12.4 C^0 and C^1 Surface Interpolation to Scattered Data

A. Purpose

Given a set of triples $(x_i,\ y_i,\ z_i)$, $i = 1, \ldots,$ NP, these routines implicitly construct a conveniently computable function $f(x, y)$ satisfying the conditions

$$z_i = f(x_i, y_i), \quad i = 1, \ldots, NP,$$

and support evaluation of $f(x, y)$ and its first partial derivatives for user-provided values of (x, y). The data $(x_i,\ y_i)$ are not assumed to lie in any special pattern, but it is assumed that $(x_i,\ y_i) = (x_j,\ y_j)$ only if $i = j$.

B. Usage

To apply this method the user must first call STGGRD to build a Delaunay triangular grid having the given $(x_i,\ y_i)$ points as nodal points. For convenience from here on we shall let $\mathcal{S}$ denote the given set of $(x_i,\ y_i)$ points. The user must specify that f is to have either C^0 or C^1 continuity. For C^1 continuity the interpolation method needs values of first partial derivatives of the desired surface at the nodal points. Most commonly the user will not have these values, in which case the subroutine STGPD can be called to compute estimated partial derivatives.

Once the grid and partial derivatives (for the C^1 method) have been established, one can evaluate the interpolation function f at any (x, y) point by calling STGFI. This evaluation operation can be repeated for any number of points. Subroutine STGFI uses STGFND to determine which triangle of the grid contains the point (x, y), and STGC0 or STGC1 to evaluate $f(x, y)$ in that triangle. If the point is outside the triangular grid STGFI calls STGEXT to compute extrapolated values.

We do not describe the separate usage of STGFND, STGEXT, STGC0, and STGC1, but a user could learn this from the comments in these subroutines and the code in STGFI, if desired for specialized applications.

In some applications one wishes to produce a table of values of the interpolated surface at points of a regular rectangular grid. The subroutine STGREC can be called to produce such a table. STGREC makes a sequence of calls to STGFI. The user must call STGGRD and optionally STGPD before calling STGREC.

The subroutine STGPGR is provided to print the data structures that define the triangular grid. This routine may also be useful as a model if one wishes to interface to a graphics system to draw the triangular grid.

The following sections address usage for the following:

B.1. STGGRD	Construct a Delaunay triangular grid
B.2. STGPD	Estimate partial derivatives at grid nodes
B.3. STGFI	Lookup and interpolate in a triangular grid
B.4. STGREC	Construct a rectangular grid of function values by interpolation in a triangular grid
B.5. STGPGR	Print the data structures defining the triangular grid

B.1 Construct a Delaunay Triangular Grid

B.1.a Program Prototype, Single Precision

INTEGER NP, IP(≥NP), MT, TRIANG(MT), MB, B(4, MB), NT, INFO(3)

REAL X(≥NP), Y(≥NP), W(≥NP)

Assign values to NP, MT, MB, X(1:NP), and Y(1:NP).

CALL STGGRD (X, Y, NP, IP, W, TRIANG, MT, B, MB, NT, INFO)

Results are returned in TRIANG(), B(,), NT, and INFO(1:3)

B.1.b Argument Definitions

X(1:NP), Y(1:NP) [in] The $(x,\ y)$ coordinates of the NP data points constituting the set $\mathcal{S}$.

NP [in] Number of data points. Require NP ≥ 3. The upper limit on NP will depend on the storage available for the array TRIANG().

IP() [scratch] Working space.

W() [scratch] Working space.

TRIANG() [out] Defines the triangular grid. Each triangle is defined by six integer indices identifying up to three neighboring triangles and three vertex points. The number of triangles will be NT = $2 \times (\text{NP} - 1) - \text{NB}$ where NB is the number of points on the boundary of the convex hull of $\mathcal{S}$.

MT [in] The declared dimension of TRIANG. Must be $\geq 6 \times [2 \times (\text{NP} - 1) - \text{NB}]$. If NB is not known, $12 \times (\text{NP} - 1)$ provides an upper bound for the required value of MT.

B(,) [out] On return this contains a two–way linked list identifying the boundary points of the grid.

MB [in] The declared second dimension of B(,). Must be larger by one than the largest number of boundary points in any of the subsets of $\mathcal{S}$ that STGGRD considers. Setting MB = NP + 1 would always be

large enough, however, a much smaller value for MB will usually suffice.
The expected number of boundary points in $\mathcal{S}$ when $\mathcal{S}$ is a random set of points uniformly distributed in a disk is $3.4 \times NP^{1/3}$ for NP > 100.
We suggest setting MB = NP + 1 for NP ≤ 15, and MB = an integer exceeding $6 \times NP^{1/3}$ for NP ≥ 16.

