Complete Listing of CWP Free Program Self-Documentations

generated by GENDOCS,
a shell script by John Stockwell
Center for Wave Phenomena
Colorado School of Mines

May 27, 2010

Center for Wave Phenomena
Colorado School of Mines

Names and Short descriptions of the Codes

CWPROOT = /usr/local/cwp

```
Mains:

In CWPROOT/src/cwp/main:
* CTRLSTRIP - Strip non-graphic characters
* DOWNFORT - change Fortran programs to lower case, preserving strings
* FCAT - fast cat with 1 read per file
* ISATTY - pass on return from isatty(2)
* MAXINTS - Compute maximum and minimum sizes for integer types
* PAUSE - prompt and wait for user signal to continue
* T - time and date for non-military types
* UPFORT - change Fortran programs to upper case, preserving strings

In CWPROOT/src/par/main:
A2B - convert ascii floats to binary
A2I - convert Ascii to binary Integers
B2A - convert binary floats to ascii
CELLAUTO - Two-dimensional CELLular AUTOmata
char* sdoc[] = {
CSHOTPLOT - convert CSHOT data to files for CWP graphers
DGWAVEFORM - make Gaussian derivative waveform
DZDV - determine depth derivative with respect to the velocity ",
FARITH - File ARITHmetic -- perform simple arithmetic with binary files
FLOAT2IBM - convert native binary FLOATS to IBM tape FLOATS
FTNSTRIP - convert a file of binary data plus record delimiters created
FTNUNSTRIP - convert C binary floats to Fortran style floats
GRM - Generalized Reciprocal refraction analysis for a single layer
H2B - convert 8 bit hexidecimal floats to binary
HTI2STIFF - convert HTI parameters alpha, beta, d(V), e(V), gamma
 HUDSON - compute  effective parameters of anisotropic solids
I2A - convert binary integers to ascii
IBM2FLOAT - convert IBM tape FLOATS to native binary FLOATS
KAPERTURE - generate the k domain of a line scatterer for a seismic array
LINRORT - linearized P-P, P-S1 and P-S2 reflection coefficients

MAKEVEL - MAKE a VELocity function v(x,y,z)
MKPARFILE - convert ascii to par file format
MRAFXZWT - Multi-Resolution Analysis of a function F(X,Z) by Wavelet
PDFHISTOGRAM - generate a HISTOGRAM of the Probability Density function
PRPLOT - PRinter PLOT of 1-D arrays f(x1) from a 2-D function f(x1,x2)
```

RANDVEL3D - Add a random velocity layer (RVL) to a gridded
RAYT2DAN -- P- and SV-wave raytracing in 2D anisotropic media
RAYT2D - traveltime Tables calculated by 2D paraxial RAY tracing
RECAST - RECAST data type (convert from one data type to another)
REFREALAZIHTI - REAL AZImuthal REFL/transm coeff for HTI media
REFREALVTI - REAL REFL/transm coeff for VTI media and symmetry-axis
REGRID3 - REwrite a [ni3][ni2][ni1] GRID to a [no3][no2][no1] 3-D grid
RESAMP - RESAMPle the 1st dimension of a 2-dimensional function f(x1,x2)
SMOOTH2 --- SMOOTH a uniformly sampled 2d array of data, within a user-
SMOOTH3D - 3D grid velocity SMOOTHing by the damped least squares
SMOOTHINT2 --- SMOOTH non-uniformly sampled INTerfaces, via the damped
STIFF2VEL - Transforms 2D elastic stiffnesses to (vp,vs,epsilon,delta)
SUBSET - select a SUBSET of the samples from a 3-dimensional file
SWAPBYTES - SWAP the BYTES of various data types
THOM2HTI - Convert Thompson parameters V_p0, V_s0, eps, gamma,
THOM2STIFF - convert Thomsen's parameters into (density normalized)
TRANSP3D - TRANSPose an n1 by n2 by n3 element matrix
TRANSP - TRANSPose an n1 by n2 element matrix
TVNMOQC - Check tnmo-vnmo pairs; create t-v column files
UNIF2ANISO - generate a 2-D UNIFormly sampled profile of elastic
UNIF2 - generate a 2-D UNIFormly sampled velocity profile from a layered
UNISAM2 - UNIformly SAMple a 2-D function f(x1,x2)
UNISAM - UNIformly SAMple a function y(x) specified as x,y pairs
UTMCONV - CONVert longitude and latitude to UTM, and vice versa
VEL2STIFF - Transforms VELocities, densities, and Thomsen or Sayers
VELCONV - VELocity CONVersion
VELPERTAN - Velocity PERTerbation analysis in ANisotropic media to
VELPERT - estimate velocity parameter perturbation from covariance

VTLVZ -- Velocity as function of Time for Linear V(Z);
WKBJ - Compute WKBJ ray theoretic parameters, via finite differencing
XY2Z - converts (X,Y)-pairs to spike Z values on a uniform grid
Z2XYZ - convert binary floats representing Z-values to ascii

In CWPROOT/src/psplot/main:
PSBBOX - change BoundingBOX of existing PostScript file
PSCONTOUR - PostScript CONTOURing of a two-dimensional function f(x1,x2)
PSCUBE - PostScript image plot with Legend of a data CUBE
PSCCONTOUR - PostScript Contour plot of a data CUBE
PSEPSI - add an EPSI formatted preview bitmap to an EPS file
PSGRAPH - PostScript GRAPHer
PSIMAGE - PostScript IMAGE plot of a uniformly-sampled function f(x1,x2)
PSLABEL - output PostScript file consisting of a single TEXT string

```
PSMANAGER - printer MANAGER for HP 4MV and HP 5Si Mx Laserjet
PSMERGE - MERGE PostScript files
PSMOVIE - PostScript MOVIE plot of a uniformly-sampled function f(x1,x2,x3)
PSWIGB - PostScript WIGgle-trace plot of f(x1,x2) via Bitmap
PSWIGP - PostScript WIGgle-trace plot of f(x1,x2) via Polygons

In CWPROOT/src/xplot/main:
* LCMAP - List Color Map of root window of default screen
* LPROP - List PROPerties of root window of default screen of display
* SCMAP - set default standard color map (RGB_DEFAULT_MAP)
XCONTOUR - X CONTOUR plot of f(x1,x2) via vector plot call
* XESPB - X windows display of Encapsulated PostScript as a single Bitmap
* XEPSP - X windows display of Encapsulated PostScript
XIMAGE - X IMAGE plot of a uniformly-sampled function f(x1,x2)
XPICKER - X wiggle-trace plot of f(x1,x2) via Bitmap with PICKing
* XPSP - Display conforming PostScript in an X-window
XWIGB - X WIGgle-trace plot of f(x1,x2) via Bitmap

In CWPROOT/src/Xtcwp/main:
XGRAPH - X GRAPHer
XGRAPH - X GRAPHer
XMOVIE - image one or more frames of a uniformly sampled function f(x1,x2)
XRECTS - plot rectangles on a two-dimensional grid

In CWPROOT/src/Xmcwp/main:
* FFTLAB - Motif-X based graphical 1D Fourier Transform

In CWPROOT/src/su/graphics/psplot:
SUPSCONTOUR - PostScript CONTOUR plot of a segy data set
SUPSCUBE - PostScript CUBE plot of a segy data set
SUPSCUBECONTOUR - PostScript CUBE plot of a segy data set
SUPSGRAPH - PostScript GRAPH plot of a segy data set
SUPSIMAGE - PostScript IMAGE plot of a segy data set
SUPSMAX - PostScript of the MAX, min, or absolute max value on each trace
SUPSMOVIE - PostScript MOVIE plot of a segy data set
SUPSWIGB - PostScript Bit-mapped WIGgle plot of a segy data set
SUPSWIGP - PostScript Polygon-filled WIGgle plot of a segy data set

In CWPROOT/src/su/main/amplitudes:
SUCENTSAMP - CENTRoid SAMPle seismic traces
SUDIPDIVCOR - Dip-dependent Divergence (spreading) correction
SUDIVCOR - Divergence (spreading) correction
SUGAIN - apply various types of gain
```

```
SUNAN - remove NaNs & Infs from the input stream
SUNORMALIZE - Trace balancing by rms, max, or median           ",
SUPGC   -   Programmed Gain Control--apply agc like function
SUWEIGHT - weight traces by header parameter, such as offset
SUZERO -- zero-out (or set constant) data within a time window

In CWPROOT/src/su/main/attributes_parameter_estimation:
SUATTRIBUTES - instantaneous trace ATTRIBUTES
SUBACKUS - calculate Thomsen anisotropy parameters from
SUBACKUSH - calculate Thomsen anisotropy parameters from

SUHISTOGRAM - create histogram of input amplitudes
SULPRIME - find appropriate Backus average length for
SUMAX - get trace by trace local/global maxima, minima, or absolute maximum
SUMEAN - get the mean values of data traces ",
SUQUANTILE - display some quantiles or ranks of a data set

In CWPROOT/src/su/main/convolution_correlation:
SUACOR - auto-correlation
SUACORFRAC -- general FRACtional Auto-CORrelation/convolution
SUCONV - convolution with user-supplied filter
SUREFCON -  Convolution of user-supplied Forward and Reverse
SUXCOR - correlation with user-supplied filter

In CWPROOT/src/su/main/data_compression:
SUPACK1 - pack segy trace data into chars
SUPACK2 - pack segy trace data into 2 byte shorts
SUUNPACK1 - unpack segy trace data from chars to floats
SUUNPACK2 - unpack segy trace data from shorts to floats

In CWPROOT/src/su/main/data_conversion:
DT1TOSU - Convert ground-penetrating radar data in the Sensors & Software
LAS2SU - convert las2 format well log curves to su traces
SEGYCLEAN - zero out unassigned portion of header
SEGYHDRMOD - replace the text header on a SEGY file
SEGYHDRS - make SEG-Y ascii and binary headers for segywrite
SEGYREAD - read an SEG-Y tape
SEGYWRITE - write an SEG-Y tape
SETBHED - SET the fields in a SEGY Binary tape HEaDer file, as would be
SUASCII - print non zero header values and data in various formats
SUINTVEL - convert stacking velocity model to interval velocity model
SUOLDTONEW - convert existing su data to xdr format
SUSTKVEL - convert constant dip layer interval velocity model to the
```

SUSWAPBYTES - SWAP the BYTES in SU data to convert data from big endian

In CWPROOT/src/su/main/datuming:
SUDATUMFD - 2D zero-offset Finite Difference acoustic wave-equation
SUDATUMK2DR - Kirchhoff datuming of receivers for 2D prestack data
SUDATUMK2DS - Kirchhoff datuming of sources for 2D prestack data

In CWPROOT/src/su/main/decon_shaping:
SUCDDECON - DECONvolution with user-supplied filter by straightforward
SUFXDECON - random noise attenuation by FX-DECONvolution
SUPEF - Wiener predictive error filtering
SUPHIDECON - PHase Inversion Deconvolution
SUSHAPE - Wiener shaping filter

In CWPROOT/src/su/main/dip_moveout:
SUDMOFK - DMO via F-K domain (log-stretch) method for common-offset gathers
SUDMOFKCW - converted-wave DMO via F-K domain (log-stretch) method for
SUDMOTIVZ - DMO for Transeversely Isotropic V(Z) media for common-offset
SUDMOTX - DMO via T-X domain (Kirchhoff) method for common-offset gathers
SUDMOVZ - DMO for V(Z) media for common-offset gathers
SUTIHALEDMO - TI Hale Dip MoveOut (based on Hale's PhD thesis)

In CWPROOT/src/su/main/filters:
SUBFILT - apply Butterworth bandpass filter
SUCCFILT - FX domain Correlation Coefficient FILTER
SUDIPFILT - DIP--or better--SLOPE Filter in f-k domain
SUFILTER - applies a zero-phase, sine-squared tapered filter
SUFRAC -- take general (fractional) time derivative or integral of
SUFWATRIM - FX domain Alpha TRIM
SUK1K2FILTER - symmetric box-like K-domain filter defined by the
SUKFILTER - radially symmetric K-domain, sin^2-tapered, polygonal
SUKFRAC - apply FRACtional powers of i|k| to data, with phase shift
SULFAF - Low frequency array forming ",
SUMEDIAN - MEDIAN filter about a user-defined polygonal curve with
SUPHASE - PHASE manipulation by linear transformation
SUSMGAUSS2 --- SMOOTH a uniformly sampled 2d array of velocities
SUTVBAND - time-variant bandpass filter (sine-squared taper)

In CWPROOT/src/su/main/headers:
BHEDTOPAR - convert a Binary tape HEaDer file to PAR file format
SU3DCHART - plot x-midpoints vs. y-midpoints for 3-D data
SUABSHW - replace header key word by its absolute value
SUADDHEAD - put headers on bare traces and set the tracl and ns fields

```
SUAZIMUTH - compute trace AZIMUTH, offset, and midpoint coordinates
SUCHART - prepare data for x vs. y plot
SUCHW - Change Header Word using one or two header word fields
SUCLIPHEAD - Clip header values
SUCOUNTKEY - COUNT the number of unique values for a given KEYword.
SUDUMPTRACE - print selected header values and data.
SUEDIT - examine segy diskfiles and edit headers
SUGETHW - sugethw writes the values of the selected key words
SUHTMATH - do unary arithmetic operation on segy traces with
SUKEYCOUNT - sukeycount writes a count of a selected key
SULCTHW - Linear Coordinate Transformation of Header Words
SULHEAD - Load information from an ascii column file into HEADERS
SUPASTE - paste existing SEGY headers on existing data
SURANGE - get max and min values for non-zero header entries
SUSEHW - Set the value the Header Word denoting trace number within
SUSHW - Set one or more Header Words using trace number, mod and
SUSTRIP - remove the SEGY headers from the traces
SUTRCOUNT - SU program to count the TRaces in infile
SUUTM - UTM projection of longitude and latitude in SU trace headers
SUXEDIT - examine segy diskfiles and edit headers

In CWPROOT/src/su/main/interp_extrap:
SUINTERP - interpolate traces using automatic event picking
SUINTERPFOWLER - interpolate output image from constant velocity panels
SUOCEXT - smaller Offset EXTrapolation via Offset Continuation

In CWPROOT/src/su/main/migration_inversion:
SUGAZMIGQ - SU version of Jeno GAZDAG's phase-shift migration
SUINVXZCO - Seismic INVersion of Common Offset data for a smooth
SUINVZCO3D - Seismic INVersion of Common Offset data with V(Z) velocity
SUKDMIG2D - Kirchhoff Depth Migration of 2D poststack/prestack data
SUKDMIG3D - Kirchhoff Depth Migration of 3D poststack/prestack data
SUKTMIG2D - prestack time migration of a common-offset section with
SUMIGFD - 45-90 degree Finite difference depth migration for
SUMIGFFD - Fourier finite difference depth migration for
SUMIGGBZOAN - MIGration via Gaussian beams ANisotropic media (P-wave)
SUMIGGBZO - MIGration via Gaussian Beams of Zero-Offset SU data
SUMIGPREFD --- The 2-D prestack common-shot 45-90 degree
SUMIGPREFFD - The 2-D prestack common-shot Fourier finite-difference
SUMIGPREPSPI --- The 2-D PREstack commom-shot Phase-Shift-Plus
SUMIGPRESP - The 2-D prestack common-shot split-step Fourier ",
SUMIGPS - MIGration by Phase Shift with turning rays
SUMIGPSPI - Gazdag's phase-shift plus interpolation depth migration
```

```
SUMIGPSTI - MIGration by Phase Shift for TI media with turning rays
SUMIGSPLIT - Split-step depth migration for zero-offset data.
SUMIGTK - MIGration via T-K domain method for common-midpoint stacked data
SUMIGTOPO2D - Kirchhoff Depth Migration of 2D postack/prestack data
SUSTOLT - Stolt migration for stacked data or common-offset gathers
SUTIFOWLER   VTI constant velocity prestack time migration

In CWPROOT/src/su/main/multicomponent:
SUALFORD - trace by trace Alford Rotation of shear wave data volumes
SUEIPOFI - EIgenimage (SVD) based POlarization FIlter for
SUHROT - Horizontal ROTation of three-component data
SULTT - trace by trace, sample by sample, rotation of shear wave data
SUPOFILT - POlarization FILTer for three-component data
SUPOLAR - POLarization analysis of three-component data

In CWPROOT/src/su/main/noise:
SUADDNOISE - add noise to traces
SUHARLAN - signal-noise separation by the invertible linear
SUJITTER - Add random time shifts to seismic traces

In CWPROOT/src/su/main/operations:
SUFLIP - flip a data set in various ways
SUFWMIX -  FX domain multidimensional Weighted Mix
SUMIX - compute weighted moving average (trace MIX) on a panel
SUOP2 - do a binary operation on two data sets
SUOP - do unary arithmetic operation on segys
SUPERMUTE - permute or transpose a 3d datacube
SUVCAT -  append one data set to another, with or without an  ",
SUVLENGTH - Adjust variable length traces to common length

In CWPROOT/src/su/main/picking:
SUFBPICKW - First break auto picker
SUFNZERO - get Time of First Non-ZERO sample by trace
SUPICKAMP - pick amplitudes within user defined and resampled window

In CWPROOT/src/su/main/stacking:
SUCVS4FOWLER --compute constant velocity stacks for input to Fowler codes
SUDIVSTACK -  Diversity Stacking using either average power or peak
SUPWS - Phase stack or phase-weighted stack (PWS) of adjacent traces
SURECIP - sum opposing offsets in prepared data (see below)
SUSTACK - stack adjacent traces having the same key header word

In CWPROOT/src/su/main/statics:
```

```
SUADDSTATICS - ADD random STATICS on seismic data
SURANDSTAT - Add RANDom time shifts STATIC errors to seismic traces
SURESSTAT - Surface consistent source and receiver statics calculation
SUSTATIC - Elevation static corrections, apply corrections from
SUSTATICRRS - Elevation STATIC corrections, apply corrections from

In CWPROOT/src/su/main/stretching_moveout_resamp:
SUILOG -- time axis inverse log-stretch of seismic traces
SULOG -- time axis log-stretch of seismic traces
SUNMO - NMO for an arbitrary velocity function of time and CDP
SUREDUCE - convert traces to display in reduced time ",
SURESAMP - Resample in time
SUSHIFT - shifted/windowed traces in time
SUTSQ -- time axis time-squared stretch of seismic traces
SUTTOZ - resample from time to depth
SUZTOT - resample from depth to time

In CWPROOT/src/su/main/supromax:
SUGET  - Connect SU program to file descriptor for input stream.
SUPUT - Connect SU program to file descriptor for output stream.

In CWPROOT/src/su/main/synthetics_waveforms_testpatterns:

SUEA2DF - SU version of (an)elastic anisotropic 2D finite difference
SUFCTANISMOD - Flux-Corrected Transport correction applied to the 2D
SUFDMOD1 - Finite difference modelling (1-D 1rst order) for the
SUFDMOD2 - Finite-Difference MODeling (2nd order) for acoustic wave equation
SUFDMOD2_PML - Finite-Difference MODeling (2nd order) for acoustic wave
SUGASSMAN - Model reflectivity change with rock/fluid properties
SUGOUPILLAUD - calculate 1D impulse response of
SUGOUPILLAUDPO - calculate Primaries-Only impulse response of a lossless
SUIMP2D - generate shot records for a line scatterer
SUIMP3D - generate inplane shot records for a point
SUIMPEDANCE - Convert reflection coefficients to impedances.
SUKDSYN2D - Kirchhoff Depth SYNthesis of 2D seismic data from a
SUNHMOSPIKE - generates SPIKE test data set with a choice of several
SUNULL - create null (all zeroes) traces
SUPLANE - create common offset data file with up to 3 planes
SURANDSPIKE - make a small data set of RANDom SPIKEs
SUSPIKE - make a small spike data set
SUSYNCZ - SYNthetic seismograms for piecewise constant V(Z) function
SUSYNLV - SYNthetic seismograms for Linear Velocity function
SUSYNLVCW - SYNthetic seismograms for Linear Velocity function
```

```
SUSYNLVFTI - SYNthetic seismograms for Linear Velocity function in a
SUSYNVXZ - SYNthetic seismograms of common offset V(X,Z) media via
SUSYNVXZCS - SYNthetic seismograms of common shot in V(X,Z) media via
SUVIBRO - Generates a Vibroseis sweep (linear, linear-segment,
SUWAVEFORM - generate a seismic wavelet

In CWPROOT/src/su/main/tapering:
SUGAUSSTAPER - Multiply traces with gaussian taper
SURAMP - Linearly taper the start and/or end of traces to zero.
SUTAPER - Taper the edge traces of a data panel to zero.
SUTXTAPER - TAPER in (X,T) the edges of a data panel to zero.

In CWPROOT/src/su/main/transforms:
SUAMP - output amp, phase, real or imag trace from
SUCCWT - Complex continuous wavelet transform of seismic traces
SUCEPSTRUM - Compute the CEPSTRUM of a seismic trace or compute the
SUCWT - generates Continous Wavelet Transform amplitude, regularity
SUFFT - fft real time traces to complex frequency traces
SUGABOR -  Outputs a time-frequency representation of seismic data via
SUHILB - Hilbert transform
SUIFFT - fft complex frequency traces to real time traces
SUPHASEVEL - Multi-mode PHASE VELocity dispersion map computed
SURADON - compute forward or reverse Radon transform or remove multiples
SUSLOWFT - Fourier Transforms by a (SLOW) DFT algorithm (Not an FFT)
SUSLOWIFT - Fourier Transforms by (SLOW) DFT algorithm (Not an FFT)
SUSPECFK - F-K Fourier SPECtrum of data set
SUSPECFX - Fourier SPECtrum (T -> F) of traces
SUSPECK1K2 - 2D (K1,K2) Fourier SPECtrum of (x1,x2) data set
SUTAUP - forwared and inverse T-X and F-K global slant stacks
SUWFFT - Weighted amplitude FFT with spectrum flattening 0->Nyquist

In CWPROOT/src/su/main/velocity_analysis:
SURELANAN - REsiduaL-moveout semblance ANalysis for ANisotropic media
SURELAN - compute residual-moveout semblance for cdp gathers based
 SUTIVEL -  SU Transversely Isotropic velocity table builder
SUVEL2DF - compute stacking VELocity semblance for a single time in
SUVELAN - compute stacking velocity semblance for cdp gathers
SUVELAN_NCCS - compute stacking VELocity panel for cdp gathers
SUVELAN_NSEL - compute stacking VELocity panel for cdp gathers
SUVELAN_UCCS - compute stacking VELocity panel for cdp gathers
SUVELAN_USEL - compute stacking velocity panel for cdp gathers

In CWPROOT/src/su/main/well_logs:
```

```
SUWELLRF - convert WELL log depth, velocity, density data into a

In CWPROOT/src/su/main/windowing_sorting_muting:
SUCOMMAND - pipe traces having the same key header word to command
SUGETGTHR - Gets su files from a directory and put them
SUGPRFB - SU program to remove First Breaks from GPR data
SUKILL - zero out traces
SUMIXGATHERS - mix two gathers
SUMUTE - MUTE above (or below) a user-defined polygonal curve with ",
SUPUTGTHR - split the stdout flow to gathers on the bases of given
SUSORT - sort on any segy header keywords
SUSORTY - make a small 2-D common shot off-end
SUSPLIT - Split traces into different output files by keyword value
SUWIND - window traces by key word
SUWINDPOLY - WINDow data to extract traces on or within a respective

In CWPROOT/src/su/graphics/psplot:
SUPSCONTOUR - PostScript CONTOUR plot of a segy data set
SUPSCUBE - PostScript CUBE plot of a segy data set
SUPSCUBECONTOUR - PostScript CUBE plot of a segy data set
SUPSGRAPH - PostScript GRAPH plot of a segy data set
SUPSIMAGE - PostScript IMAGE plot of a segy data set
SUPSMAX - PostScript of the MAX, min, or absolute max value on each trace
SUPSMOVIE - PostScript MOVIE plot of a segy data set
SUPSWIGB - PostScript Bit-mapped WIGgle plot of a segy data set
SUPSWIGP - PostScript Polygon-filled WIGgle plot of a segy data set

In CWPROOT/src/su/graphics/xplot:
SUXCONTOUR - X CONTOUR plot of Seismic UNIX tracefile via vector plot call
SUXGRAPH - X-windows GRAPH plot of a segy data set
SUXIMAGE - X-windows IMAGE plot of a segy data set
SUXMAX - X-windows graph of the MAX, min, or absolute max value on
SUXMOVIE - X MOVIE plot of a 2D or 3D segy data set
SUXPICKER - X-windows  WIGgle plot PICKER of a segy data set
SUXWIGB - X-windows Bit-mapped WIGgle plot of a segy data set

In CWPROOT/src/tri/main:
GBBEAM - Gaussian beam synthetic seismograms for a sloth model
NORMRAY - dynamic ray tracing for normal incidence rays in a sloth model
TRI2UNI - convert a TRIangulated model to UNIformly sampled model
TRIMODEL - make a triangulated sloth (1/velocity^2) model
TRIRAY - dynamic RAY tracing for a TRIangulated sloth model
TRISEIS - Gaussian beam synthetic seismograms for a sloth model
```

```
UNI2TRI - convert UNIformly sampled model to a TRIangulated model

In CWPROOT/src/xtri:
SXPLOT - X Window plot a triangulated sloth function s(x1,x2)

In CWPROOT/src/tri/graphics/psplot:
SPSPLOT - plot a triangulated sloth function s(x,z) via PostScript

In CWPROOT/src/comp/dct/main:
DCTCOMP - Compression by Discrete Cosine Transform
DCTUNCOMP - Discrete Cosine Transform Uncompression
ENTROPY - compute the ENTROPY of a signal
WPTCOMP - Compression by Wavelet Packet Compression
WPTUNCOMP - Uncompress  WPT compressed data
WTCOMP - Compression by Wavelet Transform
WTUNCOMP - UNCOMPression of WT compressed data

In CWPROOT/src/comp/dwpt/1d/main:
WPC1COMP2 --- COMPress a 2D seismic section trace-by-trace using
WPC1UNCOMP2 --- UNCOMPRESS a 2D seismic section, which has been

In CWPROOT/src/comp/dwpt/2d/main:
WPCCOMPRESS --- COMPRESS a 2D section using Wavelet Packets
WPCUNCOMPRESS --- UNCOMPRESS a 2D section

Shells:

In CWPROOT/src/cwp/shell:
# ARGV - give examples of dereferencing char **argv
# COPYRIGHT - insert CSM COPYRIGHT lines at top of files in working directory
# CPALL , RCPALL - for local and remote directory tree/file transfer
# CWPFIND - look for files with patterns in CWPROOT/src/cwp/lib
# DIRTREE - show DIRectory TREE
# FILETYPE - list all files of given type
# Grep  - recursively call egrep in pwd
# NEWCASE - Changes the case of all the filenames in a directory, dir
# OVERWRITE - copy stdin to stdout after EOF
# PRECEDENCE - give table of C precedences from Kernighan and Ritchie
# REPLACE - REPLACE string1 with string2  in files
# THIS_YEAR - print the current year
# TIME_NOW - prints time in ZULU format with no spaces
# TODAYS_DATE - prints today's date in ZULU format with no spaces
# USERNAMES - get list of all login names
```

```
# VARLIST - list variables used in a Fortran program
# WEEKDAY - prints today's WEEKDAY designation
# ZAP - kill processes by name

In CWPROOT/src/par/shell:
# GENDOCS - generate complete list of selfdocs in latex form
# STRIPTOTXT -  put files from $CWPROOT/src/doc/Stripped into a new
# UPDATEDOCALL - put self-docs in ../doc/Stripped
# UPDATEDOC - put self-docs in ../doc/Stripped and ../doc/Headers
# UPDATEHEAD - update ../doc/Headers/Headers.all

In CWPROOT/src/psplot/shell:
# MERGE2 - put 2 standard size PostScript figures on one page
# MERGE4 - put 4 standard size PostScript plots on one page

In CWPROOT/src/su/shell:
# LOOKPAR - show getpar lines in SU code with defines evaluated
# MAXDIFF - find absolute maximum difference in two segy data sets
# RECIP - sum opposing (reciprocal) offsets in cdp sorted data
# RMAXDIFF - find percentage maximum difference in two segy data sets
# SUAGC - perform agc on SU data
# SUBAND - Trapezoid-like Sin squared tapered Bandpass filter via  SUFILTER
# SUDIFF, SUSUM, SUPROD, SUQUO, SUPTDIFF, SUPTSUM,
# SUDOC - get DOC listing for code
# SUENV - Instantaneous amplitude, frequency, and phase via: SUATTRIBUTES
# SUFIND - get info from self-docs
# SUFIND - get info from self-docs
# SUGENDOCS - generate complete list of selfdocs in latex form
# SUHELP - list the CWP/SU programs and shells
# SUKEYWORD -- guide to SU keywords in segy.h
# SUNAME - get name line from self-docs
# UNGLITCH - zonk outliers in data

Libs:

In CWPROOT/src/cwp/lib:
ABEL - Functions to compute the discrete ABEL transform:
AIRY - Approximate the Airy functions  Ai(x), Bi(x) and their respective
ALLOC - Allocate and free multi-dimensional arrays
ANTIALIAS - Butterworth anti-aliasing filter
AXB - Functions to solve a linear system of equations Ax=b by LU
BIGMATRIX - Functions to manipulate 2-dimensional matrices that are too big
BUTTERWORTH - Functions to design and apply Butterworth filters:
```

