

# The Numerical Differentiation of Discrete Functions using Polynomial Interpolation Methods

James M. Hyman  
Bernard Larrouturou

Published in the  
Journal of APPLIED MATHEMATICS AND COMPUTATION  
1982; v.10, p.487-506

Published in:  
Numerical Grid Generation for Numerical  
Solution of Partial Differential Equations,  
(Elsevier North-Holland, New York)  
Joe F. Thompson, Editor, (1982) pp 487-506

# THE NUMERICAL DIFFERENTIATION OF DISCRETE FUNCTIONS USING POLYNOMIAL INTERPOLATION METHODS

JAMES M. HYMAN<sup>†</sup> AND BERNARD LARROUTUROU<sup>††</sup>

<sup>†</sup>Center for Nonlinear Studies, Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA; <sup>††</sup>Laboratoire Central des Ponts et Chaussées, 58 Bd Lefebvre, 75732 Paris, Cedex 15, France

## ABSTRACT

A FORTRAN subroutine package, called DERMOD, has been written for calculating numerical approximations for the spatial derivatives of a function defined only on a discrete set of data points. The routines are designed to complement many existing codes for solving ordinary and partial differential equations and for the interpolation of tabular data. We describe some new numerical differentiation algorithms, discuss a mapping procedure for nonuniform grids, and explain the program methodology used in designing the software.

## I. INTRODUCTION

The accurate approximation of spatial derivatives is a crucial element in the numerical solution of partial differential equations (PDEs). The derivation and implementation of accurate differentiation formulas is an error-prone and tedious process. This is especially true in two and three dimensions on nonuniform grids. For these reasons we have written a subroutine package, called DERMOD, to help reduce the effort needed to accurately and reliably differentiate discretely defined functions.

A major advantage of the package, compared with the traditional approach for solving PDEs, is that state-of-the-art numerical methods can be used with a minimum of programming effort to approximate the derivatives in large complicated PDE systems. Furthermore, the resulting programs can easily be modified for a comparison of the relative accuracy and efficiency of the different methods for solving a particular problem. The production runs can then be made using the best available methods.

The numerical analyst will also benefit from this approach. Most of the developmental analysis for numerical methods is for simple linear systems. It is important to understand the behavior of a new method in a complex situation before recommending its use to the uninitiated. New methods can now be quickly tested on any PDE system discretized using the DERMOD package.

All the spatial differentiation methods we describe will follow the same algorithmic flow. A function is defined on a discrete set of mesh points in one, two, or three space dimensions. These discrete data sets are the input to a black box type subroutine in the DERMOD library. In this subroutine the approximation for the described derivative is calculated and returned to the user.

Thus, the spatial differentiation is totally divorced from the nonlinearities of the PDE, the boundary conditions, and the time integration method. This modularity also reduces the redundancy of programming the same approximation to the spatial derivatives each time they appear in an equation. These differentiation routines are designed for no specific PDE and need to be debugged and optimized for a particular machine only once.

The data structures of the grids allowed in the current version of DERMOD, listed in increasing order of complexity and computer cost, include:

- one-argument grids: (tensor-product grids, Fig. 1a)  $\{x_i\}$ ,  $\{x_i, y_j\}$ , and  $\{x_i, y_j, z_k\}$ . Uniform grids are a special case of one-argument grids,
- multiple-argument grids: (logically rectangular, Fig. 1b)  $\{x_i\}$ ,  $\{x_{i,j}, y_{i,j}\}$ , and  $\{x_{i,j,k}, y_{i,j,k}, z_{i,j,k}\}$ ,
- neighborhood grids: (Fig. 1c)  $\{x_\ell\}$ ,  $\{x_\ell, y_\ell\}$ ,  $\{x_\ell, y_\ell, z_\ell\}$ , and  $NBRS_\ell$ . These grids are typical of finite element simplex grids.

The numerical differentiation methods described can be divided into two classes: interpolation methods and mapping methods.

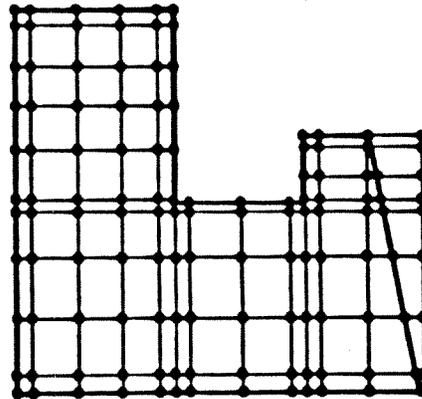
The interpolation method approximates the function with an interpolant (such as splines, or a local Lagrange polynomial), differentiates the interpolant, and evaluates the derivative at the desired location. These formulas are simple on uniform grids but are usually complicated on nonuniform grids. The interpolation method is not as sensitive to rough mesh variations as the mapping method.

In the mapping method, the nonuniform grid is mapped to a uniform reference grid. The derivatives on the nonuniform grid can then be expressed as products and sums of the derivatives of the function on the reference grid and the map. These derivatives on the reference grid are approximated using an interpolation method that is simple and efficient. The accuracy of the mapping method depends upon the smoothness of both the original function and the map. Therefore, the smoothness of the mesh variations in the nonuniform grid can strongly influence the accuracy of the derivative approximations.

