Army Research Laboratory

Terrain Elevation Program for BFM: New Development

By Teizi Henmi
Stephen F. Kirby

Information Science and Technology Directorate
Battlefield Environment Division

ARL-MR-424

October 1999

Approved for public release; distribution unlimited.

20000313 129

DTIC QUALITY INSPECTED 3

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Washington Headquarters Services, Directorate for information Operations and reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302 and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE October 1999	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Terrain Elevation Program for BFM: New Development		5. FUNDING NUMBERS
6. AUTHOR(S) Teizi Henmi and Stephen F. Kirby		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Information Science and Technology Directorate Battlefield Environment Division ATTN: AMSRL-SL-EW White Sands Missile Range, NM 88002-5501		8. PERFORMING ORGANIZATION REPORT NUMBER ARL-MR-424
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory 2800 Powder Mill Road Adelphi, MD 20783-1145		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARL-MR-424
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.		12b. DISTRIBUTION CODE A
13. ABSTRACT (Maximum 200 words)		
14. SUBJECT TERMS BFM, Terrain Elevation, GTOPO30		15. NUMBER OF PAGES 67
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF THIS REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
prescribed by ANSI Std. Z39-18-298-102

Preface

This report describes the computer program to produce the terrain data for the Battlescale Forecast Model (BFM) using the U. S. Geological Survey's (USGS) 30 arc-second elevation data (GTOPO30). The program written in the C language is also attached in appendix A.

Contents

Preface 1

Executive Summary 5

1. Introduction 7
   1.1 Background 7
   1.2 Purpose 7

2. GTOPO30 Data Set 9

3. Method of Elevation Data Creation for BFM 13

4. Program Description 15

5. Examples of Terrain Data 17

6. Limitation of the Program 21

7. Summary 23

Acronyms 25

Appendix A 27

Distribution 57

Figures

Figure 1. Map showing GTOPO30 tiles (obtained from the GTOPO30 home web page) 10
Figure 2. Terrain data showing northwestern Alaska and the eastern edge of Siberia, centered at 65° N and 165° W. Data were obtained from one tile 18
Figure 3. Terrain data for eastern Brazil entered at 8° S and 39.5° W. Data were obtained from two tiles 18
Figure 4. Terrain data for the midwestern United States centered at 40.0° N and 100°. Data were obtained from four tiles 19

Tables

Table 1. Tile and USGS names 10
Table 2. Latitude limits 21

Executive Summary

A new computer program named *read_terrain.c* is developed to generate the elevation data for the Battlescale Forecast Model (BFM) from the U. S. Geological Survey (USGS) 30 arc-second elevation data (GTOPO30 data set). Since the data set covers the entire globe, terrain data for the BFM can be obtained for any part of the world, except for the areas in high latitudes.

This report describes the GTOPO30 data set, which contains the data every 30 arc-second over the entire globe. The data was originally contained in five compact disk read only memory (CDROM)s to increase the portability and ease of use. The data were compressed and stored on a single CDROM.

The method of data extraction from the data set is mentioned, which is capable of producing the terrain data for the BFM from a multiple number (up to four) tiles. Several function subprograms are used in this "C" language program, and the role of each subprogram is described. Examples of terrain data produced by this program are shown. The program codes in "C" language are included in appendix A.

1. Introduction

Battlescale Forecast Model (BFM), developed at the U.S. Army Research Laboratory (ARL), has been extensively used to produce short-range forecasts of atmospheric conditions as a component in both the Integrated Meteorological System (IMETS) and the Computer Assisted Artillery Meteorology (CAAM) system.

1.1 Background

The elevation data is an essential input data for a mesoscale weather forecast model such as the BFM. Previously, the BFM has obtained elevation data from the Digital Terrain Elevation Data (DTED) Level 1 data set, produced by the National Imagery and Mapping Agency (formerly the Defense Mapping Agency). DTED Level 1 has a horizontal grid spacing of 3-arc seconds (approximately 90 m). The data on the compact disk read only memory (CDROM) usually covers an area of 12° of latitude by 18° of longitude. Therefore, a substantial number of CDROMs are needed to cover the entire globe. The BFM is typically used over model areas ranging from 100 x 100 km to 500 x 500 km with a model grid spacing of 5 to 10 km. For the current BFM on IMETS, elevation data for an area of 1,600 x 1,600 km is needed for a model domain of 500 x 500 km. Therefore, depending on the area of interest, a number of CDROMs are required to create elevation data for the model. A serious shortcoming of the DTED Level 1 data set is that there are regions where no data are provided, most notably various regions in the Southern Hemisphere.

The U.S. Geological Survey (USGS) GTOPO30 is a global digital elevation data set spanning the entire globe. The elevation data in GTOPO30 are regularly spaced at 30-arc seconds (approximately 1 km). Thus, it is an appropriate data source to create elevation data for BFM application in any part of the world.

1.2 Purpose

The purpose of this technical note is to describe the method and a computer program developed to create the elevation data for BFM from the GTOPO30 data set:

- section II, GTOPO30 data are described,
- section III, the method of data reduction is described,
- section IV, the role of the function subprogram in the program is described,
- section V examples of terrain data produced by the program are shown,
- section VI briefly mentions limitations of the program,
- section VII is a summary, and
- appendix A, the computer program in C language is provided.

2. GTOPO30 Data Set

GTOPO30 is a global data set covering the full latitude range (90° south to 90° north), and the full range of longitude (180° west to 180° east). The horizontal grid spacing is 30-arc seconds. The data is given in meters above mean sea level (MSL). Ocean areas are assigned a value of -9999. The data is provided as 16-bit signed integer data in a simple binary format.

GTOPO30 is divided into 33 small pieces or tiles. The area from 60°south latitude to 90° north latitude and from 180° west longitude to 180° east longitude is covered by 27 tiles, with each tile covering 50° of latitude and 40° of longitude.

Antarctica (90° south latitude to 60° south latitude and 180° west longitude to 180° east longitude) is covered by 6 tiles. Each tile covers 30° of latitude and 60° of longitude. Figure 1 shows the distribution of tiles covering the entire globe.

Each of the 27 tiles that individually cover 50°of latitude and 40°of longitude have 6,000 rows and 4,800 columns. Each of the six Antarctica tiles that individually cover 30° latitude and 60° of longitude have 3,600 rows and 7,200 columns.

Details of GTOPO30 are described in the documentation found in the universal resource locator (URL) address on the web:

http://edcwww.cr.usgs.gov/landdaac/gtopo30/

To increase the portability and ease of use of this software, the data for 33 tiles have been extracted from 5 CDROMs obtained from the USGS and compressed using the **gzip** command enabling the data for 33 tiles to be stored on a single CDROM. Through the use of the gzip utility, the total size of the global data is reduced from almost 2.72 GB to approximately 290 MB. Thus, elevation data for any part of the world can be obtained from a single CDROM. Table 1 gives the names of the tiles contained in the CDROM and the USGS names of the tiles.

Figure 1. Map showing GTOPO30 tiles (obtained from the GTOPO30 home web page).