COMPLEX - Functions to manipulate complex numbers
COMPLEXD - Functions to manipulate double-precision complex numbers
COMPLEXF - Subroutines to perform operations on complex numbers.
COMPLEXFD - Subroutines to perform operations on double complex numbers.
CONVOLUTION - Compute z = x convolved with y
CUBICSPLINE - Functions to compute CUBIC SPLINE interpolation coefficients
DBLAS - Double precision Basic Linear Algebra subroutines
DGE - Double precision Gaussian Elimination matrix subroutines adapted
PFAFFT - Functions to perform Prime Factor (PFA) FFT's, in place
*CWP_Exit - exit subroutine for CWP/SU codes
FRANNOR - functions to generate a pseudo-random float normally distributed
FRANUNI - Functions to generate a pseudo-random float uniformly distributed
HANKEL - Functions to compute discrete Hankel transforms
Hartley - routines for fast Hartley transform
HILBERT - Compute Hilbert transform y of x
HOLBERG1D - Compute coefficients of Holberg's 1st derivative filter
INTCUB - evaluate y(x), y'(x), y''(x), ... via piecewise cubic interpolation
INTL2B - bilinear interpolation of a 2-D array of bytes
INTLIN - evaluate y(x) via linear interpolation of y(x[0]), y(x[1]), ...
INTLINC - evaluate complex y(x) via linear interpolation of y(x[0]), y(x[1]), ...
INTLIRR2B - bilinear interpolation of a 2-D array of bytes
INTSINC8 - Functions to interpolate uniformly-sampled data via 8-coeff. sinc
INTTABLE8 - Interpolation of a uniformly-sampled complex function y(x)
LINEAR_REGRESSION - Compute linear regression of (y1,y2,...,yn) against
maxmin - subroutines that pertain to maximum and minimum values
MKDIFF - Make an n-th order DIFFerentiator via Taylor's series method.
MKHDIFF - Compute filter approximating the bandlimited HalF-DIFFerentiator.
MKSINC - Compute least-squares optimal sinc interpolation coefficients.
MNEWT - Solve non-linear system of equations f(x) = 0 via Newton's method
ORTHPOLYNOMIALS - compute ORTHogonal POLYNOMIALS
PFAFFT - Functions to perform Prime Factor (PFA) FFT's, in place
POLAR - Functions to map data in rectangular coordinates to polar and vise versa
PRINTERPLOT - Functions to make a printer plot of a 1-dimensional array
QUEST - Functions to ESTimate Quantiles:
RESSINC8 - Functions to resample uniformly-sampled data via 8-coefficient sinc
RFWTVA - Rasterize a Float array as Wiggle-Trace-Variable-Area.
RFWTVAINT - Rasterize a Float array as Wiggle-Trace-Variable-Area, with
SBLAS - Single precision Basic Linear Algebra Subroutines
SCAXIS - compute a readable scale for use in plotting axes
SGA - Single precision general matrix functions adapted from LINPACK FORTRAN:
SHFS8R - Shift a uniformly-sampled real-valued function y(x) via
SINC - Return SINC(x) for as floats or as doubles
SORT - Functions to sort arrays of data or arrays of indices

SQR - Single precision QR decomposition functions adapted from LINPACK FORTRAN:
STOEP - Functions to solve a symmetric Toeplitz linear system of equations
STRSTUFF -- STRing manuplation subs
SWAPBYTE - Functions to SWAP the BYTE order of binary data
SYMMEIGEN - Functions solving the eigenvalue problem for symmetric matrices
TEMPORARY_FILENAME - Creates a file name in a user-specified directory.
TRIDIAGONAL - Functions to solve tridiagonal systems of equations Tu=r for u.
VANDERMONDE - Functions to solve Vandermonde system of equations Vx=b
WAVEFORMS Subroutines to define some wavelets for modeling of seismic
WINDOW - windowing routines
wrapArray - wrap an array
XCOR - Compute z = x cross-correlated with y
XINDEX - determine index of x with respect to an array of x values
YCLIP - Clip a function y(x) defined by linear interpolation of the
YXTOXY - Compute a regularly-sampled, monotonically increasing function x(y)
ZASC - routine to translate ncharacters from ebcdic to ascii
ZEBC - routine to translate ncharacters from ascii to ebcdic

In CWPROOT/src/par/lib:
ATOPKGE - convert ascii to arithmetic and with error checking
DOCPKGE - Function to implement the CWP self-documentation facility
EALLOC - Allocate and free multi-dimensional arrays with error reports.
ERRPKGE - routines for reporting errors
FILESTAT - Functions to determine and output the type of a file from file
GETPARS - Functions to GET PARameterS from the command line. Numeric
LINCOEFF - subroutines to create linearized reflection coefficients
MINFUNC - routines to MINimize FUNCtions
MODELING - Seismic Modeling Subroutines for SUSYNLV and clones
REFANISO - Reflection coefficients for Anisotropic media
RKE - integrate a system of n-first order ordinary differential equations
SMOOTH - Functions to compute smoothing of 1-D or 2-D input data
SUBCALLS - routines for system functions with error checking
SYSCALLS - routines for SYSTEM CALLs with error checking
TAUP - Functions to perform forward and inverse taup transforms (radon or
UPWEIK - Upwind Finite Difference Eikonal Solver
VND - large out-of-core multidimensional block matrix transpose
WTLIB - Functions for wavelet transforms

In CWPROOT/src/su/lib:
FGETGTHR - get gathers from SU datafiles
fgethdr - get segy tape identification headers from the file by file pointer
FGETTR - Routines to get an SU trace from a file
FPUTGTHR - put gathers to a file

```
FPUTTR - Routines to put an SU trace to a file
HDRPKGE - routines to access the SEGY header via the hdr structure.
TABPLOT - TABPLOT selected sample points on selected trace
VALPKGE - routines to handle variables of type Value

In CWPROOT/src/psplot/lib:
BASIC - Basic C function interface to PostScript
PSAXESBOX3 -  Functions draw an axes box via PostScript, estimate bounding box
PSAXESBOX - Functions to draw PostScript axes and estimate bounding box
PSCAXESBOX - Draw an axes box for cube via PostScript
PSCONTOUR - draw contour of a two-dimensional array via PostScript
PSDRAWCURVE - Functions to draw a curve from a set of points
PSLEGENDBOX - Functions to draw PostScript axes and estimate bounding box
PSWIGGLE - draw wiggle-trace with (optional) area-fill via PostScript

In CWPROOT/src/xplot/lib:
AXESBOX - Functions to draw axes in X-windows graphics
COLORMAP - Functions to manipulate X colormaps:
DRAWCURVE - Functions to draw a curve from a set of points
IMAGE - Function for making the image in an X-windows image plot
LEGENDBOX - draw a labeled axes box for a legend (i.e. colorscale)
RUBBERBOX -  Function to draw a rubberband box in X-windows plots
WINDOW - Function to create a window in X-windows graphics
XCONTOUR - draw contour of a two-dimensional array via X vectorplot calls

In CWPROOT/src/Xtcwp/lib:
AXES - the Axes Widget
COLORMAP - Functions to manipulate X colormaps:
FX - Functions to support floating point coordinates in X
MISC - Miscellaneous X-Toolkit functions
RESCONV - general purpose resource type converters
RUBBERBOX -  Function to draw a rubberband box in X-windows plots

In CWPROOT/src/Xmcwp/lib:
RADIOBUTTONS -  convenience functions creating and using radio buttons
SAMPLES - Motif-based Graphics Functions

In CWPROOT/src/tri/lib:
CHECK - CHECK triangulated models
CIRCUM - define CIRCUMcircles for Delaunay triangulation
COLINEAR - determine if edges or vertecies are COLINEAR in triangulated
CREATE - create model, boundary edge triangles, edge face, edge vertex, add
DELETE - DELETE vertex, model, edge, or boundary edge from triangulated model
```

```
DTE - Distance to Edge
FIXEDGES - FIX or unFIX EDGES between verticies
INSIDE -  Is a vertex or point inside a circum circle, etc. of a triangulated
NEAREST - NEAREST edge or vertex in triangulated model
PROJECT - project to edge in triangulated model
READWRITE - READ or WRITE a triangulated model

In CWPROOT/src/cwputils:
CPUSEC - return cpu time (UNIX user time) in seconds
CPUTIME - return cpu time (UNIX user time) in seconds using ANSI C built-ins
WALLSEC - Functions to time processes
WALLTIME - Function to show time a process takes to run

In CWPROOT/src/comp/dct/lib:
BUFFALLOC - routines to ALLOCate/initialize and free BUFFers
DCT1 - 1D Discreet Cosine Transform Routines
DCT2 - 2D Discrete Cosine Transform Routines
DCTALLOC - ALLOCate space for transform tables for 1D DCT
GETFILTER - GET wavelet FILTER type
HUFFMAN - Routines for in memory Huffman coding/decoding
LCT1 - functions used to perform the 1D Local Cosine Transform (LCT)
LPRED - Lateral Prediction of Several Plane Waves
PENCODING - Routines to en/decode the quantized integers for lossless
QUANT - QUANTization routines
RLE - routines for in memory silence en/decoding
WAVEPACK1 - 1D wavelet packet transform
WAVEPACK2 - 2D Wavelet PACKet transform
WAVEPACK1 - 1D wavelet packet transform
WAVETRANS2 - 2D wavelet transform by tensor-product of two 1D transforms

In CWPROOT/src/comp/dct/lib:
BUFFALLOC - routines to ALLOCate/initialize and free BUFFers
DCT1 - 1D Discreet Cosine Transform Routines
DCT2 - 2D Discrete Cosine Transform Routines
DCTALLOC - ALLOCate space for transform tables for 1D DCT
GETFILTER - GET wavelet FILTER type
HUFFMAN - Routines for in memory Huffman coding/decoding
LCT1 - functions used to perform the 1D Local Cosine Transform (LCT)
LPRED - Lateral Prediction of Several Plane Waves
PENCODING - Routines to en/decode the quantized integers for lossless
QUANT - QUANTization routines
RLE - routines for in memory silence en/decoding
WAVEPACK1 - 1D wavelet packet transform
```

```
WAVEPACK2 - 2D Wavelet PACKet transform
WAVEPACK1 - 1D wavelet packet transform
WAVETRANS2 - 2D wavelet transform by tensor-product of two 1D transforms

In CWPROOT/src/comp/dwpt/1d/lib:
WBUFFALLOC -  routines to allocate/initialize and free buffers in wavelet
WPC1 - routines for compress a single seismic trace using wavelet packets
WPC1CODING - routines for encoding the integer symbols in 1D WPC
wpc1Quant - quantize
WPC1TRANS - routines to perform a 1D wavelet packet transform

In CWPROOT/src/comp/dwpt/2d/lib:
WPCBUFFAL - routines to allocate/initialize and free buffers
WPC - routines for compress a 2D seismic section using wavelet packets
WPCCODING - Routines for in memory coding of the quantized coeffiecients
WPCENDEC -  Wavelet Packet Coding, Encoding and Decoding routines
WPCHUFF -Routines for in memory Huffman coding/decoding
WPCPACK2 - routine to perform a 2D wavelet packet transform
WPCQUANT - quantization routines for WPC
WPCSILENCE - routines for in memory silence en/decoding


To search on a program name fragment, type:
      suname name_fragment <CR>

For more information type: program_name <CR>

  Items labeled with an asterisk (*) are C programs  that may
  or may not have this self documentation feature.

  Items labeled with a pound sign (#) are shell  scripts that may,
  or may not have the self documentation feature.
```

Self Documentations

Mains:

```
------
 CTRLSTRIP - Strip non-graphic characters

 ctrlstrip <dirtyfile >cleanfile



------
 DOWNFORT - change Fortran programs to lower case, preserving strings

 Usage:   downfort < infile.f > outfile.f




 Credits:
   Brian Sumner  c. 1984


------
 FCAT - fast cat with 1 read per file

 Usage: fcat file1 file2 ... > file3




 Credits:
Shuki

 This program belongs to the Center for Wave Phenomena
 Colorado School of Mines
/

------
 ISATTY - pass on return from isatty(2)

 Usage:  isatty filedes

 See:   man isatty     for further information
```

```
 Credits:
CWP: Shuki

 This program belongs to the Center for Wave Phenomena
 Colorado School of Mines


------
 MAXINTS - Compute maximum and minimum sizes for integer types
(quick and dirty)

 Usage:  maxints

 Note: These results will be in limits.h on most systems




 Credits:
CWP: Jack K. Cohen

 This program belongs to the Center for Wave Phenomena
 Colorado School of Mines


------
 PAUSE - prompt and wait for user signal to continue

 Usage:   pause [optional arguments]

 Note:
 Default prompt is "press return key to continue" which is *evoked
 by calling pause with no arguments.  The word,
 "continue", is replaced by any optional arguments handed to pause.
 Thus, the command "pause do plot" will evoke the prompt,
 "press return key to do plot".



------
 T - time and date for non-military types
```

```
 Usage: t

 Credit: Jack


------
 UPFORT - change Fortran programs to upper case, preserving strings

 Usage:   upfort < infile.f > outfile.f

 Reverse of: downfort
```

```
 A2B - convert ascii floats to binary

 a2b <stdin >stdout outpar=/dev/null

 Required parameters:
  none

 Optional parameters:
  n1=2 floats per line in input file

  outpar=/dev/null output parameter file, contains the
number of lines (n=)
  other choices for outpar are: /dev/tty,
  /dev/stderr, or a name of a disk file

 Credits:
CWP: Jack K. Cohen, Dave Hale
Hans Ecke 2002: Replaced line-wise file reading via gets() with
float-wise reading via fscanf(). This makes it
much more robust: it does not impose a specific
structure on the input file.
```

```
 A2I - convert Ascii to binary Integers

 a2i <stdin >stdout outpar=/dev/tty

 Required parameters:
  none

 Optional parameters:
  n1=2 integers per line in input file

  outpar=/dev/tty output parameter file, contains the
number of lines (n=)
```

```
 B2A - convert binary floats to ascii

 b2a <stdin >stdout

 Required parameters:
  none

 Optional parameters:
  n1=2 floats per line in output file

  outpar=/dev/tty output parameter file, contains the
number of lines (n=)
                       other choices for outpar are: /dev/tty,
                       /dev/stderr, or a name of a disk file


 Credits:
CWP: Jack K. Cohen
```

```
 CELLAUTO - Two-dimensional CELLular AUTOmata

   cellauto > stdout [optional params]

 Optional Parameters:
 n1=500 output dimensions of image (n1 x n1 pixels)
 rule=30 CA rule (Wolfram classification)
  Others: 54,60,62,90,94,102,110,122,126
                       150,158,182,188,190,220,222,225,226,250
 fill=0 Don't fill image (=1 fill image)
 f0=330 fill zero values with f0
 f1=3000 fill non-zero values with f1
 ic=1 initial condition for centered unit value at t=0
               = 2 for multiple random units
 nc=20 number of random units (if ic=2)
 tc=1 random initial units at t=0 (if ic=2)
               = 2 for initial units at random (t,x)
 verbose=0 silent operation
               = 1 echos 'porosity' of the CA in bottom half of image
 seed=from_clock     random number seed (integer)

 Notes:
 This program generates a select set of Wolframs fundamental cellular
 automata. This may be useful for constructing rough, vuggy wavespeed
 profiles. The numbering scheme follows Stephen Wolfram's.

 Example:
  cellauto rule=110 ic=2 nc=100 fill=1 f1=3000 | ximage n1=500 nx=500 &

 Here we simulate a complex near surface with air-filled
 vugs in hard country rock, with smoothing applied via smooth2

  cellauto rule=110 ic=2 nc=100 fill=1 f1=3000 n1=500 |
  smooth2 n1=500 n2=500 r1=5 r2=5 > vfile.bin


 Credits:
UHouston: Chris Liner

 Trace header fields accessed:  ns
 Trace header fields modified:  ns and delrt
```

```
char* sdoc[] = {

   CPFTREND - generate picks of the Cumulate Probability Function

 Required parameters:
  ix=      - column containing X variable
  iy=      - column containing Y variable
  min_x=   - minimum X bin
  max_x=   - maximum X bin
  min_y=   - minimum Y bin
  max_y=   - maximum Y bin

 Optional parameters:
  nx=100    - number of X bins
  ny=100    - number of Y bins
  logx=0   - =1 use logarithmic scale for X axis
  logy=0   - =1 use logarithmic scale for Y axis
  ir=       - column containing reject variable
  rmin=     - reject values below rmin
  rmax=     - reject values above rmax
              NOTE: only one, rmin or rmax, may be used
 NOTES:
  cpftrend makes picks on the 2D cumulate representing the
  probability density function of the input data.

   Commandline options allow selecting any of several normalizations
   to apply to the distributions.


  cpftrend(1) makes picks on the 2D cumulate representing the
  probability density function of the input data.

   Commandline options allow selecting any of several normalizations
   to apply to the distributions.

 Credits:  Reginald H. Beardsley                      rhb@acm.org
     Copyright 2006 Exploration Software Consultants Inc.
```

```
 CSHOTPLOT - convert CSHOT data to files for CWP graphers

 cshotplot <cshot1plot [optional parameter file]

 Required parameters:
  none

 Optional parameter:
   outpar=/dev/tty output parameter file, contains:
number of plots (n2=)
points in each plot (n1=)
colors for plots (linecolor=)
```

```
DGWAVEFORM - make Gaussian derivative waveform

dgwaveform >stdout  [optional parameters]


Optional parameters:
n=2     order of derivative (n>=1)
fpeak=35 peak frequency
nfpeak=n*n max. frequency = nfpeak * fpeak
nt=128 length of waveform
shift=0 additional time shift in s (used for plotting)
sign=1 use =-1 to change sign
verbose=0 =0 don't display diagnostic messages
              =1 display diagnostic messages
Notes:
This code computes a waveform that is the n-th order derivative of a
Gaussian. The variance of the Gaussian is specified through its peak
frequency, i.e. the frequency at which the amplitude spectrum of the
Gaussian has a maximum. nfpeak is used to compute maximum frequency,
which in turn is used to compute the sampling interval. Increasing
nfpeak gives smoother plots. In order to have a (pseudo-) causal
pulse, the program computes a time shift equal to sqrt(n)/fpeak. An
additional shift can be applied with the parameter shift. A positive
value shifts the waveform to the right.

Examples:
2-loop Ricker: dgwaveform n=1 >ricker2.su
3-loop Ricker: dgwaveform n=2 >ricker3.su
Sonic transducer pulse: dgwaveform n=10 fpeak=300 >sonic.su

To display use suxgraph. For example:
dgwaveform n=10 fpeak=300 | suxgraph style=normal &


Credits:

Werner M. Heigl, February 2007

This copyright covers parts that are not part of the original
CWP/SU: Seismic Un*x codes called by this program:

Copyright (c) 2007 by the Society of Exploration Geophysicists.
For more information, go to http://software.seg.org/2007/0004 .
```

```
 You must read and accept usage terms at:
 http://software.seg.org/disclaimer.txt before use.

 Revision history:
 Original SEG version by Werner M. Heigl, Apache E&P Technology,
 February 2007.

 Jan 2010 - subroutines deriv_n_gauss and hermite_n_polynomial moved
 to libcwp.a
/
```

```
 DZDV - determine depth derivative with respect to the velocity ",
  parameter, dz/dv,  by ratios of migrated data with the primary
  amplitude and those with the extra amplitude

 dzdv <infile afile=afile dfile=dfile>outfile [parameters]

 Required Parameters:
 infile= input of common image gathers with primary amplitude
 afile= input of common image gathers with extra amplitude
 dfile= output of imaged depths in common image gathers
 outfile= output of dz/dv at the imaged points
 nx=          number of migrated traces
 nz=         number of points in migrated traces
 dx= horizontal spacing of migrated trace
 dz=         vertical spacing of output trace
 fx=         x-coordinate of first migrated trace
 fz=         z-coordinate of first point in migrated trace
 off0=         first offset in common image gathers
 noff=         number of offsets in common image gathers
 doff=         offset increment in common image gathers
 cip=x1,z1,r1,..., cip=xn,zn,rn         description of input CIGS
x x-value of a common image point
z z-value of a common image point at zero offset
r r-parameter in a common image gather

 Optional Parameters:
 nxw, nzw=0 window widths along x- and z-directions in
which points are contributed in solving dz/dv.


 Notes:
 This program is used as part of the velocity analysis technique developed
 by Zhenyue Liu, CWP:1995.

 Author: CWP: Zhenyue Liu,  1995

 Reference:
 Liu, Z. 1995, "Migration Velocity Analysis", Ph.D. Thesis, Colorado
      School of Mines, CWP report #168.
```

```
 FARITH - File ARITHmetic -- perform simple arithmetic with binary files

 farith <infile >outfile [optional parameters]

 Optional Parameters:
 in=stdin input file
 out=stdout output file
 in2=    second input file (required for binary operations)
   if it can't be opened as a file, it might be a scalar
 n=size_of_in,  fastest dimension (used only for op=cartprod is set)
 isig=  index at which signum function acts (used only for
op=signum)
 scale= value to scale in by, used only for op=scale)
 bias= value to bias in by, used only for op=bias)

 op=noop   noop for out = in
   neg  for out = -in
   abs  for out = abs(in)
   scale for out = in *scale
   bias for out = in + bias
   exp  for out = exp(in)
   sin  for out = sin(in)
   cos  for out = cos(in)
   log  for out = log(in)
   sqrt for out = (signed) sqrt(in)
   sqr  for out = in*in
   degrad  for out = in*PI/180
   raddeg  for out = in*180/PI
   pinv  for out = (punctuated) 1 / in
   pinvsqr  for out = (punctuated) 1 /in*in
   pinvsqrt for out = (punctuated signed) 1 /sqrt(in)
   add  for out = in + in2
   sub  for out = in - in2
   mul  for out = in * in2
   div  for out = in / in2
cartprod for out = in x in2
requires: n=size_of_in, fastest dimension in output
signum for out[i] = in[i] for i< isig  and
= -in[i] for i>= isig
requires: isig=point where signum function acts
 Seismic operations:
   slowp   for  out =  1/in - 1/in2 Slowness perturbation
   slothp  for  out =  1/in^2 - 1/in2^2   Sloth perturbation
```

```
Notes:
op=sqrt takes sqrt(x) for x>=0 and -sqrt(ABS(x)) for x<0 (signed sqrt)

op=pinv takes y=1/x for x!=0,  if x=0 then y=0. (punctuated inverse)

The seismic operations assume that in and in2 are wavespeed profiles.
"Slowness" is 1/wavespeed and "sloth" is  1/wavespeed^2.
Use "suop" and "suop2" to perform unary and binary operations on
data in the SU (SEGY trace) format.

The options "pinvsq" and "pinvsqrt" are also useful for seismic
computations involving converting velocity to sloth and vice versa.

The option "cartprod" (cartesian product) requires also that the
parameter n=size_of_in be set. This will be the fastest dimension
of the rectangular array that is output.

The option "signum" causes a flip in sign for all values with index
greater than "isig" (really -1*signum(index)).

For file operations on SU format files, please use:  suop, suop2



   AUTHOR:  Dave Hale, Colorado School of Mines, 07/07/89
Zhaobo Meng added scale and cartprod, 10/01/96
Zhaobo Meng added signum, 9 May 1997
Tony Kocurko added scalar operations, August 1997
      John Stockwell added bias option 4 August 2004
```

```
 FLOAT2IBM - convert native binary FLOATS to IBM tape FLOATS

 float2ibm <stdin >stdout

 Required parameters:
  none

 Optional parameters:
 endian= byte order of your system (autodetected)
 outpar=/dev/tty output parameter file, contains the
number of values (n=)
       other choices for outpar are: /dev/tty,
       /dev/stderr, or a name of a disk file

 Notes:
 endian=1 (big endian) endian=0 (little endian) byte order
 You probably will not have to set this, as the byte order of
 your system is autodetected by the program.
 This program is usable for writing SEG Y traces with the headers
 stripped off.

 Credits:
CWP: John Stockwell, based on code by Jack K. Cohen
```

```
 FTNSTRIP - convert a file of binary data plus record delimiters created
      via Fortran to a file containing only binary values (as created via C)

 ftnstrip <ftn_data >c_data

 Caveat: this code assumes the conventional Fortran format of header
         and trailer integer containing the number of byte in the
         record.  This is overwhelmingly common, but not universal.


 Credits:
CWP: Jack K. Cohen
```

```
 FTNUNSTRIP - convert C binary floats to Fortran style floats

 ftnunstrip <stdin >stdout

 Required parameters:
  none

 Optional parameters:
  n1=1 floats per line in output file

  outpar=/dev/tty output parameter file, contains the
number of lines (n=)
  other choices for outpar are: /dev/tty,
  /dev/stderr, or a name of a disk file

 Notes: This program assumes that the record length is constant
 throughout the input and output files.
 In fortran code reading these floats, the following implied
 do loop syntax would be used:
        DO i=1,n2
                 READ (10) (someARRAY(j), j=1,n1)
        END DO
 Here n1 is the number of samples per record, n2 is the number
 of records, 10 is some default file (fort.10, for example), and
 someArray(j) is an array dimensioned to size n1


 Credits:
CWP: John Stockwell, Feb 1998,
            based on ftnstrip by: Jack K. Cohen
```

```
 GRM - Generalized Reciprocal refraction analysis for a single layer

     grm <stdin >stdout  [parameters]

 Required parameters:
 nt= Number of arrival time pairs
 dx= Geophone spacing (m)
 v0= Velocity in weathering layer (m/s)
 abtime= If set to 0, use last time as a-b, else give time (ms)

 Optional parameters:
 XY=      Value of XY if you want to override the optimum XY
  algorithm in the program. If it is not an integer multiple of
 dx, then it will be converted to the closest
 one.
XYmax   Maximum offset distance allowed when searching for
optimum XY (m)  (Default is 2*dx*10)
depthres  Size of increment in x during verical depth search(m)
  (Default is 0.5m)
 Input file:
4 column ASCII - x,y, forward time, reverse time
 Output file:
1) XYoptimum
2) apparent refractor velcocity
3) x, y, z(x,y), y-z(x,y)
z(x,y) = calculated (GRM) depth below (x y)
y-z(x,y) = GRM depth subtracted from y - absolute depth
      ..............
      4) x, y, d(x,y), y-d(x,y), (error)
d(x,y) = dip corrected depth estimate below (x,y)
y-d(x,y) = dip corrected absolute depth
error = estimated error in depth due only to the inexact
      matching of tangents to arcs in dip estimate.

      If the XY calculation is bypassed and XY specified, the values
      used will precede 1) above.  XYoptimum will still be calculated
      and displayed for reference.

 Notes:
      Uses average refactor velocity along interface.

  Credits:
```

