84783954D+0
     =,      0.84924806D+0,0.85136870D+0,0.85354964D+0,0.85731965D+0
     =,      0.86290615D+0,0.86978744D+0,0.87728315D+0,0.88443439D+0
     =,      0.89224185D+0,0.90083852D+0,0.91041328D+0,0.92292224D+0
     =,      0.93782070D+0,0.95675960D+0,0.97270741D+0,0.98583633D+0
     =,      0.99161005D+0,1.00000000D+0,108*0.00000D+0 /
      DATA (U(48,J),J=1,128) /           0.85371996D+0,0.85446912D+0
     =,      0.85583688D+0,0.85811263D+0,0.86040003D+0,0.86430864D+0
     =,      0.86999694D+0,0.87685431D+0,0.88416695D+0,0.89102154D+0
     =,      0.89840309D+0,0.90644832D+0,0.91535924D+0,0.92700918D+0
     =,      0.94101671D+0,0.95911015D+0,0.97436841D+0,0.98674482D+0
     =,      0.99217072D+0,1.00000000D+0,108*0.00000D+0 /
      END
```

Linking the components of the batch-job.

There are two differnt methods to combine SINSUR with the routine DATUXY and with the main program ("TESSIN" in the following examples) of the user.

Connecting a "loaded" DATUXY : In this case the block data routine

```
'inr000.userfort.FORT(DATUXY)'
```

had been compiled and loaded previously and is now located in the member

```
'inr000.userload.LOAD(DATUXY)'
```

.

DATUXY will be then connected to the job using the symbolic name "BERS" in the LINK-step as shown in the first job-example :

```
//inr000L  JOB (0abc,xyc,p9x9y),dombrowsky,MSGLEVEL=(1,1)
//TESSIN EXEC F7CLG
//C.SYSIN DD *
      PROGRAM TESSIN
C     TESTING SINSUR
      IMPLICIT REAL*4 (A-H,O-$)
      DIMENSION VL(60),WL(4),FF(4)
      REAL*8 V,W,FI
      NAMELIST /LALINE/ NV,NW,VL,WL
C     INITIALISING SINSUR
      DATA V,W/0.D0,0.D0/
      CALL SINSUR(V,W,FI,IQ)
  120 READ(5,LALINE)
      IF(NV .LE. 0)          GOTO 999
      IF(NW .LE. 0)          GOTO 999
      IF(NW .GT. 4)          NW=4
      WRITE(6,1000) (WL(K),K=1,NW)
      DO 201 J=1,NV
      V=VL(J)
      DO 211 I=1,NW
      W=WL(I)
      CALL SINSUR(V,W,FI,IQ)
  211 FF(I)=FI
  201 WRITE(6,2000) J,V,(FF(I),I=1,NW)
 1000 FORMAT(1H1,11X,1HV,23X,28HFI(V,W) ON THE ISOLINES  W = /
     =/20X,4G15.6 / )
 2000 FORMAT(I5,5G15.6)
          GO TO 120
C
  999 STOP
      END
//*
//L.BIKA DD DISP=SHR,DSN=INR105.KATHER.LOAD
//L.BERS DD DISP=SHR,DSN=inr000.userload.LOAD
//L.SYSIN DD *
 INCLUDE BIKA(SINSUR)
 INCLUDE BERS(DATUXY)
 ENTRY TESSIN
```

```
//*
//G.SYSPRINT DD SYSOUT=*
//G.SYSIN DD *
 &LALINE NW=4,NV=50,WL=0.220,0.250,0.270,0.300,VL=370.,410.,450.,
 475.,500.,525.,550.,575.,600.,625.,650.,675.,700.,725.,750.,775.,
 800.,825.,850.,875.,900.,925.,950.,975.,1000.,1050.,1100.,1150.,
 1200.,1250.,1300.,1325.,1350.,1365.,1380.,1390.,1400.,1415.,1430.,
 1440.,1450.,1460.,1470.,1485.,1500.,1515.,1530.,1540.,1550.,1575.,
 &END
 &LALINE NW=4,NV=45,WL=0.220,0.250,0.270,0.300,VL=1600.0,1625.0,
 1650.0,1675.0,1700.0,1725.0,1750.0,1775.0,1800.0,1825.0,1850.0,
 1875.0,1900.0,1925.0,1950.0,1975.0,2000.0,2050.0,2100.0,2150.0,
 2200.0,2250.0,2300.0,2325.0,2350.0,2375.0,2400.0,2415.0,2430.0,
 2440.0,2450.0,2460.0,2470.0,2480.0,2490.0,2495.0,2500.0,2501.0,
 2502.0,2503.0,2504.0,2505.0,2506.0,2506.5,2507.0, &END
 &LALINE NW=0,NV=0, &END
//
```

In the above job the coordinates of the points of the interpolation are given in the namelisits LALINE as WL (y) and VL (x). Part of the output of this job are shown on the figures Figure 2 - Figure 5 .

Connecting a fortran-DATUXY in the Compile-step : In the second job-example the block data routine

```
'inr000.userfort.FORT(DATUXY)'
```

will be added directly to the other routines to be compiled :

```
//inr000C  JOB (0abc,xyc,p9x9y),dombrowsky,MSGLEVEL=(1,1)
//TESSIN EXEC F7CLG
//C.SYSIN DD DSN=inr000.userfort.FORT(DATUXY),DISP=SHR
//        DD *
      PROGRAM TESSIN
C     TESTING SINSUR
      IMPLICIT REAL*4 (A-H,O-$)
      DIMENSION VL(60),WL(4),FF(4)
      REAL*8 V,W,FI
      NAMELIST /LALINE/ NV,NW,VL,WL
C     INITIALISING SINSUR
      DATA V,W/0.D0,0.D0/
      CALL SINSUR(V,W,FI,IQ)
  120 READ(5,LALINE)
      IF(NV .LE. 0)          GOTO 999
      IF(NW .LE. 0)          GOTO 999
      IF(NW .GT. 4)          NW=4
      WRITE(6,1000) (WL(K),K=1,NW)
      DO 201 J=1,NV
      V=VL(J)
      DO 211 I=1,NW
      W=WL(I)
      CALL SINSUR(V,W,FI,IQ)
  211 FF(I)=FI
  201 WRITE(6,2000) J,V,(FF(I),I=1,NW)
 1000 FORMAT(1H1,11X,1HV,23X,28HFI(V,W) ON THE ISOLINES  W = /
     =/20X,4G15.6 / )
 2000 FORMAT(I5,5G15.6)
          GO TO 120
C
  999 STOP
      END
//*
//L.BIKA DD DISP=SHR,DSN=INR105.KATHER.LOAD
//L.SYSIN DD *
 INCLUDE BIKA(SINSUR)
 ENTRY TESSIN
//*
//G.SYSPRINT DD SYSOUT=*
//G.SYSIN DD *
 &LALINE NW=4,NV=50,WL=0.220,0.250,0.270,0.300,VL=370.,410.,450.,
 475.,500.,525.,550.,575.,600.,625.,650.,675.,700.,725.,750.,775.,
 800.,825.,850.,875.,900.,925.,950.,975.,1000.,1050.,1100.,1150.,
 1200.,1250.,1300.,1325.,1350.,1365.,1380.,1390.,1400.,1415.,1430.,
 1440.,1450.,1460.,1470.,1485.,1500.,1515.,1530.,1540.,1550.,1575.,
 &END
 &LALINE NW=4,NV=45,WL=0.220,0.250,0.270,0.300,VL=1600.0,1625.0,
 1650.0,1675.0,1700.0,1725.0,1750.0,1775.0,1800.0,1825.0,1850.0,
 1875.0,1900.0,1925.0,1950.0,1975.0,2000.0,2050.0,2100.0,2150.0,
 2200.0,2250.0,2300.0,2325.0,2350.0,2375.0,2400.0,2415.0,2430.0,
 2440.0,2450.0,2460.0,2470.0,2480.0,2490.0,2495.0,2500.0,2501.0,
 2502.0,2503.0,2504.0,2505.0,2506.0,2506.5,2507.0, &END
 &LALINE NW=0,NV=0, &END
//
```

Interpolating with SINSUR in a TSO-session.

For interpolating a surface immediately a second version of SINSUR has been written in the on-line language SPEAKEASY /3/ (the language, actually used was "SPEAKEASY IV Epsilon"). The members of the SPEAKEASY-SINSUR-code (s. "Appendix A. The SPEAKEASY-Version of SINSUR.") are collected in a SPEAKEASY-namelist named also SINSUR. The user can copy this namelist from the dataset

'INR105.SPEAKHLP.TEXT(SINSUR)'.

SINSUR expects the description of the 3D-surface U(x,y) as a two-dimensional array named LAP. The array LAP must be organized as follows (s. e.g. "Appendix B. An example of the array LAP") :

0	Y(1)	Y(2)	....	Y(M)
X(1)	U(1,1)	U(1,2)	....	U(1,M)
X(2)	U(2,1)	U(2,2)	....	U(2,M)
...	...	...	....	...
...	...	...	....	...
...	...	...	....	...
X(N+1)	U(N,1)	U(N,2)	....	U(N,M)

The user can take either the whole surface given in LAP, or select only a part of it.

SINSUR calculates the interpolated values always for a whole line of the given (X,Y)-grid. Either as an x=const.-line :

$$X_i \leq V \leq X_{(i+1)} \quad , \quad W = Y_1 , Y_2 , \ldots , Y_M \quad ,$$

or as an y=const line :

$$V = X_1 , X_2 , \ldots , X_N \quad , \quad Y_j \leq W \leq Y_{(j+1)} \quad .$$

In the case of an x=const-line the user can subdivide his y-grid into K-parts :

$$Y_1 , Y_{11} , Y_{12} , Y_{13} , \ldots , Y_{1K}$$
$$\ldots$$
$$Y_j , Y_{j1} , Y_{j2} , Y_{j3} , \ldots , Y_{jK}$$
$$\ldots$$
$$Y_M , Y_{M1} , Y_{M2} , Y_{M3} , \ldots , Y_{MK}$$

In the case of an y=const-line the x-grid may be subdivided correspondingly.

The interpolated values of the functiun U (X , Y) are named FI (V , W).

In the SPEAKEASY-session - reproduced below - both objects , the namelist SINSUR and the array LAP are members of the SPEAKEASY user-dataset "MYKEPT". The user retrieves these objects from his library and call either the modul SSINEX or SSINEY. The first one interpolates along an x=const, the second along an y=const line. If SSINEX or SSINEY is the first call wich adresses the codesystem, then SINSUR starts the moduls SSINCO - ... - TAYLOR to calculate all of the coefficients $A^{(i,j)}$, , $i = 1,...,N$, $j = 1,...,M$. The calculated coefficients remain in use in the following interpolations, until the user decides to calculate new ones. This can be done by setting **ISIN = 0** .

With the interpolation terminated, the function FI (V , W) will be shown on the screen - if the user has a graphical screen and if he has correctly turned on the graphical capabilities of SPEAKEASY. The figure shows, besides FI (V , W) also the neighbouring val-

ues of the surface U (X , Y) as two different set of points (s. the fig. below). The display of the functions can be stopped by setting **IGRA = 0** .

Record of a SPEAKEASY-session in which SINSUR was used :