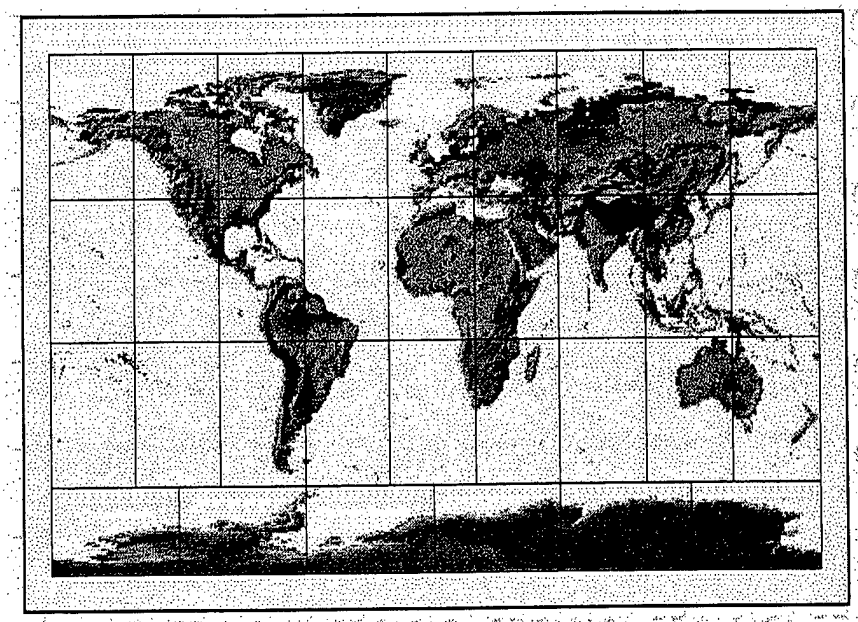


Table 1. Tile and USGS names

Number	Tile name	USGS names
1	e020n9~1	E020N90.DEM
2	e060n9~1	E060N90.DEM
3	e100n9~1	E100N90.DEM
4	e140n9~1	E140N90.DEM
5	w180n9~1	W180N90.DEM
6	w140n9~1	W140N90.DEM
7	w100n9~1	W100N90.DEM
8	w060n9~1	W060N90.DEM
9	w020n9~1	W020N90.DEM
10	e020n4~1	E020N40.DEM
11	e060n4~1	E060N40.DEM
12	e100n4~1	E100N40.DEM
13	e140n4~1	E140N40.DEM

Table 1. Tile and USGS names (continued)

Number	Tile name	USGS names
14	w180n4~1	W180N40.DEM
15	w140n4~1	W140N40.DEM
16	w100n4~1	W100N40.DEM
17	w060n4~1	W60N40.DEM
18	w020n4~1	W020N40.DEM
19	e020s1~1	E020N40.DEM
20	e060s1~1	E060S10.DEM
21	e100s1~1	E100S10.DEM
22	e140s1~1	E40S10.DEM
23	w180s1~1	W180S10.DEM
24	w140s1~1	W140S10.DEM
26	w060s1~1	W060S10.DEM
27	w020s1~1	W020S10.DEM
28	e060s6~1	E060S60.DEM
29	e120s6~1	E120S60.DEM
30	w180s6~1	W180S60.DEM
31	w120s6~1	W120S60.DEM
32	w060s6~1	W060S60.DEM
33	w000s6~1	W000S60.DEM

In table 1, each name represents the latitude and longitude of the northwest corner of tiles. The names of the USGS files changed as they were gzipped and stored on CDROM due to file name length limitations in the disk operating system (DOS). For instance, tile # 7, w100n9~1 covers longitudes from 100° west to 60° west, and latitudes from 90° north to 40° north.

3. Method of Elevation Data Creation for BFM

The procedures to create elevation data sets from the CDROM are as follows:

In the BFM, Cartesian coordinates are used for horizontal coordinates. Therefore, the latitude and longitude of model grid points (I, J) are calculated as

$$\theta(J) = \theta_0 + \left[\frac{\Delta \cdot \left(J - \left\{ \frac{J_{max}+1}{2} \right\} \right)}{R_E \cdot \left(\frac{\pi}{180} \right)} \right] \tag{1}$$

$$\phi(I,J) = \phi_0 + \left[\frac{\Delta \cdot \left(I - \left(\frac{I_{max}+1}{2} \right) \right)}{R_E \left(\frac{\pi}{180} \right) \cdot \cos\left(\left(\frac{\pi}{180} \right) \theta(J) \right)} \right] \tag{2}$$

where

R_E	=	the radius of the earth
Δ	=	the unit grid distance
ϕ_0	=	longitude of the center of a model domain
θ_0	=	latitude of the center of model domain
(I_{max}, J_{max})	=	the mamimum numbers of grid points for the (x, y) directions
(I, J)	=	grid point counters in the x and y directions

Therefore, minimum and maximum longitudes and latitudes of the model domain are given as

For $\theta_0 \geq 0$,

minimum longitude = $\phi_{min} = \phi(1, J_{max})$
maximum longitude = $\phi_{max} = \phi(I_{max}, J_{max})$

and for $\theta_0 \leq 0$,

minimum longitude = $\phi_{min} = \phi(1, 1)$
maximum longitude = $\phi_{max} = \phi(I_{max}, 1)$
minimum latitude = $\theta_{min} = \theta(1, 1) = \theta(I_{max}, 1)$

maximum latitude = $\theta_{max} = \theta(1, J_{max}) = \theta(I_{max}, J_{max})$

Since the data are given at every 30 arc second (or 1/120°), the numbers of a data point in longitudinal and latitudinal directions, i_t and j_t, for an area bounded by ϕ_{min}, ϕ_{max}, θ_{min}, and θ_{max} are calculated as

$$i_t = (\phi_{max} - \phi_{min}) \times 120 + 1 \tag{3}$$

$$j_t = (\theta_{max} - \theta_{min}) \times 120 + 1 \tag{4}$$

The longitudes and latitudes of the elevation data covering the model domain are given by

$$\phi(i) = \phi_{min} + (i - 1) \times (1/120), \quad i = 1, 2, \cdots\cdots, i_t \tag{5}$$

$$\theta(j) = \theta_{min} + (j - 1) \times (1/120), \quad j = 1, 2, \cdots\cdots, j_t \tag{6}$$

All tiles are examined to see if they contain any elevation data for $\phi(i)$, $i=1,2\cdots\cdots, i_t$, and $\theta(j)$, $j=1,\cdots\cdots, j_t$ and, any tile(s) containing the data are copied to the directory in which the program is being executed and uncompressed. Depending on the location and size of the model domain, elevation data may have to be obtained from a multiple number of tiles.

The present program is designed so that elevation data can be obtained from 1, 2, or 4 tiles. Therefore, if the model domain is located at high latitudes where longitudinal lines are densely packed, the present program may become incapable of extracting elevation data, depending on the area size. The limitations of the program will be discussed in sectionVI.

Within each tile, the data is stored in row major order (all the data for row 1, followed by the data for row 2, etc.). The northwest corner is designated as (0, 0) in the program. The portion of data in the tiles which encompass the model domain are extracted and used to interpolate elevation data for the model domains.

If a grid-point (x′, y′) of the model domain is surrounded by four grid points of the elevation data, an interpolated elevation value Z′ at (x′, y′) is then calculated using bilinear interpolation as

$$t_1 = Z(x, y) + (x' - x) \cdot [Z(x + 1, y) - Z(x, y)] \tag{7}$$

$$t_2 = Z(x, y + 1) + (x' - x) \cdot [Z(x + 1, y + 1) - Z(x, y + 1)] \tag{8}$$

$$Z'(x', y') = t_1 + (y' - y) \cdot [t_2 - t_1] \tag{9}$$

Here, (x, y) is the southwest grid of four grid points surrounding the model grid point (x′, y′), and Z (x, y) is the elevation data at (x, y). This operation is repeated for all grid points of the model domain.

4. Program Description

Several functions subprograms are used in this program. In the following, the role of each function is described:

- *main* The latitude and longitude of the center of the model domain, the number of grid points in the x and y directions, and the grid spaces in kilometers are typed in this program. Latitude and longitude values of the model domain grid points are calculated, and maximum and minimum latitudes and longitudes are passed to the function called *data30s*.

- *data30s* This subprogram examines which and how many tile(s) encompass the model domain. The tiles containing necessary data are copied, and uncompressed by the gzip -d command to the directory where the program is executed. Elevation data on the tile(s) are read from the northwest corner; (for example, all the data for row 1, followed by all the data for row 2). After one row of the data is read, the data is then passed onto the function *get_data1* or *get_data2* in order to select the portion of data that encompasses the model domain.

- *get_data1* and *get_data2* These functions examine whether the elevation data encompasses the model domain. An array of elevation data is then made in these functions and passed onto the function *zinterp*. The function *get_data1* is used for the tiles 1 to 27 and *get_data2,* for the tiles 28 to 33.

- *zinterp* Is a subprogram that performs bilinear interpolation of elevation data for the grid points of the model domain and creates "terrain_data" for the model domain.

- *cp_from_cd* The data on a tile are copied from a CDROM onto the directory where the program resides.