CWP: Steven D. Sheaffer

D. Palmer, "The Generalized Reciprocal Method of Seismic Refraction Interpretation", SEG, 1982.

```
 H2B - convert 8 bit hexidecimal floats to binary

 h2b <stdin >stdout outpar=/dev/tty

 Required parameters:
  none

 Optional parameters:
  outpar=/dev/tty output parameter file, contains the
number of lines (n=)
  other choices for outpar are: /dev/tty,
  /dev/stderr, or a name of a disk file

 Note: this code may be used to recover binary data from PostScript
 bitmaps. To do this, strip away all parts of the PSfile that
 are not the actual hexidecimal bitmap and run through h2b.

 Note: that the binary file may need to be transposed using
 "transp" to appear to be the same as input data.

 Note: output will be floats with the values 0-255
```

```
 HTI2STIFF - convert HTI parameters alpha, beta, d(V), e(V), gamma
into stiffness tensor

   hti2stiff  [optional parameter] (output is to   outpar)

 Optional Parameters
 alpha=2     isotropy-plane p-wave velocity
 beta=1      fast isotropy-plan s-wave velocity
 ev=0 e(V)
 dv=0 d(V)
 gamma=0     shear-wave splitting parameter
 rho=1 density
 sign      sign of c13+c55 ( for most materials sign=1)
 outpar=/dev/tty    output parameter file

 Output:
  c_ijkl     stiffness components for x1=symmetry axis
    x3= vertical



 Credits:   Andreas Rueger, CWP Aug 01, 1996

 Reference: Andreas Rueger, P-wave reflection coefficients for
    transverse isotropy with vertical and horizontal axis
    of symmetry,  GEOPHYSICS
```

```
 HUDSON - compute  effective parameters of anisotropic solids
  using Hudson's crack theory.

 Required paramters: <none>

 Optional parameters

 vp=4.5        p-wave velocity uncracked solid
 vs=2.53       s-wave velocity uncracked solid
 rho=2.8       density
 cdens=0 crack density
 aspect=0      aspect ratio
 fill=0        gas filled cracks
               =1 water filled
 outpar        =/dev/tty   output file

Notes:
 The cracks are assumed to be vertically aligned, penny-shaped and the
 matrix is isotropic. The resulting anisotropic solid is of HTI symmetry.

 Output:
 Computes(a) stiffness elements
         (b) density normalized stiffness components
         (c) generic Thomsen parameters (vp0,vs0,eps,delta,gamma)
         (d) equivalent VTI parameters (alpha,beta,ev,dv,gv)




 AUTHOR:: Andreas Rueger, Colorado School of Mines, 10/10/96

 Additional notes:
  The routine can be easily modified to allow for any
  filling adding attenuation is not trivial

 Technical Reference:
  Hudson's theory: Hudson, 1981: Wave speed and attenuation of elastic
                                 waves in material containing cracks.
                                 Geophys. J. R. astr. Soc 64, 133-150
                   Crampin, 1984: Effective anisotropic elastic constants
                                  for waves propagating through cracked
                                  solids:
```

Geophys. J. R. astr. Soc 76, 135-145
Equivalent VTI : Rueger, 1996: Reflection coefficients in transversely
isotropic media with vertical and
horizontal axis of symmetry: Geophysics

```
 I2A - convert binary integers to ascii

 i2a <stdin >stdout

 Required parameters:
  none

 Optional parameters:
  n1=2 floats per line in output file

  outpar=/dev/tty output parameter file, contains the
number of lines (n=)


 Credits:
 Potash Corporation: c. 2008, Balazs Nemeth,  Saskatoon, Saskatchewan.
   based on b2a.c by:  CWP: Jack K. Cohen
```

```
 IBM2FLOAT - convert IBM tape FLOATS to native binary FLOATS

 ibm2float <stdin >stdout

 Required parameters:
  none

 Optional parameters:
 endian= byte order of your system (autodetected)
 outpar=/dev/tty output parameter file, contains the
number of values (n=)
       other choices for outpar are: /dev/tty,
       /dev/stderr, or a name of a disk file

 Notes:
 endian=1 (big endian) endian=0 (little endian) byte order
 You probably will not have to set this, as the byte order of
 your system is autodetected by the program.
 This program is usable for reading SEG Y files with the headers
 stripped off.

 Credits:
CWP: John Stockwell, based on code by Jack K. Cohen
```

```
KAPERTURE - generate the k domain of a line scatterer for a seismic array

kaperture [optional parameters] >stdout

Optional parameters
 x0=1000 point scatterer location
 z0=1000 point scatterer location
 nshot=1 number of shots
 sxmin=0 first shot location
 szmin=0 first shot location
 dsx=100 x-steps in shot location
 dsz=0 z-steps in shot location
 ngeo=1 number of receivers
 gxmin=0 first receiver location
 gzmin=0 first receiver location
 dgx=100 x-steps in receiver location
 dgz=0 z-steps in receiver location
 fnyq=125 Nyquist frequency  (Hz)
 fmax=125 maximum frequency  (Hz)
 fmin=5 minimum frequency  (Hz)
 nfreq=2 number of frequencies
 both=0 = 1 gives negative freqs too
 nstep=60 points on Nyquist circle
 c=5000 speed
 outpar=/dev/tty output parameter file, contains:
 xmin, xmax, ymin, ymax
 and npairs (needed for psgraph or xgraph)
 other choices for outpar are: /dev/tty,
 /dev/stderr, or a name of a disk file
Notes:
      nfreq=1 produces fmin
      nstep=0 suppresses the Nyquist circle
 and npairs
Examples:

Default case: both=0 nfreq=2

 kaperture nshot=NSHOT ngeo=NGEO nstep=NSTEP |
 psgraph n=NPAIRS,NSTEP mark=1,0 marksize=1,0 linewidth=0,1 |...
 WHERE: NPAIRS=NSHOT*NGEO

Other cases:
```

```
both=0 nfreq=NFREQ > 2
 kaperture both=0 nfreq=NFREQ nshot=NSHOT ngeo=NGEO nstep=NSTEP |
 psgraph n=NPAIRS,NSTEP mark=1,0 marksize=1,0 linewidth=0,1 |...
 WHERE: NPAIRS=NFREQ*NSHOT*NGEO

both=1 nfreq=NFREQ > 2
 kaperture both=1 nfreq=NFREQ nshot=NSHOT ngeo=NGEO nstep=NSTEP |
 psgraph n=NPAIRS,NSTEP mark=1,0 marksize=1,0 linewidth=0,1 |...
  WHERE: NPAIRS=NFREQ*NSHOT*NGEO*2

When in doubt to the size of NPAIRS, redirect output of kaperture to
/dev/tty the first time to get npairs=:
kaperture [optional parameters] > /dev/tty
```

```
 LINRORT - linearized P-P, P-S1 and P-S2 reflection coefficients
for a horizontal interface separating two of any of the
following halfspaces: ISOTROPIC, VTI, HTI and ORTHORHOMBIC.

   linrort [optional parameters]

 hspace1=ISO medium type of the incidence halfspace:
=ISO ... isotropic
=VTI ... VTI anisotropy
=HTI ... HTI anisotropy
=ORT ... ORTHORHOMBIC anisotropy
 for ISO:
 vp1=2   P-wave velocity, halfspace1
 vs1=1 S-wave velocity, halfspace1
 rho1=2.7 density, halfspace1

 for VTI:
 vp1=2 P-wave vertical velocity (V33), halfspace1
 vs1=1 S-wave vertical velocity (V44=V55), halfspace1
 rho1=2.7 density, halfspace1
 eps1=0 Thomsen's generic epsilon, halfspace1
 delta1=0 Thomsen's generic delta, halfspace1
 gamma1=0 Thomsen's generic gamma, halfspace1 ",

 for HTI:
 vp1=2  P-wave vertical velocity (V33), halfspace1
 vs1=1  "fast" S-wave vertical velocity (V44), halfspace1
 rho1=2.7 density, halfspace1
 eps1_v=0 Tsvankin's "vertical" epsilon, halfspace1
 delta1_v=0 Tsvankin's "vertical" delta, halfspace1
 gamma1_v=0 Tsvankin's "vertical" gamma, halfspace1 ",

 for ORT:
 vp1=2 P-wave vertical velocity (V33), halfspace1
 vs1=1  x2-polarized S-wave vertical velocity (V44), halfspace1
 rho1=2.7 density, halfspace1
 eps1_1=0 Tsvankin's epsilon in [x2,x3] plane, halfspace1
 delta1_1=0 Tsvankin's delta in [x2,x3] plane, halfspace1
 gamma1_1=0 Tsvankin's gamma in [x2,x3] plane, halfspace1
 eps1_2=0 Tsvankin's epsilon in [x1,x3] plane, halfspace1
 delta1_2=0 Tsvankin's delta in [x1,x3] plane, halfspace1
 gamma1_2=0 Tsvankin's gamma in [x1,x3] plane, halfspace1
 delta1_3=0 Tsvankin's delta in [x1,x2] plane, halfspace1
```

```
 hspace2=ISO medium type of the reflecting halfspace (the same
convention as above)

 medium parameters of the 2nd halfspace follow the same convention
 as above:

 vp2=2.5  vs2=1.2 rho2=3.0
 eps2=0   delta2=0
 eps2_v=0 delta2_v=0 gamma2_v=0
 eps2_1=0 delta2_1=0 gamma2_1=0
 eps2_2=0 delta2_2=0 gamma2_2=0
 delta2_3=0

(note you do not need "gamma2" parameter for evaluation
of weak-anisotropy reflection coefficients)

 a_file=-1 the string '-1' ... incidence and azimuth angles are
generated automatically using the setup values below
a_file=file_name ... incidence and azimuth angles are
read from a file "file_name"; the program expects a
file of two columns [inc. angle, azimuth]

 in the case of a_file=-1:
 fangle=0 first incidence phase angle
 langle=30 last incidence angle
 dangle=1 incidence angle increment
 fazim=0 first azimuth (in deg)
 lazim=0 last azimuth  (in deg)
 dazim=1 azimuth increment (in deg)

 kappa=0. azimuthal rotation of the lower halfspace2 (e.t. a
symmetry axis plane for HTI, or a symmetry plane for
ORTHORHOMBIC) with respect to the x1-axis

 out_inf=info.out information output file
 out_P=Rpp.out file with Rpp reflection coefficients
 out_S=Rps.out file with Rps reflection coefficients
 out_SVSH=Rsvsh.out  file with SV and SH projections of reflection
coefficients
 out_Error=error.out file containing error estimates evaluated during
the computation of the reflection coefficients;
```

```
Output:
out_P:
inc. phase angle, azimuth, reflection coefficient; for a_file=-1, the
inc. angle is the fast dimension
out_S:
inc. phase angle, azimuth, Rps1, Rps2, cos(PHI), sin(PHI); for
a_file=-1, the inc. angle is the fast dimension ",
out_SVSH:
inc. phase angle, azimuth, Rsv, Rsh, cos(PHI), sin(PHI); for
a_file=-1, the inc. angle is the fast dimension
out_Error:
error estimates of Rpp, Rpsv and Rpsh approximations; global error is
analysed as well as partial contributions to the error due to the
isotropic velocity contrasts, and due to anisotropic  upper and lower
halfspaces. The error file is self-explanatory, see also descriptions
of subroutines P_err_2nd_order, SV_err_2nd_order and SH_err_2nd_order.


Adopted Convention:

The right-hand Cartesian coordinate system with the x3-axis pointing
upward has been chosen. The upper halfspace (halfspace1)
contains the incident P-wave. Incidence angles can vary from <0,PI/2),
azimuths are unlimited, +azimuth sense counted from x1->x2 axes
(azimuth=0 corresponds to the direction of x1-axis). In the current
version, the coordinate system is attached to the halfspace1 (e.t.
the symmetry axis plane of HTI halfspace1, or one of symmetry planes
of ORTHORHOMBIC halfspace1, is aligned with the x1-axis), however, the
halfspace2 can be arbitrarily rotated along the x3-axis with respect
to the halfspace1. The positive weak-anisotropy polarization of the
reflected P-P wave (e.t. positive P-P reflection coefficient) is close
to the direction of isotropic slowness vector of the wave (pointing
outward the interface). Similarly, weak-anisotropy S-wave reflection
coefficients are described in terms of "SV" and "SH" isotropic
polarizations, "SV" and "SH" being unit vectors in the plane
perpendicular to the isotropic slowness vector. Then, the positive
"SV" polarization vector lies in the incidence plane and points
towards the interface, and positive "SH" polarization vector is
perpendicular to the incidence plane, aligned with the positive
x2-axis, if azimuth=0. Rotation angle "PHI", characterizing a
rotation of "the best projection" of the S1-wave polarization
vector in the isotropic SV-SH plane in the incidence halfspace1, is
```

```
 counted in the positive sense from "SV" axis (PHI=0) towards the
 "SH" axis (PHI=PI/2). Of course, S2 is perpendicular to S1, and
 the projection of S1 and S2 polarizations onto the SV-SH plane
 coincides with SV and SH directions, respectively, for PHI=0.

 The units for velocities are km/s, angles I/O are in degrees

 Additional Notes:
The coefficients are computed as functions of phase incidence
angle and azimuth (determined by the incidence slowness vector).
Vertical symmetry planes of the HTI and
ORTHORHOMBIC halfspaces can be arbitrarily rotated along the
x3-axis. The linearization is based on the assumption of weak ",
contrast in elastic medium parameters across the interface,
and the assumption of weak anisotropy in both halfspaces.
See the "Adopted Convention" paragraph below for a proper
input.


  Author: Petr Jilek, CSM-CWP, December 1999.
```

```
 LORENZ - compute the LORENZ attractor

  lorenz > [stdout]

 Required Parameters: none
 Optional Parameters:
 rho=28.0 parameter for lorenz equations
 sigma=10.0 parameter for lorenz equations
 eta=1.6666667 parameter for lorenz equations
 y0=1.0 initial value of y[0]
 y1=-1.0 initial value of y[1]
 y2=1.0 initial value of y[2]
 h=.01 increment in time
 tol=1.e-08 error tolerance
 stepmax=500 maximum number of steps to compute
 mode=xy xy-pairs, =yz yz-pairs, =xz xz-pairs,
=xyz xyz-triplet, =x only, =y only, =z only
 Notes:
 This program is really just a demo showing how to use the
 differential equation solver rke_solve written by Francois
 Pinard, based on a modified form of the 4th order Runge-Kutta
 method, which employs the error checking method of R. England
 1969.

 The output consists of unformated C-style binary floats, of
 either pairs or triplets as specified by the "mode" paramerter.

 Examples:
 lorenz stepmax=1000 mode=xy | xgraph n=1000 &
 lorenz stepmax=1000 mode=yz | xgraph n=1000 &
 lorenz stepmax=1000 mode=xz | xgraph n=1000 &

 lorenz stepmax=1000 mode=x | suaddhead ns=1000 | suxwigb &
 lorenz stepmax=1000 mode=y | suaddhead ns=1000 | suxwigb &
 lorenz stepmax=1000 mode=z | suaddhead ns=1000 | suxwigb &



 The lorenz equations describe a simplified model of a convection
 cell, and are given by the autonomous system of ODE's
x'(t) = sigma * ( y - x )
y'(t) = x * ( rho - z ) - y
```

```
z'(t) = x * y - eta * z

 Author: CWP: Aug 2004: John Stockwell
```

```
MAKEVEL - MAKE a VELocity function v(x,y,z)

makevel > outfile nx= nz= [optional parameters]

Required Parameters:
nx=                     number of x samples (3rd dimension)
nz=                     number of z samples (1st dimension)

Optional Parameters:
ny=1                    number of y samples (2nd dimension)
dx=1.0                  x sampling interval
fx=0.0                  first x sample
dy=1.0                  y sampling interval
fy=0.0                  first y sample
dz=1.0                  z sampling interval
fz=0.0                  first z sample
v000=2.0                velocity at (x=0,y=0,z=0)
dvdx=0.0                velocity gradient with respect to x
dvdy=0.0                velocity gradient with respect to y
dvdz=0.0                velocity gradient with respect to z
vlens=0.0               velocity perturbation in parabolic lens
tlens=0.0               thickness of parabolic lens
dlens=0.0               diameter of parabolic lens
xlens=                  x coordinate of center of parabolic lens
ylens=                  y coordinate of center of parabolic lens
zlens=                  z coordinate of center of parabolic lens
vran=0.0 standard deviation of random perturbation
vzfile=                 file containing v(z) profile
vzran=0.0               standard deviation of random perturbation to v(z)
vzc=0.0                 v(z) chirp amplitude
z1c=fz                  z at which to begin chirp
z2c=fz+(nz-1)*dz        z at which to end chirp
l1c=dz                  wavelength at beginning of chirp
l2c=dz                  wavelength at end of chirp
exc=1.0                 exponent of chirp
```

```
 MKPARFILE - convert ascii to par file format

 mkparfile <stdin >stdout

 Optional parameters:
  string1="par1" first par string
  string2="par2" second par string

 This is a tool to convert values written line by line to parameter
 vectors in the form expected by getpar.  For example, if the input
 file looks like:
  t0 v0
  t1 v1
...
 then
mkparfile <input >output string1=tnmo string2=vnmo
 yields:
tnmo=t0,t1,...
vnmo=v0,v1,...
```

```
 MRAFXZWT - Multi-Resolution Analysis of a function F(X,Z) by Wavelet
 Transform. Modified to perform different levels of resolution
        analysis for each dimension and also to allow to transform
        back only the lower level of resolution.

    mrafxzwt [parameters] < infile > mrafile

 Required Parameters:
 n1= size of first (fast) dimension
 n2= size of second (slow) dimension

 Optional Parameters:
 p1= maximum integer such that 2^p1 <= n1
 p2= maximum integer such that 2^p2 <= n2
 order=6 order of Daubechies wavelet used (even, 4<=order<=20)
 mralevel1=3   maximum multi-resolution analysis level in dimension 1
 mralevel2=3   maximum multi-resolution analysis level in dimension 2
 trunc=0.0 truncation level (percentage) of the reconstruction
 verbose=0 =1 to print some useful information
 reconfile=    reconstructed data file to write
 reconmrafile= reconstructed data file in MRA domain to write
 dfile= difference between infile and reconfile to write
 dmrafile= difference between mrafile and reconmrafile to write
 dconly=0      =1 keep only dc component of MRA
 verbose=0     =1 to print some useful information
 if (n1 or n2 is not integer powers of 2) specify the following:
  nc1=n1/2 center of trimmed image in the 1st dimension
  nc2=n2/2 center of trimmed image in the 2nd dimension
trimfile= if given, output the trimmed file

 Notes:
 This program performs multi-resolution analysis of an input function
 f(x,z) via the wavelet transform method. Daubechies's least asymmetric
 wavelets are used. The smallest wavelet coefficient retained is given
 by trunc times the absolute maximum size coefficient in the MRA.

 The input dimensions of the data must be expressed by (p1,p2) which

  Author: Zhaobo Meng, 11/25/95, Colorado School of Mines             *
  Modified:  Carlos E. Theodoro, 06/25/97, Colorado School of Mines   *
Included options for:                                    *
- different level of resolutionf or each dimension;           *
- transform back the lower level of resolution, only. *
```

```
*
 Reference: *
 Daubechies, I., 1988, Orthonormal Bases of Compactly Supported *
 Wavelets, Communications on Pure and Applied Mathematics, Vol. XLI,  *
 909-996.   *
```

```
 PDFHISTOGRAM - generate a HISTOGRAM of the Probability Density function

  pdfhistogram < stdin > sdtout [Required params] (Optional params)

 Required parameters:
 ix= column containing X variable
 iy= column containing Y variable
 min_x= minimum X bin
 max_x= maximum X bin
 min_y= minimum Y bin
 max_y= maximum Y bin
 logx=0 =1 use logarithmic scale for X axis
 logy=0 =1 use logarithmic scale for Y axis
 norm= selected normalization type
sqrt - bin / sqrt( xnct*ycnt)
avg_cnt - 0.5* bin / (xcnt + ycnt)
avg_sum - (bin / xcnt + bin / ycnt ) / 2
xcnt - bin / xcnt
ycnt - bin / ycnt
log - log(bin)
total - bin / total
 Optional parameters:
  nx=100 - number of X bins
  ny=100 - number of Y bins
  ir= - column containing reject variable
  rmin= - reject values below rmin
  rmax= - reject values above rmax
NOTE: only one, rmin or rmax, may be used
 Notes:
 PDFHISTOGRAM creates a 2D histogram representing the probability density
 function of the input data. The output is in the form of a binary array
 that can then be plotted via ximage.
 Commandline options allow selecting any of several normalizations
 to apply to the distributions.


 Credits:
  Reginald H. Beardsley rhb@acm.org
Copyright 2006 Exploration Software Consultants Inc.
```

```
PRPLOT - PRinter PLOT of 1-D arrays f(x1) from a 2-D function f(x1,x2)

prplot <infile >outfile [optional parameters]

Optional Parameters:
n1=all                  number of samples in 1st dimension
d1=1.0                  sampling interval in 1st dimension
f1=d1                   first sample in 1st dimension
n2=all                  number of samples in 2nd dimension
d2=1.0                  sampling interval in 2nd dimension
f2=d2                   first sample in 2nd dimension
label2=Trace            label for 2nd dimension
```

```
 RANDVEL3D - Add a random velocity layer (RVL) to a gridded
           v(x,y,z) velocity model

randvel3d <infile n1= n2= >outfile [parameters]

 Required Parameters:
 n1= number of samples along 1st dimension
 n2= number of samples along 2nd dimension

 Optional Parameters:

 n3=1          number of samples along 3rd dimension

 mode=1             add single layer populated with random vels
                    =2 add nrvl layers of random thickness and vel
 seed=from_clock    random number seed (integer)

 ---->New layer geometry info
 i1beg=1       1st dimension beginning sample
 i1end=n1/5    1st dimension ending sample
 i2beg=1       2nd dimension beginning sample
 i2end=n2      2nd dimension ending sample
 i3beg=1       3rd dimension beginning sample
 i3end=n3      3rd dimension ending sample
 ---->New layer velocity info
 vlsd=v/3     range (std dev) of random velocity in layer,
               where v=v(0,0,i1) and i1=(i1beg+i1end)/2
 add=1         add random vel to original vel (v_orig) at that point
               =0 replace vel at that point with (v_orig+v_rand)
 how=0         random vels can be higher or lower than v_orig
               =1 random vels are always lower than v_orig
               =2 random vels are always higher than v_orig
 cvel=2000     layer filled with constant velocity cvel
               (overides vlsd,add,how params)
 ---->Smoothing parameters (0 = no smoothing)
 r1=0.0 1st dimension operator length in samples
 r2=0.0 2nd dimension operator length in samples
 r3=0.0 3rd dimension operator length in samples
 slowness=0 =1 smoothing on slowness; =0 smoothing on velocity

 nrvl=n1/10    number of const velocity layers to add
 pdv=10.       percentage velocity deviation (max) from input model
```

```
Notes:
1. Smoothing radii usually fall in the range of [0,20].
2. Smoothing radii can be used to set aspect ratio of random velocity
   anomalies in the new layer.  For example (r1=5,r2=0,r3=0) will
   result in vertical vel streaks that mimick vertical fracturing.
3. Smoothing on slowness works better to preserve traveltimes relative
   to the unsmoothed case.
4. Default case is a random velocity (+/-30%) near surface layer whose
   thickness is 20% of the total 2D model thickness.
5. Each layer vel is a random perturbation on input model at that level.
6. The depth dimension is assumed to be along axis 1.

Example:
1. 2D RVL with no smoothing
  makevel nz=250 nx=200 | randvel3d n1=250 n2=200 | ximage n1=250
2. 3D RVL with no smoothing
  makevel nz=250 nx=200 ny=220 |
  randvel3d n1=250 n2=200 n3=220 |
  xmovie n1=250 n2=200



Author:  U Houston: Chris Liner c. 2008
         Based on smooth3d (CWP: Zhenyue Liu  March 1995)
```

```
RAYT2DAN -- P- and SV-wave raytracing in 2D anisotropic media

rayt2dan > ttime parameterfiles= nt= nx= nz= [optional parameters]

Required Parameters:
VP0file=  name of file containing VP0(x,z)
nt=                    number of time samples for each ray
nx=                    number of samples (x) for the parameter fields
nz=                    number of samples (z) for the parameter fields

Optional Parameters:
SV=0  for P-waves, SV=1 for Shear waves

Parameters defining the velocity field
dt=0.008                time sampling interval ",
fx=0                    first lateral sample (x) in parameter field
fz=0                    first lateral sample (z) in parameter field
dx=100.0                sample spacing (x) for the parameter fields
dz=100.0                sample spacing (z) for the parameter fields

Parameters defining the takeoff angle of a ray at a source position ",
fa=-60                  first take-off angle of rays (degrees)
na=61                   number of rays ",
da=2                    increment of take-off angle
amin=0                  minimum emergence angle; must be > -90 degrees
amax=90                 maximum emergence angle; must be < 90 degrees

Parameters defining the output traveltime table
fxo=fx                  first lateral sample in traveltime table
nxo=nx                  number of later samples in traveltime table
dxo=dx                  lateral interval in traveltime table
fzo=fz                  first depth sample in traveltime table
nzo=nz                  number of depth samples in traveltime table
dzo=dz                  depth interval in traveltime table
fac=0.01                factor to determine the radius of extrap.

Parameters defining the source positions
fsx=fx                  x-coordinate of first source
nsx=1                   number of sources
dsx=2*dxo               x-coordinate increment of sources
aperx=0.5*nx*dx         ray tracing aperature in x-direction

Files for general anisotropic parameters confined to a vertical plane:
```

```
 a1111file name of file containing a1111(x,z)
 a1133file           name of file containing a1133(x,z)
 VS0file         name of file containing VS0(x,z)
 a1113file           name of file containing a1113(x,z)
 a3313file           name of file containing a3313(x,z)

 For transversely isotropic media Thomsen's parameters could be used:
 deltafile name of file containing delta(x,z)
 epsilonfile name of file containing epsilon(x,z)

 if anisotropy parameters are not given the program considers ",
 isotropic media. ",


 Credits:
      Debashish Sarkar


   Technical Reference:

Cerveny, V., 1972, Seismic rays and ray intensities
in inhomogeneous anisotropic media:
Geophys. J. R. Astr. Soc., 29, 1--13.

  Hangya, A., 1986, Gaussian beams in anisotropic elastic media:
      Geophys. J. R. Astr. Soc., 85, 473--563.

Gajewski, D. and Psencik, I., 1987, Computation of high frequency
seismic wavefields in 3-D  laterally inhomogeneous anisotropic
  media: Geophys. J. R. Astr. Soc., 91, 383-411.
```

```
 RAYT2D - traveltime Tables calculated by 2D paraxial RAY tracing

     rayt2d vfile= tfile= [optional parameters]

 Required parameters:
 vfile=stdin file containning velocity v[nx][nz]
 tfile=stdout file containning traveltime tables
t[nxs][nxo][nzo]

 Optional parameters
 dt=0.008 time sample interval in ray tracing
 nt=401 number of time samples in ray tracing

 fz=0 first depth sample in velocity
 nz=101 number of depth samples in velocity
 dz=100 depth interval in velocity
 fx=0 first lateral sample in velocity
 nx=101 number of lateral samples in velocity
 dx=100 lateral interval in velocity

 fzo=fz first depth sample in traveltime table
 nzo=nz number of depth samples in traveltime table
 dzo=dz depth interval in traveltime table
 fxo=fx first lateral sample in traveltime table
 nxo=nx number of lateral samples in traveltime table
 dxo=dx lateral interval in traveltime table

 surf="0,0;99999,0"  Recording surface "x1,z1;x2,z2;x3,z3;...
 fxs=fx x-coordinate of first source
 nxs=1 number of sources
 dxs=2*dxo x-coordinate increment of sources
 aperx=0.5*nx*dx   ray tracing aperature in x-direction

 fa=-60 first take-off angle of rays (degrees)
 na=61 number of rays
 da=2 increment of take-off angle
 amin=0 minimum emergence angle
 amax=90 maximum emergence angle

 fac=0.01 factor to determine radius for extrapolation
 ek=1 flag of implementing eikonal in shadow zones
 ms=10 print verbal information at every ms sources
 restart=n job is restarted (=y yes; =n no)
```

```
 npv=0 flag of computing quantities for velocity analysis
 if npv>0 specify the following three files
 pvfile=pvfile input file of velocity variation pv[nxo][nzo]
 tvfile=tvfile output file of traveltime variation tables
tv[nxs][nxo][nzo]
 csfile=csfile output file of cosine tables cs[nxs][nxo][nzo]

 Notes:
 1. Each traveltime table is calculated by paraxial ray tracing; then
    traveltimes in shadow zones are compensated by solving eikonal
    equation.
 2. Input velocity is uniformly sampled and smooth one preferred.
 3. Traveltime table and source ranges must be within velocity model.
 4. Ray tracing aperature can be chosen as sum of migration aperature
    plus half of maximum offset.
 5. Memory requirement for this program is about
      [nx*nz+4*mx*nz+3*nxo*nzo+na*(nx*nz+mx*nz+3*nxo*nzo)]*4 bytes
    where mx = min(nx,2*(1+aperx/dx)).


 Author:  Zhenyue Liu, 10/11/94,  Colorado School of Mines

          Trino Salinas, 01/01/96 included the option to handle nonflat
          reference surfaces.
          Subroutines from Dave Hale's modeling library were adapted in
          this code to define topography using cubic splines.

 References:

 Beydoun, W. B., and Keho, T. H., 1987, The paraxial ray method:
   Geophysics, vol. 52, 1639-1653.

 Cerveny, V., 1985, The application of ray tracing to the numerical
   modeling of seismic wavefields in complex structures, in Dohr, G.,
   ED., Seismic shear waves (part A: Theory): Geophysical Press,
   Number 15 in Handbook of Geophysical Exploration, 1-124.
```

```
 RECAST - RECAST data type (convert from one data type to another)

 recast <stdin [optional parameters]  >stdout

 Required parameters:
  none

 Optional parameters:
 in=float input type (float)
  =double (double)
  =int (int)
  =char (char)
=uchar (unsigned char)
  =short (short)
  =long (long)
  =ulong (unsigned long)

 out=double output type (double)
  =float (float)
  =int (int)
  =char (char)
  =uchar (unsigned char)
  =short (short)
  =long (long)
  =ulong (unsigned long)

 outpar=/dev/tty output parameter file, contains the
number of values (n1=)
  other choices for outpar are: /dev/tty,
  /dev/stderr, or a name of a disk file

 Notes: Converting bigger types to smaller is hazardous. For float
 or double conversions to integer types, the results are
 rounded to the nearest integer.


 Credits:

CWP: John Stockwell, Jack K. Cohen
```

```
 REFREALAZIHTI -  REAL AZImuthal REFL/transm coeff for HTI media

 refRealAziHTI  [optional parameters] >coeff.data

 Optional parameters:
 vp1=2         p-wave velocity medium 1 (with respect to symm.axes)
 vs1=1         s-wave velocity medium 1 (with respect to symm.axes)
 eps1=0        epsilon medium1
 delta1=0 delta medium 1
 gamma1=0 gamma medium 1
 rho1=2.7 density medium 1
 vp2=2         p-wave velocity medium 2 (with respect to symm.axes)
 vs2=1         s-wave velocity medium 2 (with respect to symm.axes)
 eps2=0        epsilon medium 2
 delta2=0 delta medium 2
 gamma2=0 gamma medium 2
 rho2=2.7 density medium 2
 modei=0  incident mode is qP
=1 incident mode is qSV
=2 incident mode is SP
 modet=0  scattered mode
 rort=1  reflection(1) or transmission (0)
 azimuth=0 azimuth with respect to x1-axis (clockwise)
 fangle=0 first incidence angle
 langle=45 last incidence angle
 dangle=1 angle increment
 iscale=0      default: angle in degrees
=1 angle-axis in rad
               =2 axis  horizontal slowness
               =3 sin^2 of incidence angle
 ibin=1  binary output
=0 Ascci output
 outparfile =outpar parameter file for plotting
 coeffile =coeff.data coefficient-output file
 test=1  activate testing routines in code
 info=0  output intermediate results

 Notes:
 Axes of symmetry have to coincide in both media.  This code computes
 all 6 REAL reflection/transmissions coefficients on the fly. However,
 the set-up is such Real reflection/transmission coefficients in
 HTI-media with coinciding symmetry axes.
 However, the set-up is such that currently only one coefficient is
```

```
 dumped into the output. This is easily changed.  The solution of the
 scattering problem is obtained numerically and involves the Gaussian
 elimination of a 6X6 matrix. ",


 AUTHOR:: Andreas Rueger, Colorado School of Mines, 02/10/95
               original name of code <graebnerTIH.c>
           modified, extended version of this code <refTIH3D>

  Technical references:

  Sebastian Geoltrain: Asymptotic solutions to direct
and inverse scattering in anisotropic elastic media;
CWP 082.
Graebner, M.; Geophysics, Vol 57, No 11:
Plane-wave reflection and transmission coefficients
for a transversely isotropic solid.
Cerveny, V., 1972, Seismic rays and ray intensities in inhomogeneous
anisotropic media: Geophys. J. R. astr. Soc., 29, 1-13.

.. and some derivations by Andreas Rueger.

 If propagation is perpendicular or
 parallel to the symmetry axis, the solution is analytic (see
 graebner2D.c and rtRealIso.c
```

```
 REFREALVTI -  REAL REFL/transm coeff for VTI media and symmetry-axis
                planes of HTI media

 refRealVTI  [Optional parameters]

 Optional parameters:
 vp1=2          p-wave velocity medium 1 (along symm.axes)
 vs1=1          s-wave velocity medium 1 (along symm.axes)
 eps1=0         Thomsen's epsilon medium 1
 delta1=0  Thomsen's delta medium 1
 rho1=2.7  density medium 1
 axis1=0  medium 1 is VTI
 =1 medium 1 is HTI
 vp2=2.5         p-wave velocity medium 2 (along symm.axes)
 vs2=1.2          s-wave velocity medium 2 (along symm.axes)
 eps2=0  epsilon medium2
 delta2=0 delta medium 2
 rho2=3.0 density medium 2
 axis2=0 medium 2 is VTI
=2 medium 2 is HTI
 modei=0  incident mode is qP
=1 incident mode is qSV
 modet=0  scattered mode
 rort=1  reflection(1) or transmission (0)
 fangle=0 first angle
 langle=45 last angle
 dangle=1 angle increment
 iscale=0       =1 angle-axis in rad
                =2 axis  horizontal slowness
                =3 sin^2 of incidence angle
 ibin=1  binary output
=0 Ascci output
 outparfile =outpar parameter file for plotting
 coeffile =coeff.data coefficient-output file

 Notes:
 Coefficients are based on Graebner's 1992 Geophysics paper. Note the
 mistype in the equation for K1. The algorithm can be used for VTI
 and HTI media on the incidence and scattering side.



 AUTHOR:: Andreas Rueger, Colorado School of Mines, 01/20/95
```

```
      original name of algorithm: graebner1.c

  Technical reference: Graebner, M.; Geophysics, Vol 57, No 11:
Plane-wave reflection and transmission coefficients
for a transversely isotropic solid.
Rueger, A.; Geophysics 1996 (accepted):
P-wave reflection coefficients ...
```

```
REGRID3 - REwrite a [ni3][ni2][ni1] GRID to a [no3][no2][no1] 3-D grid

regrid3 < oldgrid > newgrid [parameters]

Optional parameters:
ni1=1   fastest (3rd) dimension in input grid
ni2=1   second fastest (2nd) dimension in input grid
ni3=1   slowest (1st) dimension in input grid
no1=1   fastest (3rd) dimension in output grid
no2=1   second fastest (2nd) dimension in output grid                ",
no3=1   slowest (1st) dimension in output grid
Optional Parameters:
verbose=0 =1 print some useful information

Notes:
REGRID3 can be used to span a 1-D grid to a 2-D grid,  or a 2-D grid
to a 3-D grid; or to change grid parameters within the dimensions.
Together with MUL and UNIF3, most 3-D velocity model can be


Credits:
  CWP: Zhaobo Meng, 1996, Colorado School of Mines
```

```
RESAMP - RESAMPle the 1st dimension of a 2-dimensional function f(x1,x2)

resamp <infile >outfile [optional parameters]

Required Parameters:

Optional Parameters:
n1=all                  number of samples in 1st (fast) dimension
n2=all                  number of samples in 2nd (slow) dimension
d1=1.0                  sampling interval in 1st dimension
f1=d1                   first sample in 1st dimension
n1r=n1                  number of samples in 1st dimension after resampling
d1r=d1                  sampling interval in 1st dimension after resampling
f1r=f1                  first sample in 1st dimension after resampling

NOTE:  resamp currently performs NO ANTI-ALIAS FILTERING before resampling!
Caveat: this program resamples data that are oscillatory in the fast
   dimension only, such as seismic data with no SU headers. To resample
   other 2d data, such as velocity profiles, use "unisam" or "unisam2
```

```
 SMOOTH2 --- SMOOTH a uniformly sampled 2d array of data, within a user-
defined window, via a damped least squares technique

 smooth2 < stdin n1= n2= [optional parameters ] > stdout

 Required Parameters:
 n1= number of samples in the 1st (fast) dimension
 n2= number of samples in the 2nd (slow) dimension

 Optional Parameters:
 r1=0 smoothing parameter in the 1 direction
 r2=0 smoothing parameter in the 2 direction
 win=0,n1,0,n2 array for window range
 rw=0 smoothing parameter for window function
 efile=                 =efilename if set write relative error(x1) to
efilename

 Notes:
 Larger r1 and r2 result in a smoother data. Recommended ranges of r1  ",
 and r2 are from 1 to 20.

 The file verror gives the relative error between the original velocity
 and the smoothed one, as a function of depth. If the error is
 between 0.01 and 0.1, the smoothing parameters are suitable. Otherwise,
 consider increasing or decreasing the smoothing parameter values.

 Smoothing can be implemented in a selected window. The range of 1st
 dimension for window is from win[0] to win[1]; the range of 2nd
 dimension is from win[2] to win[3].

 Smoothing the window function (i.e. blurring the edges of the window)
 may be done by setting a nonzero value for rw, otherwise the edges
 of the window will be sharp.



Credits:
CWP: Zhen-yue Liu,
adapted for par/main by John Stockwell 1 Oct 92
Windowing feature added by Zliu on 16 Nov 1992
```

```
 SMOOTH3D - 3D grid velocity SMOOTHing by the damped least squares

smooth3d <infile >outfile [parameters]

 Required Parameters:
 n1= number of samples along 1st dimension
 n2= number of samples along 2nd dimension
 n3= number of samples along 3rd dimension

 Optional Parameters:

 Smoothing parameters (0 = no smoothing)
 r1=0.0 operator length in 1st dimension
 r2=0.0 operator length in 2nd dimension
 r3=0.0 operator length in 3rd dimension

 Sample intervals:
 d1=1.0 1st dimension
 d2=1.0 2nd dimension
 d3=1.0 3rd dimension
 iter=2 number of iteration used
 time=0 which dimension the time axis is (0 = no time axis)
 depth=1 which dimension the depth axis is (ignored when time>0)
 mu=1 the relative weight at maximum depth (or time)
 verbose=0 =1 for printing minimum wavelengths
 slowness=0 =1 smoothing on slowness; =0 smoothing on velocity
 vminc=0 velocity values below it are cliped before smoothing
 vmaxc=99999 velocity values above it are cliped before smoothing

 Notes:
 1. The larger the smoothing parameters, the smoother the output velocity.
    These parameters are lengths of smoothing operators in each dimension.
 2. iter controls the orders of derivatives to be smoothed in the output
    velocity. e.g., iter=2 means derivatives until 2nd order smoothed.
 3. mu is the multipler of smoothing parameters at the bottom compared to
    those at the surface.
 4. Minimum wavelengths of each dimension and the total may be printed
   for the resulting output velocity is. To compute these parameters for
   the input velocity, use r1=r2=r3=0.
 5. Smoothing directly on slowness works better to preserve traveltime.
   So the program optionally converts the input velocity into slowness ",
   and smooths the slowness, then converts back into velocity.
```

```
 Author:  CWP: Zhenyue Liu  March 1995

 Reference:
 Liu, Z., 1994,"A velocity smoothing technique based on damped least squares
in Project Reveiew, May 10, 1994, Consortium Project on
Seismic Inverse Methods for Complex Stuctures.
```

```
 SMOOTHINT2 --- SMOOTH non-uniformly sampled INTerfaces, via the damped
   least-squares technique

  smoothint2 <input ninf= >output [optional parameters]

 Required Parameters:
 <input                  file containing original interfaces
 >output                 file containing smoothed interfaces

 Optional Parameters:
 ninf=5                number of interfaces
 r=100 smoothing parameter
 npmax=101 maximum number of points in interfaces

 Notes:
 The input file is an ASCII file. Each interface is represented by pairs
 (non-uniform sampling) of x and z values, with one pair of values on
 each line, separated by spaces or tabs. Each interface is separated with
 an entry with a large negative z value for example: 1.0     -9999.
 There is no entry for the surface. The surface is assumed to be flat
 with z=0.
 This is similar to a CSHOT model file without a surface entry and
 without comments.

 The smoothing method is analogous to a moving window averaging process
 (but not the same) with the parameter "r" being analogous to the "width
 of the window. Thus, the size of "r" must be chosen to by compatible
 with the scale (wavelengths) of the variations of the interfaces in the
 model being smoothed.

 Example using the test data set generated by unif2:
 unif2 tfile=tfilename
 Compare the unsmoothed interface model:
 unif2 < tfilename method=interpolation_method |
  psimage n1=100 n2=100 d1=10 d2=10 | ...
 To the smoothed interface model:
 smoothint2 r=100 < tfilename | unif2 method=interpolation_method | ",
psimage n1=100 n2=100 d1=10 d2=10 | ...



 Credits:
  CWP: Zhenyue Liu, Jan 1994
```