```
INPUT...GRAPHICS(GDDM21)
INPUT...SETCOLOR(0,8)
INPUT...MARGINS(60)
INPUT...SINSUR=KEPTLIST(SINSUR)
INPUT...GET LAP

INPUT...WHATIS(SINSUR,LAP)
SINSUR IS A 19 ELEMENT NAME-LITERAL ARRAY
LAP IS A 49 BY 21 REAL ARRAY

INPUT...SINSUR

SINSUR (A 19 COMPONENT ARRAY)
SSINCO   DEFINU   DELZ     PREPARE  TORAND   DERAN
SOLVEX   SOLVEY   TAYLOR   MATRIKS  TESX     TESY
SSINEX   SSINEY   FINTER   DISTANCE FINOM    FIGEX
FIGEY
```

initiating the graphic facility, fetching SINSUR and LAP.

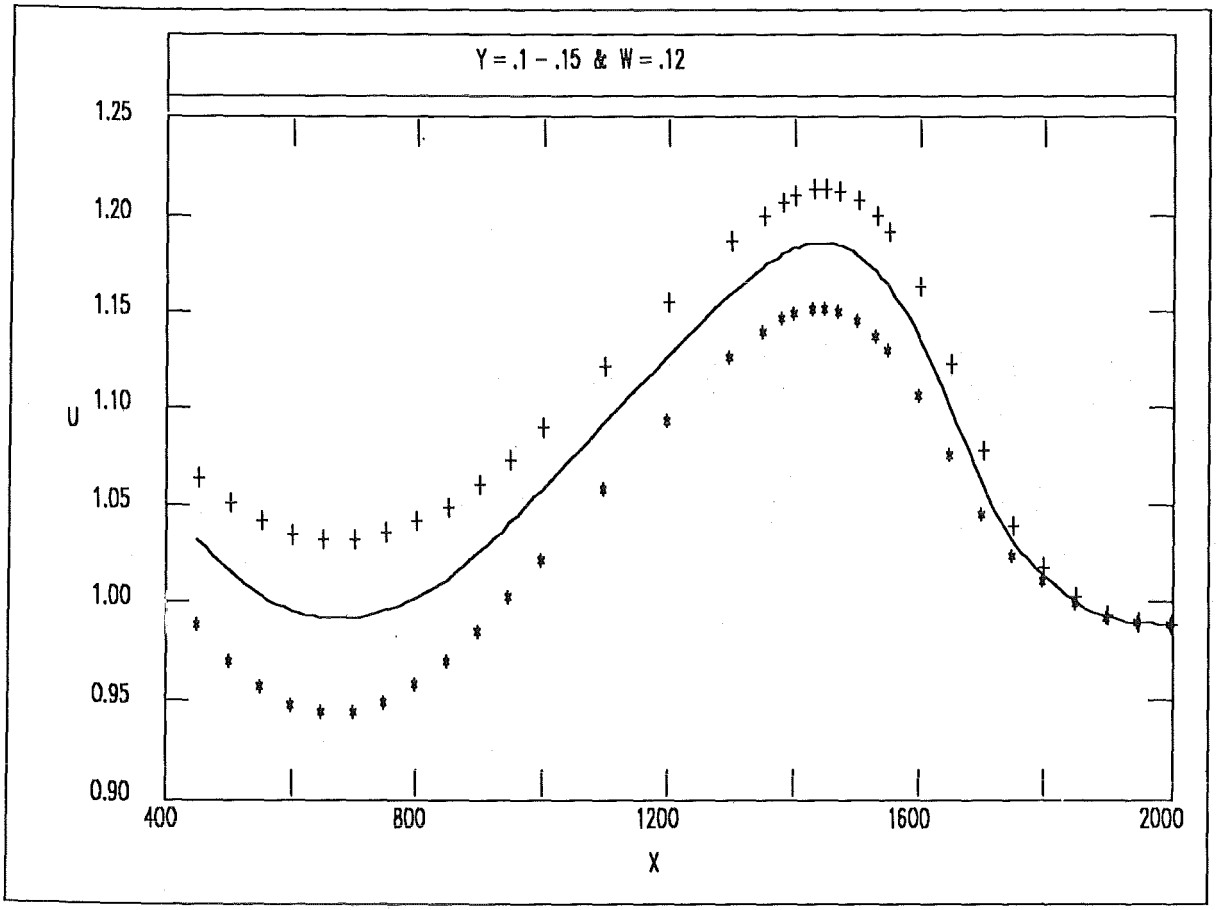
```
INPUT...SSINEY
EXECUTION STARTED

 PROCEDURE SSINCO :
 COEFFICIENTS OF A SPLINE-APPROXIMATION
 OF A FUNCTION-SURFACE  U(X,Y)

 DEFINING THE (X,Y)-SURFACE  U AND THE VARIABLES X & Y :
I1 =  1   I2 =  48
I1 = 2
I2 = 34
J1 =  1   J2 =  20
J1 = <<< NULL LINE ENTERED >>>
J1 =  1
J2 = <<< NULL LINE ENTERED >>>
J2 =  20

 CALCULATING THE BOUNDARY-VALUES FOR
          P=DU/DX , Q=DU/DY , R=D2U/DX/DY :

 PAUSE
:SSINCO><<< NULL LINE ENTERED >>>
INPUT...
PROCEDURE SOLVEX ( P , U )
PROCEDURE SOLVEY ( Q , U )
PROCEDURE SOLVEX ( R , Q )
PROCEDURE SOLVEY ( R , P )
```

```
 PAUSE
:SSINCO><<< NULL LINE ENTERED >>>
INPUT...

 PROCEDURE TAYLOR
 CALCULATING THE COEFFICIENTS OF THE SPLINE-APPROXIMATION
        A11(I,J) , A21(I,J) , ... , A44(I,J) :
```

initiating SINSUR, calculating an y-line

```
 NR     Y
 **   ****
  1   0
 19    .99

 NUMBER <=> CHOSEN Y LINE = 5.4
NR. OF THE PARTITIONS IN ONE X-CELL = 4
Y = .1 - .15  &  W = .12
```

selecting the y-coordinate of the line
subdividing the x-grid



the screen shows the interpolated isoline whith the neighboring y-lines

```
CURSOR (A 2 COMPONENT ARRAY)
  3    4.5
MANUAL MODE

INPUT...TABULATE(V,FI)