- *unzip* To unzip the command gzip -d is executed to uncompress data for a given tile.

- *rm_file* After uncompressed data have been read, data file(s) are removed from the directory of program execution.

5. Examples of Terrain Data

The computer program is capable of producing terrain data for the BFM from some multiple numbers of tiles. In this section, three different terrain data files produced from different numbers of tiles are presented:

- *Western Alaska and Bering Strait Area* Figure 2 shows the terrain data for the area of 1,000 x 1,000 km (101 x 101 grids) centered at 65.0° N and 165.0° W, and grid spacing of 10 km. These data are obtained from one tile, w180n9~1. The DTED level 1 does not have complete coverage of this area.

- *Brazil Area* The DTED Level 1 data set does not have any data coverage over Brazil. By using the present program and data set, the terrain data over Brazil can be generated. Figure 3 represents the central part of Brazil, the area centered at 8.0° S and 39.5° W covering 1,000 x 1,000 km (51 x 51 grids) with grid spacing of 20 km.

- *Midwestern US* Figure 4 shows the terrain data obtained from four tiles, w140n9~1, w100n9~1, w140n4~1, and w100n4~1. The center of this domain is located at the four corner point, 40° N and 100° W. The data covers 1,000 km x 1,000 km (101 x 101 grids) with 10 km grid spacing.

Figure 2. Terrain data showing northwestern Alaska and the eastern edge of Siberia, centered at 65° N and 165° W. Data were obtained from one tile.

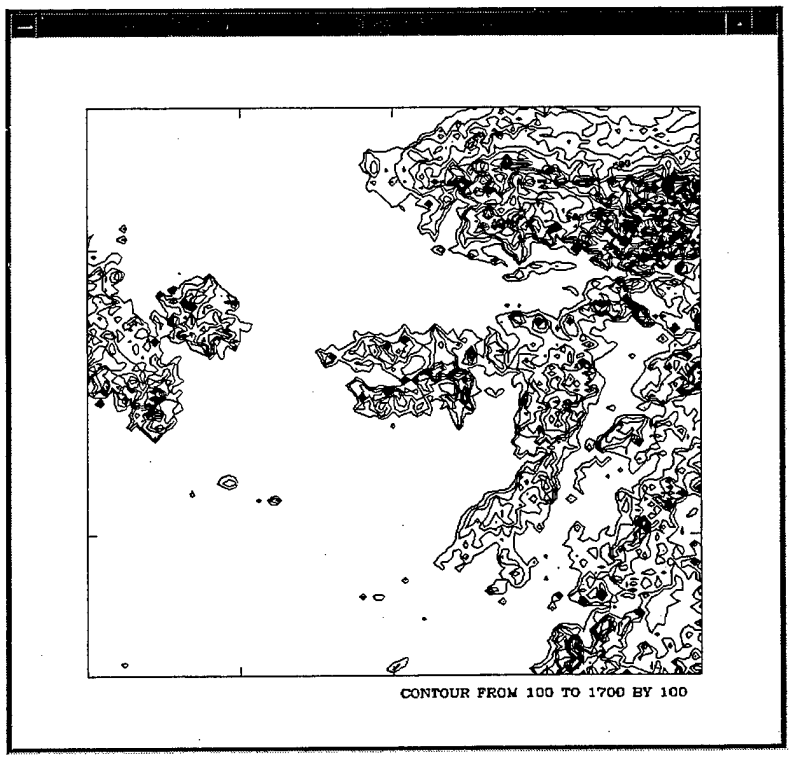


Figure 3. Terrain data for eastern Brazil entered at 8° S and 39.5° W. Data were obtained from two tiles.

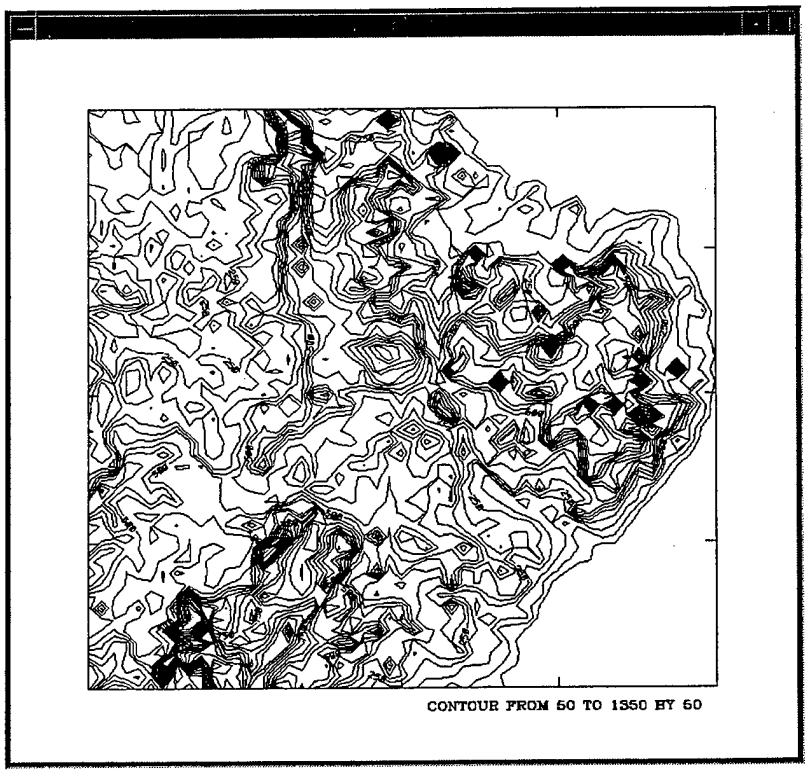
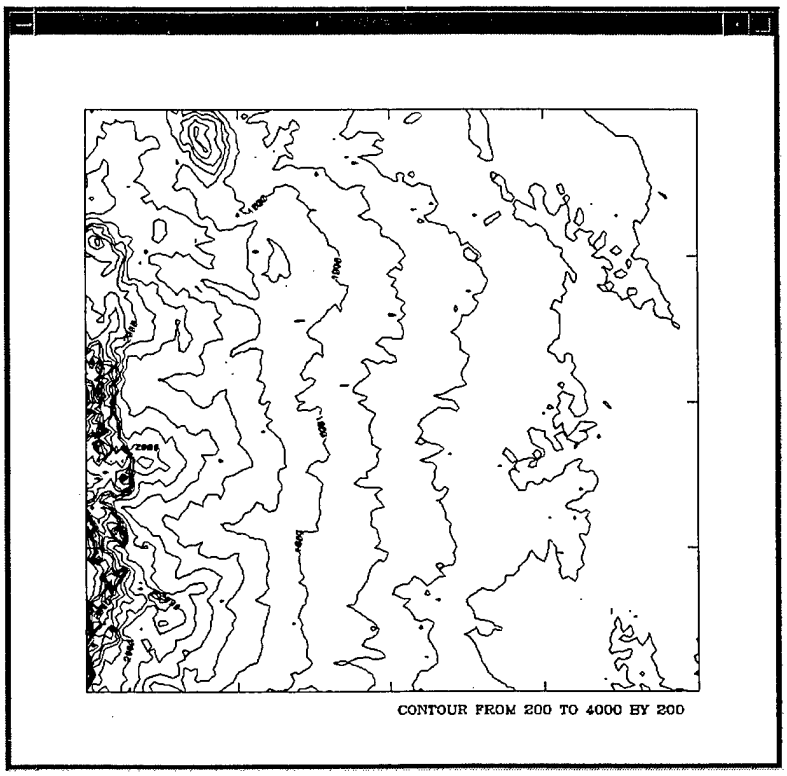


Figure 4. Terrain data for the midwestern United States centered at 40.0° N and 100°. Data were obtained from four tiles.



6. Limitation of the Program

Because the program is capable of extracting elevation data only from two consecutive tiles in both longitudinal and latitudinal directions, data in high latitudes cannot be obtained.

Table 2 shows the limits of latitudes over which elevation data cannot be obtained.