One-argument grids

$$(x_i, y_j, z_k)$$

2-D Tensor-product grid

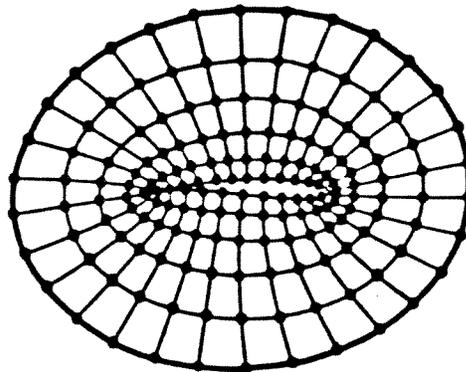


Multiple-argument grids

$$(x_{i,j}, y_{i,j})$$

$$(x_{i,j,k}, y_{i,j,k}, z_{i,j,k})$$

2-D logically rectangular grid



Neighborhood grids

$$(x, y, z)_L$$

$$XLOC(L)$$

$$YLOC(L)$$

$$ZLOC(L)$$

$$NBR(L, *)$$

2-D triangular grid

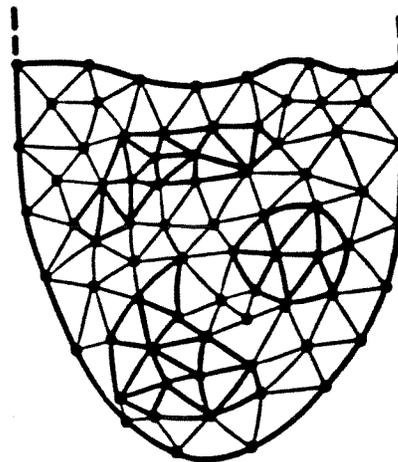


Fig. 1. Different 2-D grid data structures.

After first describing the interpolation method in greater detail we will discuss the mapping method, explain the program methodology used in designing the software, and then provide instructions for using the DERMOD subroutines effectively.

## II. INTERPOLATION METHODS

In the interpolation method one can either construct a local interpolation function based only on data points near where the derivative is desired or, by using all the data points, construct a global interpolant. Usually the local interpolants are simpler and more appropriate for handling sharp gradients and nonuniform meshes, but they are less accurate than the global interpolants. In this report we will describe only the local polynomial interpolation methods. In a later report we plan to describe on the implementation of global interpolants based on the Fourier transform or Chebyshev polynomials.

The simplest local interpolation method to approximate derivatives fits a Lagrange or least-squares polynomial through the data at the nearby mesh points, differentiates the polynomial and evaluates it at the desired mesh point.<sup>1-8</sup> This results in an approximation for the k-th derivative of f at  $x_i$ , denoted by  $f^{[k]}(x_i)$ , that is a linear combination of the nearby function values,

$$f^{[k]}(x_i) = \sum a_j f(x_j) / (\text{constant} \times h^k).$$

On uniform grids these formulas are relatively simple and are called finite difference methods. Table 1 lists some common finite difference formulas used in DERMOD. The centered difference formulas are used whenever possible and the uncentered ones are used only near boundaries when too few function values are available to use a centered scheme.

When the data is smooth, the high order Lagrange interpolation formulas in Table 1a usually provide better accuracy on a given grid than do the lower order formulas.

When the data is rough, the sensitivity of the derivative approximations to noise can be more important than the order of accuracy of the method. For these problems the least-squares formulas<sup>9</sup> in Table 1b are preferred over the Lagrange formulas. These formulas are derived by fitting a least-squares polynomial through the data points of degree two less than the Lagrange polynomial.

TABLE 1a  
LAGRANGE APPROXIMATIONS ON EQUALLY-SPACED GRIDS,  $ch^k f^{[k]} = \sum a_j f(x_i)$

Derivative	$a_{i-3}$	$a_{i-2}$	$a_{i-1}$	$a_i$	$a_{i+1}$	$a_{i+2}$	$a_{i+3}$	$a_{i+4}$	$a_{i+5}$	$a_{i+6}$	Accuracy
$2hf_x(x_i)$			-1		1						$O(h^2)$
$2hf_x(x_i)$				-3	4	-1					$O(h^2)$
$12hf_x(x_i)$		1	-8		8	-1					$O(h^4)$
$12hf_x(x_i)$			-3	-10	18	-6	1				$O(h^4)$
$12hf_x(x_i)$				-25	48	-36	16	-3			$O(h^4)$
$60hf_x(x_i)$	-1	9	-45		45	-9	1				$O(h^6)$
$60hf_x(x_i)$		2	-24	-35	80	-30	8	-1			$O(h^6)$
$60hf_x(x_i)$			-10	-77	150	-100	50	-15	2		$O(h^6)$
$60hf_x(x_i)$				-147	360	-450	400	-225	72	-10	$O(h^6)$
$hf_x(x_{i+\frac{1}{2}})$				-1	1						$O(h^2)$
$24hf_x(x_{i+\frac{1}{2}})$			1	-27	27	-1					$O(h^4)$
$24hf_x(x_{i+\frac{1}{2}})$				-23	21	3	-1				$O(h^4)$
$1920hf_x(x_{i+\frac{1}{2}})$		-9	125	-2250	2250	-125	9				$O(h^6)$
$h^2 f_{xx}(x_i)$			1	-2	1						$O(h^2)$
$h^2 f_{xx}(x_i)$				2	-5	4	-1				$O(h^2)$
$12h^2 f_{xx}(x_i)$		-1	16	-30	16	-1					$O(h^4)$
$12h^2 f_{xx}(x_i)$			10	-15	-4	14	-6	1			$O(h^4)$
$12h^2 f_{xx}(x_i)$				35	-104	114	-56	11			$O(h^3)$
$180h^2 f_{xx}(x_i)$	2	-27	270	-490	270	-27	2				$O(h^6)$
$180h^2 f_{xx}(x_i)$		-13	228	-420	200	15	-12	2			$O(h^5)$
$180h^2 f_{xx}(x_i)$			137	-147	-255	470	-285	93	-13		$O(h^5)$
$180h^2 f_{xx}(x_i)$				812	-3132	5265	-5080	2970	-972	137	$O(h^5)$
$2h^3 f_{xxx}(x_i)$		-1	2		-2	1					$O(h^2)$
$2h^3 f_{xxx}(x_i)$			-3	10	-12	6	-1				$O(h^2)$
$2h^3 f_{xxx}(x_i)$				-5	18	-24	14	-3			$O(h^2)$
$h^4 f_{xxxx}(x_i)$		1	-4	6	-4	1					$O(h^2)$
$h^4 f_{xxxx}(x_i)$			2	-9	16	-14	6	-1			$O(h^2)$
$h^4 f_{xxxx}(x_i)$				3	-14	26	-24	11	-2		$O(h^2)$