    V        FI         V        FI         V        FI
*******  ********   *******  ********   *******  ********
  450     1.0327      987.5   1.0529      1515     1.1762
 462.5    1.0285      1000    1.0573     1522.5    1.174
  475     1.0244      1025    1.066       1530     1.1716
 487.5    1.0204      1050    1.0746      1535     1.1699
  500     1.0166      1075    1.0832      1540     1.168
 512.5    1.0131      1100    1.0918      1545     1.1661
  525     1.0098      1125    1.1004      1550     1.1641
 537.5    1.0068      1150    1.109      1562.5    1.1585
  550     1.0041      1175    1.1175      1575     1.1522
 562.5    1.0017      1200    1.1261     1587.5    1.1452
  575      .99955     1225    1.1345      1600     1.1375
 587.5     .9977      1250    1.1428     1612.5    1.1292
  600      .9961      1275    1.1509      1625     1.1203
 612.5    .99472      1300    1.1587     1637.5    1.1111
  625     .99358     1312.5   1.1624      1650     1.1014
 637.5    .99269      1325    1.166      1662.5    1.0916
  650     .99207     1337.5   1.1694      1675     1.0816
 662.5    .99172      1350    1.1726     1687.5    1.0719
  675     .99161     1357.5   1.1744      1700     1.0625
 687.5    .99173      1365    1.1762     1712.5    1.0537
  700     .99204     1372.5   1.1778      1725     1.0455
 712.5    .99252      1380    1.1793     1737.5    1.0383
  725     .99319      1385    1.1802      1750     1.032
 737.5    .99407      1390    1.1811     1762.5    1.0267
  750     .99518      1395    1.1819      1775     1.0223
 762.5    .99652      1400    1.1827     1787.5    1.0184
  775      .9981     1407.5   1.1837      1800     1.0148
 787.5    .99988      1415    1.1845     1812.5    1.0112
  800     1.0018     1422.5   1.1852      1825     1.0076
 812.5    1.004       1430    1.1857     1837.5    1.0043
  825     1.0063      1435    1.1859      1850     1.0012
 837.5    1.0088      1440    1.186      1862.5     .99846
  850     1.0115      1445    1.186       1875      .99614
 862.5    1.0144      1450    1.1859     1887.5     .99419
  875     1.0176      1455    1.1858      1900      .99263
 887.5    1.021       1460    1.1855     1912.5     .99144
  900     1.0245      1465    1.1852      1925      .99058
 912.5    1.0282      1470    1.1848     1937.5     .98998
  925     1.032      1477.5   1.1839      1950      .9896
 937.5    1.0359      1485    1.1828     1962.5     .98938
  950     1.04       1492.5   1.1815      1975      .98918
 962.5    1.0442      1500    1.1799     1987.5     .98887
  975     1.0485     1507.5   1.1782      2000      .98832
```

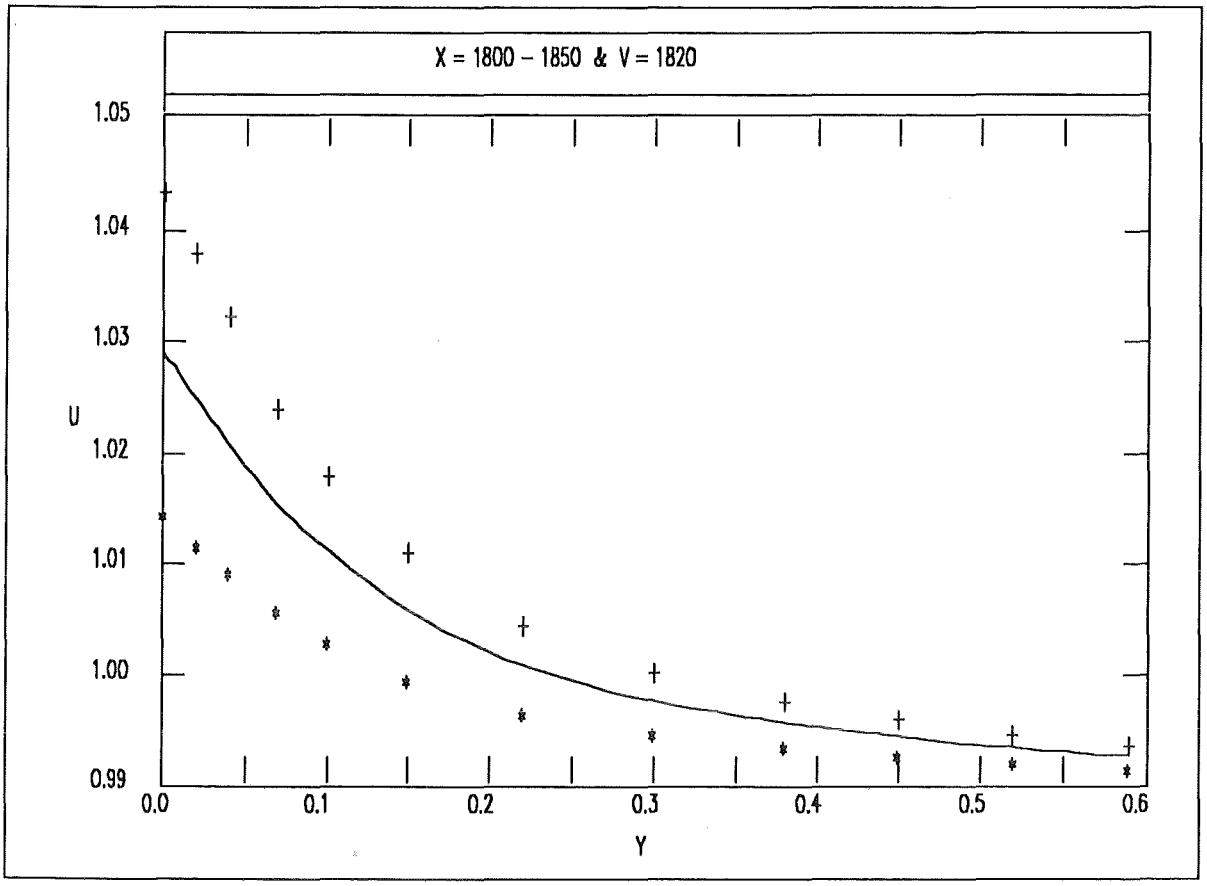
printing the interpolated y-line

```
INPUT...ISIN
ISIN =  1

INPUT...ISIN=0

INPUT...SSINEX
EXECUTION STARTED

 PROCEDURE SSINCO :
 COEFFICIENTS OF A SPLINE-APPROXIMATION
 OF A FUNCTION-SURFACE  U(X,Y)

 DEFINING THE (X,Y)-SURFACE  U AND THE VARIABLES X & Y :
I1 =  1   I2 =  48
I1 = 2
I2 = 34
J1 =  1   J2 =  20
J1 = 1
J2 = 12
 CALCULATING THE BOUNDARY-VALUES FOR
          P=DU/DX , Q=DU/DY , R=D2U/DX/DY :

 PAUSE
:SSINCO><<< NULL LINE ENTERED >>>
INPUT...
PROCEDURE SOLVEX ( P , U )
PROCEDURE SOLVEY ( Q , U )
PROCEDURE SOLVEX ( R , Q )
PROCEDURE SOLVEY ( R , P )
 PAUSE
:SSINCO><<< NULL LINE ENTERED >>>
INPUT...

 PROCEDURE TAYLOR
 CALCULATING THE COEFFICIENTS OF THE SPLINE-APPROXIMATION
        A11(I,J) , A21(I,J) , ... , A44(I,J) :
```

calculating an x-line on a new partial-surface

```
   NR      X
 *****   ****
  1.00    450
 32.00   1950

 NUMBER <=> CHOSEN X LINE = 29.4
NR. OF THE PARTITIONS IN ONE Y-CELL = 6
X = 1800 - 1850  &  V = 1820
```

selecting the x-coordinate of the line
subdividing the y-grid



the screen shows the x-isoline whith the neighborhood.

```
CURSOR (A 2 COMPONENT ARRAY)
  3     4.5

MANUAL MODE
```

```
INPUT...TABULATE(W,FI)

     W          FI          W          FI
 **********  ********  **********  ********
 0           1.0291       .19667    1.0024
  .0033333   1.0284       .20833    1.0016
  .0066667   1.0277       .22       1.0009
  .01        1.027        .23333    1.0002
  .013333    1.0263       .24667     .99962
  .016667    1.0257       .26        .99907
  .02        1.025        .27333     .99858
  .023333    1.0243       .28667     .99813
  .026667    1.0236       .3         .99771
  .03        1.0229       .31333     .99733
  .033333    1.0223       .32667     .99697
  .036667    1.0216       .34        .99662
  .04        1.0209       .35333     .9963
  .045       1.02         .36667     .996
  .05        1.019        .38        .99571
  .055       1.0181       .39167     .99546
  .06        1.0172       .40333     .99523
  .065       1.0163       .415       .995
  .07        1.0155       .42667     .99479
  .075       1.0147       .43833     .99458
  .08        1.014        .45        .99438
  .085       1.0133       .46167     .99419
  .09        1.0126       .47333     .99401
  .095       1.012        .485       .99384
  .1         1.0114       .49667     .99367
  .10833     1.0104       .50833     .99351
  .11667     1.0094       .52        .99336
  .125       1.0085       .53167     .99322
  .13333     1.0076       .54333     .99308
  .14167     1.0068       .555       .99295
  .15        1.006        .56667     .99282
  .16167     1.005        .57833     .9927
  .17333     1.004        .59        .99259
  .185       1.0032

INPUT...QUIT
 SPACE USED  112 K NOW,  150 K PEAK,  SIZE   300 K
```

printing the interpolated x-line
closing the session.

References.

/1/ C. de Boor, Bicubic Spline Interpolation,
J. Math. Phys., **41**. p. 212-218 1962.

/2/ H. Spaeth, Algorithmus 10. Zweidimensionale glatte Interpolation,
Computing **4**, p. 178-182, 1969.

/3/ S. Cohen, S. C. Pieper, The Speakeasy-3 Reference Manual,
Argonne National Laboratory, Dec. 1977.

Figures.

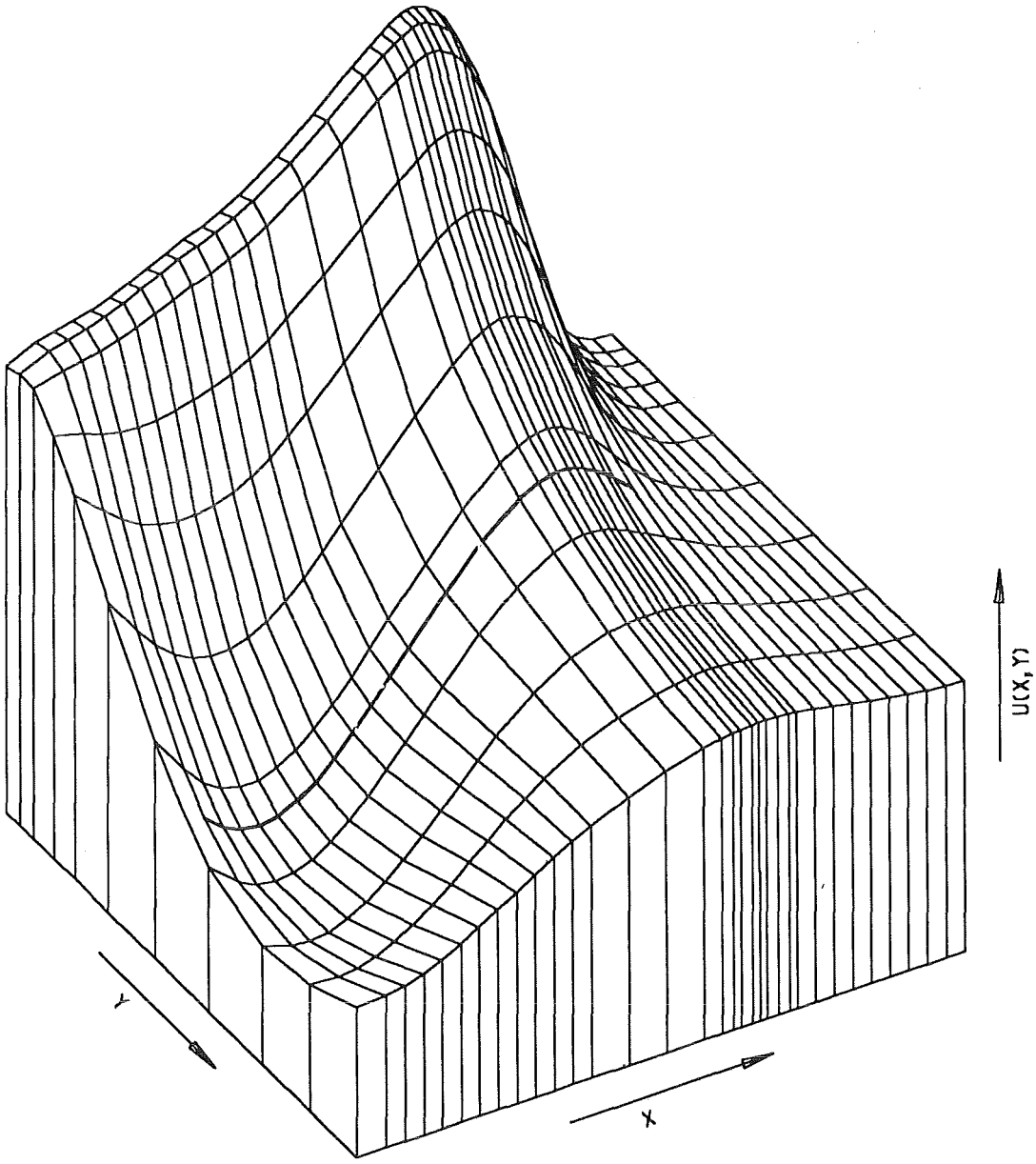


Figure 1. Smooth interpolation on a 3-D surface.