Reference:
Liu, Zhenyue, 1994, Velocity smoothing: theory and implementation, Project Review, 1994, Consortium Project on Seismic Inverse Methods for Complex Stuctures (in review)

```
STIFF2VEL - Transforms 2D elastic stiffnesses to (vp,vs,epsilon,delta)

stiff2vel  nx=  nz=  [optional parameters]

Required parameters:
nx=    number of x samples (2nd dimension)
nz=    number of z samples (1st dimension)
rho_file='rho.bin'     input file containing rho(x,z)
c11_file='c11.bin'     input file containing c11(x,z)
c13_file='c13.bin'     input file containing c13(x,z)
c33_file='c33.bin'     input file containing c33(x,z)
c44_file='c44.bin'     input file containing c44(x,z)

Optional Parameters:
vp_file='vp.bin'  output file containing P-wave velocities
vs_file='vs.bin'  output file containing S-wave velocities
rho_file='rho.bin'  output file containing densities
eps_file='eps.bin'  output file containing Thomsen epsilon
delta_file='delta.bin' output file containing Thomsen delta

Notes:
1. All quantities in MKS units
2. Isotropy implied by c11(x,z)=c33(x,z)=0
3. Vertical symmetry axis is assumed.



 Coded:
 Aramco: Chris Liner 9/25/2005
         (based on vel2stiff.c)
```

```
SUBSET - select a SUBSET of the samples from a 3-dimensional file

subset <infile >outfile [optional parameters]

Optional Parameters:
n1=nfloats              number of samples in 1st dimension
n2=nfloats/n1           number of samples in 2nd dimension
n3=nfloats/(n1*n2)      number of samples in 3rd dimension
id1s=1                  increment in samples selected in 1st dimension
if1s=0                  index of first sample selected in 1st dimension
n1s=1+(n1-if1s-1)/id1s number of samples selected in 1st dimension
ix1s=if1s,if1s+id1s,...indices of samples selected in 1st dimension
id2s=1                  increment in samples selected in 2nd dimension
if2s=0                  index of first sample selected in 2nd dimension
n2s=1+(n2-if2s-1)/id2s number of samples selected in 2nd dimension
ix2s=if2s,if2s+id2s,...indices of samples selected in 2nd dimension
id3s=1                  increment in samples selected in 3rd dimension
if3s=0                  index of first sample selected in 3rd dimension
n3s=1+(n3-if3s-1)/id3s number of samples selected in 3rd dimension
ix3s=if3s,if3s+id3s,...indices of samples selected in 3rd dimension

For the 1st dimension, output is selected from input as follows:
      output[i1s] = input[ix1s[i1s]], for i1s = 0 to n1s-1
Likewise for the 2nd and 3rd dimensions.



AUTHOR:  Dave Hale, Colorado School of Mines, 07/07/89
```

```
 SWAPBYTES - SWAP the BYTES of various  data types

 swapbytes <stdin [optional parameters]  >stdout

 Required parameters:
  none

 Optional parameters:
 in=float input type (float)
  =double (double)
  =short (short)
  =ushort (unsigned short)
  =long (long)
  =ulong (unsigned long)
  =int (int)

 outpar=/dev/tty output parameter file, contains the
number of values (n1=)
  other choices for outpar are: /dev/tty,
  /dev/stderr, or a name of a disk file

 Notes:
 The byte order of the mantissa of binary data values on PC's and DEC's
 is the reverse of so called "big endian" machines (IBM RS6000, SUN,etc.)
 hence the need for byte-swapping capability. The subroutines in this code
 have been tested for swapping between PCs and big endian machines, but
 have not been tested for DEC products.

 Caveat:
 2 byte short, 4 byte long, 4 byte float, 4 byte int,
 and 8 bit double assumed.


 Credits:
CWP: John Stockwell (Jan 1994)
 Institut fur Geophysik, Hamburg: Jens Hartmann supplied byte swapping
subroutines
```

```
THOM2HTI - Convert Thompson parameters V_p0, V_s0, eps, gamma,
             to the HTI parameters alpha, beta, epsilon(V), delta(V),
      gamma

thom2hti  [optional parameter]


vp=2         symm.axis p-wave velocity
vs=1         symm.axis s-wave velocity
eps=0        Thomsen's (generic) epsilon
gamma=0      Thomsen's generic gamma
weak=1       compute weak approximation
outpar=/dev/tty output parameter file

Outputs:
  alpha, beta, e(V), d(V), gamma

Notes:
Output is dumped to the screen and, if selected to outpar

Code can be used to find models that satisfy the constraints
that are imposed on HTI models caused by vertically fractured
layers. For definition and use of the HTI parameter set see CWP-235.


 Credits: Andreas Rueger, CWP
 For definition and use of the HTI parameter set see CWP-235.
```

```
THOM2STIFF - convert Thomsen's parameters into (density normalized)
                stiffness components for transversely isotropic
    material with in-plane-tilted axis of symmetry

thom2stiff [optional parameter]

vp=2          symm.axis p-wave velocity
vs=1          symm.axis s-wave velocity
eps=0         Thomsen's (generic) epsilon
delta=0       Thomsen's (generic) delta
gamma=0       Thomsen's (generic) gamma
rho=1         density
phi=0         angle(DEG) vertical --> symmetry axes (clockwise)
sign=1        sign of c11+c44 (generally sign=1)
outpar=/dev/tty output parameter file

Output:
aijkl,cijkl (density normalized) stiffness components


Author: CWP: Andreas Rueger  1995
```

```
TRANSP3D - TRANSPose an n1 by n2 by n3 element matrix

transp3d <infile >outfile n1= n2= [optional parameters]

Required Parameters:
n1 number of elements in 1st (fast) dimension of matrix
n2 number of elements in 2nd (middle) dimension of matrix

Optional Parameters:
n3=all number of elements in 3rd (slow) dimension of matrix
perm=231 desired output axis ordering
nbpe=sizeof(float) number of bytes per matrix element
scratchdir=/tmp directory for scratch files
scratchstem=foo stem prefix for scratch file names
verbose=0 =1 for diagnostic information
```

```
TRANSP - TRANSPose an n1 by n2 element matrix

transp <infile >outfile n1= [optional parameters]

Required Parameters:
n1                      number of elements in 1st (fast) dimension of matrix

Optional Parameters:
n2=all                  number of elements in 2nd (slow) dimension of matrix
nbpe=sizeof(float)      number of bytes per matrix element
verbose=0               =1 for diagnostic information
```

```
TVNMOQC - Check tnmo-vnmo pairs; create t-v column files

tvnmoqc [parameters] cdp=... tnmo=... vnmo=...

  Example:
tvnmoqc mode=1 \\
cdp=15,35 \\
tnmo=0.0091,0.2501,0.5001,0.7501,0.9941 \\
vnmo=1497.0,2000.0,2500.0,3000.0,3500.0 \\
tnmo=0.0082,0.2402,0.4902,0.7402,0.9842 \\
vnmo=1495.0,1900.0,2400.0,2900.0,3400.0

Required Parameter:
  prefix=        Prefix of output t-v file(s)
                 Required only for mode=2

Optional Parameter:
  mode=1         1=qc: check that tnmo values increase
                 2=qc and output t-v files

mode=1
  TVNMOQC checks that there is a tnmo and vnmo series for each CDP
    and it checks that each tnmo series increases in time.

mode=2
  TVNMOQC does mode=1 checking, plus ...

  TVNMOQC converts par (MKPARFILE) values written as:

         cdp=15,35,...,95 \\
         tnmo=t151,t152,...,t15n \\
         vnmo=v151,v152,...,v15n \\
         tnmo=t351,t352,...,t35n \\
         vnmo=v351,v352,...,v35n \\
         tnmo=... \\
         vnmo=... \\
         tnmo=t951,t952,...,t95n \\
         vnmo=v951,v952,...,v95n \\

  to column format. The format of each output file is:

         t1 v1
         t2 v2
```

```
          ...
         tn vn
```

One file is output for each input pair of tnmo-vnmo series.

A CDP VALUE MUST BE SUPPLIED FOR EACH TNMO-VNMO ROW PAIR.