Table 2. Latitude limits

	Upper Limit of the Center of Domain	
Area Width (km)	Northern Hemisphere	Southern Hemisphere
1,600	60.3°	69.0°
1,000	71.8°	76.3°
500	80.0°	83.6°

In the attached program code (appendix A) the greatest number of grids for the model domain is currently set to 161 x 161. Therefore, if this grid number is exceeded, the program must be modified. The program is developed for the UNIX operating system. One additional limitation is that the terrain data with grid spacing smaller than 30 arc-second (about 1 km) cannot be produced.

7. Summary

The present program is developed to produce terrain data for the BFM from the USGS GTOPO30 (global digital elevation data set). GTOPO30 data are extracted from five CDROMs, compressed by the UNIX utility "grip" and archived onto a single CDROM. Thus, terrain data for most of the BFM application can be obtained from the CDROM by using the present program.

Acronyms

ARL	U.S. Army Research Laboratory
BFM	Battlescale Forecast Model
CAAM	Computer Assisted Artillery Meterology
CDROM	compact disk read only memory
DOS	disk operating system
DTED	Digital Terrain Elevation Data
IMETS	Integrated Meterological System
MSL	mean sea level
URL	universal resource locator
ARL	U.S. Army Research Laboratory
USGS	U.S. Geological Survey