NT [out] The number of triangles in the triangular grid. NT, NB, and NP are related by the equation NT = 2 × (NP − 1) − NB.

INFO(1:3) [out] Termination information.

INFO(1) Status on termination. The grid is complete if INFO(1) = 0 and incomplete or not optimal otherwise.

=0 Normal termination, the triangular grid is complete and optimized.

=1 A complete triangular grid has been constructed, but the process of optimizing the shapes of the triangular cells did not complete. This should not happen.

=2 Either all given points are colinear or else some pair of the points are duplicates.

=3 Some pair of the points are duplicates.

=4 The dimension MB is not large enough. Setting MB = NP + 1 will suffice.

=5 The dimension MT is not large enough. Setting MT = 12 × (NP − 1) will suffice.

INFO(2) = NB, the number of boundary points in the convex hull of $\mathcal{S}$.

INFO(3) = The minimum value that would have sufficed for the dimension MB on this execution of STGGRD.

B.2 Estimate Partial Derivatives at Grid Nodes

B.2.a Program Prototype, Single Precision

INTEGER NP, TRIANG(≥ 6×NP), NT, IWORK(≥NP)

REAL X(≥NP), Y(≥NP), Z(≥NP), DZ(2, ≥NP)

If one does not have partial derivative values available, this subroutine can be used to compute estimates of the partial derivatives after using subroutine STGGRD to construct a triangular grid. The values of X(), Y(), and NP should be the same as were input to STGGRD. The values of TRIANG() and NT should be those produced by STGGRD. Values must be assigned to Z(1:NP).

CALL STGPD (X, Y, Z, DZ, NP, TRIANG, NT, IWORK)

Results are returned in DZ(1:2, 1:NP).

B.2.b Argument Definitions

X(1:NP), Y(1:NP), Z(1:NP) [in] The user supplied data; Z(i) is the function value associated with the point (X(i), Y(i)).

DZ(1:2, 1:NP) [out] Estimates of the first partial derivatives, stored as:
DZ(1,i) = $\partial f/\partial x$ at (X(i), Y(i))
DZ(2,i) = $\partial f/\partial y$ at (X(i), Y(i))

NP [in] Number of data points.

TRIANG(), NT [in] Defined as for STGGRD.

IWORK(1:NP) [scratch] Working space.

B.3 Lookup and Interpolate in a Triangular Grid

B.3.a Program Prototype, Single Precision

INTEGER TRIANG(≥ 6×NT), NT, NCONT, MODE, MB, B(4, MB)

REAL X(≥NP), Y(≥NP), Z(≥NP), DZ(2,≥NP), Q(2), ZOUT, DZOUT(2), SAVWRK(28)

LOGICAL WANTDZ

This subroutine is intended to be used after a triangular grid has been constructed using subroutine STGGRD. The values of X(), Y(), and MB should be the same as were input to STGGRD. The values of TRIANG(), NT, and B(), should be those produced by STGGRD. The value of NP (used only in the text here) should be the same as was used in the call to STGGRD. Values must be assigned to Z(1:NP), NCONT, Q(1:2), WANTDZ, and SAVWRK(1). If C^1 interpolation is being requested, (NCONT = 1), values must be assigned to DZ(1:2, 1:NP), possibly by use of subroutine STGPD.

CALL STGFI (X, Y, Z, DZ, TRIANG, NT, B, MB, NCONT, Q, ZOUT, WANTDZ, DZOUT, MODE, SAVWRK)

Results are returned in ZOUT, DZOUT(1:2), and MODE, and the contents of SAVWRK(1:28) will generally be changed.

B.3.b Argument Definitions

X(1:NP), Y(1:NP) [in] The $(x,\ y)$ coordinates of the data points constituting the set $\mathcal{S}$, and thus the nodes of the triangular grid.

Z(1:NP) [in] Z(i) is the value at (X(i), Y(i)) of the data to be interpolated.

DZ(1:2, 1:NP) [in] Values, or estimates of values, of first partial derivatives at the nodal points. These are needed only when NCONT = 1. Otherwise this

array will not be referenced.
DZ(1,i) = $\partial f/\partial x$ at (X(i), Y(i))
DZ(2,i) = $\partial f/\partial y$ at (X(i), Y(i))

TRIANG(), NT [in] Defined as for STGGRD.

B(), MB [in] Defined as for STGGRD.

NCONT [in] Set NCONT = 0 for C^0 interpolation, and = 1 for C^1 interpolation.

Q(1:2) [in] Lookup and interpolation will be done for the point $q = (x, y)$ = (Q(1), Q(2)).

ZOUT [out] Interpolated value computed by this subroutine.