Prefix of each output file is the user-supplied value of
parameter PREFIX.
Suffix of each output file is the cdp value.
For the example above, output files names are:
PREFIX.15 PREFIX.35 ... PREFIX.95

Credits:
MTU: David Forel (adapted from SUNMO)

```
 UNIF2ANISO - generate a 2-D UNIFormly sampled profile of elastic
constants from a layered model.

  unif2aniso < infile [Parameters]

 Required Parameters:
 none

 Optional Parameters:
 ninf=5 number of interfaces
 nx=100 number of x samples (2nd dimension)
 nz=100 number of z samples (1st dimension)
 dx=10 x sampling interval
 dz=10 z sampling interval

 npmax=201 maximum number of points on interfaces

 fx=0.0 first x sample
 fz=0.0 first z sample


 x0=0.0,0.0,...,  distance x at which vp00 and vs00 are specified
 z0=0.0,0.0,...,  depth z at which vp00 and vs00 are specified

 vp00=1500,2000,..., P-velocity at each x0,z0 (m/sec)
 vs00=866,1155..., S-velocity at each x0,z0 (m/sec)
 rho00=1000,1100,..., density at each x0,z0 (kg/m^3)
 q00=110,120,130,.., attenuation Q, at each x0,z0 (kg/m^3)

 eps00=0,0,0..., Thomsen or Sayers epsilon
 delta00=0,0,0..., Thomsen or Sayers delta
 gamma00=0,0,0..., Thomsen or Sayers gamma

 dqdx=0.0,0.0,..., x-derivative of Q (d q/dx)
 dqdz=0.0,0.0,..., z-derivative of Q (d q/dz)

 drdx=0.0,0.0,..., x-derivative of density (d rho/dx)
 drdz=0.0,0.0,..., z-derivative of density (d rho/dz)

 dvpdx=0.0,0.0,..., x-derivative of P-velocity (dvp/dx)
 dvpdz=0.0,0.0,..., z-derivative of P-velocity (dvs/dz)

 dvsdx=0.0,0.0,..., x-derivative of S-velocity (dvs/dx)
```

```
 dvsdz=0.0,0.0,..., z-derivative of S-velocity (dvs/dz)

 dedx=0.0,0.0,..., x-derivative of epsilon (de/dx)
 dedz=0.0,0.0,..., z-derivative of epsilon with depth z (de/dz)

 dddx=0.0,0.0,..., x-derivative of delta (dd/dx)
 dddz=0.0,0.0,..., z-derivative of delta (dd/dz)

 dgdz=0.0,0.0,..., x-derivative of gamma (dg/dz)
 dgdx=0.0,0.0,..., z-derivative of gamma (dg/dx)

 phi00=0,0,...,  rotation angle(s) in each layer

 ...output filenames
 c11_file=c11_file output filename for c11 values
 c13_file=c13_file output filename for c13 values
 c15_file=c15_file output filename for c15 values
 c33_file=c33_file output filename for c33 values
 c35_file=c35_file output filename for c35 values
 c44_file=c44_file output filename for c44 values
 c55_file=c55_file output filename for c55 values
 c66_file=c66_file output filename for c66 values

 rho_file=rho_file output filename for density values
 q_file=q_file output filename for Q values

 paramtype=1   =1 Thomsen parameters, =0 Sayers parameters(see below)
 method=linear for linear interpolation of interface
  =mono for monotonic cubic interpolation of interface
=akima for Akima's cubic interpolation of interface
=spline for cubic spline interpolation of interface

 tfile= =testfilename  if set, a sample input dataset is
   output to "testfilename".

 Notes:
 The input file is an ASCII file containing x z values representing a
 piecewise continuous velocity model with a flat surface on top.

 The surface and each successive boundary between media is represented
 by a list of selected x z pairs written column form. The first and
 last x values must be the same for all boundaries. Use the entry
 1.0  -99999  to separate the entries for successive boundaries. No
```

```
boundary may cross another. Note that the choice of the method of
interpolation may cause boundaries to cross that do not appear to
cross in the input data file.

The number of interfaces is specified by the parameter "ninf". This
number does not include the top surface of the model. The input data
format is the same as a CSHOT model file with all comments removed.

The algorithm works by transforming the P-wavespeed , S-wavespeed,
density and the Thomsen or Sayers parameters epsilon, delta, and gamma
into elastic stiffness coefficients. Furthermore, the user can specify
rotations, phi, to the elasticity tensor in each layer.

Common ranges of Thomsen parameters are
 epsilon:  0.0 -> 0.5
 delta:   -0.2 -> 0.4
 gamma:    0.0 -> 0.4

If only P-wave, S-wave velocities and density is given as input,
the model is, by definition,  isotropic.

If files containing Thomsen/Sayers parameters are given, the model
will be assumed to have VTI symmetry.

Example using test input file generating feature:
unif2aniso tfile=testfilename  produces a 5 interface demonstration model
unif2aniso < testfilename
ximage < c11_file n1=100 n2=100
ximage < c13_file n1=100 n2=100
ximage < c15_file n1=100 n2=100
ximage < c33_file n1=100 n2=100
ximage < c35_file n1=100 n2=100
ximage < c44_file n1=100 n2=100
ximage < c55_file n1=100 n2=100
ximage < c66_file n1=100 n2=100
ximage < rho_file n1=100 n2=100
ximage < q_file   n1=100 n2=100



Credits:
     CWP: John Stockwell, April 2005.
 CWP: based on program unif2 by Zhenyue Liu, 1994
```

```
 UNIF2 - generate a 2-D UNIFormly sampled velocity profile from a layered
    model. In each layer, velocity is a linear function of position.

  unif2 < infile > outfile [parameters]

 Required parameters:
 none

 Optional Parameters:
 ninf=5 number of interfaces
 nx=100 number of x samples (2nd dimension)
 nz=100 number of z samples (1st dimension)
 dx=10 x sampling interval
 dz=10 z sampling interval

 npmax=201 maximum number of points on interfaces

 fx=0.0 first x sample
 fz=0.0 first z sample

 x0=0.0,0.0,...,  distance x at which v00 is specified
 z0=0.0,0.0,...,  depth z at which v00 is specified
 v00=1500,2000,2500..., velocity at each x0,z0 (m/sec)
 dvdx=0.0,0.0,..., derivative of velocity with distance x (dv/dx)
 dvdz=0.0,0.0,..., derivative of velocity with depth z (dv/dz)

 method=linear for linear interpolation of interface
  =mono for monotonic cubic interpolation of interface
=akima for Akima's cubic interpolation of interface
=spline for cubic spline interpolation of interface

 tfile= =testfilename  if set, a sample input dataset is
   output to "testfilename".

 Notes:
 The input file is an ASCII file containing x z values representing a
 piecewise continuous velocity model with a flat surface on top. The surface
 and each successive boundary between media are represented by a list of
 selected x z pairs written column form. The first and last x values must
 be the same for all boundaries. Use the entry   1.0  -99999  to separate
 entries for successive boundaries. No boundary may cross another. Note
 that the choice of the method of interpolation may cause boundaries
 to cross that do not appear to cross in the input data file.
```

```
The number of interfaces is specified by the parameter "ninf". This
number does not include the top surface of the model. The input data
format is the same as a CSHOT model file with all comments removed.

Example using test input file generating feature:
unif2 tfile=testfilename    produces a 5 interface demonstration model
unif2 < testfilename | psimage n1=100 n2=100 d1=10 d2=10 | ...



Credits:
 CWP: Zhenyue Liu, 1994
     CWP: John Stockwell, 1994, added demonstration model stuff.
```

```
UNISAM2 - UNIformly SAMple a 2-D function f(x1,x2)

unisam2 [optional parameters] <inputfile >outputfile

Required Parameters:
none
Optional Parameters:
x1=             array of x1 values at which input f(x1,x2) is sampled
... Or specify a unform linear set of values for x1 via:
nx1=1           number of input samples in 1st dimension
dx1=1           input sampling interval in 1st dimension
fx1=0           first input sample in 1st dimension
...
n1=1            number of output samples in 1st dimension
d1=             output sampling interval in 1st dimension
f1=             first output sample in 1st dimension
x2=             array of x2 values at which input f(x1,x2) is sampled
... Or specify a unform linear set of values for x2 via:
nx2=1           number of input samples in 2nd dimension
dx2=1           input sampling interval in 2nd dimension
fx2=0           first input sample in 2nd dimension
...
n2=1            number of output samples in 2nd dimension
d2=             output sampling interval in 2nd dimension
f2=             first output sample in 2nd dimension
...
method1=linear  =linear for linear interpolation
                =mono for monotonic bicubic interpolation
                =akima for Akima bicubic interpolation
                =spline for bicubic spline interpolation
method2=linear  =linear for linear interpolation
                =mono for monotonic bicubic interpolation
                =akima for Akima bicubic interpolation
                =spline for bicubic spline interpolation

NOTES:
The number of input samples is the number of x1 values times the
number of x2 values.  The number of output samples is n1 times n2.
The output sampling intervals (d1 and d2) and first samples (f1 and f2)
default to span the range of input x1 and x2 values.  In other words,
d1=(x1max-x1min)/(n1-1) and f1=x1min; likewise for d2 and f2.

Interpolation is first performed along the 2nd dimension for each
```

```
value of x1 specified.  Interpolation is then performed along the
1st dimension.


AUTHOR:  Dave Hale, Colorado School of Mines, 01/12/91\n"
```

```
UNISAM - UNIformly SAMple a function y(x) specified as x,y pairs

  unisam xin= yin= nout= [optional parameters] >binaryfile
   ... or ...
  unisam xfile= yfile= npairs= nout= [optional parameters] >binaryfile
   ... or ...
  unisam xyfile= npairs= nout= [optional parameters] >binaryfile

Required Parameters:
xin=,,, array of x values (number of xin = number of yin)
yin=,,, array of y values (number of yin = number of xin)
 ... or
xfile= binary file of x values
yfile= binary file of y values
 ... or
xyfile= binary file of x,y pairs
npairs= number of pairs input (active only if xfile= and yfile=
 or xyfile= are set)

nout=  number of y values output to binary file

Optional Parameters:
dxout=1.0  output x sampling interval
fxout=0.0  output first x
method=linear  =linear for linear interpolation (continuous y)
=mono for monotonic cubic interpolation (continuous y')
=akima for Akima's cubic interpolation (continuous y')
=spline for cubic spline interpolation (continuous y'')
isint=,,,  where these sine interpolations to apply
amp=,,,  amplitude of sine interpolations
phase0=,,,  starting phase (defaults: 0,0,0,...,0)
totalphase=,,, total phase (default pi,pi,pi,...,pi.)
nwidth=0       apply window smoothing if nwidth>0


AUTHOR:  Dave Hale, Colorado School of Mines, 07/07/89
         Zhaobo Meng, Colorado School of Mines,
     added sine interpolation and window smoothing, 09/16/96
         CWP: John Stockwell,  added file input options, 24 Nov 1997

Remarks: In interpolation, suppose you need 2 pieces of
     sine interpolation before index 3 to 4, and index 20 to 21
   then set: isint=3,20. The sine interpolations use a sine
```

```
function with starting phase being phase0, total phase
being totalphase (i.e. ending phase being phase0+totalphase
for each interpolation).
```

```
UTMCONV - CONVert longitude and latitude to UTM, and vice versa

utmconv <stdin >stdout [optional parameters]

Optional parameters:
   idx=23          reference ellipsoid index (default is WGS 1984)
   format=%.3f     output number format (printf style for one float)
   a=(from idx)    user-specified semimajor axis of ellipsoid
   f=(from idx)    user-specified flattening of ellipsoid
   letter=0        =1: use UTM letter designator for latitude/Northing
   invert=0        =0: convert latitude and longitude to UTM
                   =1: convert UTM to latitude and longitude
   verbose=0       =1: echo parameters and number of converted coords

   lon0=           central meridian for TM projection in degrees
                   (default uses the 60 standard UTM longitude zones)
   xoff=500000     false Easting (default: UTM)
   ysoff=10000000  false Northing, southern hemisphere (default: UTM)
   ynoff=0         false Northing, northern hemisphere (default: UTM)

Notes:
   Universal Transverse Mercator (UTM) coordinates are defined between
   latitudes 80S (-80) and 84N (84). Longitude values must be between
   -180 degrees (west) and 179.999... degrees (east).

   Input and output is in ASCII format. For a conversion from lon/lat
   to UTM (invert=0), input is a two-column table of longitude and
   latitude in decimal degrees. Output is a three-column table of
   UTM Easting, UTM Northing, and UTM zone. The zone is given either
   by the zone number only (default, negative on southern hemisphere)
   or by the positive zone number plus a letter designator (letter=1).

Example:
   Convert 40.822N, 14.125E to UTM with zone number and letter,
   output values rounded to nearest integer:
       echo 14.125 40.822 | utmconv letter=1 format=%.0f
   The output is "426213 4519366 33T" (Easting, Northing, UTM zone).

Reference ellipsoids:
   An ellipsoid may be specified by its semimajor axis a and its
   flattening f, or one of the following ellipsoids may be selected
   by its index idx (semimajor axes in meters):
    0  Sphere with radius of 6371000 m
```

```
 1  Airy 1830
 2  Australian National 1965
 3  Bessel 1841 (Ethiopia, Indonesia, Japan, Korea)
 4  Bessel 1841 (Namibia)
 5  Clarke 1866
 6  Clarke 1880
 7  Everest (Brunei, E. Malaysia)
 8  Everest (India 1830)
 9  Everest (India 1956)
10  Everest (Pakistan)
11  Everest (W. Malaysia, Singapore 1948)
12  Everest (W. Malaysia 1969)
13  Geodetic Reference System 1980 (GRS 1980)
14  Helmert 1906
15  Hough 1960
16  Indonesian 1974
17  International 1924 / Hayford 1909
18  Krassovsky 1940
19  Modified Airy
20  Modified Fischer 1960
21  South American 1969
22  World Geodetic System 1972 (WGS 1972)
23  World Geodetic System 1984 (WGS 1984) / NAD 1983
```

UTM grid:
The Universal Transverse Mercator (UTM) system is a world wide coordinate system defined between 80S and 84N. It divides the Earth into 60 six-degree zones. Zone number 1 has its central meridian at 177W (-177 degrees), and numbers increase eastward.

Within each zone, an Easting of 500,000 m is assigned to its central meridian to avoid negative coordinates. On the northern hemisphere, Northings start at 0 m at the equator and increase northward. On the southern hemisphere a false Northing of 10,000,000 m is applied, i.e. Northings start at 10,000,000 m at the equator and decrease southward. Letters are sometimes used to identify different zones of latitude. The letters C-M indicate zones on the southern and the letters N-X zones on the northern hemisphere.

Author:
Nils Maercklin, RISSC, University of Naples, Italy, March 2007

References:
NIMA (2000). Department of Defense World Geodetic System 1984 -
its definition and relationships with local geodetic systems.
Technical Report TR8350.2. National Imagery and Mapping Agency,
Geodesy and Geophysics Department, St. Louis, MO. 3rd edition.
J. P. Snyder (1987). Map Projections - A Working Manual.
U.S. Geological Survey Professional Paper 1395, 383 pages.
U.S. Government Printing Office.

```
 VEL2STIFF - Transforms VELocities, densities, and Thomsen or Sayers
parameters to elastic STIFFnesses

 vel2stiff  [Required parameters] [Optional Parameters] > stdout

 Required parameters:
 vpfile= file with P-wave velocities
 vsfile= file with S-wave velocities
 rhofile= file with densities

 Optional Parameters:
 epsfile= file with Thomsen/Sayers epsilon
 deltafile= file with Thomsen/Sayers delta
 gammafile= file with Thomsen/Sayers gamma
 phi_file= angle of axis of symmetry from vertical (radians)

 c11_file=c11_file     output filename for c11 values
 c13_file=c13_file     output filename for c13 values
 c15_file=c15_file     output filename for c15 values
 c33_file=c33_file     output filename for c33 values
 c35_file=c35_file     output filename for c35 values
 c44_file=c44_file     output filename for c44 values
 c55_file=c55_file     output filename for c55 values
 c66_file=c66_file     output filename for c66 values

 paramtype=1  (1) Thomsen parameters, (0) Sayers parameters(see below)

 nx=101 number of x samples 2nd (slow) dimension
 nz=101 number of z samples 1st (fast) dimension

 Notes:
 Transforms velocities, density and Thomsen/Sayers parameters
 epsilon, delta, and gamma into elastic stiffness coefficients.

 If only P-wave, S-wave velocities and density is given as input,
 the model is assumed to be isotropic.

 If files containing Thomsen/Sayers parameters are given, the model
 will be assumed to have VTI symmetry.

 All input files  vpfile, vsfile, rhofile etc. are assumed to consist
 only of C style binary floating point numbers representing the
 corresponding  material values of vp, vs, rho etc. Similarly, the output
```

files consist of the coresponding stiffnesses as C style binary floats.
If the output files are to be used as input for a modeling program,
such as suea2df, then further, the contents are assumed be arrays of
floating point numbers of the form of Array[n2][n1], where the fast
dimension, dimension 1, represents depth.

Author:
CWP: Sverre Brandsberg-Dahl 1999

Extended:
CWP: Stig-Kyrre Foss 2001
- to include the option to use the parameters by Sayers (1995)
instead of the Thomsen parameters

Technical reference:
Sayers, C. M.: Simplified anisotropy parameters for transversely
isotropic sedimentary rocks. Geophysics 1995, pages 1933-1935.

```
VELCONV - VELocity CONVersion

velconv <infile >outfile intype= outtype= [optional parameters]

Required Parameters:
intype=                 input data type (see valid types below)
outtype=                output data type (see valid types below)

Valid types for input and output data are:
vintt           interval velocity as a function of time
vrmst           RMS velocity as a function of time
vintz           velocity as a function of depth
zt              depth as a function of time
tz              time as a function of depth

Optional Parameters:
nt=all                  number of time samples
dt=1.0                  time sampling interval
ft=0.0                  first time
nz=all                  number of depth samples
dz=1.0                  depth sampling interval
fz=0.0                  first depth
nx=all                  number of traces

Example:  "intype=vintz outtype=vrmst" converts an interval velocity
          function of depth to an RMS velocity function of time.

Notes:  nt, dt, and ft are used only for input and output functions
        of time; you need specify these only for vintt, vrmst, orzt.
        Likewise, nz, dz, and fz are used only for input and output
        functions of depth.

The input and output data formats are C-style binary floats.


 AUTHOR:  Dave Hale, Colorado School of Mines, 07/07/89
```

```
 VELPERTAN - Velocity PERTerbation analysis in ANisotropic media to
             determine the model update required to flatten image gathers",

 velpertan boundary= par=cig.par refl1= refl2= npicks1=
npicks2= cdp1= cdp2= vfile= efile= dfile= nx= dx= fx=
ncdp= dcdp= fcdp= off0= noff= doff= >outfile [parameters]

 Required Parameters:
 refl1= file with picks on the 1st reflector
 refl2= file with picks on the 2nd reflector
 vfile= file defining VP0 at all grid points from prev. iter. ",
 efile= file defining eps at all grid points from prev. iter. ",
 dfile= file defining del at all grid points from prev. iter. ",
 boundary= file defining the boundary above which
         parameters are known; update is done below this ",
boundary ",
 npicks1= number of picks on the 1st reflector
 npicks2= number of picks on the 2nd reflector
 ncdp= number of cdp's
 dcdp= cdp spacing
 fcdp= first cdp
 off0= first offset in common image gathers
 noff= number of offsets in common image gathers
 doff= offset increment in common image gathers
 cip1=x1,r1,r2,..., cip=xn,r1n,r2n         description of input CIGS
 cip2=x2,r1,r2,..., cip=xn,r1n,r2n         description of input CIGS
x x-value of a common image point
r1 hyperbolic component of the residual moveout
r2 non-hyperbolic component of residual moveout
 Optional Parameters:
 method=akima         for linear interpolation of the interface
                       =mono for monotonic cubic interpolation of interface
                       =akima for Akima's cubic interpolation of interface
                       =spline for cubic spline interpolation of interface
 VP0=2000 Starting value for vertical velocity at a point in the
target layer
 x00=0.0 x-coordinate at which VP0 is defined
 z00=0.0 z-coordinate at which VP0 is defined
 eps=0.0 Starting value for Thomsen's parameter epsilon
 del=0.0 Starting value for Thomsen's parameter delta
 kz=0.0 Starting value for the vertical gradient in VP0
 kx=0.0 Starting value for the lateral gradient in VP0
 nx=100 number of nodes in the horizontal direction for the
```

```
velocity grid
 nz=100 number of nodes in the vertical direction for the
velocity grid
 dx=10 horizontal grid increment
 dz=10 vertical grid increment
 fx=0 first horizontal grid point
 fz=0 first vertical grid point
 dt=0.008 traveltime increment
 nt=500 no. of points on the ray
 amax=360 max. angle of emergence
 amin=0 min. angle of emergence

 Smoothing parameters:
 r1=0                  smoothing parameter in the 1 direction
 r2=0                  smoothing parameter in the 2 direction
 win=0,n1,0,n2         array for window range
 rw=0                  smoothing parameter for window function
 nbound=2 number of points picked on the boundary
 tol=0.1 tolerance in computing the offset (m)
 Notes:
 This program is used as part of the velocity analysis technique developed
 by Debashish Sarkar, CWP:2003.

 Notes:
 The output par file contains the coefficients describing the residual
 moveout. This program is used in conjunction with surelanan.


 Author: CSM: Debashish Sarkar, December 2003
 based on program: velpert.c written by Zhenuye Liu.
```

```
VELPERT - estimate velocity parameter perturbation from covariance
of imaged depths in common image gathers (CIGs)

verpert <dfile dzfile=dzfile >outfile [parameters]

Required Parameters:
dfile input of imaged depths in CIGs
dzfile=dzfile input of dz/dv at the imaged depths in CIGs
outfile output of the estimated parameter
noff number of offsets
ncip number of common image gathers

Optional Parameters:
moff=noff number of first offsets used in velocity estimation

Notes:
1. This program is part of Zhenyue Liu's velocity analysis technique.
   The input dzdv values are computed using the program dzdv.
2. For given depths, using moff smaller than noff may avoid poor
   values of dz/dv at far offsets. However, a too small moff used
   will the sensitivity of velocity error to the imaged depth.
3. Outfile contains three parts:
   dlambda correction of the velocity paramter. dlambda plus
    the initial parameter (used in migration) will be
the updated one.
   deviation to measure how close imaged depths are to each other
    in CIGs. Old deviation corresponds to the initial
parameter; new deviation corresponds to the updated one.
   sensitivity  to predict how sensitive the error in the estimated
parameter is to an error in the measurement of imaged
depths.

      error of parameter <= sensitivity * error of depth.


Author:  Zhenyue Liu, 12/29/93,  Colorado School of Mines

Reference:
Liu, Z. 1995, "Migration Velocity Analysis", Ph.D. Thesis, Colorado
     School of Mines, CWP report #168.
```

```
 VERHULST - solve the VERHULST logistic equation

  verhulst > [stdout]

 Required Parameters: none
 Optional Parameters:
 a1=1.0 parameter for verhulst equation
 a2=2000 parameter for verhulst equation
 y0=10 initial value of y[0]
 h=.01 increment in time
 tol=1.e-08 error tolerance
 stepmax=2000 maximum number of steps to compute
 mode=x xy-pairs, =yz yz-pairs, =xz xz-pairs,
=xyz xyz-triplet, =x only, =y only, =z only
 Notes:
 This program is really just a demo showing how to use the
 differential equation solver rke_solve written by Francois
 Pinard, based on a modified form of the 4th order Runge-Kutta
 method, which employs the error checking method of R. England
 1969.

 The output consists of unformated C-style binary floats, of
 either pairs or triplets as specified by the "mode" paramerter.

 Examples:
 x is the population
 verhulst stepmax=2000 mode=x | suaddhead ns=2000 | suxwigb &
 y is dx/dt, the rate of growth of the population
 verhulst stepmax=2000 mode=y | suaddhead ns=2000 | suxwigb &

 In the Verhulst equation, a1 is the reproduction rate and
 a2 is the carrying capacity
  x'(t) = a1 * x * ( 1 - x/a2 )


 The verhulst equation describes a simplified model of a population
 reproducing in an environment with limited resources,
 and are given by the autonomous system of ODE's
y'(t) = a1 * y ( 1 - y/a2 )

 Author: CWP: Aug 2009: John Stockwell
```

```
VTLVZ -- Velocity as function of Time for Linear V(Z);
         writes out a vector of velocity = v0 exp(a t/2)

vtlvz > velfile nt= dt= v0= a=

Required parameters
nt= number of time samples
dt= time sampling interval
v0= velocity at the surface
a= velocity gradient
```

```
 WKBJ - Compute WKBJ ray theoretic parameters, via finite differencing

 wkbj <vfile >tfile nx= nz= xs= zs= [optional parameters]

 Required Parameters:
 <vfile file containing velocities v[nx][nz]
 nx= number of x samples (2nd dimension)
 nz= number of z samples (1st dimension)
 xs= x coordinate of source
 zs= z coordinate of source

 Optional Parameters:
 dx=1.0  x sampling interval
 fx=0.0  first x sample
 dz=1.0  z sampling interval
 fz=0.0  first z sample
 sfile=sfile file containing sigmas sg[nx][nz]
 bfile=bfile file containing incident angles bet[nx][nz]
 afile=afile file containing propagation angles a[nx][nz]

 Notes:
 Traveltimes, propagation angles, sigmas, and incident angles in WKBJ
 by finite differences  in polar coordinates. Traveltimes are calculated
 by upwind scheme; sigmas and incident angles by a Crank-Nicolson scheme.

 Credits:
CWP: Zhenyue Liu, Dave Hale, pre 1992.
```

```
 XY2Z - converts (X,Y)-pairs to spike Z values on a uniform grid

    xy2z < stdin npairs= [optional parameters] >stdout

 Required parameter:
 npairs=  number of pairs input

 Optional parameter:
 scale=1.0 value to scale spikes by
 nx1=100  dimension of first (fast) dimension of output array
 nx2=100  dimension of second (slow) dimension of output array
 x1pad=2 zero padding in x1 dimension
 x2pad=2 zero padding in x2 dimension
 yx=0 assume (x,y) pairs
  =1 assume (y,x) pairs

 Notes:
 Converts ordered (x,y) pairs to spike x1values, of height=scale on a
 uniform grid.


 Credits:
CWP: John Stockwell, Nov 1995
```

```
 Z2XYZ - convert binary floats representing Z-values to ascii
     form in X Y Z ordered triples

    z2xyz <stdin >stdout

 Required parameters:
 n1= number of floats in 1st (fast) dimension

 Optional parameters:

 outpar=/dev/tty  output parameter file

 Notes: This program is useful for converting panels of float
 data (representing evenly spaced z values) to the x y z
 ordered triples required for certain 3D plotting packages.

 Example of NXplot3d usage on a NeXT:
 suplane | sufilter | z2xyz n1=64 > junk.ascii

 Now open junk.ascii as a mesh data file with NXplot3d.
 (NXplot3d is a NeXTStep-only utility for viewing 3d data sets


 Credits:
CWP: John Stockwell based on "b2a" by Jack Cohen
```

```
 SUPSCONTOUR - PostScript CONTOUR plot of a segy data set

 supscontour <stdin [optional parameters] | ...

 Optional parameters:

 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 See the pscontour selfdoc for the remaining parameters.

 On NeXT: supscontour < infile [optional parameters]  | open

 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (pscontour, etc.)
   Jack Cohen and John Stockwell (supscontour, etc.)
      Delphi: Alexander Koek, added support for irregularly spaced traces
      Aarhus University: Morten W. Pedersen copied everything from the xwigb
                         source and just replaced all occurencies of the word

 Notes:
When the number of traces isn't known, we need to count
the traces for pscontour.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

If your system does make a disk file, consider altering
the code to remove the file on interrupt.  This could be
done either by trapping the interrupt with "signal"
or by using the "tmpnam" routine followed by an immediate
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUPSCUBECONTOUR - PostScript CUBE plot of a segy data set

supscubecontour <stdin [optional parameters] | ...

Optional parameters:

n2 is the number of traces per frame.  If not getparred then it
is the total number of traces in the data set.

n3 is the number of frames.  If not getparred then it
is the total number of frames in the data set measured by ntr/n2

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the pscubecontour selfdoc for the remaining parameters.

example:   supscubecontour < infile [optional parameters]  | gv -
```

```
 Credits:

CWP: Dave Hale and Zhiming Li (pscube)
     Jack K. Cohen  (suxmovie)
     John Stockwell (supscubecontour)

 Notes:
When n2 isn't getparred, we need to count the traces
for pscube. Although we compute ntr, we don't allocate a 2-d array
and content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUPSCUBE - PostScript CUBE plot of a segy data set

supscube <stdin [optional parameters] | ...

Optional parameters:

n2 is the number of traces per frame.  If not getparred then it
is the total number of traces in the data set.

n3 is the number of frames.  If not getparred then it
is the total number of frames in the data set measured by ntr/n2

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the pscube selfdoc for the remaining parameters.

On NeXT:     supscube < infile [optional parameters]  | open
```

```
 Credits:

CWP: Dave Hale and Zhiming Li (pscube)
     Jack K. Cohen  (suxmovie)
     John Stockwell (supscube)

 Notes:
When n2 isn't getparred, we need to count the traces
for pscube. Although we compute ntr, we don't allocate a 2-d array
and content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSGRAPH - PostScript GRAPH plot of a segy data set

 supsgraph <stdin [optional parameters] | ...

 Optional parameters:
 style=seismic seismic is default here, =normal is alternate
(see psgraph selfdoc for style definitions)

 nplot is the number of traces (ntr is an acceptable alias for nplot)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to over-ride the header values if
 not.

 See the psgraph selfdoc for the remaining parameters.

 On NeXT:     supsgraph < infile [optional parameters]  | open

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (pswigp, etc.)
   Jack Cohen and John Stockwell (supswigp, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigp.  You can make this value "known"
either by getparring nplot or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSIMAGE - PostScript IMAGE plot of a segy data set

 supsimage <stdin [optional parameters] | ...

 Optional parameters:

 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 See the psimage selfdoc for the remaining parameters.

 On NeXT: supsimage < infile [optional parameters]  | open

 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (psimage, etc.)
   Jack Cohen and John Stockwell (supsimage, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for psimage.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUPSMAX - PostScript of the MAX, min, or absolute max value on each trace
    of a SEGY (SU) data set

  supsmax <stdin >postscript file [optional parameters]

Optional parameters:
mode=max max value
 =min min value
 =abs absolute max value

n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the sumax selfdoc for additional parameter.
See the psgraph selfdoc for the remaining parameters.
```

```
 Credits:

CWP: John Stockwell, based on Jack Cohen's SU JACKet

 Notes:
When the number of traces isn't known, we need to count
the traces for psgraph.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.
```

```
SUPSMOVIE - PostScript MOVIE plot of a segy data set

supsmovie <stdin [optional parameters] | ...

Optional parameters:

n2 is the number of traces per frame.  If not getparred then it
is the total number of traces in the data set.

n3 is the number of frames.  If not getparred then it
is the total number of frames in the data set measured by ntr/n2

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the psmovie selfdoc for the remaining parameters.

On NeXT:     supsmovie < infile [optional parameters]  | open
```

```
 Credits:

CWP: Dave Hale and Zhiming Li (psmovie)
     Jack K. Cohen  (suxmovie)
     John Stockwell (supsmovie)

 Notes:
When n2 isn't getparred, we need to count the traces
for psmovie.  In this case:
we are using tmpfile because on many machines it is
implemented as a memory area instead of a disk file.
Although we compute ntr, we don't allocate a 2-d array
and content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSWIGB - PostScript Bit-mapped WIGgle plot of a segy data set

 supswigb <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
 vsp (same as normal with axis 2 reversed)
 Note: vsp requires use of a keyword

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type   sukeyword -o   to see the complete list of SU keywords.

 This program is really just a wrapper for the plotting program: pswigb
 See the pswigb selfdoc for the remaining parameters.
```

```
 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2, keyword (if set)

 Credits:

CWP: Dave Hale and Zhiming Li (pswigb, etc.)
   Jack Cohen and John Stockwell (supswigb, etc.)
      Delphi: Alexander Koek, added support for irregularly spaced traces

Modified by Brian Zook, Southwest Research Institute, to honor
 scale factors, added vsp style

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigb.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSWIGP - PostScript Polygon-filled WIGgle plot of a segy data set

 supswigp <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
 vsp (same as normal with axis 2 reversed)
 Note: vsp requires use of a keyword

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type   sukeyword -o   to see the complete list of SU keywords.

 This program is really just a wrapper for the plotting program: pswigp
 See the pswigp selfdoc for the remaining parameters.
```

```
 On NeXT: supswigp < infile [optional parameters]  | open

 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2, key specified by key

 Credits:

CWP: Dave Hale and Zhiming Li (pswigp, etc.)
   Jack Cohen and John Stockwell (supswigp, etc.)
Delphi: Alexander Koek, added support for irregularly spaced traces

Modified by Brian Zook, Southwest Research Institute, to honor
 scale factors, added vsp style

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigp.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

If your system does make a disk file, consider altering
the code to remove the file on interrupt.  This could be
done either by trapping the interrupt with "signal"
or by using the "tmpnam" routine followed by an immediate
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXCONTOUR - X CONTOUR plot of Seismic UNIX tracefile via vector plot call

 suxwigb <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type   sukeyword -o   to see the complete list of SU keywords.

 This program is really just a wrapper for the plotting program: xcontour
 See the xcontour selfdoc for the remaining parameters.



 Credits:
```

```
CWP: Dave Hale and Zhiming Li (xwigb, etc.)
   Jack Cohen and John Stockwell (suxwigb, etc.)
      Delphi: Alexander Koek, added support for irregularly spaced traces
      Aarhus University: Morten W. Pedersen copied everything from the xwigb
                         source and just replaced all occurencies of the word
                         xwigb with xcountour ;-)

 Notes:
When the number of traces isn't known, we need to count
the traces for xcontour.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXGRAPH - X-windows GRAPH plot of a segy data set

 suxgraph <stdin [optional parameters] | ...

 Optional parameters:
 style=seismic seismic is default here, =normal is alternate
(see xgraph selfdoc for style definitions)

 nplot= number of traces (ntr is an acceptable alias for nplot)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to over-ride the header values if
 not.

 See the xgraph selfdoc for the remaining parameters.

 On NeXT:     suxgraph < infile [optional parameters]  | open

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (pswigp, etc.)
   Jack Cohen and John Stockwell (supswigp, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigp.  You can make this value "known"
either by getparring nplot or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUXIMAGE - X-windows IMAGE plot of a segy data set

suximage infile= [optional parameters] | ...  (direct I/O)
 or
suximage <stdin [optional parameters] | ... (sequential I/O)

Optional parameters:

infile=NULL SU data to be ploted, default stdin with sequential access
            if 'infile' provided, su data read by (fast) direct access

     with ftr,dtr and n2 suximage will pass a subset of data
     to the plotting program-ximage:
ftr=1       First trace to be plotted
dtr=1       Trace increment to be plotted
n2=tr.ntr   (Max) number of traces to be plotted (ntr is an alias for n2)
     Priority: first try to read from parameter line;
       if not provided, check trace header tr.ntr;
       if still not provided, figure it out using ftello

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set or was set to 0)

key= key for annotating d2 (slow dimension)
 If annotation is not at proper increment, try
 setting d2; only first trace's key value is read

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0             =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()
```

```
 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 See the ximage selfdoc for the remaining parameters.


 Credits:

CWP: Dave Hale and Zhiming Li (ximage, etc.)
   Jack Cohen and John Stockwell (suximage, etc.)
MTU: David Forel, June 2004, added key for annotating d2
      ConocoPhillips: Zhaobo Meng, Dec 2004, added direct I/O

 Notes:

      When provide ftr and dtr and infile, suximage can be used to plot
      multi-dimensional volumes efficiently.  For example, for a Offset-CDP
      dataset with 32 offsets, the command line
      suximage infile=volume3d.su ftr=1 dtr=32 ... &
      will display the zero-offset common offset data with ranrom access.
      It is highly recommend to use infile= to view large datasets, since
      using stdin only allows sequential access, which is very slow for
      large datasets.

When the number of traces isn't known, we need to count
the traces for ximage.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXMAX - X-windows graph of the MAX, min, or absolute max value on
each trace of a SEGY (SU) data set

   suxmax <stdin [optional parameters]

 Optional parameters:
 mode=max max value
  =min min value
  =abs absolute max value

 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to over-ride the header values if
 not.

 See the sumax selfdoc for additional parameter.
 See the xgraph selfdoc for the remaining parameters.
```

```
 Credits:

CWP: John Stockwell, based on Jack Cohen's SU JACKet

 Notes:
When the number of traces isn't known, we need to count
the traces for xgraph.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.
```

```
SUXMOVIE - X MOVIE plot of a 2D or 3D segy data set

suxmovie <stdin [optional parameters]

Optional parameters:

n1=tr.ns                number of samples per trace
ntr=tr.ntr            number of traces in the data set
n2=tr.shortpad or tr.ntr number of traces in inline direction
n3=ntr/n2            number of traces in crossline direction

d1=tr.d1 or tr.dt/10^6    sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2     sampling interval in the slow dimension
  =1.0  (if not set)

d3=1.0     sampling interval in the slowest dimension

f1=tr.f1 or 0.0       first sample in the z dimension
f2=tr.f2 or 1.0           first sample in the x dimension
f3=1.0

mode=0          0= x,z slice movie through y dimension (in line)
                1= y,z slice movie through x dimension (cross line)
                2= x,y slice movie through z dimension (time slice)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Notes:
For seismic data, the "fast dimension" is either time or
depth and the "slow dimension" is usually trace number.
The 3D data set is expected to have n3 sets of n2 traces representing
the horizontal coverage of n2*d2 in x  and n3*d3 in y direction.

The data is read to memory with and piped to xmovie with the
respective sampling parameters.
```