Appendix A

Program: read_terrain.c

/* read_terrain.c */

```
/* Program to extract and interpolate an elevation */
 /* data file from up to 4 USGS GTOPO30 data tiles, depending */
/* on how many boundaries are overlapped. In     */
/* these files, -999 represents water so it is    */
/* replaced with 0 as it is read.              */

/* Steve Kirby (30 July 1999) */

#include <stdio.h>
#include <math.h>

#define ip 161
#define jp 161
#define i_x 4000
#define j_y 4000
#define n1 4800
#define n2 6000
#define n3 7200
#define n4 3600

#define conv_factor 3.1416/180.
#define re 6371.22

FILE *fp,*fp1,*fp2,*fp3,*fp4,*fpout,*fpcheck;
float longi,lati,gridsp;
float origlon;
int val,min_val,max_val,numx,numy;
int i,j,imax=4800,jmax=6000,ip1,jp1;
float lat_b[jp],lon_b[ip][jp],gdis,zs[ip][jp],lato,lono;
float center=1.0/120.0;
float z[i_x][j_y],lat[j_y],lon[i_x],hi_2[i_x][j_y],
lat_o_2[j_y];
float lon_o_2[i_x],lon_o2[2*n3];
signed short hi[2*n1];
signed short hi2[2*n3];
int i_lon,j_lat;
float xminlon,xmaxlon,xminlat,xmaxlat;
float min_lat,min_lon,max_lat,max_lon;
float lon_o1[2*n1];

/* prototypes */
void data30s(float,float,float,float,float);
void get_data1(float,float,float,float,int,int,float,
```

```
float,int*);
void get_data2(float,float,float,float,int,int,float,
float,int*);
void zinterp(int,int,float [],float []);
void cp_from_cd(char[]);
void unzip(char []);
void rm_file(char []);
int main(){

/* user input for center of domain */
printf("enter the center lat/lon of the domain
(- in SH and WH): ");
scanf("%f %f",&lato,&lono);

/* need to save input lon for output */
/* in case it's negative */
origlon=lono;
/* user input for size of grid */
printf("enter the number of grid points in the
x- and y-direction: ");
scanf("%d %d",&ip1,&jp1);
/* user input for grid spacing in kms */
printf("enter the grid spacing (kms): ");
scanf("%f",&gdis);

printf("\n\ncenter lat | lon:%5.2f %6.2f\n
size of grid:%dx%d\n grid spacing:%6.2f kms\n",lato,lono,ip1,jp1,gdis);

/* formulate lat | lons of BFM grid points */
for(j=0;j<jp1;j++){
lat_b[j]=lato + ((float)(2*(j+1)-jp1-1)/2.) * (gdis/(re*conv_factor));
for(i=0;i<ip1;i++){
lon_b[i][j]=lono+(float)(2*(i+1)-ip1-1)/2.*(gdis/(re*conv_factor*
cos(lat_b[j]*conv_factor)));
        }
        }
min_lat=lat_b[0];
max_lat=lat_b[jp1-1];
if(lato<0.0)
        {
max_lon=lon_b[ip1-1][0];
min_lon=lon_b[0][0];
        }
        else
        {
min_lon=lon_b[0][jp1-1];
max_lon=lon_b[ip1-1][jp1-1];
        }
```

```
data30s(max_lat,min_lat,max_lon,min_lon,center);
return 0;
        }
void data30s(float xmaxlat,float xminlat, float
xmaxlon,float xminlon,float center)
        {
float dlat0[33],dlon0[33],dlat1[33],dlon1[33];
signed short hi_max=-1000,hi_min=1000;
int kd[33];
int lon_index;
int nrs,it,jt,nk,k,i,j;
float lon_f,lat_f;
char lon_f_str[4],lat_f_str[3];
float lat[j_y],lon[i_x];
float xlon0,xlat0,xlon1,xlat1;
float xx,yy;
int imax1,jmax1,imax2,jmax2,imax,jmax;
float lat_o1[2*n3],lat_o2[2*n3];
int index_region,index_r,index,i_switch,j_switch;
float dlat_max,dlat_min,dlon_max,dlon_min;
char *elev_data[4];
int jj;
float dlon00[4];
char *dem_file[]={
"e020n9~1","e060n9~1","e100n9~1",
"e140n9~1","w180n9~1","w140n9~1",
"w100n9~1","w060n9~1","w020n9~1",
"e020n4~1","e060n4~1","e100n4~1",
"e140n4~1","w180n4~1","w140n4~1",
"w100n4~1","w060n4~1","w020n4~1",
"e020s1~1","e060s1~1","e100s1~1",
"e140s1~1","w180s1~1","w140s1~1",
"w100s1~1","w060s1~1","w020s1~1",
"e060s6~1","e120s6~1","w180s6~1",
"w120s6~1","w060s6~1","w000s6~1"};
char show_file[60];
printf("\nbounding region-- LAT:%5.2f %5.2f LON:%5.2f
%5.2f\n",xminlat,xmaxlat,
xminlon,xmaxlon);
nrs=(int)(1./center + 0.5);
it=(xmaxlon-xminlon)*nrs;
jt=(xmaxlat-xminlat)*nrs;
```

```
nk=-1;
for(k=0;k<33;k++){
lon_f_str[0]=dem_file[k][1];
lon_f_str[1]=dem_file[k][2];
lon_f_str[2]=dem_file[k][3];
lon_f_str[3]='\0';
lon_f=(float)atoi(lon_f_str);
lat_f_str[0]=dem_file[k][5];
lat_f_str[1]='0';
lat_f_str[2]='\0';
lat_f=(float)atoi(lat_f_str);
if(dem_file[k][0] == 'w')
xlon0=-lon_f;
        else
xlon0=lon_f;

if(dem_file[k][4] == 's')
xlat1=-lat_f;
        else
xlat1=lat_f;
if(k <= 26){
xlon1=xlon0+(float)(n1-1)/120.;
if(xlon1 >= 360.0)
xlon1-=360.;
xlat0=xlat1-(float)(n2-1)/120.;
        }
        else{
xlon1=xlon0+(float)(n3-1)/120.;
if(xlon1 >= 360.0)
xlon1-=360.;
xlat0=xlat1-(float)(n4-1)/120.;
        }

if((xminlon > xlon1) && (xminlat > xlat1))
goto label20;
if((xminlon == -180.) && (xmaxlon == 180.)){
/* the pole is inside the domain */
if((xmaxlat == 90.) && (k <= 8)){
nk+=1;
kd[nk]=k;
dlat0[nk]=xlat0;
dlon0[nk]=xlon0;
dlat1[nk]=xlat1;
dlon1[nk]=xlon1;
            }
else if((xminlat == -90.) &&
(k > 26)){
/* South Pole */
```

```
nk+=1;
kd[nk]=k;
dlat0[nk]=xlat0;
dlon0[nk]=xlon0;
dlat1[nk]=xlat1;
dlon1[nk]=xlon1;
                }
                }
                else{
/* the pole is outside the domain */
for(i=0;i<it;i++){
xx=xminlon+i*center;
for(j=0;j<jt;j++){
yy=xminlat+j*center;
if((xx>=xlon0) &&
(xx<xlon1) &&
(yy>xlat0) &&
(yy<xlat1) &&
(y>=xminlat) &&
(yy < xmaxlat) &&
nk+=1;
dlat0[nk]=xlat0;
dlon0[nk]=xlon0;
dlat1[nk]=xlat1;
dlon1[nk]=xlon1;
kd[nk]=k;
goto label20;
                }
                }
                }
                }
label20:
continue;
                }
printf("\nneed %d 30-sec tile(s):\n",nk+1);
for(k=0;k<=nk;k++){
strcpy(show_file,dem_file[kd[k]]);
show_file[6]='0';
show_file[7]='\0';
printf("%d>> %s LAT: %5.2f %5.2f LON:
%6.2f %6.2f\n",k+1,show_file,dlat0[k],
dlat1[k],dlon0[k],dlon1[k]);
        }
```

```
lon_index=1;
if((nk == 1 && dlon0[0] == 20.0 && dlon0[1]
== -20.0) || (nk == 1 && dlon0[0] == 60.0 && dlon0[1] == 0.0)){
lon_index=2;
elev_data[0]=dem_file[kd[1]];
elev_data[1]=dem_file[kd[0]];

dlon00[0]=dlon0[1];
dlon00[1]=dlon0[0];

if(nk ==2 && dlon0[1] == 60.0 && dlon0[2]
== 0.0){
lon_index=2;
elev_data[0]=dem_file[kd[0]];
elev_data[1]=dem_file[kd[2]];
elev_data[2]=dem_file[kd[1]];

dlon00[1]=dlon0[2];
dlon00[2]=dlon0[1];
        }
else if(nk == 3 && dlon0[0] == 20.0 &&
dlon0[1] == -20.0 && dlon0[2]== 20.0 &&
dlon0[3]==-20.0){
lon_index=2;
elev_data[0]=dem_file[kd[1]];
elev_data[1]=dem_file[kd[0]];
elev_data[2]=dem_file[kd[3]];
elev_data[3]=dem_file[kd[2]];

dlon00[0]=dlon0[1];
dlon00[1]=dlon0[0];
        }
else if(nk == 3 && dlon0[2] == 60.0
&& dlon0[3] == 0.0){
lon_index=2;
elev_data[2]=dem_file[kd[3]];
elev_data[3]=dem_file[kd[2]];

dlon00[2]=dlon0[3];
dlon00[3]=dlon0[2];
        }
        else{
for(k=0;k<=nk;k++)
elev_data[k]=dem_file[kd[k]];
        }
        }
        else{
```

```
for(k=0;k<=nk;k++)
elev_data[k]=dem_file[kd[k]];
        }
for(k=0;k<=nk;k++){
if(dlat0[k] < -60.){
index_region=2;
goto label22;
        }
        else{
index_region=1;
jmax=n2;
imax=n1;
        }
        }

label22:
if(index_region == 1){
if(nk == 0){
/* lat | lon of elevation data */
label5000:
for(j=0;j<jmax;j++)
lat_o1[j]=dlat0[nk]+
center*(float)j;
for(i=0;i<imax;i++)
lon_o1[i]=dlon0[nk]+
center*(float)i;
/* open the elevation data */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
if(!(fp=fopen(elev_data[0],"r"))){
printf("USGS DEM file
%s unavailable ...
exiting...\n",dem_file[kd[nk]]
);
exit(0);
        }
j_lat=-1;
j=jmax-1;
while(j>=1){
for(i=0;i<imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp);
if(hi[i] == -9999)
hi[i]=0;
if(hi[i]>hi_max)
hi_max=hi[i];
if(hi[i]<hi_min)
hi_min=hi[i];
```

```
        }
index_r=1;
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,imax,lat_o1[j],lat_o1[j-1],&index_r);
if(index_r == 2)
goto label210;
j--;
    }
label210:
fclose(fp);
rm_file(elev_data[0]);
    }
else if(nk == 1){
/* tile 1 is above, and
tile 2 is below */
if(dlon0[0] == dlon0[1] &&
dlon1[0] == dlon1[1]){
printf("case tile 1 above,
tile 2 below\n");
for(j=0;j<2*jmax;j++)
lat_o1[j]=dlat0[1]+
center*(float)j;

for(i=0;i<imax;i++)
lon_o1[i]=dlon0[1]+
center*(float)i;

/* open elevation data */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
cp_from_cd(elev_data[1]);
unzip(elev_data[1]);
if(!(fp1=