WANTDZ [in] Turns on or off the computation of DZOUT(1:2).

DZOUT(1:2) [out] If WANTDZ = .true., interpolated values of the partial derivatives at the point q will be computed and returned in DZOUT(1:2). No partial derivative interpolation is done if WANTDZ = .false.

MODE [out] Indicates status on return.

- =0 Point q is within the convex hull of the set $\mathcal{S}$, or possibly outside this triangle by a small tolerance. Interpolated results are returned.
- =1 Point q is outside the convex hull of $\mathcal{S}$ by more than the built-in small tolerance. Extrapolated results are returned. Accuracy degrades rapidly as the evaluation points move away from the convex hull.
- =2 Error condition. STGFND has tested the point q against NT triangles without resolving its status. This would indicate an error in the definition of the triangular grid or in the design or coding of the subroutine. No results are returned.

SAVWRK(1:28) [inout] Array used as work space and to save values that, in some circumstances, can be reused to reduce the amount of computation on subsequent calls. The user must set SAVWRK(1) = 0.0 on the first call to this subroutine to signal that SAVWRK() does not contain any saved information at that time. Generally SAVWRK(1) will be nonzero on returns from STGFI. If subsequent calls do not involve changes to X, Y, Z, DZ, TRIANG, NT, B, MB, or NCONT, the user should not alter SAVWRK() on the subsequent calls. If the user changes any of these quantities, the user should reset SAVWRK(1) = 0.0 to indicate that saved information should be ignored. See Section C for further discussion of SAVWRK().

B.4 Construct a Rectangular Grid of Function Values by Interpolation in a Triangular Grid

B.4.a Program Prototype, Single Precision

LOGICAL WANTPD

INTEGER NP, TRIANG(≥ 6×NT), NT, NX, NY, MX, MY, NCONT, MB, B(4, MB)

REAL X(≥NP), Y(≥NP), Z(≥NP), DZ(2, ≥NP), XYLIM(4), ZFILL, ZVALS(MX, MY), DZVALS(MX, MY, 2)

Assign values to all of the subroutine arguments except ZVALS() and DZVALS(). STGGRD should be used to set values for TRIANG(), NT and B(). Subroutine STGPD can be used to set values for DZ(,).

CALL STGREC (X, Y, Z, DZ, NP, TRIANG, NT, B, NB, XYLIM, NX, NY, ZFILL, ZVALS, MX, MY, NCONT, WANTPD, DZVALS)

Results are returned in ZVALS(,) and DZVALS(,,1:2)

B.4.b Argument Definitions

X(1:NP), Y(1:NP), Z(1:NP), DZ(1:2, 1:NP) [in] Values of x, y, z, $\partial z/\partial x$, and $\partial z/\partial y$ at the NP points of the triangular grid. Values are needed in DZ() only if NCONT = 1. See subroutine STGPD for computation of DZ().

NP [in] Number of data points.

TRIANG(), NT [in] Defined as for STGGRD.

B(), MB [in] Defined as for STGGRD.

XYLIM(), NX, NY [in] Specifies the desired rectangular grid. The x grid values will run from XYLIM(1) to XYLIM(2) with NX − 2 equally spaced intermediate values. The y grid values will run from XYLIM(3) to XYLIM(4) with NY − 2 equally spaced intermediate values.

ZFILL [in] Determines the action to be taken when a point of the rectangular grid is outside the convex region covered by the triangular grid. If ZFILL is zero, extrapolated values will be returned. Otherwise the value, ZFILL, will be returned as a function value in ZVALS() and optionally as a partial derivative value in DZVALS().

ZVALS(,) [out] An MX by MY array in which STGREC stores the NX by NY set of values of the function $z = f(x, y)$ defined by interpolation in the triangular grid. Thus for $i = 1, \ldots,$ NX, $j = 1, \ldots,$ NY, STGREC computes
x_i = XYLIM(1) + $(i - 1)\times$

[XYLIM(2) − XYLIM(1)]/(NX − 1)
y_j = XYLIM(3) + (j − 1)×
[XYLIM(4) − XYLIM(3)]/(NY − 1)
IF ((x_i, y_j) is in the region covered by the triangular grid) THEN
ZVAL$(i, j) = f(x_i, y_j)$ (using interpolation)
ELSE
IF (ZFILL .eq. 0.0E0) THEN
ZVAL$(i, j) = f(x_i, y_j)$ (using extrapolation).
ELSE
ZVAL(i, j) = ZFILL
END IF
END IF

MX, MY [in] Dimensions for the array ZVALS(,) and first two dimensions for DZVALS(,,1:2). Require MX ≥ NX and MY ≥ NY.

NCONT [in] Set NCONT = 0 for C^0 interpolation, and = 1 for C^1 interpolation.