```
 MVS/XA DFP VER 2 LINKAGE EDITOR                    13:47:11  THU  JUL 06, 1989
 JOB inr000c    STEP TESSIN     PROCEDURE L
 INVOCATION PARAMETERS - LIST
 ACTUAL SIZE=(317440,86016)
          SYSPRINT DEFAULT BLOCKING USED   1 - 1
 OUTPUT DATA SET SYS89187.T134634.RA000.inr000c.GOSET IS ON VOLUME
IEW0000     INCLUDE BIKA(SINSUR)
IEW0000     INCLUDE BERS(DATUXY)
IEW0000     ENTRY TESSIN
 ** MAIN     DID NOT PREVIOUSLY EXIST BUT WAS ADDED AND HAS AMODE 31
 ** LOAD MODULE HAS RMODE ANY
 ** AUTHORIZATION CODE IS         0.
```

Figure 2. Job inr000c . Page L-1.

```
 THE VARIABLES  X , Y  AND THE SURFACE U(X,Y) ARE NOW LOADED .
 THE BOUNDARY-VALUES OF THE U-DERIVATIVES ARE NOW SET .
 THE CALCULATION OF THE FUNCTION  P = DU/DX IS NOW COMPLETED .
 THE CALCULATION OF THE FUNCTION  Q = DU/DY IS NOW COMPLETED .
 FIRST AND LAST COLUMNS OF THE FUNCTION  R = D2U/DXDY ARE CALCULATED .
 THE CALCULATION OF THE FUNCTION  R = D2U/DXDY IS NOW COMPLETED .
```

Figure 3. Job inr000c . Page G-1.

	V	FI(V,W) ON THE ISOLINES W =			
		0.220000	0.250000	0.270000	0.300000
1	370.000	0.940770	0.912551	0.896591	0.877393
2	410.000	0.923258	0.896624	0.882066	0.865291
3	450.000	0.907117	0.882467	0.869517	0.855341
4	475.000	0.898353	0.875045	0.863138	0.850585
5	500.000	0.890825	0.868899	0.858038	0.847067
6	525.000	0.884644	0.864115	0.854285	0.844833
7	550.000	0.879911	0.860773	0.851942	0.843924
8	575.000	0.876687	0.858918	0.851037	0.844343
9	600.000	0.874896	0.858457	0.851464	0.845971
10	625.000	0.874442	0.859286	0.853108	0.848679
11	650.000	0.875321	0.861383	0.855940	0.852434
12	675.000	0.877518	0.864716	0.859925	0.857203
13	700.000	0.880899	0.869134	0.864910	0.862842
14	725.000	0.885322	0.874482	0.870741	0.869202
15	750.000	0.890738	0.880707	0.877364	0.876234
16	775.000	0.897098	0.887756	0.884727	0.883892
17	800.000	0.904248	0.895476	0.892682	0.892036
18	825.000	0.912042	0.903718	0.901086	0.900534
19	850.000	0.920442	0.912452	0.909915	0.909368
20	875.000	0.929411	0.921651	0.919146	0.918524
21	900.000	0.938795	0.931174	0.928653	0.927886
22	925.000	0.948433	0.940880	0.938304	0.937333
23	950.000	0.958267	0.950720	0.948060	0.946839
24	975.000	0.968234	0.960645	0.957881	0.956374
25	1000.00	0.978167	0.970504	0.967620	0.965797
26	1050.00	0.997457	0.989578	0.986404	0.983840
27	1100.00	1.01611	1.00784	1.00421	1.00066
28	1150.00	1.03428	1.02532	1.02099	1.01609
29	1200.00	1.05157	1.04158	1.03631	1.02975
30	1250.00	1.06739	1.05599	1.04960	1.04118
31	1300.00	1.08079	1.06770	1.06010	1.04977
32	1325.00	1.08626	1.07228	1.06406	1.05281
33	1350.00	1.09075	1.07586	1.06705	1.05495
34	1365.00	1.09294	1.07750	1.06834	1.05578
35	1380.00	1.09468	1.07871	1.06921	1.05625
36	1390.00	1.09556	1.07926	1.06955	1.05635
37	1400.00	1.09623	1.07959	1.06969	1.05628
38	1415.00	1.09680	1.07968	1.06950	1.05585
39	1430.00	1.09679	1.07924	1.06884	1.05503
40	1440.00	1.09643	1.07865	1.06812	1.05427
41	1450.00	1.09580	1.07781	1.06720	1.05336
42	1460.00	1.09491	1.07676	1.06609	1.05231
43	1470.00	1.09374	1.07548	1.06480	1.05113
44	1485.00	1.09142	1.07313	1.06251	1.04914
45	1500.00	1.08847	1.07030	1.05986	1.04692
46	1515.00	1.08492	1.06704	1.05692	1.04455
47	1530.00	1.08088	1.06348	1.05379	1.04214
48	1540.00	1.07795	1.06098	1.05165	1.04053
49	1550.00	1.07482	1.05838	1.04946	1.03893
50	1575.00	1.06619	1.05143	1.04374	1.03481

Figure 4. Job inr000c . Page G-2.

	V	FI(V,W) ON THE ISOLINES W =			
		0.220000	0.250000	0.270000	0.300000
1	1600.00	1.05690	1.04422	1.03785	1.03056
2	1625.00	1.04764	1.03718	1.03207	1.02622
3	1650.00	1.03894	1.03064	1.02659	1.02185
4	1675.00	1.03123	1.02480	1.02151	1.01753
5	1700.00	1.02443	1.01949	1.01677	1.01335
6	1725.00	1.01840	1.01451	1.01227	1.00940
7	1750.00	1.01316	1.01001	1.00819	1.00587
8	1775.00	1.00863	1.00609	1.00466	1.00285
9	1800.00	1.00438	1.00243	1.00136	1.00003
10	1825.00	1.00009	0.998756	0.998040	0.997142
11	1850.00	0.996246	0.995459	0.995035	0.994488
12	1875.00	0.993402	0.992984	0.992742	0.992408
13	1900.00	0.991476	0.991257	0.991108	0.990886
14	1925.00	0.990217	0.990069	0.989960	0.989797
15	1950.00	0.989336	0.989193	0.989099	0.988963
16	1975.00	0.988564	0.988421	0.988333	0.988210
17	2000.00	0.987761	0.987627	0.987545	0.987434
18	2050.00	0.985812	0.985730	0.985682	0.985621
19	2100.00	0.983580	0.983569	0.983566	0.983569
20	2150.00	0.981243	0.981287	0.981319	0.981373
21	2200.00	0.978565	0.978621	0.978660	0.978722
22	2250.00	0.975056	0.975079	0.975095	0.975121
23	2300.00	0.969642	0.969672	0.969693	0.969732
24	2325.00	0.965854	0.965927	0.965981	0.966070
25	2350.00	0.961231	0.961350	0.961436	0.961573
26	2375.00	0.955480	0.955622	0.955720	0.955872
27	2400.00	0.947350	0.947501	0.947605	0.947765
28	2415.00	0.940633	0.940798	0.940915	0.941100
29	2430.00	0.931736	0.931964	0.932129	0.932398
30	2440.00	0.924216	0.924538	0.924771	0.925150
31	2450.00	0.915221	0.915696	0.916037	0.916588
32	2460.00	0.904630	0.905333	0.905832	0.906631
33	2470.00	0.892462	0.893482	0.894199	0.895334
34	2480.00	0.879038	0.880472	0.881471	0.883036
35	2490.00	0.865736	0.867671	0.869005	0.871079
36	2495.00	0.859942	0.862145	0.863659	0.866002
37	2500.00	0.856575	0.859011	0.860680	0.863254
38	2501.00	0.856391	0.858865	0.860559	0.863171
39	2502.00	0.856542	0.859052	0.860769	0.863415
40	2503.00	0.857134	0.859676	0.861414	0.864089
41	2504.00	0.858075	0.860630	0.862376	0.865063
42	2505.00	0.859531	0.862068	0.863805	0.866479
43	2506.00	0.862906	0.865421	0.867140	0.869787
44	2506.50	0.865859	0.868371	0.870085	0.872718
45	2507.00	0.869997	0.872520	0.874234	0.876854

Figure 5. Job inr000c . Page G-3.

MODULE TREES

```
SSINCO   DEFINU    DELZ      _
         TORAND    DERAN     _
         PREPARE   _
         SOLVEX    _
         SOLVEY    _
         TAYLOR    MATRIKS   _
```

Figure 6. MODULE TREE FOR THE PROCEDURE SSINCO

```
SSINEX ( SSINCO )
         FINOM     _
         FINTER    DISTANCE  _
         FIGEX     _
```

Figure 7. MODULE TREE FOR THE PROCEDURE SSINEX