fopen(elev_data[0],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[0]);
exit(0);
        }
if(!(fp2=
fopen(elev_data[1],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[1]);
exit(0);
        }

j_lat=-1;
j=2*jmax-1;
while(j>=1){
if(j>=jmax){
for(i=0;i<imax;i++){
fread(&hi[i],
```

```
sizeof(signed short),1,fp1);
if(hi[i] == -9999)
hi[i]=0;
        }
        }
        else{
for(i=0;i<imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] ==
-9999)
hi[i]=0;
        }
        }
index_r=1;
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,imax,
lat_o1[j],
lat_o1[j-1],&index_r);
if(index_r == 2)
goto label310;
j--;
        }

label310:
fclose(fp1);
rm_file(elev_data[0]);
fclose(fp2);
rm_file(elev_data[1]);
        }
/* tile 1 is to the left;
tile 2 is to the right */
else if(dlat0[0] == dlat0[1]
&& dlat1[0] == dlat1[1]){
printf("case tile 1 to
left tile 2 to right\n");
label6000:
for(j=0;j<jmax;j++)
lat_o1[j]=dlat0[0]
+ center*(float)j;

if(lon_index ==1){
for(i=0;i<2*imax;i++)
lon_o1[i]=
dlon0[0] +
center*(float)i;
        }
else if(lon_index == 2){
for(i=0;i<2*imax;i++)
```

```
lon_o1[i]=
dlon00[0] + center*(float)i;
        }
/* open the elevation data */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
cp_from_cd(elev_data[1]);
unzip(elev_data[1]);

if(!(fp1=
fopen(elev_data[0],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[0]);
exit(0);
        }
if(!(fp2=
fopen(elev_data[1],"r"))){
print f("USGS DEM file %s unavailable ... exiting ...\n",elev_data[1]);
exit(0);
        }

j_lat=-1;
j=jmax-1;
while(j>=1){
for(i=0;i<imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp1);
if(hi[i] == -9999)
hi[i]=0;
        }
for(i=imax;i<2*imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] == -9999)
hi[i]=0;
        }
index_r=1;
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,2*imax,

lat_o1[j],
lat_o1[j-1],&index_r);
if(index_r == 2)
goto label410;
j--;
        }
```

```
label410:
fclose(fp1);
rm_file(elev_data[0]);
fclose(fp2);
rm_file(elev_data[1]);
        }
        }
else if(nk == 3){
/* 4-tiles */
for(j=0;j<2*jmax;j++)
lat_o1[j]=dlat0[2]+
center*(float)j;
if(lon_index == 1){
for(i=0;i<2*imax;i++)
lon_o1[i]=dlon0[0]
 + center*(float)i;
        }
else if(lon_index == 2){
for(i=0;i<2*imax;i++)
lon_o1[i]=
dlon00[0] +
center * (float)i;
        }

/* open elevation data */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
cp_from_cd(elev_data[1]);
unzip(elev_data[1]);
cp_from_cd(elev_data[2]);
unzip(elev_data[2]);
cp_from_cd(elev_data[3]);
unzip(elev_data[3]);

if(!(fp1=
fopen(elev_data[0],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[0]);
exit(0);
        }
if(!(fp2=
fopen(elev_data[1],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[1]);
exit(0);
        }
if(!(fp3=
fopen(elev_data[2],"r"))){
printf("USGS DEM file %s unavailable ... exiting
exit(0);
        }
```

```
if(!(fp4=
fopen(elev_data[3],"r"))){
printf("USGS DEM file %s unavailable ... exiting
...\n",elev_data[3]);
exit(0);
        }
j_lat=-1;
j=2*jmax-1;
while(j>=1){
if(j>=jmax){
for(i=0;i<imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp1);
if(hi[i] == -9999)
hi[i]=0;
        }

for(i=imax;i<2*imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] == -9999)
hi[i]=0;
        }
        }
        else{
for(i=0;i<imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp3);
if(hi[i] == -9999)
hi[i]=0;
        }
for(i=imax;i<2*imax;i++){
fread(&hi[i],
sizeof(signed short),1,fp4);
if(hi[i] == -9999)
hi[i]=0;
        }
        }
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,2*imax,
lat_o1[j],lat_o1[j-1],&index_r);
if(index_r == 2)
goto label1310;
        j--;
        }
```

```
label1310:
fclose(fp1);
rm_file(elev_data[0]);
fclose(fp2);
rm_file(elev_data[1]);
fclose(fp3);
rm_file(elev_data[2]);
fclose(fp4);
rm_file(elev_data[3]);
        }
        }
else if(index_region == 2){
jmax1=n2;
imax1=n1;
jmax2=n4;
imax2=n3;
/* 1-tile */
if(nk == 0){
jmax=jmax2;
imax=imax2;
goto label5000;
        }
/* 2-tiles */
else if(nk == 1){
if(dlat0[0]==dlat0[1] && dlat1[0]
== dlat1[1]){
jmax=jmax2;
imax=imax2;
goto label6000;
        }
        else{
jmax1=n2;
imax1=n1;
jmax2=n4;
imax2=n3;
for(j=0;j<jmax1;j++)
/* top tile */
lat_o1[j]=dlat0[0]+
center*(float)j;
for(j=0;j<jmax2;j++)
/* bottom tile */
lat_o2[j]=dlat0[1]+
center*(float)j;
for(i=0;i<imax1;i++)
/* top tile */
lon_o1[i]=dlon0[0]+
center*(float)i;
for(i=0;i<imax2;i++)
```

```
/* bottom tile */
lon_o2[i]=dlon0[1]+
center*(float)i;
/* open elevation data */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
cp_from_cd(elev_data[1]);
unzip(elev_data[1]);
if(!(fp1=
fopen(elev_data[0],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[0]);
exit(0);
        }
if(!(fp2=
fopen(elev_data[1],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[1]);
exit(0);
        }

j_lat=-1;
j=jmax1-1;
while(j>=1){
for(i=0;i<imax1;i++){
fread(&hi[i],
sizeof(signed short),1,fp1);
if(hi[i]==-9999)
hi[i]=0;
        }
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,imax1,
lat_o1[j],
lat_o1[j-1],&index_r);
if(index_r == 2)
goto label3310;
j--;
        }
label3310:
fclose(fp1);
rm_file(elev_data[0]);
j=jmax2-1;
while(j>=1){
for(i=0;i<imax2;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] == -9999)
hi[i]=0;
        }
if(index_r == 2)
```

```
goto label4310;
j--;
        }
label4310:
fclose(fp2);
rm_file(elev_data[1]);
        }
        }
/* 3-tiles */
else if(nk == 2){
jmax1=n2;
imax1=n1;

jmax2=n4;
imax2=n3;

/* open elevation data */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
cp_from_cd(elev_data[1]);
unzip(elev_data[1]);
cp_from_cd(elev_data[2]);
unzip(elev_data[2]);
if(!(fp1=fopen(elev_data[0],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[0]);
exit(0);
        }

if(!(fp2=fopen(elev_data[1],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[1]);
exit(0);
        }
if(!(fp3=fopen(elev_data[2],"r"))){
printf("USGS DEM file %s unavailable ... exiting
...\n",elev_data[2]);
exit(0);
        }

if(dlat0[0]==dlat0[1]){
for(j=0;j<jmax1;j++)
/* top tile */
lat_o1[j]=dlat0[0]+

center*(float)j;
for(j=0;j<jmax2;j++)
/* bottom tile */
lat_o2[j]=dlat0[2]+
center*(float)j;
```

```
for(i=0;i<2*imax1;i++)
/* top tile */
lon_o1[i]=dlon0[0]+
center*(float)i;
for(i=0;i<imax2;i++)
/* bottom tile */
lon_o2[i]=dlon0[2]+
center*(float)i;
j_lat=-1;
j=jmax1-1;
while(j>=1){
for(i=0;i<imax1;i++){
fread(&hi[i],
sizeof(signed short),1,fp1);
if(hi[i] == -9999)
hi[i]=0;
        }
for(i=imax1;
i<2*imax1;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] == -9999)
hi[i]=0;
        }
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,2*imax1,lat_o1[j],
lat_o1[j-1],&index_r);
if(index_r == 2)
goto label5310;
j--;
        }

label5310:
fclose(fp1);
rm_file(elev_data[0]);
fclose(fp2);
rm_file(elev_data[1]);

j=jmax2-1;
while(j>=1){
for(i=0;i<imax2;i++){
fread(&hi2[i],
sizeof(signed short),1,fp3);
if(hi2[i] == -9999)
hi2[i]=0;
        }
if(index_r == 2)
goto label6310;
j--;
        }
```

```
label6310:
fclose(fp3);
rm_file(elev_data[2]);
        }
else if(dlat1[1] == dlat1[2]){
if(lon_index == 1){
for(j=0;j<jmax1;j++)
/* top tile */
lat_o1[j]=dlat0[0]+
center*(float)j;
for(j=0;j<jmax2;j++)
/* bottom tile */
lat_o2[j]=dlat0[1]+
center*(float)j;

for(i=0;i<imax1;i++)
/* top tile */
lon_o1[i]=dlon0[0]
+ center * (float)i;
for(i=0;i<2*imax2;i++)
/* bottom tile */
lon_o2[i]=dlon0[1]
+ center * (float)i;
        }
else if(lon_index == 2){
for(j=0;j<jmax1;j++)
/* top tile */
lat_o1[j]=dlat0[0]
+ center * (float)j;
for(j=0;j<jmax2;j++)
/* bottom tile */
lat_o2[j]=dlat0[1]
+ center * (float)j;
for(i=0;i<imax1;i++)
/* top tile */
lon_o1[i]=dlon0[0]
+ center * (float)i;
for(i=0;i<2*imax2;i++)
/* bottom tile */
lon_o2[i]=dlon00[1] + center * (float)i;
        }
j_lat=-1;
j=jmax1-1;
while(j>=1){
for(i=0;i<imax1;i++){
fread(&hi[i],
```

```
sizeof(signed short),1,fp1);
if(hi[i] == -9999)
hi[i]=0;
        }
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,2*imax,