WANTPD [in] Set to .TRUE. if computation of DZVALS(,,1:2) is desired in addition to ZVALS(,) and to .FALSE. if only ZVALS(,) is desired.

DZVALS(,,1:2) [out] An MX by MY by 2 array in which the NX by NY by 2 set of values of $\partial z/\partial x$ and $\partial z/\partial y$ will be stored if WANTPD = .TRUE. As in the computation of ZVALS(,), the value ZFILL determines the action when an (x_i, y_j) point is outside the triangular grid.
If WANTPD = .FALSE. the array DZVALS() will not be referenced and the user can reduce all dimensions of DZVALS() to 1.

B.5 Print the Data Structures defining a Triangular Grid

B.5.a Program Prototype, Single Precision

INTEGER NP, TRIANG(≥ 6×NT), B(4, MB), NB, NT

REAL X(≥NP), Y(≥NP)

CALL STGPRG (X, Y, NP, TRIANG, B, NB, NT)

B.5.b Argument Definitions

All arguments are defined as for STGGRD, and should have the values they had on a return from subroutine STGGRD (NB = INFO(2)).

B.6 Modifications for Double Precision

Change the first letter of every subroutine name from "S" to "D", and change all REAL type statements to DOUBLE PRECISION.

C. Examples and Remarks

Example. The program DRSTGFI1 contains a set of data: $(x_i,\ y_i), i = 1, \ldots, 28$. The program calls STGGRD to generate a triangular grid having these points as nodal points. The program runs two cases of C^1 interpolation to data over this grid. The interpolation is done at a set of points along a line cutting across the region covered by the grid.

For the first case, data are generated at the nodal points using a quadratic function. Mathematically, the partial derivative estimation method in STGPD and the C^1 interpolation method in STGC1 are exact when the given z_i data are a quadratic function of the (x_i, y_i) data. Thus we observe that the errors in the interpolated values are of the order of magnitude of the precision of the machine arithmetic, which was IEEE single precision arithmetic for the output listed with this writeup.

For the second case, data are generated using an exponential function. The partial derivative estimation method in STGPD and the C^1 interpolation method in STGC1 will both introduce errors in processing this data, with the result that the interpolated values will deviate from the exponential function at points away from the nodal points.

The first and last points at which interpolation is requested in each case are outside the convex hull of the given (x_i, y_i) data. The program used extrapolation for these points.

Note that this program can be altered to run ten cases rather than just two by changing the comment status of the loop control lines involving the loop indices "ncont" and "KF". A version of this program called DRSTGFI has this change made.

Choosing between C^0 and C^1 interpolation. Since these are interpolation, and not smoothing, methods, they are most appropriate for data that has very little noise. The C^1 method requires first partial derivative data, whereas the C^0 method does not. If the first partial derivative data have been estimated from the z_i data by using STGPD, they will have errors that are larger as the data deviate more from having a locally quadratic shape. Estimates of first partial derivative values at points (x_i, y_i) on the boundary of the convex hull of $\mathcal{S}$ may be particularly bad. The interpolation surface defined by the C^1 method can easily have unwanted sharp peaks and valleys due to errors in the first partial derivative data, or simply due to the possibility that the shape of the data cannot be well represented by the type of piecewise cubic surface produced by this method.

The C^0 method is more stable than the C^1 method, in the sense that it will not produce maxima higher than

the data or minima lower than the data, however, being piecewise linear it will have a choppy appearance due to corners along all the edges of the triangular grid.

Using multiple copies of SAVWRK().

Let us say the settings of X(), Y(), Z(), DZ(,), NP, TRIANG(), NT, B(,), and MB define a *grid and data*. Applications may arise in which the user wishes to alternate between interpolations using STGFI with different *grid and data* sets. In such cases one can simply reset SAVWRK(1) = 0.0 each time the *grid and data* is changed. For more computational efficiency, however, one could use a distinct SAVWRK() array for each distinct *grid and data*, say SAVW1(), SAVW2(), ... One could set SAVW1(1) = 0, SAVW2(1) = 0, ..., only once initially. Then on each call to STGFI, use the SAVWn() array associated with the current *grid and data*. This permits some saving of computational time in lookups and C^1 interpolation.

D. Functional Description

Delaunay Triangulation. Given a finite set, $\mathcal{S}$, of points in the plane there are generally many ways one could connect pairs of these points to form a triangular grid, i.e., a grid having triangular cells. The Delaunay triangulation, constructed by subroutine STGGRD, has a number of favorable mathematical properties, specifically the max-min angle property, the empty circumcircle property, and duality with a proximity region tessellation.