```
SSINEY ( SSINCO )
         FINOM     _
         FINTER    DISTANCE  _
         FIGEY     _
```

Figure 8. MODULE TREE FOR THE PROCEDURE SSINEY

PREPARING THE COEFFICIENTS

PROGRAM SSINCO

```
 1 PROGRAM
 2 SPACE; " PROCEDURE SSINCO : "
 3 " COEFFICIENTS OF A SPLINE-APPROXIMATION "
 4 " OF A FUNCTION-SURFACE  U(X,Y)  "; SPACE
 5 $   LAP    =   A2D(N+1,M+1:Y,X,U(X,Y))
 6 $ LAP(1)   =    0  ,  Y(1)  , ... ,  Y(M)
 7 $ LAP(2)   =  X(1) , U(1,1) , ... , U(1,M)
 8 $  ...       ...    ...     ...    ...
 9 $ LAP(N+1) =  X(N) , U(N,1) , ... , U(N,M)
10 IF(KIND(IPROTEC) .EQ. 0)        GOTO L1
11 PRINTSPECS(NR: DECIMALS 2)
12 UNPROTECT X0 Y0 U N M
13 L1: DEFINU(LAP,U,X0,Y0,DX,DY); IF(KIND(U) .EQ. 0) RETURN
14 PROTECT X0 Y0 U N M
15 R=U; TORAND(X0,Y0,P,Q,R); PAUSE
16 IF(N .LE. 2 .OR. M .LE. 2)  GOTO LT
17 IF(N .GT. 3)  PREPARE(DX,DX1,DX2,SX,WX)
18 IF(M .GT. 3)  PREPARE(DY,DY1,DY2,SY,WY)
19 DM=1; SOLVEX(P,U,DX,DX1,DX2,SX,WX,DM)
```

```
20 SOLVEY(Q,U,DY,DY1,DY2,SY,WY)
21 DM=M-1; SOLVEX(R,Q,DX,DX1,DX2,SX,WX,DM)
22 SOLVEY(R,P,DY,DY1,DY2,SY,WY); PAUSE
23 LT: TAYLOR
24 ISIN=1; IPROTEC=1
25 END
```

SUBROUTINE DEFINU

```
 1 SUBROUTINE DEFINU(LAP,U,X,Y,DX,DY)
 2 " DEFINING THE (X,Y)-SURFACE  U AND THE VARIABLES X & Y :"
 3 IF(CLASS(LAP) .EQ. 6) GOTO L0
 4 WHATIS LAP; "ERROR : LAP MUST BE A 2.DIM ARRAY"; RETURN
 5 L0: I1=1;I2=NOROWS(LAP)-1
 6 J1=1;J2=NOCOLS(LAP)-1
 7 I1,I2; REQUEST(I1,I2)
 8 J1,J2; REQUEST(J1,J2)
 9 J=INTS(J1+1,J2+1)
10 I=INTS(I1+1,I2+1)
11 IF(NOELS(J) .GE. 2 .AND. NOELS(I) .GE. 2)  GOTO L1
12 "ERROR : THE DOMAIN IS TO SMALL :" I, J; RETURN
13 L1: X=LAP(I,1); DELZ(X,DX,N)
14 IF(N .LE. 0)       RETURN
15 Y=LAP(1,J); DELZ(Y,DY,M)
16 IF(M .LE. 0)       RETURN
17 MAKEGLOBAL N M
18 U=LAP(I,J)
19 END
```

SUBROUTINE DELZ

```
 1 SUBROUTINE DELZ(Z,DZ,L)
 2 L=NOELS(Z); I=INTS(L-1)
 3 DZ=Z(I+1)-Z(I)
 4 D0=MIN(DZ); IF(D0 .GT. 0) RETURN
 5 TYPE("ERROR : ",ARGNAME(1),"MUST BE RANKED "); L=0
 6 TYPE(ARGNAME(2)," = ",DZ)
 7 END
```

SUBROUTINE PREPARE

```
 1 SUBROUTINE PREPARE(DZ,DZ1,DZ2,SZ,WZ)
 2 $ CALCULATING AUXILIARY FUNCTIONS FOR SOLVEX / SOLVEY
 3 L0: K1=NOELS(DZ); K2=K1-1
 4 DZ1=A1D(K2:);WZ=DZ1;SZ=DZ1;DZ2=DZ1
 5 J=INTS(2,K1); DZ1(J)=DZ(J-1)/DZ(J); DZ2(J)=1/DZ1(J)
 6 WZ(1)=0.5/(DZ(1)+DZ(2)); SZ(1)=-DZ(1)*WZ(1)
 7 FOR J=2,K2,1
 8 WZ(J)=1/(2*(DZ(J)+DZ(J+1))+DZ(J+1)*SZ(J-1))
 9 SZ(J)=-DZ(J)*WZ(J)
10 NEXT J
11 END
```

SUBROUTINE DERAN

```
 1 SUBROUTINE DERAN(Z,FZ,FZ1)
 2 $ FZ1 = NUMERICAL DERIVATIVE DFZ/DZ
 3 I=NOELS(FZ)
 4 IF(I .GT. 1)       GOTO L0
 5 " ERROR : "; NOELS(FZ); RETURN
 6 L0: I1=I-1; J1=INTS(I1)
 7 DZ=Z(J1+1)-Z(J1)
 8 Z1=0.5*(Z(J1+1)+Z(J1))
 9 EZ1=(FZ(J1+1)-FZ(J1))/DZ
10 FZ3=A1D(I1:)
11 IF(I .GT. 2)       GOTO L1
12 FZ1=EZ1,EZ1;     RETURN
13 L1: I2=I1-1; J2=INTS(I2)
14 DZ1=Z1(J2+1)-Z1(J2)
15 Z2=0.5*(Z1(J2+1)+Z1(J2))
16 EZ2=(EZ1(J2+1)-EZ1(J2))/DZ1
17 IF(I .GT. 3)       GOTO L2
18 FZ2=EZ2,EZ2;     GOTO L8
19 L2: I3=I2-1; J3=INTS(I3)
20 DZ2=Z2(J3+1)-Z2(J3)
21 Z3=0.5*(Z2(J3+1)+Z2(J3))
22 FZ3=(EZ2(J3+1)-EZ2(J3))/DZ2
23 IF(I .GT. 4)       GOTO L5
24 FU=FZ3(1); FO=FZ3(I3)
25 GOTO L6
26 L5: FU=2*FZ3(1)-FZ3(2);FO=2*FZ3(I3)-FZ3(I3-1)
27 L6: FZ3=FU,FZ3,FO
28 GZ3=0.5*(FZ3(J2+1)+FZ3(J2))
29 FZ2(1)=EZ2(1)-0.5*DZ1(1)*GZ3(1)
30 FZ2(J2+1)=EZ2(J2)+0.5*DZ1(J2)*GZ3(J2)
31 L8: FZ1(1)=EZ1(1)-0.5*DZ(1)*(FZ2(1)-0.25*DZ(1)*FZ3(1))
32 FZ1(J1+1)=EZ1(J1)+0.5*DZ(J1)*(FZ2(J1)+0.25*DZ(J1)*FZ3(J1))
33 END
```

SUBROUTINE TORAND

```
 1 SUBROUTINE TORAND(X,Y,P,Q,R)
 2 " CALCULATING THE BOUNDARY-VALUES FOR          "
 3 "          P=DU/DX , Q=DU/DY , R=D2U/DX/DY : "
 4 SPACE; U=R;P=R-R;Q=P;R=Q
 5 IF(N .LE. 10)      IX=INTS(N)
 6 IF(N .GT. 10)      IX=INTS(5),INTS(N-4,N)
 7 IM=LOCS(1 .LT. IX .AND. IX .LT. N)
 8 IF(MAX(IM) .GT. 0) IM=IX(IM)
 9 Z=X(IX)
10 FOR J=1,M
11 F=U(IX,J);DERAN(Z,F,F1);P(IX,J)=F1
12 NEXT J
13 IF(MAX(IM) .GT. 0)        P(IM)=P(IM)*0
14 FREE F1
15 IF(M .LE. 10)      IY=INTS(M)
16 IF(M .GT. 10)      IY=INTS(5),INTS(M-4,M)
17 JM=LOCS(1 .LT. IY .AND. IY .LT. M)
18 IF(MAX(JM) .GT. 0) JM=IY(JM)
19 Z=Y(IY)
```

```
20 FOR I = 1,N
21 F = U(I,IY);DERAN(Z,F,F1);Q(I,IY) = F1
22 NEXT I
23 IF(MAX(JM) .GT. 0)        Q(,JM) = Q(,JM)*0
24 FREE F1
25 L = 1,M
26 F = P(1);DERAN(Y,F,F1);R(1,L) = F1(L)
27 F = P(N);DERAN(Y,F,F1);R(N,L) = F1(L)
28 END
```

SUBROUTINE SOLVEX

```
 1 SUBROUTINE SOLVEX(P,U,DX,DX1,DX2,SX,WX,DM)
 2 TYPE(" PROCEDURE SOLVEX (" ARGNAME(P) "," ARGNAME(U) ")" )
 3 J = INTS(1,M,DM)
 4 IF(N .GT. 3)  GOTO L3
 5 A1 = 3*(U(3,J)-U(2,J))/DX(2)-P(3,J)
 6 A2 = 3*(U(2,J)-U(1,J))/DX(1)-P(1,J)
 7 P(2,J) = (DX(1)*A1 + DX(2)*A2)/(2*(DX(2) + DX(1)))
 8 RETURN
 9 L3: I = INTS(2,N1);IP = I + 1;IN = I-1
10 UP = U-U;UN = UP;F = UP
11 UP(I,J) = U(IP,J)-U(I,J)
12 UN(I,J) = U(I,J)-U(IN,J)
13 P(I,J) = 3*(DX1(I)*UP(I,J) + DX2(I)*UN(I,J))
14 P(2,J)  = P(2,J)-DX(2)*P(1,J)
15 P(N1,J) = P(N1,J)-DX(N2)*P(N,J)
16 F(1,J) = P(2,J)*WX(1)
17 FOR L = 2,N2,1
18 F(L,J) = (P(L + 1,J)-DX(L + 1)*F(L-1,J))*WX(L)
19 NEXT L
20 P(N1,J) = F(N2,J)
21 FOR L = N2,2,-1
22 P(L,J) = SX(L-1)*P(L + 1,J) + F(L-1,J)
23 NEXT L
24 END
```

SUBROUTINE SOLVEY