lat_o1[j],
lat_o1[j-1],&index_r);
if(index_r == 2)
goto label7310;
j--;
        }

label7310:
fclose(fp1);
rm_file(elev_data[0]);

j=jmax2-1;
while(j>=1){
for(i=0;i<imax2;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] == -9999)
hi[i] = 0;
        }
for(i=imax2;
i<2*imax2;i++){
fread(&hi[i],
sizeof(signed short),1,fp3);
if(hi[i] == -9999)
hi[i] = 0;
        }
if(index_r == 2)
goto label7410;
j--;
        }

label7410:
fclose(fp2);
rm_file(elev_data[1]);
fclose(fp3);
rm_file(elev_data[2]);
        }
        }
else if(nk == 3){
/* 4-tiles */
cp_from_cd(elev_data[0]);
unzip(elev_data[0]);
```

```
cp_from_cd(elev_data[1]);
unzip(elev_data[1]);
cp_from_cd(elev_data[2]);
unzip(elev_data[2]);
cp_from_cd(elev_data[3]);
unzip(elev_data[3]);
if(!(fp1=fopen(elev_data[0],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[0]);
exit(0);
        }
if(!(fp2=fopen(elev_data[1],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[1]);
exit(0);
        }
if(!(fp3=fopen(elev_data[2],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[2]);
exit(0);
        }
if(!(fp4=fopen(elev_data[3],"r"))){
printf("USGS DEM file %s unavailable ... exiting ...\n",elev_data[3]);
exit(0);
        }
for(j=0;j<jmax1;j++)
lat_o1[j]=dlat0[0]+
center*(float)j; /* top tile */
for(j=0;j<jmax2;j++)
lat_o2[j]=dlat0[2]+
center*(float)j;/*bottom tile */
for(i=0;i<2*imax1;i++)
lon_o1[i]=dlon0[0]+
center*(float)i; /* top tile */
for(i=0;i<2*imax2;i++)
lon_o2[i]=dlon0[2]+
center*(float)i;/*bottom tile */
j_lat=-1;
j=jmax1-1;
while(j>=1){
for(i=0;i<imax1;i++){
fread(&hi[i],
sizeof(signed short),1,fp1);
if(hi[i] == -9999)
hi[i]=0;
        }
for(i=imax1;i<2*imax1;i++){
fread(&hi[i],
sizeof(signed short),1,fp2);
if(hi[i] == -9999)
hi[i]=0;
```

```
        }
get_data1(xminlat,xmaxlat,
xminlon,xmaxlon,0,2*imax1,
lat_o1[j],lat_o1[j-1],&index_r);
if(index_r == 2)
goto label8310;
j--;
        }

label8310:
fclose(fp1);
rm_file(elev_data[0]);
fclose(fp2);
rm_file(elev_data[1]);

j=jmax2-1;
while(j>=1){
for(i=0;i<imax2;i++){
fread(&hi2[i],sizeof(signed short),1,fp3);
if(hi2[i] == -9999)
hi2[i] = 0;
        }
for(i=imax2;i<2*imax2;i++){
fread(&hi2[i],sizeof(signed short),1,fp4);
if(hi2[i] == -9999)
hi2[i] = 0;
        }
if(index_r == 2)
goto label8410;
j--;
        }

label8410:
fclose(fp3);
rm_file(elev_data[2]);
fclose(fp4);
rm_file(elev_data[3]);
        }
        }
/* reverse the order of the elevation data
in the y-direction */
jj=-1;
j=j_lat;
while(j >= 0){
jj++;
for(i=0;i<i_lon;i++){
z[i][jj]=hi_2[i][j];
lat[jj]=lat_o_2[j];
```

```
lon[i]=lon_o_2[i];
        }
j--;
        }

/* interpolation of z to model lat | lon */
zinterp(i_lon,j_lat,lat,lon);
fpout=fopen("terrain_data","w");
fprintf(fpout," %11.6f  %9.6f
%d %d   %6.1f\n",origlon,lato,ip1,jp1,gdis);
for(i=0;i<ip1;i++){
for(j=0;j<jp1;j++){
if(j==jp1-1)
fprintf(fpout," %4d\n",(int)
(zs[i][j]+0.5));
else
fprintf(fpout," %4d",(int)
(zs[i][j]+0.5));
        }
        }
fclose(fpout);

fpcheck=fopen("terrain_data","r");
fscanf(fpcheck,"%f %f
%d %d %f",&longi,&lati,&numx,&numy,&gridsp);

min_val=99999;
max_val=-99999;
for(i=0;i<ip1;i++){
for(j=0;j<jp1;j++){
fscanf(fpcheck,"%d",&val);
if(val<min_val)min_val=val;
if(val>max_val)max_val=val;
        }
        }
fclose(fpcheck);
printf("\ninterpolated min=%dm
interpolated max=%dm\n",min_val,max_val);

        }
void zinterp(int i_l,int j_l,float lat[],float lon[])
        {
int i,j,m,n;
float x,y;
float x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4;
float zd,zdd,yd,ydd;

int found;
```

```
float biggest=-99999.,smallest=99999.;
float big=-99999.,small=99999.;

i=-1;
j=-1;
for(j=0;j<jp1;j++){
y=lat_b[j];
for(i=0;i<ip1;i++){
x=lon_b[i][j];
n=-1;
do{
n++;
m=-1;
do{
m++;
if(lat_b[j]>=
lat[n] && lat_b[j] <
lat[n+1] &&
lon_b[i][j]>=
lon[m] && lon_b[i][j]<lon[m+1]){
if(z[m][n]<smallest)
smallest=z[m][n];
if(z[m][n]>biggest)
biggest=z[m][n];

x1=lon[m];
y1=lat[n];
x2=lon[m+1];
y2=lat[n];
x3=lon[m];
y3=lat[n+1];
x4=lon[m+1];
y4=lat[n+1];
z1=z[m][n];
z2=z[m+1][n];
z3=z[m][n+1];
z4=z[m+1][n+1];

yd=y1+((y2-y1)
/(x2-x1))*(x-x1);
ydd=y3+((y4-y3)
/(x4-x3))*(x-x3);

zd=z1+((z2-z1)
/(x2-x1))*(x-x1);
zdd=z3+((z4-z3)
/(x4-x3))*(x-x3);
```

```
zs[i][j]=zd+
(zdd-zd)/(ydd-yd)*(y-yd);
found=1;
        }
else
found=0;
}while(m<i_l-1 && !found);
}while(n<j_l-1 && !found);
        }
        }
        }

void get_data1(float xminlat,float xmaxlat,float xminlon,float
xmaxlon,int io,int ie,float lat_1,float lat_2,int *index_r)
        {
int i;
*index_r=1;
if(lat_1 >= xmaxlat && lat_2 < xmaxlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o1[i] <= xminlon
&& lon_o1[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
        }
else if(lon_o1[i] > xminlon
&& lon_o1[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
        }
else if(lon_o1[i] >= xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
break;
        }
        }
        }
else if(lat_1 < xmaxlat && lat_2 > xminlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o1[i] <= xminlon
```

```
&& lon_o1[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
        }
else if(lon_o1[i] > xminlon
&& lon_o1[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
        }
else if(lon_o1[i] >= xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
break;
        }
        }
        }
else if(lat_1 <= xminlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o1[i] <= xminlon &&
lon_o1[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
        }
else if(lon_o1[i] >
xminlon && lon_o1[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
        }
else if(lon_o1[i] >= xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi[i];
lon_o_2[i_lon]=lon_o1[i];
*index_r=2;
break;
        }
        }
        }
        }
```

```
void get_data2(float xminlat,float xmaxlat,float xminlon,float
xmaxlon,int io,int ie,float lat_1,float lat_2,int *index_r)
        {
*index_r=1;

if(lat_1 >= xmaxlat && lat_2 < xmaxlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o2[i] <= xminlon &&
lon_o2[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >
xminlon && lon_o2[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >= xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
break;
        }
        }
        }
else if(lat_1 < xmaxlat && lat_2 > xminlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o2[i] <= xminlon &&
lon_o2[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >
xminlon && lon_o2[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >= xmaxlon){
```

```
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
break;
        }
        }
        }
else if(lat_1 < xmaxlat && lat_2 > xminlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o2[i] <= xminlon &&
lon_o2[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >
xminlon && lon_o2[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >= xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
break;
        }
        }
        }
else if(lat_1 <= xminlat){
j_lat+=1;
lat_o_2[j_lat]=lat_1;
i_lon=-1;
for(i=io;i<ie-1;i++){
if(lon_o2[i] <= xminlon &&
lon_o2[i+1] > xminlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
        }
else if(lon_o2[i] >
xminlon && lon_o2[i+1] < xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
```

```
        }
else if(lon_o2[i] >= xmaxlon){
i_lon+=1;
hi_2[i_lon][j_lat]=hi2[i];
lon_o_2[i_lon]=lon_o2[i];
*index_r=2;
break;
        }
        }
        }
        }

void cp_from_cd(char str[])
        {
char system_call[60];
strcpy(system_call,"cp ");
strcat(system_call,"/cdrom/990701_1519/");
strcat(system_call,str);
strcat(system_call,".gz");
strcat(system_call," .");
system(system_call);
        }

void unzip(char str[])

        {
char system_call[60];
strcpy(system_call,"gzip -d ");
strcat(system_call,str);
strcat(system_call,".gz");
system(system_call);
        }

void rm_file(char str[])

        {
char system_call[60];
strcpy(system_call,"rm -f ");
strcat(system_call,str);
system(system_call);
        }
```