The max-min angle property has both a local and a global implication. Locally, in any pair of triangles that share a side and whose union is a strictly convex quadrilateral, the size of the smallest of the six interior angles of the two triangles would not be increased by replacing these two triangles by two triangles formed by partitioning the quadrilateral using its other diagonal. Globally, the smallest interior angle occurring in the whole triangulation could not be made larger by any other triangulation.

The empty circumcircle property means that for each triangle of the grid, its circumcircle, i.e., the unique circle passing through its three vertices, contains no points of the set $\mathcal{S}$ in its interior.

The Delaunay triangulation of a set $\mathcal{S}$ is unique, with the exception that if any set of four or more points of $\mathcal{S}$ lie on a circle that contains no points of $\mathcal{S}$ in its interior, it is arbitrary how triangles are formed using these vertices.

A Delaunay triangulation is dual to a proximity tessellation associated with the names Dirichlet, Voronoi, and Thiessen. A Dirichlet tessellation associated with a set $\mathcal{S}$ is a partitioning of the whole plane into regions with the property that the interior of region i contains all the points of the plane that are closer to point (x_i, y_i) of $\mathcal{S}$ than to any other point of $\mathcal{S}$. These regions will be convex, polygonal, and some will be of infinite extent. Clearly a one-to-one correspondence exists between regions of a Dirichlet tessellation and nodes of a Delaunay triangulation. It also happens that a correspondence exists between triangles of a Delaunay triangulation and nodes of a Dirichlet tessellation. Furthermore edges in a Dirichlet tessellation are perpendicular bisectors of edges in a Delaunay triangulation.

Algorithms. The triangulation algorithm implemented in STGGRD is described in [1]. From both theoretical analysis and timing tests it appears that the execution time for STGGRD is proportional to $NP^{4/3}$.

The method of estimating partial derivatives implemented in STGPD is described in [1]. To estimate the partial derivatives at a point (x_i, y_i) of $\mathcal{S}$, this method locates up to 16 points of $\mathcal{S}$ that are near to (x_i, y_i) in the sense of the grid connectivity, and does a uniformly weighted least squares fit of a 6-parameter quadratic polynomial to the grid data at these points. A modification of this method is used if NP < 6. The first partial derivatives of this polynomial at (x_i, y_i) are returned as the estimated partial derivatives.

The lookup method used in STGFND is described in [1]. This method starts by testing the given point q for containment in the triangle whose index is INDTRI (or 1 if INDTRI is out of range.) If q is outside of this triangle relative to side j, the search moves to the triangle adjacent to the current triangle across side j. This process is repeated until either a triangle is found that contains q, or q is found to be outside some triangle relative to an edge that is a boundary edge of the convex hull of $\mathcal{S}$.

The tests for containment produce values that are also needed in both the C^0 and C^1 interpolation methods. To avoid recomputation, these quantities are stored in an array by STGFND for subsequent use in STGC0 or STGC1.

This method is quite efficient in general, and is particularly efficient when a sequence of lookups is done at points that are near to each other relative to the grid connectivity, and with each lookup being started in the triangle at which the previous lookup terminated.

C^0 continuity implies continuity of values but allows the possibility of discontinuity of first partial derivatives, whereas C^1 continuity implies continuity of values and first partial derivatives, but allows the possibility of discontinuity of second partial derivatives.

The C^0 interpolation method implemented in STGC0 defines a planar surface over each triangular cell of the

grid. Each such planar surface matches the values at the three vertices of its triangle. This gives a piecewise linear surface over the whole grid. The surface is continuous over the whole grid, but generally has jumps in its first derivative values when crossing from one triangle to another.

For C^1 interpolation we use the Clough-Tocher method, [2], used in finite element computations. The particular implementation in STGC1 is described in [3]. Additional properties of the induced surface are described in [4].

For one triangular cell of a grid this method implicitly partitions the triangle into three subtriangles, by inserting new edges from the vertices to the centroid. Over each of these subtriangles the method implicitly defines a cubic polynomial having C^1 continuity across the newly inserted internal edges. Along each of the original edges of a triangle the values and first partial derivatives depend only on the values and first partial derivatives given at the two vertices of that edge. This assures that two triangles that share an edge will have the same values and first partial derivatives along the common edge, and thus the surface over the whole triangular grid will have C^1 continuity.

The C^0 method is exact for linear data both in interpolation and extrapolation. The C^1 method is exact for quadratic data in interpolation and for linear data in extrapolation.

References

1. C. L. Lawson, *Software for C^1 surface interpolation*, in John R. Rice, editor, **Mathematical Software III**, 161–194, Academic Press, New York (1977). Also JPL Publication 77–30, August 1977.