```
 1 SUBROUTINE SOLVEY(Q,U,DY,DY1,DY2,SY,WY)
 2 TYPE(" PROCEDURE SOLVEY (" ARGNAME(Q) "," ARGNAME(U) ")" )
 3 I = INTS(1,N)
 4 IF(M .GT. 3)  GOTO L3
 5 A1 = 3*(U(I,3)-U(I,2))/DY(2)-Q(I,3)
 6 A2 = 3*(U(I,2)-U(I,1))/DY(1)-Q(I,1)
 7 Q(I,2) = (DY(1)*A1 + DY(2)*A2)/(2*(DY(2) + DY(1)))
 8 RETURN
 9 L3: J = INTS(2,M1);JP = J + 1;JM = J-1
10 UP = U-U;UM = UP;F = UP
11 UP(I,J) = U(I,JP)-U(I,J)
12 UM(I,J) = U(I,J)-U(I,JM)
13 Q(I,J) = 3*(UP(I,J)*DY1(J) + UM(I,J)*DY2(J))
14 Q(I,2) =  Q(I,2)-DY(2)*Q(I,1)
15 Q(I,M1) = Q(I,M1)-DY(M2)*Q(I,M)
16 F(I,1) = Q(I,2)*WY(1)
17 FOR L = 2,M2,1
```

```
18 F(I,L) = (Q(I,L + 1)-DY(L + 1)*F(I,L-1))*WY(L)
19 NEXT L
20 Q(I,M1) = F(I,M2)
21 FOR L = M2,2,-1
22 Q(I,L) = SY(L-1)*Q(I,L + 1) + F(I,L-1)
23 NEXT L
24 END
```

PROGRAM TAYLOR

```
 1 PROGRAM
 2 SPACE; " PROCEDURE TAYLOR "
 3 " CALCULATING THE COEFFICIENTS OF THE SPLINE-APPROXIMATION "
 4 "        A11(I,J) , A21(I,J) , ... , A44(I,J) :"
 5 IF(KIND(IPROTEC) .EQ. 0) GOTO L1
 6 UNPROTECT A11 A21 A31 A41 A12 A22 A32 A42
 7 UNPROTECT A13 A23 A33 A43 A14 A24 A34 A44
 8 L1: FOR I = 1,N-1,1
 9 MATRIKS(1/DX(I),BI); I1 = I + 1
10 FOR J = 1,M-1,1
11 MATRIKS(1/DY(J),BJ)
12 KIJ = MATRIX(4,4:); J1 = J + 1
13 KIJ(1) = (U(I,J),Q(I,J),U(I,J1),Q(I,J1))
14 KIJ(2) = (P(I,J),R(I,J),P(I,J1),R(I,J1))
15 KIJ(3) = (U(I1,J),Q(I1,J),U(I1,J1),Q(I1,J1))
16 KIJ(4) = (P(I1,J),R(I1,J),P(I1,J1),R(I1,J1))
17 AA = BI*KIJ*TRANSPOSE(BJ)
18 A11(I,J) = AA(1,1);A21(I,J) = AA(2,1)
19 A31(I,J) = AA(3,1);A41(I,J) = AA(4,1)
20 A12(I,J) = AA(1,2);A22(I,J) = AA(2,2)
21 A32(I,J) = AA(3,2);A42(I,J) = AA(4,2)
22 A13(I,J) = AA(1,3);A23(I,J) = AA(2,3)
23 A33(I,J) = AA(3,3);A43(I,J) = AA(4,3)
24 A14(I,J) = AA(1,4);A24(I,J) = AA(2,4)
25 A34(I,J) = AA(3,4);A44(I,J) = AA(4,4)
26 NEXT J
27 NEXT I
28 FREE AA KIJ BI BJ DX DY DX1 DY1 P Q R
29 PROTECT A11 A21 A31 A41 A12 A22 A32 A42
30 PROTECT A13 A23 A33 A43 A14 A24 A34 A44
31 END
```

SUBROUTINE MATRIKS

```
 1 SUBROUTINE MATRIKS(H,B)
 2 IF(CLASS(H) .EQ. 0) GOTO L0
 3 H; "ERROR : H MUST BE A SCALAR "; RETURN
 4 L0: B = MATRIX(4,4:)
 5 B(1,1) = 1;B(2,2) = 1
 6 SM = H*H
 7 B(3,1) = -3*SM;  B(3,2) = -2*H; B(3,3) = -B(3,1); B(3,4) = -H
 8 B(4,1) = 2*SM*H; B(4,2) = SM;   B(4,3) = -B(4,1); B(4,4) = SM
 9 END
```

PROGRAM SSINEX

```
 1 PROGRAM
 2 $ INTERPOLATING ALONG AN X = CONST. LINE
 3 IF(KIND(ISIN) .LE. 0)       ISIN=0
 4 IF(KIND(IGRA) .LE. 0)       IGRA=1
 5 IF(ISIN .LE. 0)             SSINCO
 6 NR=1,N-1; X=X0(NR); TABULATE(NR,X)
 7 ASK(" NUMBER <=> CHOSEN X LINE =","ZU =")
 8 IU=INTPART(ZU);RU=FRACPART(ZU)
 9 IF(IU .GE. 1 .AND. IU .LT. N)        GOTO L0
10 " ERROR "; IU; RETURN
11 L0: V0=X0(IU); V1=X0(IU+1); V=V0+RU*(V1-V0)
12 ASK("NR. OF THE PARTITIONS IN ONE Y-CELL =","IGR =","IGR=2")
13 FINOM(Y0,W,MI,IGR)
14 FINTER
15 BT="X = "TYPE(V0)" - "TYPE(V1)" & V = "TYPE(V)
16 BT; IF(IGRA .EQ. 0) RETURN
17 FIGEX
18 END
```

PROGRAM SSINEY

```
 1 PROGRAM
 2 $ INTERPOLATING ALONG AN Y = CONST. LINE
 3 IF(KIND(ISIN) .LE. 0)       ISIN=0
 4 IF(KIND(IGRA) .LE. 0)       IGRA=1
 5 IF(ISIN .LE. 0)             SSINCO
 6 NR=1,M-1; Y=Y0(NR); TABULATE(NR,Y)
 7 ASK(" NUMBER <=> CHOSEN Y LINE =","ZU =")
 8 IU=INTPART(ZU);RU=FRACPART(ZU)
 9 IF(IU .GE. 1 .AND. IU .LT. M)        GOTO L0
10 " ERROR "; IU; RETURN
11 L0: W0=Y0(IU); W1=Y0(IU+1); W=W0+RU*(W1-W0)
12 ASK("NR. OF THE PARTITIONS IN ONE X-CELL =","IGR =","IGR=2")
13 FINOM(X0,V,MI,IGR)
14 FINTER
15 BT="Y = "TYPE(W0)" - "TYPE(W1)" & W = "TYPE(W)
16 BT; IF(IGRA .EQ. 0) RETURN
17 FIGEY
18 END
```

PROGRAM FINTER

```
 1 PROGRAM
 2 $ CALCULATIG INTERPOLATED VALUES AT (V,W) :
 3 DISTANCE(X,V,DV,IV); MV=NOELS(V)
 4 DISTANCE(Y,W,DW,IW); MW=NOELS(W)
 5 FI=NULLSET
 6 FOR K=1,MV
 7 I=IV(K)
 8 FOR L=1,MW
 9 J=IW(L)
```

```
10 B1=((A14(I,J)*DW(L)+A13(I,J))*DW(L)+A12(I,J))*DW(L)+A11(I,J)
11 B2=((A24(I,J)*DW(L)+A23(I,J))*DW(L)+A22(I,J))*DW(L)+A21(I,J)
12 B3=((A34(I,J)*DW(L)+A33(I,J))*DW(L)+A32(I,J))*DW(L)+A31(I,J)
13 B4=((A44(I,J)*DW(L)+A43(I,J))*DW(L)+A42(I,J))*DW(L)+A41(I,J)
14 FI=FI,(((B4*DV(K)+B3)*DV(K)+B2)*DV(K)+B1)
15 NEXT L
16 NEXT K
17 END
```

SUBROUTINE DISTANCE

```
 1 SUBROUTINE DISTANCE(X,V,DV,IV)
 2 $ NEAREST GRIDPOINT X(IV) TO V , DV = V-X(IV)
 3 IF(CLASS(X) .EQ. 5)        GOTO L0
 4 " ERROR :"; WHATIS X; RETURN
 5 L0: MX=NOELS(X); XX=X
 6 IF(CLASS(V) .EQ. 0)        V=ARRAY(V)
 7 L1: IF(CLASS(V) .EQ. 1 .OR.
 8 &     CLASS(V) .EQ. 5)     GOTO L2
 9 " ERROR :"; WHATIS V; RETURN
10 L2: FOR J=1,NOELS(V)
11 VX=V(J)
12 IF(VX .GE. X(MX))          XX(MX)=VX+1
13 L=LOCS(VX .LT. XX:FIRST)-1
14 WHERE(L .LT. 1)  L=1
15 DV(J)=VX-X(L)
16 IV(J)=L
17 NEXT J
18 END
```

SUBROUTINE FINOM