TABLE 1b  
 LEAST-SQUARES APPROXIMATIONS ON EQUALLY-SPACED GRIDS,  $ch^k f^{[k]} = \sum a_j f(x_i)$

Approximation	$a_{i-3}$	$a_{i-2}$	$a_{i-1}$	$a_i$	$a_{i+1}$	$a_{i+2}$	$a_{i+3}$	$a_{i+4}$	$a_{i+5}$	$a_{i+6}$	Accuracy
$35f(x_i)$		-3	12	17	12	-3					$O(h^4)$
$35f(x_i)$			9	13	12	6	-5				$O(h^3)$
$35f(x_i)$				31	9	-3	-5	3			$O(h^3)$
-----											
$10hf_x(x_i)$		-2	-1		1	2					$O(h^2)$
$70hf_x(x_i)$			-34	3	20	17	-6				$O(h^2)$
$70hf_x(x_i)$				-54	13	40	27	-26			$O(h^2)$
-----											
$7h^2_{f_{xx}}(x_i)$		2	-1	-2	-1	2					$O(h^2)$
$14h^2_{f_{xx}}(x_i)$			11	-16	-4	12	-3				$O(h^2)$
$7h^2_{f_{xx}}(x_i)$				9	-15	-2	13	-5			$O(h^2)$
-----											
$231f(x_i)$	5	-30	75	131	75	-30	5				$O(h^6)$
$462f(x_i)$		-35	155	212	150	25	-65	20			$O(h^5)$
$462f(x_i)$			25	356	155	-60	-65	70	-19		$O(h^5)$
$462f(x_i)$				456	25	-35	10	20	-19	5	$O(h^5)$
-----											
$252hf_x(x_i)$	22	-67	-58		58	67	-22				$O(h^4)$
$2772hf_x(x_i)$		158	-1619	-50	1218	764	-607	136			$O(h^4)$
$252hf_x(x_i)$			-104	-25	68	84	16	-59	20		$O(h^4)$
$2772hf_x(x_i)$				-4420	5059	1504	-2394	-1378	2375	-746	$O(h^4)$
-----											
$132h^2_{f_{xx}}(x_i)$	-13	67	-19	-70	-19	67	-13				$O(h^4)$
$132h^2_{f_{xx}}(x_i)$		27	3	-35	-34	9	47	-17			$O(h^3)$
$132h^2_{f_{xx}}(x_i)$			103	-145	-39	74	49	-57	15		$O(h^3)$
$132h^2_{f_{xx}}(x_i)$				215	-377	-31	254	101	-245	83	$O(h^3)$

On an equally spaced mesh it is easily seen that the centered first derivative approximations in Table 1 are conservative.<sup>10</sup> On an unequally spaced mesh the interpolation formulas are, in general, not conservative.

To compute more complicated derivatives such as  $f_{xy}$  or  $(df_x)_x$  the formulas are applied in a two-step process that ensures that the resulting formula will be as compact as possible. For example, to compute  $(df_x)_x$ , first  $f_x$  is computed at the half-points  $x_{i+1/2} = \frac{1}{2}(x_i + x_{i+1})$ . Then  $d$  is defined at these points using the harmonic mean,

$$d_{i+1/2} = \left[ \frac{1}{\Delta x_{i+1/2}} \int_{x_i}^{x_{i+1}} d^{-1}(x) dx \right]^{-1} \doteq 2d_i d_{i+1} / (d_i + d_{i+1}) .$$

The harmonic rather than the arithmetic mean is used in order to preserve the flux continuity  $(df_x)$  across discontinuities in  $d$ .<sup>11</sup> The product  $df_x$  is then differentiated and evaluated at the mesh points. On nonuniform grids special care must be taken because the centers of the midpoints are not the mesh points.

The three-point derivative approximations on nonuniform grids using a parabolic interpolant are listed in Table 2. The five-point quintic Lagrange interpolation methods are straightforward<sup>1</sup> and are also available in the

TABLE 2  
QUADRATIC APPROXIMATIONS TO  $f_x$  AND  $f_{xx}$

$$f_x(x_i) \doteq (\Delta x_{i-1/2} S_{i+1/2} + \Delta x_{i+1/2} S_{i-1/2}) / (\Delta x_{i+1/2} + \Delta x_{i-1/2})$$

$$f_x(x_i) \doteq [(2\Delta x_{i+1/2} + \Delta x_{i+3/2}) S_{i+1/2} - \Delta x_{i+1/2} S_{i+3/2}] / (\Delta x_{i+1/2} + \Delta x_{i+3/2})$$

$$f_{xx}(x_i) \doteq 2(S_{i+1/2} - S_{i-1/2}) / (\Delta x_{i+1/2} + \Delta x_{i-1/2})$$

where

$$\Delta x_{i+1/2} = x_{i+1} - x_i$$

$$S_{i+1/2} = \Delta f_{i+1/2} / \Delta x_{i+1/2} .$$

package. The coefficients  $a_j$  for the quintic interpolation methods are computed and saved on the first call to the package. By using this information, later derivative calculations on the same nonuniform grid cost little more than the approximations on an equally spaced grid.