2. R. W. Clough and J. L. Tocher. *Finite element stiffness matrices for analysis of plates in bending.* in **Proc. Conf. Matrix Methods in Struct. Mech.**, Wright–Patterson AFB, Ohio, (1965). Air Force Inst. of Tech.

3. C. L. Lawson, **C^1–Compatible interpolation over a triangle**. JPL TM 33–770, Jet Propulsion Laboratory, Pasadena, CA (May 1976) 34 pages.

4. C. L. Lawson, **Integrals of a C^1–Compatible Triangular Surface Element**. JPL TM 33–808, Jet Propulsion Laboratory, Pasadena, CA (Dec. 1976) 15 pages.

E. Error Procedures and Restrictions

STGGRD returns status information in INFO(1). STGFI returns status information in MODE. The user should encode tests of these status flags. IERV1 is used for the output of some error messages.

F. Supporting Information

The source language is ANSI Fortran 77. A common block STGCOM is used in the program units in the file STGPD. A common block DTGCOM is used in the program units in the file DTGPD.

Entry	Required Files
STGGRD	SSORTP, STGGRD, STGSET
STGPD	SROT, SROTG, STGPD, STGSET
STGFI	ERFIN, ERMSG, IERM1, IERV1, STGC0, STGC1, STGEXT, STGFI, STGFND, STGSET
STGREC	ERFIN, ERMSG, IERM1, IERV1, STGC0, STGC1, STGEXT, STGFI, STGFND, STGREC, STGSET
STGPRG	STGPRG, STGSET
DTGGRD	DSORTP, DTGGRD, DTGSET
DTGPD	DROT, DROTG, DTGPD, DTGSET
DTGFI	DTGC0, DTGC1, DTGEXT, DTGFI, DTGFND, DTGSET, ERFIN, ERMSG, IERM1, IERV1
DTGREC	DTGC0, DTGC1, DTGEXT, DTGFI, DTGFND, DTGREC, DTGSET, ERFIN, ERMSG, IERM1, IERV1
DTGPRG	DTGPRG, DTGSET

Algorithm developed and subroutine designed and written by Charles L. Lawson, JPL, 1976. Adapted to Fortran 77 and the MATH77 library, 1991 and January, 1996.