See the xmovie selfdoc for the remaining parameters and X functions.

```
 SUXPICKER - X-windows  WIGgle plot PICKER of a segy data set

 suxpicker <stdin [optional parameters] | ...

 Optional parameters:

 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information


 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type  sukeyword -o to see the complete list of SU keywords.
```

```
 See the xpicker selfdoc for the remaining parameters.


 Credits:

CWP: Dave Hale and Zhiming Li (xpicker, etc.)
   Jack Cohen and John Stockwell (suxpicker, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for xpicker.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

If your system does make a disk file, consider altering
the code to remove the file on interrupt.  This could be
done either by trapping the interrupt with "signal"
or by using the "tmpnam" routine followed by an immediate
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXWIGB - X-windows Bit-mapped WIGgle plot of a segy data set
 This is a modified suxwigb that uses the depth or coordinate scaling
 when such values are used as keys.

 suxwigb <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0 (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
 vsp (same as normal with axis 2 reversed)
 Note: vsp requires use of a keyword
 verbose=0              =1 to print some useful information


 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type  sukeyword -o to see the complete list of SU keywords.
```

```
 This program is really just a wrapper for the plotting program: xwigb
 See the xwigb selfdoc for the remaining parameters.


 Credits:

CWP: Dave Hale and Zhiming Li (xwigb, etc.)
   Jack Cohen and John Stockwell (suxwigb, etc.)
Delphi: Alexander Koek, added support for irregularly spaced traces

Modified by Brian Zook, Southwest Research Institute, to honor
 scale factors, added vsp style

 Notes:
When the number of traces isn't known, we need to count
the traces for xwigb.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
PSBBOX - change BoundingBOX of existing PostScript file

psbbox < PostScriptfile [optional parameters] > PostScriptfile

Optional Parameters:
llx= new llx
lly= new lly
urx= new urx
ury= new ury
verbose=1 =1 for info printed on stderr (0 for no info)
```

```
PSCONTOUR - PostScript CONTOURing of a two-dimensional function f(x1,x2)

pscontour n1= [optional parameters] <binaryfile >postscriptfile

Required Parameters:
n1                      number of samples in 1st (fast) dimension

Optional Parameters:
d1=1.0                  sampling interval in 1st dimension
f1=d1                   first sample in 1st dimension
x1=f1,f1+d1,...         array of monotonic sampled values in 1st dimension
n2=all                  number of samples in 2nd (slow) dimension
d2=1.0                  sampling interval in 2nd dimension
f2=d2                   first sample in 2nd dimension
x2=f2,f2+d2,...         array of monotonic sampled values in 2nd dimension
nc=5                    number of contour values
dc=(zmax-zmin)/nc       contour interval
fc=min+dc               first contour
c=fc,fc+dc,...          array of contour values
cwidth=1.0,...          array of contour line widths
cgray=0.0,...           array of contour grays (0.0=black to 1.0=white)
ccolor=none,...         array of contour colors; none means use cgray
cdash=0.0,...           array of dash spacings (0.0 for solid)
labelcf=1               first labeled contour (1,2,3,...)
labelcper=1             label every labelcper-th contour
nlabelc=nc              number of labeled contours (0 no contour label)
nplaces=6               number of decimal places in contour label
xbox=1.5                offset in inches of left side of axes box
ybox=1.5                offset in inches of bottom side of axes box
wbox=6.0                width in inches of axes box
hbox=8.0                height in inches of axes box
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
```

```
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelfont=Helvetica     font name for axes labels
labelsize=18            font size for axes labels
title=                  title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24            font size for title
labelcfont=Helvetica-Bold font name for contour labels
labelcsize=6            font size of contour labels
labelccolor=black       color of contour labels
titlecolor=black        color of title
axescolor=black         color of axes
gridcolor=black         color of grid
axeswidth=1             width (in points) of axes
ticwidth=axeswidth      width (in points) of tic marks
gridwidth=axeswidth     width (in points) of grid lines
style=seismic           normal (axis 1 horizontal, axis 2 vertical) or
                        seismic (axis 1 vertical, axis 2 horizontal)

Note.
The line width of unlabeled contours is designed as a quarter of that
of labeled contours.

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic
type:   sudoc pscontour    for more information
```