On the multiple-argument or neighborhood grids the local Lagrange interpolant is more cumbersome and frequently there is no unique formulation. For example, in two dimensions, the typical Lagrange quadratic interpolant is uniquely defined with six data points, but the  $(i,j)$ -th mesh point in two-argument grid has nine data points next to it. A possible approach is illustrated in Fig. 2. First, an orthogonal  $(x,y)$  coordinate system is set up and  $f$  is interpolated linearly to give values at the on-axis points A, B, C, and D using the function values at the neighboring points. The one-argument grid quadratic interpolation formulas are then used. This procedure, implemented in DERMOD, is not as accurate as it could be, since the interpolated values are only  $O(h^2)$ . The first derivative approximations are only  $O(h)$  accurate and the second derivative approximations may be only  $O(1)$ , and thus they may be inconsistent.

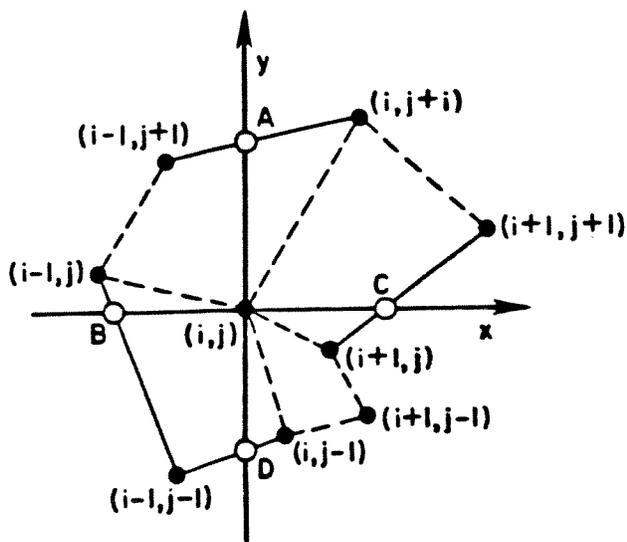


Fig. 2a. Two-argument grid.

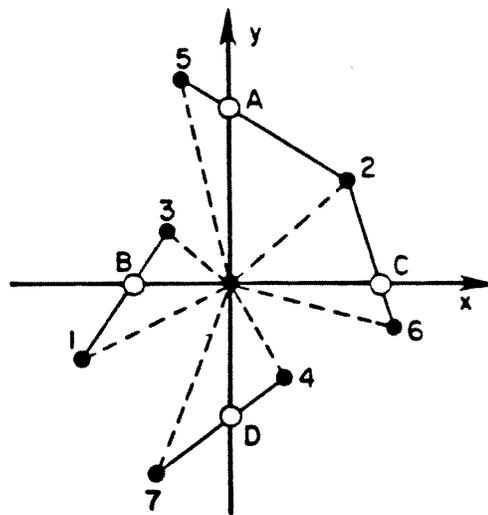


Fig. 2b. Neighborhood grid.

Fig. 2. Interpolation to an underlying orthogonal reference grid.

We have considered two other approaches to overcome this dilemma on multiple-argument and neighborhood grids. The first is to fit a least-squares quadratic polynomial through the nearby data points and differentiate it at the desired location. We expect this method to be accurate and stable. We are currently implementing this approach in DERMOD.

The lumped finite element method is another interpolation method that can also be used to generate local approximations to the derivatives. These formulas work well on uniform grids, but appear to have little advantage over the least squares approach on rough grids. In fact, using a triangulation of the grid in Fig. 2, the approximations thus generated to the second derivatives are pointwise inconsistent. The Minerbo approximation to the Laplacian<sup>12</sup> is an accurate special formula that avoids this inconsistency.

### III. MAPPING METHODS

A simpler approach to numerical differentiation on nonuniform grids is the mapping method. In the mapping method, the physical mesh points  $(x,y,z)$  are mapped to reference mesh points  $(\xi,\eta,\zeta)$ . The derivatives in the physical space are then expressed in terms of the derivatives of the map, called the mesh metrics, and the derivatives of the function on the reference grid.

The mapping method in one space dimension<sup>13</sup> is always nonsingular since the nonuniform mesh  $\{x_i\}$  forms a strictly monotonic sequence. That is, there exists a one-to-one map from  $x_i$  onto the reference grid  $\xi_i$  (for example,  $\xi_i = i$ ). The derivatives of a function defined on  $\{x_i\}$  can then be expressed as

$$f_x = f_\xi \xi_x = f_\xi / x_\xi \quad (4.1)$$

and

$$f_{xx} = f_{\xi\xi} \xi_x^2 + f_\xi \xi_{xx} = f_{\xi\xi} (x_\xi)^{-2} + \frac{1}{2} f_\xi ((x_\xi)^{-2})_\xi \quad (4.2)$$

The derivatives on the right side of these equations are derivatives on the reference mesh. These can be approximated with any of the interpolation methods, all of which have a much simpler formulation on regular reference grids. For example, if fourth-order Lagrange finite differences are used in (4.1), then

$$\begin{aligned} f_x(x_i) &= \left( \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta\xi} \right) \left( \frac{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}{12\Delta\xi} \right)^{-1} \\ &= \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}} \end{aligned}$$

Note that the reference grid points need not be evenly spaced. They could, for example, be the Gauss or Chebyshev points depending upon the interpolating functions being used.