DRSTGFI1

```
c   File: DRSTGFI1.[F|FOR] CONTAINS DRSTGFI1 AND STGFCN.
      program DRSTGFI1
c>> 2001-07-16 DRSTGFI1 Krogh Added exponent 0 to some constants.
c>> 2001-05-22 DRSTGFI1 Krogh Minor change for making .f90 version.
c>> 1997-07-01 DRSTGFI1 Krogh Reversed subscripts in B (CLL suggestion)
c>> 1997-06-19 DRSTGFI1 Krogh Minor changes to get proper C conversion.
c>> 1997-06-18 DRSTGFI1 CLL
c>> 1996-03-04 DRSTGFI1 CLL
c>> 1996-02-02 DRSTGFI1 CLL
c>> 1995-10-31 DRSTGFI1 CLL
c   Demo driver for STGFI, STGGRD, STGPD, etc.
c     ------------------------------------------------------------------
c--S replaces "?": DR?TGFI1, ?TGFI, ?TGGRD, ?TGPD, ?TGFCN
c     ------------------------------------------------------------------
      integer mb, mp, meval, mt
      parameter(mb= 16, mp= 28, mt= 336)
      parameter(meval = 5)
      integer Bdry(4,mb), i, ieval, INFO(3), IP(mp), kf
      integer mode, ncont, np, nt, TRIANG(mt)
      real              DZ(2,mp), dzout(2)
      real              dztemp(2), DZTRUE(2)
      real              q(2)
      real              savwrk(28)
      real              W(mp), X(mp)
      real              Y(mp)
      real              Z(mp), zout, ztrue
      logical  WANTDZ
      character title*75
      data X /
     *-0.76059E0,-0.02286E0,-0.44790E0, 0.15068E0,-0.87287E0,-0.23390E0,
     * 0.06093E0,-0.84142E0,-0.69173E0,-0.56613E0,-0.42243E0,-0.17249E0,
     *-0.06484E0, 0.48286E0,-0.88784E0, 0.10277E0, 0.69087E0,-0.84292E0,
     * 0.08784E0,-0.95068E0, 0.02496E0, 0.94973E0, 0.04588E0,-0.51667E0,
     *-0.77561E0, 0.90000E0,-0.70830E0,-0.40500E0 /
      data Y /
     *-0.31421E0, 0.75657E0, 0.14321E0, 0.42353E0,-0.62983E0, 0.12326E0,
     * 0.34054E0,-0.55144E0, 0.34158E0,-0.67143E0, 0.43087E0, 0.96081E0,
     *-0.68130E0, 0.12095E0,-0.10663E0, 0.29219E0, 0.31028E0, 0.12934E0,
     * 0.10709E0,-0.42307E0, 0.49895E0, 0.68597E0,-0.78215E0,-0.12362E0,
     *-0.88827E0,-0.60000E0,-0.88620E0, 0.08600E0 /
      data np / mp /
      data wantdz / .true. /
c     ------------------------------------------------------------------
      write(*,'(a/a)') ' Program DRSTGFI1.  Demo driver for STGFI,',
     *                 ',                   STGGRD, STGPD, etc.'
      call STGGRD(X,Y,NP,IP,W,TRIANG,MT,Bdry,MB,NT, INFO)
      if (INFO(1) .NE. 0) then
         write(*,'(a,i5)')
     *      ' ERROR return FROM STGGRD. INFO(1) =', INFO(1)
         stop
      endif
c
c     do 200 ncont = 0,1
      do 200 ncont = 1,1
c     do 100 KF= 0,4
      do 100 KF= 2,4,2
```

```
      write(*,'(//a,i1,a)') ' New Case.  Interpolation with C',
     *    ncont,' continuity.'
      do 20 I=1,NP
         call STGFCN(KF,X(I),Y(I),title,Z(I),DZtemp(1),DZtemp(2))
   20 continue
      write(*,'(/a)') title
      call STGPD(X,Y,Z,DZ,NP, TRIANG, NT, IP)
      savwrk(1) = 0.0e0
      write(*,'(/3x,a//a,a/16x,a/16x,a/)')
     *    ' VALUES AND PARTIAL DERIVS INTERPOLATED ALONG A PATH.',
     *    '     I     X      Y',
     *    '     Z_INTERP      Z_TRUE      Z_ERR',
     *    '  DZ1_INTERP    DZ1_TRUE    DZ1_ERR',
     *    '  DZ2_INTERP    DZ2_TRUE    DZ2_ERR'
      do 30 ieval = 0, meval-1
         q(1) = -1.0e0 + ieval*2.0e0/(meval-1)
         q(2) =  0.8e0 * q(1)
         call STGFI( X, Y, Z, DZ, TRIANG, NT, Bdry, Mb,
     *               ncont, Q, zout, WANTDZ, DZout, MODE, SAVWRK)
         if(mode .ge. 0) then
            call STGFCN(KF,  q(1), q(2),
     *          title, ztrue, dztrue(1), dztrue(2))
            write(*, '(1X,I4,2F6.2,2F11.6,E10.2)') ieval,q,
     *          zout, ztrue, zout - ztrue
            write(*,'(17x,2F11.6,E10.2)') dzout(1), dztrue(1),
     *          dzout(1) - dztrue(1)
            write(*,'(17x,2F11.6,E10.2)') dzout(2), dztrue(2),
     *          dzout(2) - dztrue(2)
         else
            write(*,'(1x,i4,2f6.2,a)')
     *         ieval, q, '   Error.'
         endif
   30 continue
  100 continue
  200 continue
      end
c     ==================================================================
      subroutine STGFCN  (KF,X,Y, TITLE, Z,ZX,ZY)
c>> 1995-09-26 DRSTGFI1 CLL Editing for inclusion into MATH77.
c     C.L.Lawson, JPL, 1976 Dec 10.  Edited comments 1979 Mar 3.
c     This subr evaluates a function and its first partial derivs as
c     selected by KF.  KF can be from 0 to 4.
c     Input is KF, X, and Y.  Output is TITLE, Z, ZX, and ZY.
c     ------------------------------------------------------------------
      integer KF
      real              X,Y,Z,ZX,ZY
      character title*75
      character title1(0:4)*56, title2(0:4)*19
      integer I
      data (title1(i),title2(i),i=0,4) /
     * '  CONSTANT FUNCTION      Z = 2', ' ',
     * '  LINEAR FUNCTION        Z = ( 1 + 2*X + 3*Y ) / 6', ' ',
     * '  QUADRATIC FUNCTION     Z = ( -1 + 2*X - 3*Y + 4*X**2 - ',
     *                              'X*Y + 9*Y**2 ) / 10',
     * '  CUBIC FUNCTION     Z = ( 9*X**3 - 2*(X**2)*Y + 3*X*Y**2',
     *                            ' - 4 * Y**3 ) / 10',
     * '  EXPONENTIAL FUNCTION    Z = EXP( -2 * (X**2 + Y**2) )', ' '/
c     ------------------------------------------------------------------
      title = title1(kf) // title2(kf)
```

```
      if(kf .eq. 0) then
c                                        KF=0  CONSTANT FCN.
        Z=2.0e0
        ZX=0.0e0
        ZY=0.0e0
      elseif(kf .eq. 1) then
c                                        KF=1  LINEAR FCN.
        Z=(1.0e0 + 2.0e0*X + 3.0e0*Y) / 6.0e0
        ZX=2.0e0/6.0e0
        ZY=3.0e0/6.0e0
      elseif(kf .eq. 2) then
c                                        KF=2  QUADRATIC FCN.
        Z=(-1.0e0 + 2.0e0*X - 3.0e0*Y +4.0e0*X**2 -X*Y + 9.0e0*Y**2)*
     *       0.1e0
        ZX =(2.0e0 + 8.0e0*X -Y)* 0.1e0
        ZY =(-3.0e0 -X + 18.0e0*Y)* 0.1e0
      elseif(kf .eq. 3) then
c                                        KF=3  CUBIC FCN.
        Z=(9.0e0*X**3 - 2.0e0*(X**2)*Y + 3.0e0*X*Y**2 - 4.0e0*Y**3)*
     *       0.1e0
        ZX=(27.0e0*X**2 - 4.0e0*X*Y + 3.0e0*Y**2)* 0.1e0
        ZY=(-2.0e0*X**2 + 6.0e0*X*Y - 12.0e0*Y**2)* 0.1e0
      elseif(kf .eq. 4) then
c                                        KF=4  EXPONENTIAL FCN.
        Z=  EXP(-(X**2 + Y**2) * 2.0e0)
c
c            NOTE THAT THE INFLECTION POINT OF THIS FCN IN ANY
c            RADIAL DIRECTION FROM THE ORIGIN IS AT R = .5
c
        ZX= -4.0e0*X*Z
        ZY= -4.0e0*Y*Z
      endif
      return
      end
```