Notes:

```
For nice even-numbered contours, use the parameters  fc and dc

Example: if the range of the z values of a data set range between
approximately -1000 and +1000, then use fc=-1000 nc=10 and dc=100
to get contours spaced by even 100's.



AUTHOR:  Dave Hale, Colorado School of Mines, 05/29/90
MODIFIED:  Craig Artley, Colorado School of Mines, 08/30/91
           BoundingBox moved to top of PostScript output
MODIFIED:  Zhenyue Liu, Colorado School of Mines, 08/26/93
     Values are labeled on contours
MODIFIED:  Craig Artley, Colorado School of Mines, 12/16/93
           Added color options (Courtesy of Dave Hale, Advance Geophysical).
Modified: Morten Wendell Pedersen, Aarhus University, 23/3-97
          Added ticwidth,axeswidth, gridwidth parameters
```

```
 PSCCONTOUR - PostScript Contour plot of a data CUBE

 pscubecontour n1= n2= n3= [optional parameters] <binaryfile >postscriptfile
    or
 pscubecontour n1= n2= n3= front= side= top= [optional parameters] >postscriptfile

 Data formats supported:
1. Entire cube read from stdin (n1*n2*n3 floats) [default format]
2. Faces read from stdin (n1*n2 floats for front, followed by n1*n3
   floats for side, and n2*n3 floats for top) [specify faces=1]
3. Faces read from separate data files [specify filenames]

 Required Parameters:
 n1                    number of samples in 1st (fastest) dimension
 n2                    number of samples in 2nd dimension
 n3                    number of samples in 3rd (slowest) dimension

 Optional Parameters:
 front                 name of file containing front panel
 side                  name of file containing side panel
 top                   name of file containing top panel
 faces=0               =1 to read faces from stdin (data format 2)
 d1=1.0                sampling interval in 1st dimension
 f1=0.0                first sample in 1st dimension
 d2=1.0                sampling interval in 2nd dimension
 f2=0.0                first sample in 2nd dimension
 d3=1.0                sampling interval in 3rd dimension
 f3=0.0                first sample in 3rd dimension
 d1s=1.0               factor by which to scale d1 before imaging
 d2s=1.0               factor by which to scale d2 before imaging
 d3s=1.0               factor by which to scale d3 before imaging
 nc=5                  number of contour values
 dc=(zmax-zmin)/nc     contour interval
 fc=min+dc             first contour
 c=fc,fc+dc,...        array of contour values
 cwidth=1.0,...        array of contour line widths
 cgray=0.0,...         array of contour grays (0.0=black to 1.0=white)
 ccolor=none,...       array of contour colors; none means use cgray
 cdash=0.0,...         array of dash spacings (0.0 for solid)
 labelcf=1             first labeled contour (1,2,3,...)
 labelcper=1           label every labelcper-th contour
 nlabelc=nc            number of labeled contours (0 no contour label)
 nplaces=6             number of decimal places in contour label
```

```
xbox=1.5                offset in inches of left side of axes box
ybox=1.5                offset in inches of bottom side of axes box
size1=4.0               size in inches of 1st axes (vertical)
size2=4.0               size in inches of 2nd axes (horizontal)
size3=3.0               size in inches of 3rd axes (projected)
angle=45                projection angle of cube in degrees (0<angle<90)
                        (angle between 2nd axis and projected 3rd axis)
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
x2beg=x2min             value at which axis 2 begins
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
x3end=x3max             value at which axis 3 ends
d3num=0.0               numbered tic interval on axis 3 (0.0 for automatic)
f3num=x3min             first numbered tic on axis 3 (used if d3num not 0.0)
n3tic=1                 number of tics per numbered tic on axis 3
grid3=none              grid lines on axis 3 - none, dot, dash, or solid
label3=                 label on axis 3
labelfont=Helvetica     font name for axes labels
labelsize=18            font size for axes labels
title=                  title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24            font size for title
titlecolor=black        color of title
labelcfont=Helvetica-Bold font name for contour labels
labelcsize=6            font size of contour labels
labelccolor=black       color of contour labels
axescolor=black         color of axes
gridcolor=black         color of grid

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Note: The values of x1beg=x1min, x2end=x2max and x3beg=x3min cannot
be changed.
```

```
Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic



(Original codes pscontour and pscube)

AUTHOR:  Craig Artley, Colorado School of Mines, 03/12/93
NOTE:  Original written by Zhiming Li & Dave Hale, CSM, 07/01/90
 Completely rewritten, the code now bears more similarity to
 psimage than the previous pscube.  Faces of cube now rendered
 as three separate images, rather than as a single image.  The
 output no longer suffers from stretching artifacts, and the
 code is simpler.  -Craig
MODIFIED:  Craig Artley, Colorado School of Mines, 12/17/93
   Added color options.

PSCCONTOUR: mashed together from pscube and pscontour
to generate 3d contour plots by Claudia Vanelle, Institute of Geophysics,
University of Hamburg, Germany somewhen in 2000

PSCUBE was "merged" with PSCONTOUR to create PSCUBECONTOUR
by Claudia Vanelle, Applied Geophysics Group Hamburg
somewhen in 2000
```

```
 PSCUBE - PostScript image plot with Legend of a data CUBE

 pscube n1= n2= n3= [optional parameters] <binaryfile >postscriptfile
    or
 pscube n1= n2= n3= front= side= top= [optional parameters] >postscriptfile

 Data formats supported:
1. Entire cube read from stdin (n1*n2*n3 floats) [default format]
2. Faces read from stdin (n1*n2 floats for front, followed by n1*n3
   floats for side, and n2*n3 floats for top) [specify faces=1]
3. Faces read from separate data files [specify filenames]

 Required Parameters:
 n1                     number of samples in 1st (fastest) dimension
 n2                     number of samples in 2nd dimension
 n3                     number of samples in 3rd (slowest) dimension

 Optional Parameters:
 front                  name of file containing front panel
 side                   name of file containing side panel
 top                    name of file containing top panel
 faces=0                =1 to read faces from stdin (data format 2)
 d1=1.0                 sampling interval in 1st dimension
 f1=0.0                 first sample in 1st dimension
 d2=1.0                 sampling interval in 2nd dimension
 f2=0.0                 first sample in 2nd dimension
 d3=1.0                 sampling interval in 3rd dimension
 f3=0.0                 first sample in 3rd dimension
 perc=100.0             percentile used to determine clip
 clip=(perc percentile) clip used to determine bclip and wclip
 bperc=perc             percentile for determining black clip value
 wperc=100.0-perc       percentile for determining white clip value
 bclip=clip             data values outside of [bclip,wclip] are clipped
 wclip=-clip            data values outside of [bclip,wclip] are clipped
 brgb=0.0,0.0,0.0       red, green, blue values corresponding to black
 wrgb=1.0,1.0,1.0       red, green, blue values corresponding to white
 bhls=0.0,0.0,0.0       hue, lightness, saturation corresponding to black
 whls=0.0,1.0,0.0       hue, lightness, saturation corresponding to white
 bps=12                 bits per sample for color plots, either 12 or 24
 d1s=1.0                factor by which to scale d1 before imaging
 d2s=1.0                factor by which to scale d2 before imaging
 d3s=1.0                factor by which to scale d3 before imaging
 verbose=1              =1 for info printed on stderr (0 for no info)
```

```
xbox=1.5                offset in inches of left side of axes box
ybox=1.5                offset in inches of bottom side of axes box
size1=4.0               size in inches of 1st axes (vertical)
size2=4.0               size in inches of 2nd axes (horizontal)
size3=3.0               size in inches of 3rd axes (projected)
angle=45                projection angle of cube in degrees (0<angle<90)
                        (angle between 2nd axis and projected 3rd axis)
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
x2beg=x2min             value at which axis 2 begins
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
x3end=x3max             value at which axis 3 ends
d3num=0.0               numbered tic interval on axis 3 (0.0 for automatic)
f3num=x3min             first numbered tic on axis 3 (used if d3num not 0.0)
n3tic=1                 number of tics per numbered tic on axis 3
grid3=none              grid lines on axis 3 - none, dot, dash, or solid
label3=                 label on axis 3
labelfont=Helvetica     font name for axes labels
labelsize=18            font size for axes labels
title=                  title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24            font size for title
titlecolor=black        color of title
axescolor=black         color of axes
gridcolor=black         color of grid
legend=0                =1 display the color scale
                        if ==1, resize xbox,ybox,width,height
lstyle=vertleft        Vertical, axis label on left side
                        =vertright (Vertical, axis label on right side)
                        =horibottom (Horizontal, axis label on bottom)
units=                  unit label for legend
legendfont=times_roman10    font name for title
following are defaults for lstyle=0. They are changed for other lstyles
lwidth=1.2              colorscale (legend) width in inches
lheight=height/3        colorscale (legend) height in inches
```

```
lx=1.0                  colorscale (legend) x-position in inches
ly=(height-lheight)/2+xybox    colorscale (legend) y-position in pixels
lbeg= lmin or wclip-5*perc    value at which legend axis begins
lend= lmax or bclip+5*perc    value at which legend axis ends
ldnum=0.0      numbered tic interval on legend axis (0.0 for automatic)
lfnum=lmin     first numbered tic on legend axis (used if d1num not 0.0)
lntic=1        number of tics per numbered tic on legend axis
lgrid=none     grid lines on legend axis - none, dot, dash, or solid

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic
Note: The values of x1beg=x1min, x2end=x2max and x3beg=x3min cannot
be changed.
```

```
PSEPSI - add an EPSI formatted preview bitmap to an EPS file

psepsi <epsfile >epsifile

Note:
This application requires
(1) that gs (the Ghostscript interpreter) exist, and
(2) that the input EPS file contain a BoundingBox and EndComments.
Ghostscript is used to build the preview bitmap, which is then
merged with the input EPS file to make the output EPSI file.
```

```
 PSGRAPH - PostScript GRAPHer
 Graphs n[i] pairs of (x,y) coordinates, for i = 1 to nplot.

 psgraph n= [optional parameters] <binaryfile >postscriptfile

 Required Parameters:
 n                      array containing number of points per plot

 Data formats supported:
1.a. x1,y1,x2,y2,...,xn,yn
  b. x1,x2,...,xn,y1,y2,...,yn (must set pairs=0)
2.   y1,y2,...,yn (must give non-zero d1[]=)
3.   x1,x2,...,xn (must give non-zero d2[]=)
4.   nil (must give non-zero d1[]= and non-zero d2[]=)
  The formats may be repeated and mixed in any order, but if
  formats 2-4 are used, the d1 and d2 arrays must be specified including
  d1[]=0.0 d2[]=0.0 entries for any internal occurences of format 1.
  Similarly, the pairs array must contain place-keeping entries for
  plots of formats 2-4 if they are mixed with both formats 1.a and 1.b.
  Also, if formats 2-4 are used with non-zero f1[] or f2[] entries, then
  the corresponding array(s) must be fully specified including f1[]=0.0
  and/or f2[]=0.0 entries for any internal occurences of format 1 or
  formats 2-4 where the zero entries are desired.

  Available colors are all the common ones and many more. The complete
  list of 68 colors is in the file $CWPROOT/src/psplot/basic.c.

 Optional Parameters:
 nplot=number of n's    number of plots
 d1=0.0,...             x sampling intervals (0.0 if x coordinates input)
 f1=0.0,...             first x values (not used if x coordinates input)
 d2=0.0,...             y sampling intervals (0.0 if y coordinates input)
 f2=0.0,...             first y values (not used if y coordinates input)
 pairs=1,...            =1 for data pairs in format 1.a, =0 for format 1.b
 linewidth=1.0,...      line widths (in points) (0.0 for no lines)
 linegray=0.0,...       line gray levels (black=0.0 to white=1.0)
 linecolor=none,...     line colors; none means use linegray
                        Typical use: linecolor=red,yellow,blue,...
 lineon=1.0,...         length of line segments for dashed lines (in points)
 lineoff=0.0,...        spacing between dashes (0.0 for solid line)
 mark=0,1,2,3,...       indices of marks used to represent plotted points
 marksize=0.0,0.0,...   size of marks (0.0 for no marks)
 xbox=1.5               offset in inches of left side of axes box
```

```
ybox=1.5                offset in inches of bottom side of axes box
wbox=6.0                width in inches of axes box
hbox=8.0                height in inches of axes box
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelfont=Helvetica     font name for axes labels
labelsize=18            font size for axes labels
title=                  title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24            font size for title
titlecolor=black        color of title
axescolor=black         color of axes
gridcolor=black         color of grid
axeswidth=1             width (in points) of axes
ticwidth=axeswidth      width (in points) of tic marks
gridwidth=axeswidth     width (in points) of grid lines
style=normal            normal (axis 1 horizontal, axis 2 vertical) or
                        seismic (axis 1 vertical, axis 2 horizontal)
reverse=0               =1 to reverse sequence of plotting curves      ",
Note: n1 and n2 are acceptable aliases for n and nplot, respectively.

mark index:
1. asterisk
2. x-cross
3. open triangle
4. open square
5. open circle
6. solid triangle
7. solid square
8. solid circle
```

```
All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Example:
psgraph n=50,100,20 d1=2.5,1,0.33 <datafile >psfile
 plots three curves with equally spaced x values in one plot frame
 x1-coordinates are x1(i) = f1+i*d1 for i = 1 to n (f1=0 by default)
 number of x2's and then x2-coordinates for each curve are read
 sequentially from datafile.

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic
```

```
PSIMAGE - PostScript IMAGE plot of a uniformly-sampled function f(x1,x2)

psimage n1= [optional parameters] <binaryfile >postscriptfile

Required Parameters:
n1  number of samples in 1st (fast) dimension

Optional Parameters:
d1=1.0  sampling interval in 1st dimension
f1=0.0  first sample in 1st dimension
n2=all  number of samples in 2nd (slow) dimension
d2=1.0  sampling interval in 2nd dimension
f2=0.0  first sample in 2nd dimension
perc=100.0  percentile used to determine clip
clip=(perc percentile) clip used to determine bclip and wclip
bperc=perc  percentile for determining black clip value
wperc=100.0-perc  percentile for determining white clip value
bclip=clip  data values outside of [bclip,wclip] are clipped
wclip=-clip  data values outside of [bclip,wclip] are clipped
                      bclip and wclip will be set to be inside
                      [lbeg,lend] if lbeg and/or lend are supplied
threecolor=1  supply 3 color values instead of only two,
                      using not only black and white, but f.e. red,
                      green and blue
brgb=0.0,0.0,0.0  red, green, blue values corresponding to black
grgb=1.0,1.0,1.0  red, green, blue values corresponding to grey
wrgb=1.0,1.0,1.0  red, green, blue values corresponding to white
bhls=0.0,0.0,0.0  hue, lightness, saturation corresponding to black
ghls=0.0,1.0,0.0  hue, lightness, saturation corresponding to grey
whls=0.0,1.0,0.0  hue, lightness, saturation corresponding to white
bps=12  bits per sample for color plots, either 12 or 24
d1s=1.0  factor by which to scale d1 before imaging
d2s=1.0  factor by which to scale d2 before imaging
verbose=1  =1 for info printed on stderr (0 for no info)
xbox=1.5  offset in inches of left side of axes box
ybox=1.5  offset in inches of bottom side of axes box
width=6.0  width in inches of axes box
height=8.0  height in inches of axes box
x1beg=x1min  value at which axis 1 begins
x1end=x1max  value at which axis 1 ends
d1num=0.0  numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min  first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1  number of tics per numbered tic on axis 1
```

```
grid1=none  grid lines on axis 1 - none, dot, dash, or solid
label1=  label on axis 1
x2beg=x2min  value at which axis 2 begins
x2end=x2max  value at which axis 2 ends
d2num=0.0  numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min  first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1  number of tics per numbered tic on axis 2
grid2=none  grid lines on axis 2 - none, dot, dash, or solid
label2=  label on axis 2
labelfont=Helvetica  font name for axes labels
labelsize=18  font size for axes labels
title=  title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24   font size for title
titlecolor=black  color of title
axescolor=black  color of axes
gridcolor=black  color of grid
axeswidth=1            width (in points) of axes
ticwidth=axeswidth     width (in points) of tic marks
gridwidth=axeswidth    width (in points) of grid lines
style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
seismic (axis 1 vertical, axis 2 horizontal)
legend=0          =1 display the color scale
lnice=0                =1 nice legend arrangement
                       (overrides ybox,lx,width,height parameters)
lstyle=vertleft  Vertical, axis label on left side
=vertright (Vertical, axis label on right side)
=horibottom (Horizontal, axis label on bottom)
units=  unit label for legend
legendfont=times_roman10    font name for title
following are defaults for lstyle=0. They are changed for other lstyles
lwidth=1.2  colorscale (legend) width in inches
lheight=height/3       colorscale (legend) height in inches
lx=1.0  colorscale (legend) x-position in inches
ly=(height-lheight)/2+xybox    colorscale (legend) y-position in pixels
lbeg= lmin or wclip-5*perc    value at which legend axis begins
lend= lmax or bclip+5*perc    value at which legend axis ends
ldnum=0.0  numbered tic interval on legend axis (0.0 for automatic)
lfnum=lmin  first numbered tic on legend axis (used if d1num not 0.0)
lntic=1  number of tics per numbered tic on legend axis
lgrid=none  grid lines on legend axis - none, dot, dash, or solid

curve=curve1,curve2,...  file(s) containing points to draw curve(s)
```

```
npair=n1,n2,n2,...            number(s) of pairs in each file
curvecolor=black,..  color of curve(s)
curvewidth=axeswidth  width (in points) of curve(s)
curvedash=0            solid curve(s), dash indices 1,...,11 produce
                       curve(s) with various dash styles

NOTES:
The curve file is an ascii file with the points specified as x1 x2
pairs, one pair to a line.  A "vector" of curve files and curve
colors may be specified as curvefile=file1,file2,etc.
and curvecolor=color1,color2,etc, and the number of pairs of values
in each file as npair=npair1,npair2,... .

You may eliminate the blocky appearance of psimages by adjusting the
d1s= and d2s= parameters, so that psimages appear similar to ximages.

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Some example colormap settings:
red white blue: wrgb=1.0,0,0 grgb=1.0,1.0,1.0 brgb=0,0,1.0
white red blue: wrgb=1.0,1.0,1.0 grgb=1.0,0.0,0.0 brgb=0,0,1.0
blue red white: wrgb=0.0,0.0,1.0 grgb=1.0,0.0,0.0 brgb=1.0,1.0,1.0
red green blue: wrgb=1.0,0,0 grgb=0,1.0,0 brgb=0,0,1.0
orange light-blue green: wrgb=1.0,.5,0 grgb=0,.7,1.0 brgb=0,1.0,0
red light-blue dark blue: wrgb=0.0,0,1.0 grgb=0,1.0,1.0 brgb=0,0,1.0

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic



AUTHOR:  Dave Hale, Colorado School of Mines, 05/29/90
```

```
MODIFIED:  Craig Artley, Colorado School of Mines, 08/30/91
   BoundingBox moved to top of PostScript output
MODIFIED:  Craig Artley, Colorado School of Mines, 12/16/93
   Added color options (Courtesy of Dave Hale, Advance Geophysical).
Modified: Morten Wendell Pedersen, Aarhus University, 23/3-97
          Added ticwidth,axeswidth, gridwidth parameters
MODIFIED: Torsten Schoenfelder, Koeln, Germany 006/07/97
         colorbar (legend) (as in ximage (by Berend Scheffers, Delft))
MODIFIED: Brian K. Macy, Phillips Petroleum, 01/14/99
   Added curve plotting option
MODIFIED: Torsten Schoenfelder, Koeln, Germany 02/10/99
         color scale with interpolation of three colors
MODIFIED: Ekkehart Tessmer, University of Hamburg, Germany, 08/22/2007
         Added dashing option to curve plotting
```

```
 PSLABEL - output PostScript file consisting of a single TEXT string
          on a specified background. (Use with psmerge to label plots.)

 pslabel t= [t=] [optional parameters] > epsfile

Required Parameters (can have multiple specifications to mix fonts):
  t=                 text string to write to output

Optional Parameters:
  f=Times-Bold       font for text string
                      (multiple specifications for each t)
  size=30            size of characters in points (72 points/inch)
  tcolor=black       color of text string
  bcolor=white       color of background box
  nsub=0             number of characters to subtract when
                     computing size of background box (not all
                     characters are the same size so the
                     background box may be too big at times.)

 Example:
 pslabel t="(a) " f=Times-Bold t="h" f=Symbol t="=0.04" nsub=3 > epsfile

 This example yields the PostScript equivalent of the string
  (written here in LaTeX notation) $ (a)\\; \\eta=0.04 $

 Notes:
 This program produces a (color if desired) PostScript text string that
 can be positioned and pasted on a PostScript plot using   psmerge
     see selfdoc of   psmerge for further information.

 Possible fonts:   Helvetica, Helvetica-Oblique, Helvetica-Bold,
  Helvetica-BoldOblique,Times-Roman,Times-Italic,Times-Bold,
  Times-BoldItalic,Courier,Courier-Bold,Courier-Oblique,
  Courier-BoldOblique,Symbol

 Possible colors:  greenyellow,yellow,goldenrod,dandelion,apricot,
  peach,melon,yelloworange,orange,burntorange,bittersweet,redorange,
  mahogany,maroon,brickred,red,orangered,rubinered,wildstrawberry,
  salmon,carnationpink,magenta,violetred,rhodamine,mulberry,redviolet,
  fuchsia,lavender,thistle,orchid,darkorchid,purple,plum,violet,royalpurple,
  blueviolet,periwinkle,cadetblue,cornflowerblue,midnightblue,naveblue,
  royalblue,blue,cerulean,cyan,processblue,skyblue,turquoise,tealblue,
  aquamarine,bluegreen,emerald,junglegreen,seagreen,green,forestgreen,
```

```
 pinegreen,limegreen,yellowgreen,springgreen,olivegreen,rawsienna,sepia,
 brown,tan,white,black,gray

All color specifications may also be made in X Window style Hex format
example:   tcolor=#255

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic



AUTHOR:  John E. Anderson, Visiting Scientist from Mobil, 1994
```

```
PSMANAGER - printer MANAGER for HP 4MV and HP 5Si Mx Laserjet
               PostScript printing

  psmanager < stdin  [optional parameters] > stdout

Required Parameters:
 none
Optional Parameters:
papersize=0 paper size  (US Letter default)
 =1       US Legal
 =2  A4
 =3       11x17

orient=0 paper orientation (Portrait default)
  =1    Landscape

tray=3        printing tray (Bottom tray default)
  =1 tray 1 (multipurpose slot)
  =2 tray 2

manual=0 no manual feed
  =1     (Manual Feed)

media=0 regular paper
  =1     Transparency
  =2     Letterhead
  =3     Card Stock
  =4     Bond
  =5     Labels
  =6     Prepunched
  =7     Recyled
  =8     Preprinted
  =9     Color (printing on colored paper)

Notes:
The option manual=1 implies tray=1. The media options apply
only to the HP LaserJet 5Si MX model printer.

Examples:
  overheads:
   psmanager <  postscript_file manual=1 media=1 | lpr
  labels:
   psmanager <  postscript_file manual=1 media=5 | lpr
```

```
 Notes:  This code was reverse engineered using output from
         the NeXTStep  printer manager.

 Author:  John Stockwell, June 1995, October 1997

 Reference:
PostScript Printer Description File Format Specification,
version 4.2, Adobe Systems Incorporated
```

```
PSMERGE - MERGE PostScript files

psmerge in= [optional parameters] >postscriptfile

Required Parameters:
in=                     postscript file to merge

Optional Parameters:
origin=0.0,0.0          x,y origin in inches
scale=1.0,1.0           x,y scale factors
rotate=0.0              rotation angle in degrees
translate=0.0,0.0       x,y translation in inches

Notes:
More than one set of in, origin, scale, rotate, and translate
parameters may be specified.  Output x and y coordinates are
determined by:
         x = tx + (x-ox)*sx*cos(d) - (y-oy)*sy*sin(d)
         y = ty + (x-ox)*sx*sin(d) + (y-oy)*sy*cos(d)
where tx,ty are translate coordinates, ox,oy are origin coordinates,
sx,sy are scale factors, and d is the rotation angle.  Note that the
order of operations is shift (origin), scale, rotate, and translate.

If the number of occurrences of a given parameter is less than the number
of input files, then the last occurrence of that parameter will apply to
all subsequent files.
```

```
PSMOVIE - PostScript MOVIE plot of a uniformly-sampled function f(x1,x2,x3)

psmovie n1= [optional parameters] <binaryfile >postscriptfile

Required Parameters:
n1                      number of samples in 1st (fast) dimension

Optional Parameters:
d1=1.0                  sampling interval in 1st dimension
f1=0.0                  first sample in 1st dimension
n2=all                  number of samples in 2nd (slow) dimension
d2=1.0                  sampling interval in 2nd dimension
f2=0.0                  first sample in 2nd dimension
perc=100.0              percentile used to determine clip
clip=(perc percentile) clip used to determine bclip and wclip
bperc=perc              percentile for determining black clip value
wperc=100.0-perc        percentile for determining white clip value
bclip=clip              data values outside of [bclip,wclip] are clipped
wclip=-clip             data values outside of [bclip,wclip] are clipped
d1s=1.0                 factor by which to scale d1 before imaging
d2s=1.0                 factor by which to scale d2 before imaging
verbose=1               =1 for info printed on stderr (0 for no info)
xbox=1.0                offset in inches of left side of axes box
ybox=1.5                offset in inches of bottom side of axes box
wbox=6.0                width in inches of axes box
hbox=8.0                height in inches of axes box
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelfont=Helvetica     font name for axes labels
labelsize=18            font size for axes labels
title=                  title of plot
```

```
titlefont=Helvetica-Bold font name for title
titlesize=24           font size for title
style=seismic          normal (axis 1 horizontal, axis 2 vertical) or
                       seismic (axis 1 vertical, axis 2 horizontal)
n3=1                   number of samples in third dimension
title2=                second title to annotate different frames
loopdsp=3              display loop type (1=loop over n1; 2=loop over n2;
                                          3=loop over n3)
d3=1.0                 sampling interval in 3rd dimension
f3=d3                  first sample in 3rd dimension

NeXT: view movie via:   psmovie < infile n1= [optional params...] | open
Note: currently only the Preview Application can handle the multipage
      PostScript output by this program.

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic
```

```
PSWIGB - PostScript WIGgle-trace plot of f(x1,x2) via Bitmap
Best for many traces.  Use PSWIGP (Polygon version) for few traces.

pswigb n1= [optional parameters] <binaryfile >postscriptfile

Required Parameters:
n1                      number of samples in 1st (fast) dimension

Optional Parameters:
d1=1.0                  sampling interval in 1st dimension
f1=0.0                  first sample in 1st dimension
n2=all                  number of samples in 2nd (slow) dimension
d2=1.0                  sampling interval in 2nd dimension
f2=0.0                  first sample in 2nd dimension
x2=f2,f2+d2,...         array of sampled values in 2nd dimension
bias=0.0                data value corresponding to location along axis 2
perc=100.0              percentile for determining clip
clip=(perc percentile) data values < bias+clip and > bias-clip are clipped
xcur=1.0                wiggle excursion in traces corresponding to clip
wt=1                    =0 for no wiggle-trace; =1 for wiggle-trace
va=1                    =0 for no variable-area; =1 for variable-area fill
                        =2 for variable area, solid/grey fill
                        SHADING: 2<= va <=5  va=2 lightgrey, va=5 black",
nbpi=72                 number of bits per inch at which to rasterize
verbose=1               =1 for info printed on stderr (0 for no info)
xbox=1.5                offset in inches of left side of axes box
ybox=1.5                offset in inches of bottom side of axes box
wbox=6.0                width in inches of axes box
hbox=8.0                height in inches of axes box
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
```

```
labelfont=Helvetica    font name for axes labels
labelsize=18           font size for axes labels
title=                 title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24           font size for title
titlecolor=black       color of title
axescolor=black        color of axes
gridcolor=black        color of grid
axeswidth=1            width (in points) of axes
ticwidth=axeswidth     width (in points) of tic marks
gridwidth=axeswidth    width (in points) of grid lines
style=seismic          normal (axis 1 horizontal, axis 2 vertical) or
                       seismic (axis 1 vertical, axis 2 horizontal)
interp=0  no display interpolation
=1 use 8 point sinc interpolation
curve=curve1,curve2,...  file(s) containing points to draw curve(s)
npair=n1,n2,n2,...             number(s) of pairs in each file
curvecolor=black,..    color of curve(s)
curvewidth=axeswidth   width (in points) of curve(s)
curvedash=0            solid curve(s), dash indices 1,...,11 produce
                       curve(s) with various dash styles

Notes:
The interp option may be useful for high nbpi values, however, it
tacitly assumes that the data are purely oscillatory. Non-oscillatory
data will not be represented correctly when this option is set.

The curve file is an ascii file with the points specified as x1 x2
pairs, one pair to a line.  A "vector" of curve files and curve
colors may be specified as curvefile=file1,file2,etc. and
curvecolor=color1,color2,etc, and the number of pairs of values in each
file as npair=npair1,npair2,... .

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCentrySchlbk-Bold"
```

```
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic
```

```
 PSWIGP - PostScript WIGgle-trace plot of f(x1,x2) via Polygons
 Best for few traces.  Use PSWIGB (Bitmap version) for many traces.

 pswigp n1= [optional parameters] <binaryfile >postscriptfile

 Required Parameters:
 n1                     number of samples in 1st (fast) dimension

 Optional Parameters:
 d1=1.0                 sampling interval in 1st dimension
 f1=0.0                 first sample in 1st dimension
 n2=all                 number of samples in 2nd (slow) dimension
 d2=1.0                 sampling interval in 2nd dimension
 f2=0.0                 first sample in 2nd dimension
 x2=f2,f2+d2,...        array of sampled values in 2nd dimension
 bias=0.0               data value corresponding to location along axis 2
 perc=100.0             percentile for determining clip
 clip=(perc percentile) data values < bias+clip and > bias-clip are clipped
 xcur=1.0               wiggle excursion in traces corresponding to clip
 fill=1 =0 for no fill;
>0 for pos. fill;
<0 for neg. fill
                               =2 for pos. fill solid, neg. fill grey
                               =-2for neg. fill solid, pos. fill grey
                       SHADING: 2<=abs(fill)<=5  2=lightgrey 5=black
 linewidth=1.0         linewidth in points (0.0 for thinest visible line)
 tracecolor=black       color of traces; should contrast with background
 backcolor=none         color of background; none means no background
 verbose=1              =1 for info printed on stderr (0 for no info)
 xbox=1.5               offset in inches of left side of axes box
 ybox=1.5               offset in inches of bottom side of axes box
 wbox=6.0               width in inches of axes box
 hbox=8.0               height in inches of axes box
 x1beg=x1min            value at which axis 1 begins
 x1end=x1max            value at which axis 1 ends
 d1num=0.0              numbered tic interval on axis 1 (0.0 for automatic)
 f1num=x1min            first numbered tic on axis 1 (used if d1num not 0.0)
 n1tic=1                number of tics per numbered tic on axis 1
 grid1=none             grid lines on axis 1 - none, dot, dash, or solid
 label1=                label on axis 1
 x2beg=x2min            value at which axis 2 begins
 x2end=x2max            value at which axis 2 ends
 d2num=0.0              numbered tic interval on axis 2 (0.0 for automatic)
```

```
f2num=x2min            first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                number of tics per numbered tic on axis 2
grid2=none             grid lines on axis 2 - none, dot, dash, or solid
label2=                label on axis 2
labelfont=Helvetica    font name for axes labels
labelsize=18           font size for axes labels
title=                 title of plot
titlefont=Helvetica-Bold font name for title
titlesize=24           font size for title
titlecolor=black       color of title
axescolor=black        color of axes
gridcolor=black        color of grid
axeswidth=1            width (in points) of axes
ticwidth=axeswidth     width (in points) of tic marks
gridwidth=axeswidth    width (in points) of grid lines
style=seismic          normal (axis 1 horizontal, axis 2 vertical) or
                       seismic (axis 1 vertical, axis 2 horizontal)

curve=curve1,curve2,...  file(s) containing points to draw curve(s)
npair=n1,n2,n2,...             number(s) of pairs in each file
curvecolor=black,..    color of curve(s)
curvewidth=axeswidth   width (in points) of curve(s)
curvedash=0            solid curve(s), dash indices 1,...,11 produce
                       curve(s) with various dash styles

Note:  linewidth=0.0 produces the thinest possible line on the output.
device.  Thus the result is device-dependent, put generally looks the
best for seismic traces.

The curve file is an ascii file with the points specified as x1 x2 pairs,
one pair to a line.  A "vector" of curve files and curve colors may
be specified as curvefile=file1,file2,etc. and similarly
curvecolor=color1,color2,etc, and the number of pairs of values
in each file as npair=npair1,npair2,... .

All color specifications may also be made in X Window style Hex format
example:   axescolor=#255

Legal font names are:
AvantGarde-Book AvantGarde-BookOblique AvantGarde-Demi AvantGarde-DemiOblique"
Bookman-Demi Bookman-DemiItalic Bookman-Light Bookman-LightItalic
Courier Courier-Bold Courier-BoldOblique Courier-Oblique
Helvetica Helvetica-Bold Helvetica-BoldOblique Helvetica-Oblique
```

```
Helvetica-Narrow Helvetica-Narrow-Bold Helvetica-Narrow-BoldOblique
Helvetica-Narrow-Oblique NewCenturySchlbk-Bold"
NewCenturySchlbk-BoldItalic NewCenturySchlbk-Roman Palatino-Bold
Palatino-BoldItalic Palatino-Italics Palatino-Roman
SanSerif-Bold SanSerif-BoldItalic SanSerif-Roman
Symbol Times-Bold Times-BoldItalic
Times-Roman Times-Italic ZapfChancery-MediumItalic
```

```
LCMAP - List Color Map of root window of default screen

Usage:   lcmap
```

```
LPROP - List PROPerties of root window of default screen of display

Usage:  lprop
```

```
SCMAP - set default standard color map (RGB_DEFAULT_MAP)

Usage:   scmap
```

```
XCONTOUR - X CONTOUR plot of f(x1,x2) via vector plot call

xcontour n1= [optional parameters] <binaryfile [>psplotfile]

X Functionality:
Button 1 Zoom with rubberband box
Button 2 Show mouse (x1,x2) coordinates while pressed
q or Q key Quit
s key Save current mouse (x1,x2) location to file
p or P key Plot current window with pswigb (only from disk files)

Required Parameters:
n1                      number of samples in 1st (fast) dimension

Optional Parameters:
d1=1.0                  sampling interval in 1st dimension
f1=0.0                  first sample in 1st dimension
x1=f1,f1+d1,...         array of sampled values in 1nd dimension
n2=all                  number of samples in 2nd (slow) dimension
d2=1.0                  sampling interval in 2nd dimension
f2=0.0                  first sample in 2nd dimension
x2=f2,f2+d2,...         array of sampled values in 2nd dimension
mpicks=/dev/tty         file to save mouse picks in
verbose=1               =1 for info printed on stderr (0 for no info)
nc=5                    number of contour values
dc=(zmax-zmin)/nc       contour interval
fc=min+dc               first contour
c=fc,fc+dc,...          array of contour values
cwidth=1.0,...          array of contour line widths
ccolor=none,...         array of contour colors; none means use cgray
cdash=0.0,...           array of dash spacings (0.0 for solid)
labelcf=1               first labeled contour (1,2,3,...)
labelcper=1             label every labelcper-th contour
nlabelc=nc              number of labeled contours (0 no contour label)
nplaces=6               number of decimal places in contour labeling
xbox=50                 x in pixels of upper left corner of window
ybox=50                 y in pixels of upper left corner of window
wbox=550                width in pixels of window
hbox=700                height in pixels of window
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
d1num=0.0               numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min             first numbered tic on axis 1 (used if d1num not 0.0)
```

```
n1tic=1                 number of tics per numbered tic on axis 1
grid1=none              grid lines on axis 1 - none, dot, dash, or solid
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
d2num=0.0               numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min             first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1                 number of tics per numbered tic on axis 2
grid2=none              grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelfont=Erg14         font name for axes labels
title=                  title of plot
titlefont=Rom22         font name for title
windowtitle=xwigb       title on window
labelcolor=blue         color for axes labels
titlecolor=red          color for title
gridcolor=blue          color for grid lines
labelccolor=black       color of contour labels                          ",
labelcfont=fixed        font name for contour labels
style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
seismic (axis 1 vertical, axis 2 horizontal)


Notes:
For some reason the contour might slight differ from ones generated
by pscontour (propably due to the pixel nature of the plot
coordinates)

The line width of unlabeled contours is designed as a quarter of that
of labeled contours.


AUTHOR: Morten Wendell Pedersen, Aarhus University

All the coding is based on snippets taken from xwigb, ximage and pscontour
All I have done is put the parts together and put in some bugs ;-)

So credits should go to the authors of these packages...

Caveats and Notes:
The code has been developed under Linux 1.3.20/Xfree 3.1.2E (X11 6.1)
with gcc-2.7.0 But hopefully it should work on other platforms as well

Since all the contours are drawn by Vector plot call's everytime the
```

Window is exposed, the exposing can be darn slow
OOPS This should be history... Now I keep my window content with backing store so I won't have to redraw my window unless I really have to...

Portability Question: I guess I should check if the display supports backingstore and redraw if it doesn't (see DoesBackingStore(3))
I have to be able to use CWBackingStore==Always (other values can be NonUseful and WhenMapped

Since I put the contour labels everytime I draw one contour level the area that contains the label will be crossed by the the next contour lines...
(this bug also seems to be present in pscontour)
To fix this I have to redraw all the labels after been through all the contour calls
Right now I can't see a way to fix this without actually to through the entire label positioning again....Overkill I would say

The relative short length of the contour segments will propably mask the cdash settings
which means it is disposable (but I know how to draw dashed lines :)
A way of fixing this could be to get all connected point and then use XDrawlines or XDrawSegments... just an idea...No idea if it'll work.

I think there is a bug in xContour since my plot coordinates increase North and west ward instead of south and eastward

I need to check the Self Doc so if the right parameters are described
(I have been through it a couple of times but....)

All functions need a heavy cleanup for unused variables
I suppose there is a couple of memory leaks due to missing free'ing of numerous pointers (especially fonts,GC's & colors could be a problem...

I have to browse through the internal pscontour call... basically what I have done is just putting pscontour instead of pswigb... Instead of repositioning the input file pointer (which doesnt work with pipes) one should consider the use of temporary file
or write your zoombox to pscontour (...how one does that?)

Wish List:
The use of cgray's unused until now... I guess I'll need to allocate a gray Colormap -> meaning that the code not will run at other display

than 8 bit Pseudocolor :((with the use of present version of the colormap library (code in $CWPROOT/src/xplot/lib))

The format of contour label should be open for the user..