Some smoothness in the function being approximated is assumed in deriving all the high order differentiation formulas. For this reason, the order of accuracy is bounded by the smoothness of  $f$ , the smoothness of the map, and the order of accuracy of the differentiation formula on the uniform mesh. Therefore the mapping methods are usually less accurate than the interpolation methods on grids with nonsmoothly varying mesh spacing.

Analytically,  $\xi_x$  can never vanish. However, on rough grids (where the mapping method is inappropriate) the numerical approximation to  $\xi_x$  may vanish or, equally bad, change sign. When this occurs, either an interpolation method or a lower order mapping method should be used.

On two-argument grids the formulas are more complicated,<sup>14</sup> but the derivatives can still be expressed as a function of the derivatives on a regular reference grid and the mesh metrics of the map from the physical  $(x,y)$  grid to the reference  $(\xi,\eta)$  grid. For this case we have

$$\begin{bmatrix} f_x \\ f_y \\ f_{xx} \\ f_{xy} \\ f_{yy} \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x & 0 & 0 & 0 \\ \xi_y & \eta_y & 0 & 0 & 0 \\ \xi_{xx} & \eta_{xx} & \xi_x^2 & 2\xi_x\eta_x & \eta_x^2 \\ \xi_{xy} & \eta_{xy} & \xi_x\xi_y & \xi_x\eta_y + \xi_y\eta_x & \eta_x\eta_y \\ \xi_{yy} & \eta_{yy} & \xi_y^2 & 2\xi_y\eta_x & \eta_y^2 \end{bmatrix} \begin{bmatrix} f_\xi \\ f_\eta \\ f_{\xi\xi} \\ f_{\xi\eta} \\ f_{\eta\eta} \end{bmatrix} \quad (4.3)$$

Using the Jacobian  $J$  of the map and its derivatives,

$$J = x_\xi y_\eta - x_\eta y_\xi,$$

$$J_\xi = x_{\xi\xi} y_\eta + x_\xi y_{\xi\eta} - x_{\xi\eta} y_\xi - x_\eta y_{\xi\xi},$$

and

$$J_\eta = x_{\xi\eta} y_\eta + x_\xi y_{\eta\eta} - x_{\eta\eta} y_\xi - x_\eta y_{\xi\eta},$$

the mesh metrics can be easily expressed as derivatives on the  $(\xi, \eta)$  reference grid.

$$\xi_x = y_\eta / J \quad ,$$

$$\xi_y = -x_\eta / J \quad ,$$

$$\eta_x = -y_\xi / J \quad ,$$

$$\eta_y = x_\xi / J \quad ,$$

$$\xi_{xx} = (-J_\xi y_\eta^2 + J y_\eta y_{\xi\eta} + J_\eta y_\xi y_\eta - J y_{\xi\eta} y_\eta) / J^3 \quad ,$$

$$\xi_{xy} = (J_\xi x_\eta y_\eta - J x_{\xi\eta} y_\eta - J_\eta x_\eta y_\xi + J x_\eta y_{\xi\eta}) / J^3 \quad ,$$

$$\xi_{yy} = (-J_\xi x_\eta^2 + J x_\eta x_{\xi\eta} + J_\eta x_\xi x_\eta - J x_{\xi\eta} x_\eta) / J^3 \quad ,$$

$$\eta_{xx} = (-J_\eta y_\xi^2 + J y_\xi y_{\xi\eta} + J_\xi y_\xi y_\eta - J y_{\xi\eta} y_\eta) / J^3 \quad ,$$

$$\eta_{xy} = (J_\eta x_\xi y_\xi - J x_{\xi\eta} y_\xi - J_\xi x_\xi y_\eta + J x_{\xi\eta} y_\eta) / J^3 \quad ,$$

and

$$\eta_{yy} = (-J_\eta x_\xi^2 + J x_\xi x_{\xi\eta} + J_\xi x_\xi x_\eta - J x_{\xi\eta} x_\eta) / J^3 \quad .$$

If the derivatives of many different functions must be calculated on the same mesh, then the mesh metrics need only be calculated once and saved. Usually this means after the initial derivative calculation, additional derivatives on the same nonuniform mesh, using the mapping method, cost only slightly more than derivative approximations on a uniform mesh. The package does this automatically.

The mapping method for three-argument grids is similar to two-argument grids but this has not been implemented in DERMOD, yet.

#### IV. SOFTWARE DESIGN

In designing DERMOD we placed a priority on making the code reliable, modular and easy to use. All the programs were extensively documented and verified as the code was developed and performance claims were tested and reconfirmed. During execution, the input is consistently checked for reasonability.

The routines are modular and as independent of each other as possible. Minimal internal communication allows sophisticated users to easily experiment and modify a subroutine for special purposes without causing unexpected errors to ripple through the other routines. Modularity also allows the programs using DERMOD to be easily upgraded and to make use of improved methods and implementations as they become available. We anticipate that by using the package the software development time and maintenance of codes may be significantly reduced, especially for lengthy 3-D programs.

The machine architecture largely determines the efficiency of many of the numerical differentiation methods. We have opted for the routines to differentiate along one line at a time. This requires only one-dimensional work arrays and allows the code to be easily vectorized on machines such as the CRAY-1 and CDC Cyber-205.