```
 1 SUBROUTINE FINOM(Y,YU,MU,GR)
 2 $ YU = PARTITION ( Y ) ; MU = INDEX( Y IN YU )
 3 $ ( ) => 1 ( ) ... GR ( )
 4 IF(CLASS(Y) .EQ. 5)  GOTO L0
 5 Y; "ERROR : Y MUST BE AN 1D ARRAY"; RETURN
 6 L0: IF(GR .GT. 1) GOTO L1
 7 YU=Y; MU=INDEXER(YU); RETURN
 9 L1: NY=NOELS(Y); GL=GR-1
10 IF(NY .GT. 1) GOTO L2
11 NY; "ERROR : THERE MUST BE BE AT LEAST 2 POINTS"; RETURN
12 L2: R=ELIMELS(Y,NY); S=ELIMELS(Y,1)
13 DR=(S-R)/GR; YU=Y
14 FOR I=1,GL
15 YU=YU,(R+I*DR)
16 NEXT I
17 YU=RANKED(YU)
18 IF(Y(1) .GT. Y(NY))  YU=REFLECT(YU)
19 MU=INDEXER(R)
20 MU=1,GR*MU+1
21 END
```

PROGRAM FIGEX

```
 1 PROGRAM
 2 $ GRAPHICAL WINDOW FOR THE U(Y)-FUNCTIONS
 3 SETTITLE(BT); SETYLABEL(" U ")
 4 SETXLABEL(" Y ")
 5 YSM1 = MIN(U(IU),U(IU + 1)); YSM2 = MAX(U(IU),U(IU + 1))
 6 SETYSCALE(AUTOMATIC INCLUDE: YSM1 THROUGH: YSM2 )
 7 LINECODE = -2
 8 GRAPH(U(IU):Y0)
 9 IF(RU .GT. 0)  LINECODE = -5
10 IF(RU .GT. 0)  ADDCURVE(U(IU + 1):Y0)
11 LINECODE = 1
12 ADDCURVE(FI:W)
13 CURSOR; FREE MI YSM1 YSM2; SETYSCALE(AUTOMATIC)
14 END
```

PROGRAM FIGEY

```
 1 PROGRAM
 2 $ GRAPHICAL WINDOW FOR THE U(X)-FUNCTIONS
 3 SETTITLE(BT); SETYLABEL(" U ")
 4 SETXLABEL(" X ")
 5 YSM1 = MIN(U(,IU),U(,IU + 1)); YSM2 = MAX(U(,IU),U(,IU + 1))
 6 SETYSCALE(AUTOMATIC INCLUDE: YSM1 THROUGH: YSM2 )
 7 LINECODE = -2
 8 GRAPH(U(,IU):X0)
 9 IF(RU .GT. 0)  LINECODE = -5
10 IF(RU .GT. 0)  ADDCURVE(U(,IU + 1):X0)
11 LINECODE = 1
12 ADDCURVE(FI:V)
13 CURSOR; FREE YSM1 YSM2; SETYSCALE(AUTOMATIC)
14 END
```

TESTING

PROGRAM TESX

```
 1 SUBROUTINE TESX(X,U,P,J)
 2 REQUEST I; REQUEST J
 3 $ RIGHT SIDE
 4 XP = X(I + 1)-X(I)
 5 XM = X(I)-X(I-1)
 6 DUP = U(I + 1,J)-U(I,J)
 7 DUM = U(I,J)-U(I-1,J)
 8 RS1 = (XM/XP)*DUP
 9 RS2 = (XP/XM)*DUM
10 RS = 3*(RS1 + RS2)
11 " RIGHT SIDE  =  "; TYPE(RS)
12 $ LEFT SIDE
13 LS1 = XM*P(I + 1,J)
14 LS3 = XP*P(I-1,J)
15 LS2 = (X(I + 1)-X(I-1))*P(I,J)
16 LS = LS1 + 2*LS2 + LS3
17 " LEFT SIDE  =  "; TYPE(LS)
```