ODSTGFI1

```
Program DRSTGFI1.  Demo driver for STGFI,
                   STGGRD, STGPD, etc.


New Case.  Interpolation with C1 continuity.

 QUADRATIC FUNCTION     Z = ( -1 + 2*X - 3*Y + 4*X**2 - X*Y + 9*Y**2 ) / 10

   VALUES AND PARTIAL DERIVS INTERPOLATED ALONG A PATH.

   I     X       Y      Z_INTERP      Z_TRUE      Z_ERR
                      DZ1_INTERP    DZ1_TRUE    DZ1_ERR
                      DZ2_INTERP    DZ2_TRUE    DZ2_ERR

   0  -1.00   -0.80     0.821840     0.836000  -0.14E-01
                       -0.367478    -0.520000   0.15E+00
                       -1.600898    -1.640000   0.39E-01
   1  -0.50   -0.40     0.144000     0.144000  -0.15E-07
                       -0.160000    -0.160000  -0.15E-07
                       -0.970000    -0.970000   0.60E-07
   2   0.00    0.00    -0.100000    -0.100000  -0.75E-08
                        0.200000     0.200000  -0.25E-06
                       -0.300000    -0.300000  -0.15E-06
   3   0.50    0.40     0.104000     0.104000   0.45E-07
                        0.560000     0.560000   0.60E-07
                        0.370000     0.370000  -0.51E-06
   4   1.00    0.80     0.743860     0.756000  -0.12E-01
                        0.891188     0.920000  -0.29E-01
                        0.839773     1.040000  -0.20E+00


New Case.  Interpolation with C1 continuity.

 EXPONENTIAL FUNCTION     Z = EXP( -2 * (X**2 + Y**2) )

   VALUES AND PARTIAL DERIVS INTERPOLATED ALONG A PATH.

   I     X       Y      Z_INTERP      Z_TRUE      Z_ERR
                      DZ1_INTERP    DZ1_TRUE    DZ1_ERR
                      DZ2_INTERP    DZ2_TRUE    DZ2_ERR

   0  -1.00   -0.80     0.021016     0.037628  -0.17E-01
                        0.343730     0.150513   0.19E+00
                        0.134924     0.120410   0.15E-01
   1  -0.50   -0.40     0.421890     0.440432  -0.19E-01
                        0.903598     0.880863   0.23E-01
                        0.580212     0.704691  -0.12E+00
   2   0.00    0.00     1.033885     1.000000   0.34E-01
                        0.258727    -0.000000   0.26E+00
                       -0.414570    -0.000000  -0.41E+00
   3   0.50    0.40     0.415133     0.440432  -0.25E-01
                       -1.002802    -0.880863  -0.12E+00
                       -0.500310    -0.704691   0.20E+00
   4   1.00    0.80     0.036043     0.037628  -0.16E-02
                       -0.535420    -0.150513  -0.38E+00
                       -0.011262    -0.120410   0.11E+00
```