The subroutine names in the package have six characters. These are chosen according to the following convention:

first letter:

- A - define all the indicated derivatives
- X - sweep in the x direction (first index sweep)
- Y - sweep in the y direction (second index sweep)
- Z - sweep in the z direction (third index sweep)

second letter:

- 1 - first derivative
- 2 - second derivative
- 3 - third derivative
- 4 - fourth derivative
- X,Y,Z or L - see special cases listed below

third letter:

- X - derivative in the x coordinate direction
- Y - derivative in the y coordinate direction
- Z - derivative in the z coordinate direction
- C,D,R,S,Y or Z - see special cases listed below

fourth letter:

P - polynomial approximation (finite differences)  
F - Fourier transform (pseudo-spectral method)  
C - Chebychev transform (pseudo-spectral method)  
R - rational function Padé approximation (implicit method)

fifth letter:

E - equally spaced grid  
I - interpolation method (unequally spaced grid)  
M - mapping method (unequally spaced grid)  
G - Gauss points  
C - Chebyshev points

sixth letter:

1 - one-argument (tensor product) grid X(I), Y(I), Z(K)  
2 - two-argument grid X(I,J), Y(I,J)  
3 - three-argument grid X(I,J,K), Y(I,J,K), Z(I,J,K)  
T - triangular neighborhood grid X(L), Y(L) (two dimensions)  
P - pyramid neighborhood grid X(L), Y(L), Z(L) (three dimensions)  
H - one-argument staggered grid (derivatives at the half points)

special cases for the second and third letter:

XY - mixed xy derivative  
XZ - mixed xz derivative  
YZ - mixed yz derivative

LR - Laplacian in rectangular geometry ( $u_{xx} + u_{yy} + u_{zz}$ )  
LC - Laplacian in cylindrical geometry ( $x^{-1}(xu_x)_x + x^{-2}u_{yy} + u_{zz}$ )  
LS - Laplacian in spherical geometry

XD - compute  $(df_x)_x$   
YD - compute  $(df_y)_y$   
ZD - compute  $(df_z)_z$

For example, subroutine X2YPM2 computes the second derivative of  $f$  with respect to  $y$ ,  $f_{yy}$ , at the mesh points along an X coordinate line using a local polynomial interpolant or finite difference method, listed in Table 1, after mapping the unequally spaced two-argument grid to a uniform grid. The mesh metrics are computed using the same order finite difference methods.

At this time the available routines are:

X1XPE1, X2XPE1, X3XPE1, X4XPE1, X1YPE1, X2YPE1, X3YPE1, X4YPE1, X1ZPE1, X2ZPE1, X3ZPE1, X4ZPE1, X1XPI1, X2XPI1, X1YPI1, X2YPI1, X1ZPI1, X2ZPI1, X1XPM1, X2XPM1, X1YPM1, X2YPM1, X1ZPM1, X2ZPM1, XXYPE1, XXZPE1, XXYPI1, XXZPI1, XXYPM1, XXZPM1, X1XPEH, X1YPEH, X1ZPEH, X1XPMH, X1YPMH, X1ZPMH, XXDPE1, XYDPE1, XZDPE1, XXDPI1, XYDPI1, XZDPI1, X1XPI2, X1YPI2, X1XPM2, X2XPM2, X1YPM2, X2YPM2, and XXYPM2.

The nomenclature used by the package will be useful in describing the capabilities of the routines. These variables (used in the above X sweep routines) and their meanings are:

Input Variables:

U - array of the function values to be differentiated

in one space dimension the function  $u(x)$  must be defined at  $U(I)$  where  $I$  is between  $NXBX$  and  $NXEX$

in two space dimensions the function  $u(x,y)$  must be defined at  $U(I,J)$  where

$I$  is between  $NXBX$  and  $NXEX$

$J$  is between  $NYBX$  and  $NYEX$

in three space dimensions the function  $u(x,y,z)$  must be defined at  $U(I,J,K)$  where

$I$  is between  $NXBX$  and  $NXEX$

$J$  is between  $NYBX$  and  $NYEX$

$K$  is between  $NZBX$  and  $NZEX$

On neighborhood grids the function  $u$  must be defined at  $U(L)$  where  $L$  is between  $NLBX$  and  $NLEX$ .

X - the array containing the mesh point locations in the first coordinate diction. The element  $X(I)$  in one-argument grids,  $X(I,J)$  or  $X(I,J,K)$  on multiple-argument grids, or  $X(L)$  on neighborhood grids must be defined for the same indices  $I$ ,  $J$ ,  $K$ , or  $L$  as those where  $U$  is defined.

Y - the array containing the mesh point locations in the first coordinate diction. The element  $Y(J)$  in one-argument grids,  $Y(I,J)$  or  $Y(I,J,K)$  on multiple-argument grids or  $Y(L)$  on neighborhood grids must be defined for the same indices  $I$ ,  $J$ ,  $K$ , or  $L$  as those where  $U$  is defined.

Z - the array containing the mesh point locations in the first coordinate diction. The element  $Z(K)$  in one-argument grids or  $Z(I,J,K)$  on multiple-argument grids or  $Z(L)$  on neighborhood grids must be defined for the same indices  $I$ ,  $J$ ,  $K$ , or  $L$  as those where  $U$  is defined.

D - array of the diffusion coefficients for the second derivatives. This array must be defined where  $U$  is defined and has the same data structure as  $U$ .

$NXBX$  - index of the first X point where  $U$  is defined.