```
18 " DIFFERENCE = "; TYPE(LS-RS)
19 END
```

PROGRAM TESY

```
 1 SUBROUTINE TESY(Y,U,Q,I)
 2 REQUEST I; REQUEST J
 3 $ RIGHT SIDE
 4 YP = Y(J + 1)-Y(J)
 5 YM = Y(J)-Y(J-1)
 6 DUP = U(I,J + 1)-U(I,J)
 7 DUM = U(I,J)-U(I,J-1)
 8 RS1 = (YM/YP)*DUP
 9 RS2 = (YP/YM)*DUM
10 RS = 3*(RS1 + RS2)
11 " RIGHT SIDE = "; TYPE(RS)
12 $ LEFT SIDE
13 LS1 = YM*Q(I,J + 1)
14 LS3 = YP*Q(I,J-1)
15 LS2 = (Y(J + 1)-Y(J-1))*Q(I,J)
16 LS = LS1 + 2*LS2 + LS3
17 " LEFT SIDE = "; TYPE(LS)
18 " DIFFERENCE = "; TYPE(LS-RS)
19 END
```

Appendix B. An example of the array LAP

LAP , rows = 1 — 7							
	0.00	0.02	0.04	0.07	0.10	0.15	0.22
0.00	0.30	0.38	0.45	0.52	0.59	0.66	0.74
	0.82	0.90	0.95	0.98	0.99	1.00	
	1.13939200	1.14453840	1.14337520	1.11976420	1.08523460	1.02199780	0.94077050
370.	0.87739300	0.85444148	0.86074543	0.88092019	0.90474383	0.92264243	0.94269215
	0.96208716	0.97877703	0.98807533	0.99348199	0.99555696	1.00000000	
	1.15199500	1.15534200	1.15032230	1.11296780	1.06445420	0.98844373	0.90711709
450.	0.85534127	0.84517983	0.85848585	0.88223876	0.90604695	0.92309414	0.94369788
	0.96273704	0.97904157	0.98811988	0.99341784	0.99548460	1.00000000	
	1.15574200	1.15972460	1.15335320	1.10753810	1.05146180	0.97013286	0.89082513
500.	0.84706679	0.84382015	0.86045229	0.88541168	0.90839914	0.92475414	0.94524295
	0.96374516	0.97955702	0.98836103	0.99351324	0.99554363	1.00000000	
	1.15921070	1.16401490	1.15627310	1.10333610	1.04121560	0.95623902	0.87991067
550.	0.84392370	0.84629641	0.86528552	0.89046581	0.91195417	0.92753108	0.94743745
	0.96511489	0.98020431	0.98861021	0.99355173	0.99553350	1.00000000	
	1.16352000	1.16937720	1.16093940	1.10193490	1.03494540	0.94749922	0.87489571
600.	0.84597130	0.85255110	0.87292922	0.89738099	0.91678111	0.93151771	0.95042250
	0.96701742	0.98117823	0.98907406	0.99373982	0.99564307	1.00000000	
	1.16903110	1.17547240	1.16595940	1.10244740	1.03201830	0.94337850	0.87532121
650.	0.85243426	0.86186575	0.88270130	0.90554977	0.92240677	0.93628534	0.95384349
	0.96914610	0.98221210	0.98950883	0.99385388	0.99567387	1.00000000	

LAP , rows = 8 — 14							
700.	1.17552920	1.18260610	1.17264770	1.10542190	1.03249550	0.94394506	0.88089892
	0.86284159	0.87390014	0.89430108	0.91480475	0.92884931	0.94181262	0.95778621
	0.97164123	0.98348843	0.99011916	0.99410306	0.99581700	1.00000000	
750.	1.18283590	1.18977010	1.17877130	1.10938670	1.03534520	0.94865409	0.89073814
	0.87623431	0.88789187	0.90697392	0.92451206	0.93568926	0.94764739	0.96187938
	0.97418504	0.98473071	0.99065281	0.99425676	0.99586931	1.00000000	
800.	1.19012710	1.19702160	1.18570350	1.11476990	1.04097850	0.95741631	0.90424854
	0.89203652	0.90344167	0.92040261	0.93452139	0.94298730	0.95381027	0.96621963
	0.97692634	0.98613002	0.99132189	0.99453038	0.99602631	1.00000000	
850.	1.19699880	1.20310650	1.19086480	1.12074730	1.04898610	0.96956366	0.92044161
	0.90936819	0.91975172	0.93387693	0.94425635	0.95039957	0.95989962	0.97045745
	0.97955528	0.98741007	0.99187030	0.99468846	0.99608096	1.00000000	
900.	1.20278530	1.20910640	1.19771470	1.12934450	1.06021350	0.98490511	0.93879500
	0.92788557	0.93650599	0.94722087	0.95370480	0.95803591	0.96601887	0.97473985
	0.98225478	0.98878329	0.99252450	0.99495551	0.99623510	1.00000000	
950.	1.20987770	1.21625690	1.20519480	1.13983990	1.07396610	1.00239580	0.95826706
	0.94683921	0.95291364	0.95977444	0.96242685	0.96546152	0.97179448	0.97872904
	0.98471991	0.98997190	0.99302340	0.99509065	0.99627738	1.00000000	
1000.	1.21920620	1.22579370	1.21572640	1.15292280	1.08974170	1.02120530	0.97816721
	0.96579663	0.96861256	0.97137375	0.97055381	0.97264405	0.97733648	0.98258003
	0.98714218	0.99119647	0.99360176	0.99532526	0.99641440	1.00000000	

LAP , rows = 15 — 21							
1100.	1.24146750	1.24728070	1.23787860	1.18007180	1.12193280	1.05802550	1.01611130
	1.00065740	0.99536544	0.98988431	0.98451205	0.98499228	0.98680419	0.98907735
	0.99116915	0.99315174	0.99444376	0.99559225	0.99653827	1.00000000	
1200.	1.26675820	1.27266940	1.26392690	1.20965280	1.15473510	1.09357670	1.05157340
	1.02975160	1.01494740	1.00263350	0.99645863	0.99479710	0.99430155	0.99418068
	0.99430254	0.99462789	0.99503631	0.99574945	0.99660273	1.00000000	
1300.	1.29558880	1.29970980	1.29050390	1.23850110	1.18617400	1.12674730	1.08078950
	1.04977320	1.02590110	1.01157430	1.00551770	1.00199420	0.99973174	0.99784208
	0.99650852	0.99560397	0.99536441	0.99578932	0.99660354	1.00000000	
1350.	1.30826760	1.31156760	1.30159260	1.25131370	1.20007040	1.14010400	1.09075460
	1.05495070	1.02813110	1.01495070	1.00854530	1.00441820	1.00150630	0.99899276
	0.99714451	0.99580259	0.99533821	0.99568703	0.99652117	1.00000000	
1380.	1.31553750	1.31833340	1.30810580	1.25811500	1.20696900	1.14621870	1.09467760
	1.05624650	1.02872350	1.01651780	1.00989700	1.00551650	1.00230980	0.99951524
	0.99743391	0.99589271	0.99532745	0.99564978	0.99649531	1.00000000	
1400.	1.32037690	1.32146610	1.31185840	1.26169760	1.21045390	1.14913690	1.09622830
	1.05627800	1.02887190	1.01732360	1.01061260	1.00611480	1.00276250	0.99982704
	0.99762804	0.99598495	0.99536837	0.99567823	0.99651890	1.00000000	
1430.	1.32612240	1.32693330	1.31571450	1.26513160	1.21375010	1.15159110	1.09679160
	1.05502950	1.02874030	1.01802100	1.01126370	1.00664970	1.00314250	1.00005490
	0.99772721	0.99597201	0.99530269	0.99561661	0.99647896	1.00000000	

LAP , rows = 22 — 28							
1450.	1.32820800	1.32799710	1.31606640	1.26541180	1.21408800	1.15157500	1.09580230
	1.05335750	1.02839730	1.01811840	1.01141720	1.00678070	1.00322420	1.00008570
	0.99771256	0.99592478	0.99523872	0.99556716	0.99644833	1.00000000	
1470.	1.32872340	1.32742090	1.31474620	1.26408450	1.21292080	1.15022970	1.09373860
	1.05113200	1.02785330	1.01796410	1.01138270	1.00676530	1.00319860	1.00004540
	0.99765678	0.99584506	0.99515678	0.99550912	0.99641258	1.00000000	
1500.	1.32579030	1.32171550	1.30868530	1.25844000	1.20786970	1.14541900	1.08846870
	1.04692020	1.02656350	1.01723470	1.01095100	1.00643770	1.00293130	0.99982664
	0.99747040	0.99568318	0.99501726	0.99542115	0.99636106	1.00000000	
1530.	1.31808520	1.31336170	1.29807020	1.24860910	1.19907740	1.13744180	1.08088050
	1.04213580	1.02466520	1.01594120	1.01004840	1.00571370	1.00233700	0.99934569
	0.99707327	0.99535645	0.99474041	0.99522430	0.99623583	1.00000000	
1550.	1.31098990	1.30489320	1.28943420	1.24006760	1.19121070	1.13042910	1.07482430
	1.03893080	1.02320530	1.01495930	1.00937800	1.00521380	1.00196620	0.99908988
	0.99690560	0.99526192	0.99469085	0.99522203	0.99624263	1.00000000	
1600.	1.27948480	1.27135710	1.25477250	1.20815080	1.16276240	1.10614800	1.05689920
	1.03056310	1.01832230	1.01151690	1.00681710	1.00322710	1.00042010	0.99794131
	0.99606444	0.99468285	0.99427386	0.99498014	0.99609978	1.00000000	
1650.	1.23244520	1.22211990	1.20458110	1.16300500	1.12326680	1.07529640	1.03894450
	1.02184990	1.01248920	1.00729940	1.00355110	1.00066190	0.99840338	0.99642330
	0.99492847	0.99388446	0.99368569	0.99461660	0.99588176	1.00000000	

LAP , rows = 29 — 35							
1700.	1.16954510	1.15883150	1.14275840	1.10843260	1.07811080	1.04515040	1.02442600
	1.01334680	1.00676480	1.00307020	1.00028380	0.99812679	0.99644783	0.99499697
	0.99391113	0.99323123	0.99325857	0.99441275	0.99576962	1.00000000	
1750.	1.09915070	1.08989500	1.07764530	1.05647260	1.03973200	1.02398410	1.01315920
	1.00587120	1.00162240	0.99905830	0.99709295	0.99557079	0.99440009	0.99341433
	0.99268739	0.99233893	0.99259122	0.99400696	0.99552747	1.00000000	
1800.	1.04334170	1.03794530	1.03221910	1.02403690	1.01795250	1.01093870	1.00438250
	1.00002730	0.99744241	0.99576472	0.99447316	0.99348313	0.99274340	0.99215183
	0.99173121	0.99167434	0.99213650	0.99379107	0.99540993	1.00000000	
1850.	1.01427750	1.01150950	1.00893620	1.00556390	1.00275590	0.99922593	0.99624596
	0.99448765	0.99322801	0.99238417	0.99173586	0.99125875	0.99093555	0.99071502
	0.99060371	0.99083694	0.99153074	0.99346393	0.99522349	1.00000000	
1900.	0.99777327	0.99652328	0.99541875	0.99409275	0.99311811	0.99211382	0.99147583
	0.99088566	0.99035570	0.98998767	0.98971813	0.98954941	0.98947767	0.98947075
	0.98956048	0.98998935	0.99088965	0.99310890	0.99501878	1.00000000	
1950.	0.99030181	0.99003317	0.98983445	0.98966735	0.98960827	0.98958893	0.98933594
	0.98896284	0.98864811	0.98844811	0.98832692	0.98828602	0.98831725	0.98837942
	0.98857817	0.98912294	0.99021891	0.99273769	0.99480396	1.00000000	
2000.	0.98865809	0.98866667	0.98878733	0.98862481	0.98844335	0.98813174	0.98776058
	0.98743424	0.98721168	0.98709987	0.98706294	0.98709483	0.98717127	0.98727225
	0.98755915	0.98820666	0.98952968	0.99237484	0.99460227	1.00000000	

LAP , rows = 36 — 42							
2100.	0.98394366	0.98388965	0.98383991	0.98377321	0.98371602	0.98364152	0.98357997
	0.98356929	0.98362035	0.98371221	0.98383964	0.98397029	0.98409181	0.98434905
	0.98482530	0.98578096	0.98777548	0.99147078	0.99409644	1.00000000	
2200.	0.97828379	0.97829912	0.97831638	0.97834626	0.97838091	0.97844901	0.97856480
	0.97872183	0.97889452	0.97902019	0.97912313	0.97927116	0.97952906	0.97997986
	0.98065577	0.98225941	0.98539743	0.99029865	0.99345363	1.00000000	
2300.	0.96937403	0.96913979	0.96910292	0.96930607	0.96942074	0.96954775	0.96964186
	0.96973155	0.96988892	0.97011819	0.97043593	0.97083104	0.97130436	0.97202872
	0.97334395	0.97666210	0.98191383	0.98866636	0.99257484	1.00000000	
2350.	0.96073438	0.96075421	0.96077772	0.96082221	0.96087863	0.96100070	0.96123109
	0.96157280	0.96198255	0.96238463	0.96282699	0.96333545	0.96398862	0.96515689
	0.96741646	0.97251512	0.97949481	0.98758775	0.99201170	1.00000000	
2400.	0.94639408	0.94643071	0.94650252	0.94664062	0.94677740	0.94701101	0.94735002
	0.94776522	0.94823807	0.94874288	0.94939893	0.95031627	0.95167452	0.95416510
	0.95849958	0.96667956	0.97622575	0.98614920	0.99125384	1.00000000	
2430.	0.93052629	0.93059769	0.93069232	0.93083806	0.93099274	0.93127458	0.93173567
	0.93239779	0.93327366	0.93429756	0.93566825	0.93753475	0.94012015	0.94441435
	0.95105325	0.96211777	0.97380312	0.98516255	0.99076652	1.00000000	
2450.	0.91272227	0.91289892	0.91307931	0.91336809	0.91368041	0.91426038	0.91522107
	0.91658783	0.91833159	0.92026700	0.92270802	0.92582067	0.92984572	0.93604540
	0.94489339	0.95849284	0.97193961	0.98441881	0.99040389	1.00000000	

LAP , rows = 43 — 49							
2470.	0.88664412	0.88703914	0.88749457	0.88820459	0.88895725	0.89031524	0.89246251
	0.89533354	0.89875107	0.90229917	0.90649951	0.91152559	0.91762499	0.92639942
	0.93804046	0.95463852	0.97004232	0.98369560	0.99005908	1.00000000	
2490.	0.85403506	0.85462036	0.85559715	0.85713387	0.85872587	0.86151493	0.86573634
	0.87107948	0.87707769	0.88297199	0.88959841	0.89712404	0.90578822	0.91755834
	0.93222792	0.95180207	0.96892874	0.98345233	0.99002688	1.00000000	
2500.	0.84078611	0.84210621	0.84343112	0.84546492	0.84755707	0.85118215	0.85657497
	0.86325437	0.87057988	0.87762133	0.88537335	0.89399417	0.90370813	0.91658928
	0.93221668	0.95246539	0.96971412	0.98404520	0.99045497	1.00000000	
2502.	0.84017258	0.84154950	0.84292762	0.84504247	0.84721477	0.85097104	0.85654188
	0.86341515	0.87092196	0.87810916	0.88599183	0.89472421	0.90452503	0.91746428
	0.93308487	0.95321202	0.97027318	0.98440136	0.99069760	1.00000000	
2504.	0.84135738	0.84272810	0.84414851	0.84631950	0.84854624	0.85238985	0.85807462
	0.86506352	0.87266642	0.87991761	0.88784096	0.89658510	0.90636141	0.91921474
	0.93465919	0.95445696	0.97114311	0.98491151	0.99103114	1.00000000	
2506.	0.84680346	0.84783954	0.84924806	0.85136870	0.85354964	0.85731965	0.86290615
	0.86978744	0.87728315	0.88443439	0.89224185	0.90083852	0.91041328	0.92292224
	0.93782070	0.95675960	0.97270741	0.98583633	0.99161005	1.00000000	
2507.	0.85371996	0.85446912	0.85583688	0.85811263	0.86040003	0.86430864	0.86999694
	0.87685431	0.88416695	0.89102154	0.89840309	0.90644832	0.91535924	0.92700918
	0.94101671	0.95911015	0.97436841	0.98674482	0.99217072	1.00000000	