Distribution

	Copies
NASA MARSHALL SPACE FLT CTR ATMOSPHERIC SCIENCES DIV E501 ATTN DR FICHTL HUNTSVILLE AL 35802	1
NASA SPACE FLT CTR ATMOSPHERIC SCIENCES DIV CODE ED 41 1 HUNTSVILLE AL 35812	1
US ARMY MISSILE CMND AMSMI RD AC AD ATTN DR PETERSON REDSTONE ARSENAL AL 35898-5242	1
US ARMY MISSILE CMND AMSMI RD AS SS ATTN MR H F ANDERSON REDSTONE ARSENAL AL 35898-5253	1
US ARMY MISSILE CMND AMSMI RD AS SS ATTN MR B WILLIAMS REDSTONE ARSENAL AL 35898-5253	1
US ARMY MISSILE CMND AMSMI RD DE SE ATTN MR GORDON LILL JR REDSTONE ARSENAL AL 35898-5245	1
US ARMY MISSILE CMND REDSTONE SCI INFO CTR AMSMI RD CS R DOC REDSTONE ARSENAL AL 35898-5241	1
US ARMY MISSILE CMND AMSMI REDSTONE ARSENAL AL 35898-5253	1
PACIFIC MISSILE TEST CTR GEOPHYSICS DIV ATTN CODE 3250 POINT MUGU CA 93042-5000	1
NAVAL OCEAN SYST CTR CODE 54 ATTN DR RICHTER SAN DIEGO CA 52152-5000	1
METEOROLOGIST IN CHARGE KWAJALEIN MISSILE RANGE PO BOX 67 APO SAN FRANCISCO CA 96555	1

DEPT OF COMMERCE CTR MOUNTAIN ADMINISTRATION SPPRT CTR LIBRARY R 51 325 S BROADWAY BOULDER CO 80303	1
DR HANS J LIEBE NTIA ITS S 3 325 S BROADWAY BOULDER CO 80303	1
NCAR LIBRARY SERIALS NATL CTR FOR ATMOS RSCH PO BOX 3000 BOULDER CO 80307-3000	1
DEPT OF COMMERCE CTR 325 S BROADWAY BOULDER CO 80303	1
HEADQUARTERS DEPT OF ARMY DAMI POI ATTN LEE PAGE WASHINGTON DC 20310-1067	1
MIL ASST FOR ENV SCI OFC OF THE UNDERSEC OF DEFNS FOR RSCH & ENGR R&AT E LS PENTAGON ROOM 3D129 WASHINGTON DC 20301-3080	1
DEAN RMD ATTN DR GOMEZ WASHINGTON DC 20314	1
US ARMY INFANTRY ATSH CD CS OR ATTN DR E DUTOIT FT BENNING GA 30905-5090	1
HQ AFWA/DNX 106 PEACEKEEPER DR STE 2N3 OFFUTT AFB NE 68113-4039	1
PHILLIPS LABORATORY PL LYP ATTN MR CHISHOLM HANSCOM AFB MA 01731-5000	1
ATMOSPHERIC SCI DIV GEOPHYISCS DIRCTRT PHILLIPS LABORATORY HANSCOM AFB MA 01731-5000	1
PHILLIPS LABORATORY PL LYP 3 HANSCOM AFB MA 01731-5000	1

US ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY
ATTN MR H COHEN
APG MD 21005-5071

US ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY AT
ATTN MR CAMPBELL
APG MD 21005-5071

US ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMXSY CR
ATTN MR MARCHET
APG MD 21005-5071

ARL CHEMICAL BIOLOGY 1
NUC EFFECTS DIV
AMSRL SL CO
APG MD 21010-5423

US ARMY MATERIEL SYST 1
ANALYSIS ACTIVITY
AMSXY
APG MD 21005-5071

ARMY RESEARCH LABORATORY 1
AMSRL D
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL OP CI SD TL
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL CI LL
ADELPHI MD 20703-1197

ARMY RESEARCH LABORATORY 1
AMSRL SS SH
ATTN DR SZTANKAY
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

ARMY RESEARCH LABORATORY 1
AMSRL IS
ATTN J GANTT
2800 POWDER MILL ROAD
ADELPHI MD 20783-1197

ARMY RESEARCH LABORATORY 1
AMSRL DD
ATTN J ROCCHIO
2800 POWDER MILL ROAD
ADELPHI MD 20783

ARMY RESEARCH LABORATORY 1
AMSRL
2800 POWDER MILL ROAD
ADELPHI MD 20783-1145

NATIONAL SECURITY AGCY W21 1
ATTN DR LONGBOTHUM
9800 SAVAGE ROAD
FT GEORGE G MEADE MD 20755-6000

US ARMY RSRC OFC 1
ATTN AMXRO GS DR BACH
PO BOX 12211
RTP NC 27009

DR JERRY DAVIS 1
NCSU
PO BOX 8208
RALEIGH NC 27650-8208

US ARMY CECRL 1
CECRL GP
ATTN DR DETSCH
HANOVER NH 03755-1290

US ARMY ARDEC 1
SMCAR IMI I BLDG 59
DOVER NJ 07806-5000

ARMY DUGWAY PROVING GRD 1
STEDP MT DA L 3
DUGWAY UT 84022-5000

ARMY DUGWAY PROVING GRD 1
STEDP MT M
ATTN MR BOWERS
DUGWAY UT 84022-5000

DEPT OF THE AIR FORCE 1
OL A 2D WEATHER SQUAD MAC
HOLLOMAN AFB NM 88330-5000

PL WE 1
KIRTLAND AFB NM 87118-6008

USAF ROME LAB TECH 1
CORRIDOR W STE 262 RL SUL
26 ELECTR PKWY BLD 106
GRIFFISS AFB NY 13441-4514

AFMC DOW 1
WRIGHT PATTERSON AFB OH 45433-5000

US ARMY FIELD ARTILLERY SCHOOL 1
ATSF TSM TA
FT SILL OK 73503-5600

US ARMY FOREIGN SCI TECH CTR CM 220 7TH STREET NE CHARLOTTESVILLE VA 22448-5000	1
NAVAL SURFACE WEAPONS CTR CODE G63 DAHLGREN VA 22448-5000	1
US ARMY OEC CSTE EFS PARK CENTER IV 4501 FORD AVE ALEXANDRIA VA 22302-1458	1
US ARMY CORPS OF ENGRS ENGR TOPOGRAPHICS LAB ETL GS LB FT BELVOIR VA 22060	1
US ARMY TOPO ENGR CTR CETEC ZC 1 FT BELVOIR VA 22060-5546	1
SCI AND TECHNOLOGY 101 RESEARCH DRIVE HAMPTON VA 23666-1340	1
US ARMY NUCLEAR CML AGCY MONA ZB BLDG 2073 SPRINGFIELD VA 22150-3198	1
USATRADOC ATCD FA FT MONROE VA 23651-5170	1
ATRC WSS R WSMR NM 88002-5502	1
ARMY RESEARCH LABORATORY AMSRL IS S INFO SCI & TECH DIR WSMR NM 88002-5501	1
ARMY RESEARCH LABORATORY AMSRL IS E INFO SCI & TECH DIR WSMR NM 88002-5501	1
ARMY RESEARCH LABORATORY AMSRL IS W INFO SCI & TECH DIR WSMR NM 88002-5501	1
DTIC 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218	1

US ARMY MISSILE CMND AMSMI REDSTONE ARSENAL AL 35898-5243	1
US ARMY DUGWAY PROVING GRD STEDP3 DUGWAY UT 84022-5000	1
USTRADOC ATCD FA FT MONROE VA 23651-5170	1
WSMR TECH LIBRARY BR STEWS IM IT WSMR NM 88002	1
ARMY RESEARCH LABORATORY AMSRL SL EW ATTN MR HEMNI INFO SCI & TECH DIR WSMR NM 88002-5501	1
ARMY RESEARCH LABORATORY AMSRL SL EW ATTN MR KIRBY INFO SCI & TECH DIR WSMR NM 88002-5501	1
Record copy	3
TOTAL	69