$NXB$  - index of the first X mesh point where the derivatives of  $U$  are to be calculated.

$NXE$  - index of the last X mesh point where the derivatives of  $U$  are to be calculated.

NXEX - index of the last X point where U is defined.  
 NXD - dimension of the first index of U and the mesh arrays.  
 NYBX - index of the first Y point where U is defined.  
 NY - index of the Y mesh point where the derivatives of U are to be calculated.  
 NYEX - index of the last Y point where U is defined.  
 NYD - dimension of the second index of U and the mesh arrays.  
 NZBX - index of the first Z point where U is defined.  
 NZ - index of the Z mesh point where the derivatives of U are to be calculated.  
 NZEX - index of the last Z point where U is defined.  
 MORD - method order parameter. The method should be asymptotically MORD-th order

Workspace Variables:

IWS - index to indicate whether the work space array contains information on the grid such as the mesh metrics (IWS = 0 on first call using the grid, IWS = 1 on later calls).  
 WS - array of workspace used for internal calculations. This array may be input or output.

Output Variables:

U\*\* - array of the derivatives defined at U\*\*(I) for I between NXB and NXE. The second and third letters are the same as these in the subroutine-naming convention.  
 MORD - The derivative returned is asymptotically MORD-th order. This will be less than or equal to the requested value. MORD returns equal to zero if no calculation was possible.

The variables for the routines that sweep in the Y and Z lines are similarly named.

A sample program to compute the derivative of  $\sin(x)$  for  $x$  between zero and one, using a sixth-order finite difference (polynomial interpolation) method is:

```

DIMENSION X(11),U(11),U1X(11)
NXBX=1
NXEX=11
DX=1.0/(NXEX-NXBX)
DO 10 I=NXBX,NXEX
X(I)=(I-1)*DX
10 U(I)=SIN(X(I))
  
```

```

MORD=6
NXB=NXBX
NXE=NXEX
CALL X1XPE1(U,X,NXBX,NXB,NXE,NXEX,MORD,U1X)
PRINT 20
20 FORMAT("      X      U      U1X      ANS")
DO 30 I=NXB,NXE
ANS=COS(X(I))
30 PRINT 40,X(I),U(I),U1X(I),ANS
40 FORMAT(4F10.6)
CALL EXIT
END

```

The output is:

<u>X</u>	<u>U</u>	<u>U1X</u>	<u>ANS</u>
0.000000	0.000000	.999980	1.000000
.100000	.099833	.995009	.995004
.200000	.198669	.980067	.980067
.300000	.295520	.955336	.955336
.400000	.389418	.921061	.921061
.500000	.479426	.877583	.877583
.600000	.564642	.825336	.825336
.700000	.644218	.764842	.764842
.800000	.717356	.696707	.696707
.900000	.783327	.621613	.621610
1.000000	.841471	.540289	.540302

Note that in this example the derivative approximations near the boundaries where the uncentered difference formulas are used are less accurate than where centered differences can be used. These errors could be avoided by defining U on a domain greater than that of the desired derivatives such as:  $NXBX < NSB - 2$  and  $NXEX > XNE + 2$ . This is also convenient when using fictitious points to incorporate the effects of the boundary conditions into a discrete approximation<sup>15</sup> and when constructing a local Hermite interpolant and sampling the interior of a table.

The workspace needed by the code is limited to one-dimensional arrays of length NXE. These arrays contain the finite-difference coefficients for a particular mesh line, and can be used to define the coefficient matrix needed in solving the linear systems arising from implicit methods. When the user's program is constrained by computer CPU time and not storage, then by saving workspace arrays of length NXE on one-argument grids or NXE·NYE on two-argument grids, the difference coefficients need only be computed once for the entire problem.

## V. USAGE

We expect the package to be used most frequently for calculating derivatives directly for explicit approximations to differential equations<sup>15</sup> and for defect correction improvements of low-order implicit approximations.<sup>16</sup> When used this way a crude estimate of the error can be obtained by comparing the derivatives with those obtained by a different method or on a coarser grid. The structure of the package makes this easy to do.

The direct usage is straightforward; the function to be differentiated is defined at the mesh points and the derivatives are calculated as described in the example in the previous section. We expect this to be the most common usage for explicit integration methods for PDEs and for constructing interpolants.

The indirect defect correction usage occurs most often in the iterative solution of algebraic equations arising in the numerical approximation to differential equations. These equations occur in steady state or time independent problems and on each time step in the implicit integration of time dependent problems.

These systems can be written

$$A(v) - b = 0 \quad , \quad (5.1)$$

where  $A$  is a nonlinear discrete operator,  $b$  is a known vector, and the discrete solution vector is  $v$ . The sparseness of  $A$  depends upon the numerical differentiation method used. The high-order methods result in less sparse, more complicated systems than the lower-order methods.

Often the solution of Eq. (5.1) is difficult to obtain directly, but the residual error,

$$r = A(w) - b \quad (5.2)$$

for an approximate solution  $w$ , is easy to evaluate. In many complicated PDE problems, one is less likely to introduce errors in evaluating  $r$  than in constructing  $A$  and solving Eq. (5.1). This is particularly true for high-order approximations of nonlinear systems on irregular domains.

If there is a related system

$$P(w) - b = 0 \quad (5.3)$$

that approximates Eq. (5.1) and is easier to solve, the defect correction

algorithm may be appropriate. The operator  $P$  may be a lower order, simpler approximation to the same system.

Given an approximation  $v^n$  (where  $n$  is the iteration parameter) near a root  $v^{n+1}$  of Eq. (5.1), we can expand Eq. (5.1) using the Taylor series to get

$$\begin{aligned}
 0 &= A(v^{n+1}) - b \\
 &= A(v^{n+1}) - b + P(v^{n+1}) - P(v^{n+1}) \\
 &= A(v^n) - b + P(v^{n+1}) - P(v^n) - (J_P - J_A)(v^{n+1} - v^n) + O(\varepsilon^2) \quad ,
 \end{aligned}
 \tag{5.4}$$

where  $\varepsilon = v^{n+1} - v^n$ . The defect correction is any  $O(\varepsilon)$  approximation to Eq. (5.4); that is, to

$$P(v^{n+1}) = P(v^n) - A(v^n) + b \quad . \tag{5.5}$$

The iteration will converge if  $v^n$  and  $J_P$  (the Jacobian of  $P$ ), are near enough to  $v^{n+1}$  and  $J_A$ , respectively. This will usually be the case if both  $A$  and  $P$  are different discretizations of the same equation.

The approximate operation  $P$  can also be chosen to make Eq. (5.5) even easier to solve using an SOR, ADI, ILU or multigrid approximation.<sup>16</sup> When this is done, the residuals need to be computed with the high-order formula only in the last few iterations when the iteration is almost converged. The cost of the high-order approximations in the residual calculations are often small and more than justified in light of the resulting increase in accuracy.

The defect correction iteration can often be speeded up by using an acceleration technique such as a Chebyshev or conjugate gradient method.

## VI. SUMMARY

We have used a modular approach to design a subroutine package calculating numerical approximations to the spatial derivatives of a function defined only at a discrete set of points. The routines are flexible, easy to use, and compatible to further expansions of the package. We hope that the software development and maintenance time of future PDE and interpolation codes using the package will be substantially reduced.

We are extending the package to include some pseudo-spectral methods and some better interpolation methods on two- and three-argument grids. We encourage

others to develop compatible subroutines that could be added to DERMOD. We will gratefully consider including into DERMOD any code sent to us that has been programmed using standard FORTRAN and the same supporting routines as the current package. For further information please contact J. M. Hyman.

#### ACKNOWLEDGMENTS

We thank Blair Swartz and John Van Rosendale for computing the lumped finite element method formulas on the two-argument grids and pointing out that the second derivative approximations are pointwise inconsistent. This work was supported by the US Department of Energy under contract W-7405-ENG-36.

#### REFERENCES

1. "Numerical Interpolation, Differentiation, and Integration," Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, M. Abramowitz and I. A. Stegun, Eds. June 1965, No. 55, pp. 877-899.
2. W. G. Bickley, "Numerical Differentiation Formulae for Numerical Differentiation," The Mathematical Gazette, 25, 18-27 (1941).
3. W. G. Bickley, "Finite Difference Formulae for the Square Lattice," Q. J. Mech. Appl. Math., 16, No. 1, 35-42 (1948).
4. G. Birkhoff and S. Gulati, "Optimal Few-Point Discretizations of Linear Source Problems," SIAM J. Numer. Anal., 11, No. 4 (September 1974).
5. Tony F. C. Chan, "Comparison of Numerical Methods for Initial Value Problems," Computer Science Department, Stanford University report STAN-CS-78-672 (November 1978).
6. L. Collatz, The Numerical Treatment of Differential Equations, 3rd ed. (Springer-Verlag, Berlin, 1960), Appendix.
7. H. B. Keller and V. Pereyra, "Symbolic Generation of Finite Difference Formulas," Mathematics of Computation, 32, No. 144, 955-971 (October 1978).
8. R. E. Lynch and J. R. Rice, "A High-Order Difference Method for Differential Equations," Mathematics of Computation, 34, No. 150, 333-372 (April 1980).
9. J. M. Hyman and T. Manteuffel, "Local Polynomial Smoothing Methods for Noisy Data," Los Alamos National Laboratory informal report LA-UR-82-660 (1982).
10. S. T. Zalesak, "Very High Order and Pseudospectral Flux-Corrected Transport (FCT) Algorithms for Conservation Laws," Proceedings of the Fourth IMACS International Symposium on Computer Methods for Partial Differential Equations, Lehigh University, Bethlehem, PA, June 30-July 2, 1981.
11. J. E. Osborn, "The Numerical Solution of Differential Equations with Rough Coefficients," in Advances in Computer Methods for PDEs-IV, R. Vichnevetsky and R. S. Stapleman, Eds., (IMACS, New Brunswick, NJ, 1981), pp. 9-13.
12. Gerald N. Minerbo, "Another 9-Point Scheme (for Solving the Diffusion Equation on a Lagrangian Hydrodynamic Mesh)," Lecture notes, Los Alamos National Laboratory, 1979.
13. D. Cunsolo and P. Orlandi, "Accuracy in Non-Orthogonal Grid Reference Systems," Numerical Methods in Laminar and Turbulent Flow, (John Wiley & Sons, New York/Toronto, 1978), pp. 899-912.

14. J. F. Thompson and C. W. Mastin, "Grid Generation Using Differential Systems Techniques," Numerical Grid Generation Techniques, Hampton, Virginia, 1980), NASA CP. 2160, pp. 37-72, Institute for Computer Applications in Science and Engineering.
15. J. M. Hyman, "The Method of Lines Solution of Partial Differential Equations," Courant Institute of Mathematical Sciences report COO-3077-139 (1976).
16. J. M. Hyman, "Numerical Methods for Nonlinear Differential Equations," Los Alamos National Laboratory report LA-8927-MS (1981).