It could be nice if one could choose to have a pixmap (like ximage) underlying the contours... this should be defined either by the input data or by a seperate file
eg useful for viewing traveltime contours on top a plot of the velocity field

```
XESPB - X windows display of Encapsulated PostScript as a single Bitmap

Usage:   xepsb < stdin

Caveat: your system must have Display PostScript Graphics

NOTE: This program is included as a demo of EPS -> X programming.
See:  xepsp and xpsp these are more advanced versions
```

```
XEPSP - X windows display of Encapsulated PostScript

Usage:   xepsp < stdin

Caveat: your system must have Display PostScript Graphics
Note:
this program is a more advanced version of   xepsb. See also:  xepsp
```

```
XIMAGE - X IMAGE plot of a uniformly-sampled function f(x1,x2)

ximage n1= [optional parameters] <binaryfile

X Functionality:
Button 1 Zoom with rubberband box
Button 2 Show mouse (x1,x2) coordinates while pressed
q or Q key Quit
s key Save current mouse (x1,x2) location to file
p or P key Plot current window with pswigb (only from disk files)
a or page up keys enhance clipping by 10%
c or page down keys reduce clipping by 10%
up,down,left,right keys move zoom window by half width/height
i or +(keypad)  zoom in by factor 2
o or -(keypad)  zoom out by factor 2

... change colormap interactively
r       install next RGB - colormap
R       install previous RGB - colormap
h       install next HSV - colormap
H       install previous HSV - colormap
H       install previous HSV - colormap
(Move mouse cursor out and back into window for r,R,h,H to take effect)

Required Parameters:
n1  number of samples in 1st (fast) dimension

Optional Parameters:
d1=1.0  sampling interval in 1st dimension
f1=0.0  first sample in 1st dimension
n2=all  number of samples in 2nd (slow) dimension
d2=1.0  sampling interval in 2nd dimension
f2=0.0  first sample in 2nd dimension
mpicks=/dev/tty  file to save mouse picks in
perc=100.0  percentile used to determine clip
clip=(perc percentile) clip used to determine bclip and wclip
bperc=perc  percentile for determining black clip value
wperc=100.0-perc  percentile for determining white clip value
bclip=clip  data values outside of [bclip,wclip] are clipped
wclip=-clip  data values outside of [bclip,wclip] are clipped
balance=0  bclip & wclip individually
=1 set them to the same abs value
  if specified via perc (avoids colorbar skew)
```

```
 cmap=hsv\'n\' or rgb\'m\' \'n\' is a number from 0 to 13
\'m\' is a number from 0 to 11
cmap=rgb0 is equal to cmap=gray
cmap=hsv1 is equal to cmap=hue
(compatibility to older versions)
 legend=0         =1 display the color scale
 units= unit label for legend
 legendfont=times_roman10    font name for title
 verbose=1 =1 for info printed on stderr (0 for no info)
 xbox=50 x in pixels of upper left corner of window
 ybox=50 y in pixels of upper left corner of window
 wbox=550 width in pixels of window
 hbox=700 height in pixels of window
 lwidth=16 colorscale (legend) width in pixels
 lheight=hbox/3 colorscale (legend) height in pixels
 lx=3 colorscale (legend) x-position in pixels
 ly=(hbox-lheight)/3   colorscale (legend) y-position in pixels
 x1beg=x1min value at which axis 1 begins
 x1end=x1max value at which axis 1 ends
 d1num=0.0 numbered tic interval on axis 1 (0.0 for automatic)
 f1num=x1min first numbered tic on axis 1 (used if d1num not 0.0)
 n1tic=1 number of tics per numbered tic on axis 1
 grid1=none grid lines on axis 1 - none, dot, dash, or solid
 label1= label on axis 1
 x2beg=x2min value at which axis 2 begins
 x2end=x2max value at which axis 2 ends
 d2num=0.0 numbered tic interval on axis 2 (0.0 for automatic)
 f2num=x2min first numbered tic on axis 2 (used if d2num not 0.0)
 n2tic=1 number of tics per numbered tic on axis 2
 grid2=none grid lines on axis 2 - none, dot, dash, or solid
 label2= label on axis 2
 labelfont=Erg14 font name for axes labels
 title= title of plot
 titlefont=Rom22 font name for title
 windowtitle=ximage title on window
 labelcolor=blue color for axes labels
 titlecolor=red color for title
 gridcolor=blue color for grid lines
 style=seismic         normal (axis 1 horizontal, axis 2 vertical) or
seismic (axis 1 vertical, axis 2 horizontal)
 blank=0 This indicates what portion of the lower range
to blank out (make the background color).  The
value should range from 0 to 1.
```

```
 plotfile=plotfile.ps  filename for interactive ploting (P)
 curve=curve1,curve2,...  file(s) containing points to draw curve(s)
 npair=n1,n2,n2,...            number(s) of pairs in each file
 curvecolor=color1,color2,...  color(s) for curve(s)
 blockinterp=0       whether to use block interpolation (0=no, 1=yes)


 NOTES:
 The curve file is an ascii file with the points  specified as x1 x2
 pairs separated by a space, one pair to a line.  A "vector" of curve
 files and curve colors may be specified as curvefile=file1,file2,etc.
 and curvecolor=color1,color2,etc, and the number of pairs of values
 in each file as npair=npair1,npair2,... .


 AUTHOR:  Dave Hale, Colorado School of Mines, 08/09/90

 Stewart A. Levin, Mobil - Added ps print option

 Brian Zook, Southwest Research Institute, 6/27/96, added blank option

 Toralf Foerster, Baltic Sea Research Institute, 9/15/96, new colormaps

 Berend Scheffers, Delft, colorbar (legend)

 Brian K. Macy, Phillips Petroleum, 11/27/98, added curve plotting option

 G.Klein, GEOMAR Kiel, 2004-03-12, added cursor scrolling and
                                   interactive change of zoom and clipping.

 Zhaobo Meng, ConocoPhillips, 12/02/04, added amplitude display

 Garry Perratt, Geocon, 08/04/05, modified perc handling to center colorbar if balan

INTL2B_block - blocky interpolation of a 2-D array of bytes

intl2b_block blocky interpolation of a 2-D array of bytes

Function Prototype:
void intl2b_block(int nxin, float dxin, float fxin,
int nyin, float dyin, float fyin, unsigned char *zin,
int nxout, float dxout, float fxout,
int nyout, float dyout, float fyout, unsigned char *zout);
```

```
Input:
nxin number of x samples input (fast dimension of zin)
dxin x sampling interval input
fxin first x sample input
nyin number of y samples input (slow dimension of zin)
dyin y sampling interval input
fyin first y sample input
zin array[nyin][nxin] of input samples (see notes)
nxout number of x samples output (fast dimension of zout)
dxout x sampling interval output
fxout first x sample output
nyout number of y samples output (slow dimension of zout)
dyout y sampling interval output
fyout first y sample output

Output:
zout array[nyout][nxout] of output samples (see notes)

Notes:
The arrays zin and zout must passed as pointers to the first element of
a two-dimensional contiguous array of unsigned char values.

Constant extrapolation of zin is used to compute zout for
output x and y outside the range of input x and y.

Author:  James Gunning, CSIRO Petroleum 1999. Hacked from
intl2b() by Dave Hale, Colorado School of Mines, c. 1989-1991
```

```
 XPICKER - X wiggle-trace plot of f(x1,x2) via Bitmap with PICKing

 xpicker n1= [optional parameters] <binaryfile

 X Menu functionality:
    Pick Filename Window default is pick_file
    Load load an existing Pick Filename
    Save save to Pick Filename
    View only/Pick default is View, click to enable Picking
    Add/Delete default is Add, click to delete picks
    Cross off/on default is Cross off, click to enable Crosshairs

 In View mode:
    a or page up keys          enhance clipping by 10%
    c or page down keys        reduce clipping by 10%
    up,down,left,right keys    move zoom window by half width/height
    i or +(keypad)             zoom in by factor 2
    o or -(keypad)             zoom out by factor 2
    l  lock the zoom while moving the coursor
    u  unlock the zoom

 Notes:
Menu selections and toggles ("clicks") are made with button 1
Pick selections are made with button 3
Edit a pick selection by dragging it with button 3 down or
by making a new pick on that trace
Reaching the window limits while moving within changes the zoom
factor in this direction. The use of zoom locking(l) disables it

 Other X Mouse functionality:
 Mouse Button 1 Zoom with rubberbox
 Mouse Button 2 Show mouse (x1,x2) coordinates while pressed

 The following keys are active in View Only mode:

 Required Parameters:
 n1= number of samples in 1st (fast) dimension

 Optional Parameters:
 mpicks=pick_file name of output (input) pick file
 d1=1.0 sampling interval in 1st dimension
 f1=d1   first sample in 1st dimension
 n2=all  number of samples in 2nd (slow) dimension
```

```
 d2=1.0  sampling interval in 2nd dimension
 f2=d2   first sample in 2nd dimension
 x2=f2,f2+d2,... array of sampled values in 2nd dimension
 bias=0.0        data value corresponding to location along axis 2
 perc=100.0      percentile for determining clip
 clip=(perc percentile) data values < bias+clip and > bias-clip are clipped
 xcur=1.0        wiggle excursion in traces corresponding to clip
 wt=1    =0 for no wiggle-trace; =1 for wiggle-trace
 va=1    =0 for no variable-area; =1 for variable-area fill
                      =2 for variable area, solid/grey fill
                      SHADING: 2<=va<=5  va=2 light grey, va=5 black
 verbose=1       =1 for info printed on stderr (0 for no info)
 xbox=50 x in pixels of upper left corner of window
 ybox=50 y in pixels of upper left corner of window
 wbox=550        width in pixels of window
 hbox=700 height in pixels of window
 x1beg=x1min value at which axis 1 begins
 x1end=x1max value at which axis 1 ends
 d1num=0.0  numbered tic interval on axis 1 (0.0 for automatic)
 f1num=x1min first numbered tic on axis 1 (used if d1num not 0.0)
 n1tic=1 number of tics per numbered tic on axis 1
 grid1=none grid lines on axis 1 - none, dot, dash, or solid
 label1= label on axis 1
 x2beg=x2min value at which axis 2 begins
 x2end=x2max value at which axis 2 ends
 d2num=0.0  numbered tic interval on axis 2 (0.0 for automatic)
 f2num=x2min first numbered tic on axis 2 (used if d2num not 0.0)
 n2tic=1 number of tics per numbered tic on axis 2
 grid2=none grid lines on axis 2 - none, dot, dash, or solid
 label2= label on axis 2
 labelfont=Erg14 font name for axes labels
 title= title of plot
 titlefont=Rom22 font name for title
 labelcolor=blue color for axes labels
 titlecolor=red color for title
 gridcolor=blue color for grid lines
 style=seismic normal (axis 1 horizontal, axis 2 vertical) or
       seismic (axis 1 vertical, axis 2 horizontal)
 endian= =0 little endian, =1 big endian
 interp=0 no sinc interpolation
=1 perform sinc interpolation
 x2file= file of "acceptable" x2 values
 x1x2=1 save picks in the order (x1,x2)
```

```
 =0 save picks in the order (x2,x1)

Notes:
Xpicker will try to detect the endian value of the X-display and will
set it to the right value. If it gets obviously wrong information the
endian value will be set to the endian value of the machine that is
given at compile time as the value of CWPENDIAN defined in cwp.h
and set via the compile time flag ENDIANFLAG in Makefile.config.

The only time that you might want to change the value of the endian
variable is if you are viewing traces on a machine with a different
byte order than the machine you are creating the traces on AND if for
some reason the automaic detection of the display byte order fails.
Set endian to that of the machine you are viewing the traces on.

The interp flag is useful for making better quality wiggle trace for
making plots from screen dumps. However, this flag assumes that the
data are purely oscillatory. This option may not be appropriate for all
data sets.

If the x2file=  option is set, then the values from the specified file
will define the set of "acceptable" values of x2 for xpicker to
output. The format is a single column of ASCII values. The number of
specified values is arbitrary.

Such a file can be built from an SU data set via:
    sugethw < sudata key=offset output=geom > x2example

If the value of x2file= is not set, then
xpicker will use the values specified via: x2=.,.,.,. or those that are",
computed from the values of f2=  and d2= as being the "acceptible
values.

See the selfdoc of  suxpicker  for information on using key fields from
the SU trace headers directly.



AUTHOR:  Dave Hale, Colorado School of Mines, 08/09/90
with picking by Wenying Cai of University of Utah.
Endian stuff by Morten Pedersen and John Stockwell of CWP.
Interp stuff by Tony Kocurko of Memorial University of Newfoundland
Modified to include acceptable values by Bill Lutter of the
```

```
    Department of Geology, University of Wisconsin 10/96
 MODIFIED:  P. Michaels, Boise State Univeristy  29 December 2000
            Added solid/grey color scheme for peaks/troughs

 G.Klein, IFG Kiel University, 2003-09-29, added cursor scrolling and
            interactive change of zoom and clipping.

 NOTES:
 Interactive picker improved to allow x-axis of picks to be
 coordinated with "key=header" parameter set in driver routine
 suxpicker. Multiple picks per trace are now allowed.

  Input:
  The command line of suxpicker is unchanged.  The parameter"key=header"
  set in  suxpicker controls a) trace x-axis  displayed via xpicker and
  b) the header values in the first column of a pick file either read in
     or written out from xpicker c) header values expected in optional file
      or written out from xpicker c) header values expected in optional file
     x2file= which reads into xpicker allowable trace x-axis values.

   a) example command line:  suxpicker key=offset < shot10.plotpik

   b) pick file format:
x-axis_value_1 time_1
x-axis_value_2 time_2
x-axis_value_3 time_3
etc.
x-axis_value_n time_n

pick file example:
         1000.000000 0.500000
         2000.000000 1.000000
         3000.000000 1.500000
         4000.000000 2.000000
         5000.000000 2.500000

  c)  format of optional file x2file=:
        header_value_1
     header_value_2
   etc.
   header_val_m

       If file "x2file=" exists in directory from which suxpicker is
```

```
    invoked, then these trace header x-axis values are the only allowable
    x-axis pick values used in the pick "add" or "delete" menu operation.
    Header values do not need to be sorted or 1 to 1 with input traces.
    Further, pick file x-axis values can be read into xpicker via load
    operation without having to match key_pickx1_val x-axis values and
    can also be rewritten out an output pickfile.  As indicated, only
    the "add" and "delete" pick operations are influenced by existence
     of this file.

    Offset header values for "x2file=" can be generated by the
    command line:

    sugethw < su_segyfile key=offset output=geom >  x2examplefile=

    Output: Only change is in format of pick_file (format described above).
    If x2file= file exists then x-axis value of added picks
    will be forced to nearest allowable trace x-axis value (input values
    of x2file= file). If x2file= is not set, then the values of x2
    that are either assigned uniformly to the traces via f2 and d2,
    or by the vector of values of x2=.,.,.,.    will be the "acceptable"
    values.

  Strategy:
 a) malloc() and realloc() used to dynamically allocate memory
for array of x-axis value read in from optional file
x2file=.  This is done in function read_keyval().

b) The pick file dimensions are set in main program via malloc()
and then initialized (*apick)[i].picked = FALSE) in function
init_picks().  The pick file is declared as pick_t **apick, in
order to use realloc() as needed in functions load_picks where the
pick file is read in and edit_picks where picks are added.  The
call to realloc() and further initializing is performed in
function realloc_picks().

c) If x2file= file exists the mouse derived x-axis value
for a pick to be added is checked against allowable x-axis values
to find the closest match via function add_pick called from
edit_picks.  If the pick is to be deleted, first a search is done
to find the closest x-axis value, then the existing pick values
are searched to find the closest radial value (x**2 + t**2) via
function del_pick() invoked from edit_picks.
```

d) Code modifications are limited to above mentioned functions, except for additional parameters passed to functions edit_picks, load_picks, save_picks, and check_buttons.

```
XPSP - Display conforming PostScript in an X-window

xpsp < stdin

Note: this is the most advanced version of xepsb and xepsp.
Caveat: your  system must have Display PostScript Graphics
```

```
 XWIGB - X WIGgle-trace plot of f(x1,x2) via Bitmap

 xwigb n1= [optional parameters] <binaryfile

 X Functionality:
 Button 1 Zoom with rubberband box
 Button 2 Show mouse (x1,x2) coordinates while pressed
 q or Q key Quit
 s key Save current mouse (x1,x2) location to file
 p or P key Plot current window with pswigb (only from disk files)
 a or page up keys enhance clipping by 10%
 c or page down keys reduce clipping by 10%
 up,down,left,right keys move zoom window by half width/height
 i or +(keypad)  zoom in by factor 2
 o or -(keypad)  zoom out by factor 2
    l  lock the zoom while moving the coursor
    u  unlock the zoom
 1,2,...,9 Zoom/Move factor of the window size

 Notes:
Reaching the window limits while moving within changes the zoom
factor in this direction. The use of zoom locking(l) disables it

 Required Parameters:
 n1  number of samples in 1st (fast) dimension

 Optional Parameters:
 d1=1.0  sampling interval in 1st dimension
 f1=0.0  first sample in 1st dimension
 n2=all  number of samples in 2nd (slow) dimension
 d2=1.0  sampling interval in 2nd dimension
 f2=0.0  first sample in 2nd dimension
 x2=f2,f2+d2,...  array of sampled values in 2nd dimension
 mpicks=/dev/tty  file to save mouse picks in
 bias=0.0  data value corresponding to location along axis 2
 perc=100.0  percentile for determining clip
 clip=(perc percentile) data values < bias+clip and > bias-clip are clipped
 xcur=1.0  wiggle excursion in traces corresponding to clip
 wt=1  =0 for no wiggle-trace; =1 for wiggle-trace
 va=1  =0 for no variable-area; =1 for variable-area fill
                        =2 for variable area, solid/grey fill
                        SHADING: 2<=va<=5  va=2 light grey, va=5 black
 verbose=0  =1 for info printed on stderr (0 for no info)
```

```
xbox=50  x in pixels of upper left corner of window
ybox=50  y in pixels of upper left corner of window
wbox=550  width in pixels of window
hbox=700  height in pixels of window
x1beg=x1min  value at which axis 1 begins
x1end=x1max  value at which axis 1 ends
d1num=0.0  numbered tic interval on axis 1 (0.0 for automatic)
f1num=x1min  first numbered tic on axis 1 (used if d1num not 0.0)
n1tic=1  number of tics per numbered tic on axis 1
grid1=none  grid lines on axis 1 - none, dot, dash, or solid
x2beg=x2min  value at which axis 2 begins
x2end=x2max  value at which axis 2 ends
d2num=0.0  numbered tic interval on axis 2 (0.0 for automatic)
f2num=x2min  first numbered tic on axis 2 (used if d2num not 0.0)
n2tic=1  number of tics per numbered tic on axis 2
grid2=none  grid lines on axis 2 - none, dot, dash, or solid
label2=  label on axis 2
labelfont=Erg14  font name for axes labels
title=  title of plot
titlefont=Rom22  font name for title
windowtitle=xwigb  title on window
labelcolor=blue  color for axes labels
titlecolor=red  color for title
gridcolor=blue  color for grid lines
style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
seismic (axis 1 vertical, axis 2 horizontal)
endian=  =0 little endian =1 big endian
interp=0  no interpolation in display
=1 use 8 point sinc interpolation
wigclip=0  If 0, the plot box is expanded to accommodate
the larger wiggles created by xcur>1. If this
flag is non-zero, the extra-large wiggles are
are clipped at the boundary of the plot box.
plotfile=plotfile.ps   filename for interactive ploting (P)
curve=curve1,curve2,...  file(s) containing points to draw curve(s)
npair=n1,n2,n2,...            number(s) of pairs in each file
curvecolor=color1,color2,...  color(s) for curve(s)

Notes:
Xwigb will try to detect the endian value of the X-display and will
set it to the right value. If it gets obviously wrong information the
endian value will be set to the endian value of the machine that is
given at compile time as the value of CWPENDIAN defined in cwp.h
```

```
and set via the compile time flag ENDIANFLAG in Makefile.config.

The only time that you might want to change the value of the endian
variable is if you are viewing traces on a machine with a different
byte order than the machine you are creating the traces on AND if for
some reason the automaic detection of the display byte order fails.
Set endian to that of the machine you are viewing the traces on.

The interp flag is useful for making better quality wiggle trace for
making plots from screen dumps. However, this flag assumes that the
data are purely oscillatory. This option may not be appropriate for all
data sets.

The curve file is an ascii file with the points specified as x1 x2
pairs, separated by a space, one pair to a line.  A "vector" of curve
files and curve colors may be specified as curvefile=file1,file2,etc.
and curvecolor=color1,color2,etc, and the number of pairs of values
in each file as npair=npair1,npair2,... .


AUTHOR:  Dave Hale, Colorado School of Mines, 08/09/90

Endian stuff by:
   Morten Wendell Pedersen, Aarhus University (visiting CSM, June 1995)
 & John Stockwell, Colorado School of Mines, 5 June 1995

Stewart A. Levin, Mobil - Added ps print option
John Stockwell - Added optional sinc interpolation
Stewart A. Levin, Mobil - protect title, labels in pswigb call

Brian J. Zook, SwRI - Added style=normal and wigclip flag

Brian K. Macy, Phillips Petroleum, 11/27/98, added curve plotting option
Curve plotting notes:
MODIFIED:  P. Michaels, Boise State Univeristy  29 December 2000
           Added solid/grey color scheme for peaks/troughs

G.Klein, IFG Kiel University, 2002-09-29, added cursor scrolling and
           interactive change of zoom and clipping.
         IFM-GEOMAR Kiel, 2004-03-12, added zoom locking
         IFM-GEOMAR Kiel, 2004-03-25, interactive plotting fixed
```

```
XGRAPH - X GRAPHer
Graphs n[i] pairs of (x,y) coordinates, for i = 1 to nplot.

xgraph n= [optional parameters] <binaryfile

X Functionality:
q or Q key    Quit

Required Parameters:
n                       array containing number of points per plot

Optional Parameters:
nplot=number of n's     number of plots
d1=0.0,...              x sampling intervals (0.0 if x coordinates input)
f1=0.0,...              first x values (not used if x coordinates input)
d2=0.0,...              y sampling intervals (0.0 if y coordinates input)
f2=0.0,...              first y values (not used if y coordinates input)
pairs=1,...             =1 for data pairs in format 1.a, =0 for format 1.b
linewidth=1,1,...       line widths in pixels (0 for no lines)
linecolor=2,3,...       line colors (black=0, white=1, 2,3,4 = RGB, ...)
mark=0,1,2,3,...        indices of marks used to represent plotted points
marksize=0,0,...        size of marks in pixels (0 for no marks)
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends

Optional resource parameters (defaults taken from resource database):
windowtitle=        title on window
width=                  width in pixels of window
height=                 height in pixels of window
nTic1=                  number of tics per numbered tic on axis 1
grid1=                  grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
nTic2=                  number of tics per numbered tic on axis 2
grid2=                  grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelFont=              font name for axes labels
title=                  title of plot
titleFont=              font name for title
titleColor=             color for title
axesColor=              color for axes
gridColor=              color for grid lines
```

```
style=                  normal (axis 1 horizontal, axis 2 vertical) or
                        seismic (axis 1 vertical, axis 2 horizontal)

Data formats supported:
 1.a. x1,y1,x2,y2,...,xn,yn
   b. x1,x2,...,xn,y1,y2,...,yn
 2. y1,y2,...,yn (must give non-zero d1[]=)
 3. x1,x2,...,xn (must give non-zero d2[]=)
 4. nil (must give non-zero d1[]= and non-zero d2[]=)
  The formats may be repeated and mixed in any order, but if
  formats 2-4 are used, the d1 and d2 arrays must be specified including
  d1[]=0.0 d2[]=0.0 entries for any internal occurences of format 1.
  Similarly, the pairs array must contain place-keeping entries for
  plots of formats 2-4 if they are mixed with both formats 1.a and 1.b.
  Also, if formats 2-4 are used with non-zero f1[] or f2[] entries, then
  the corresponding array(s) must be fully specified including f1[]=0.0
  and/or f2[]=0.0 entries for any internal occurences of format 1 or
  formats 2-4 where the zero entries are desired.

Note: n1 and n2 are acceptable aliases for n and nplot, respectively.

Example:
xgraph n=50,100,20 d1=2.5,1,0.33 <datafile
  plots three curves with equally spaced x values in one plot frame
  x1-coordinates are x1(i) = f1+i*d1 for i = 1 to n (f1=0 by default)
  number of x2's and then x2-coordinates for each curve are read
  sequentially from datafile.
```

```
XGRAPH - X GRAPHer
Graphs n[i] pairs of (x,y) coordinates, for i = 1 to nplot.

xgraph n= [optional parameters] <binaryfile

X Functionality:
q or Q key    Quit

Required Parameters:
n                       array containing number of points per plot

Optional Parameters:
nplot=number of n's     number of plots
d1=0.0,...              x sampling intervals (0.0 if x coordinates input)
f1=0.0,...              first x values (not used if x coordinates input)
d2=0.0,...              y sampling intervals (0.0 if y coordinates input)
f2=0.0,...              first y values (not used if y coordinates input)
pairs=1,...             =1 for data pairs in format 1.a, =0 for format 1.b
linewidth=1,1,...       line widths in pixels (0 for no lines)
linecolor=2,3,...       line colors (black=0, white=1, 2,3,4 = RGB, ...)
mark=0,1,2,3,...        indices of marks used to represent plotted points
marksize=0,0,...        size of marks in pixels (0 for no marks)
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
reverse=0               =1 to reverse sequence of plotting curves ",

Optional resource parameters (defaults taken from resource database):
windowtitle=        title on window
width=                  width in pixels of window
height=                 height in pixels of window
nTic1=                  number of tics per numbered tic on axis 1
grid1=                  grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
nTic2=                  number of tics per numbered tic on axis 2
grid2=                  grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelFont=              font name for axes labels
title=                  title of plot
titleFont=              font name for title
titleColor=             color for title
axesColor=              color for axes
```

```
gridColor=              color for grid lines
style=                  normal (axis 1 horizontal, axis 2 vertical) or
                        seismic (axis 1 vertical, axis 2 horizontal)

Data formats supported:
 1.a. x1,y1,x2,y2,...,xn,yn
   b. x1,x2,...,xn,y1,y2,...,yn
 2. y1,y2,...,yn (must give non-zero d1[]=)
 3. x1,x2,...,xn (must give non-zero d2[]=)
 4. nil (must give non-zero d1[]= and non-zero d2[]=)
  The formats may be repeated and mixed in any order, but if
  formats 2-4 are used, the d1 and d2 arrays must be specified including
  d1[]=0.0 d2[]=0.0 entries for any internal occurences of format 1.
  Similarly, the pairs array must contain place-keeping entries for
  plots of formats 2-4 if they are mixed with both formats 1.a and 1.b.
  Also, if formats 2-4 are used with non-zero f1[] or f2[] entries, then
  the corresponding array(s) must be fully specified including f1[]=0.0
  and/or f2[]=0.0 entries for any internal occurences of format 1 or
  formats 2-4 where the zero entries are desired.
mark index:
1. asterisk
2. x-cross
3. open triangle
4. open square
5. open circle
6. solid triangle
7. solid square
8. solid circle

Note: n1 and n2 are acceptable aliases for n and nplot, respectively.

Example:
xgraph n=50,100,20 d1=2.5,1,0.33 <datafile
  plots three curves with equally spaced x values in one plot frame
  x1-coordinates are x1(i) = f1+i*d1 for i = 1 to n (f1=0 by default)
  number of x2's and then x2-coordinates for each curve are read
  sequentially from datafile.
```

```
XMOVIE - image one or more frames of a uniformly sampled function f(x1,x2)

xmovie n1= n2= [optional parameters] <fileoffloats

X Functionality:
Button 1 Zoom with rubberband box
Button 2  reverse the direction of the movie.
Button 3  stop and start the movie.
q or Q key Quit
s or S key stop display and switch to Step mode
b or B key set frame direction to Backward
f or F key set frame direction to Forward
n or N key same as 'f'
c or C key set display mode to Continuous mode

Required Parameters:
n1=                     number of samples in 1st (fast) dimension
n2=                     number of samples in 2nd (slow) dimension

Optional Parameters:
d1=1.0                  sampling interval in 1st dimension
f1=0.0                  first sample in 1st dimension
d2=1.0                  sampling interval in 2nd dimension
f2=0.0                  first sample in 2nd dimension
perc=100.0              percentile used to determine clip
clip=(perc percentile) clip used to determine bclip and wclip
bperc=perc              percentile for determining black clip value
wperc=100.0-perc        percentile for determining white clip value
bclip=clip              data values outside of [bclip,wclip] are clipped
wclip=-clip             data values outside of [bclip,wclip] are clipped
x1beg=x1min             value at which axis 1 begins
x1end=x1max             value at which axis 1 ends
x2beg=x2min             value at which axis 2 begins
x2end=x2max             value at which axis 2 ends
fframe=1                value corresponding to first frame
dframe=1                frame sampling interval
loop=0                  =1 to loop over frames after last frame is input
                        =2 to run movie back and forth
interp=1                =0 for a non-interpolated, blocky image
verbose=1               =1 for info printed on stderr (0 for no info)
idm=0                   =1 to set initial display mode to stepmode

Optional resource parameters (defaults taken from resource database):
```

```
windowtitle=        title on window and icon
width=                 width in pixels of window
height=                height in pixels of window
nTic1=                 number of tics per numbered tic on axis 1
grid1=                 grid lines on axis 1 - none, dot, dash, or solid
label1=                label on axis 1
nTic2=                 number of tics per numbered tic on axis 2
grid2=                 grid lines on axis 2 - none, dot, dash, or solid
label2=                label on axis 2
labelFont=             font name for axes labels
title=                 title of plot
titleFont=             font name for title
titleColor=            color for title
axesColor=             color for axes
gridColor=             color for grid lines
style=                 normal (axis 1 horizontal, axis 2 vertical) or
                       seismic (axis 1 vertical, axis 2 horizontal)
sleep=                 delay between frames in microseconds

Color options:
cmap=gray     gray, hue, saturation, or default colormaps may be specified
bhue=0        hue mapped to bclip (hue and saturation maps)
whue=240      hue mapped to wclip (hue and saturation maps)
sat=1         saturation (hue map only)
bright=1      brightness (hue and saturation maps)
white=(bclip+wclip)/2  data value mapped to white (saturation map only)

Notes:
Colors are specified using the HSV color wheel model:
  Hue:  0=360=red, 60=yellow, 120=green, 180=cyan, 240=blue, 300=magenta
  Saturation:  0=white, 1=pure color
  Value (brightness):  0=black, 1=maximum intensity
For the saturation mapping (cmap=sat), data values between white and bclip
  are mapped to bhue, with saturation varying from white to the pure color.
  Values between wclip and white are similarly mapped to whue.
For the hue mapping (cmap=hue), data values between wclip and bclip are
  mapped to hues between whue and bhue.  Intermediate hues are found by
  moving counterclockwise around the circle from bhue to whue.  To reverse
  the polarity of the image, exchange bhue and whue.  Equivalently,
  exchange bclip and wclip (setting perc=0 is an easy way to do this).
  Hues in excess of 360 degrees can be specified in order to reach the
  opposite side of the color circle, or to wrap around the circle more
  than once.
```

```
The title string may contain a C printf format string containing a
  conversion character for the frame number.  The frame number is
  computed from dframe and fframe.  E.g., try setting title="Frame %g".
```

```
XRECTS - plot rectangles on a two-dimensional grid

xrects x1min= x1max= x2min= x2max= [optional parameters] <rectangles

Required Parameters:
x1min                   minimum x1 coordinate
x1max                   maximum x1 coordinate
x2min                   minimum x2 coordinate
x2max                   maximum x2 coordinate

Optional Parameters:
color=red               color used for rectangules

Optional resource parameters (defaults taken from resource database):
width=                  width in pixels of window
height=                 height in pixels of window
nTic1=                  number of tics per numbered tic on axis 1
grid1=                  grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
nTic2=                  number of tics per numbered tic on axis 2
grid2=                  grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelFont=              font name for axes labels
title=                  title of plot
titleFont=              font name for title
titleColor=             color for title
axesColor=              color for axes
gridColor=              color for grid lines
style=                  normal (axis 1 horizontal, axis 2 vertical) or
                        seismic (axis 1 vertical, axis 2 horizontal)
```

```
FFTLAB - Motif-X based graphical 1D Fourier Transform

Usage:  fftlab

Caveat: you must have the Motif Developer's package to install this code
```

```
 SUPSCONTOUR - PostScript CONTOUR plot of a segy data set

 supscontour <stdin [optional parameters] | ...

 Optional parameters:

 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 See the pscontour selfdoc for the remaining parameters.

 On NeXT: supscontour < infile [optional parameters]  | open

 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (pscontour, etc.)
   Jack Cohen and John Stockwell (supscontour, etc.)
      Delphi: Alexander Koek, added support for irregularly spaced traces
      Aarhus University: Morten W. Pedersen copied everything from the xwigb
                         source and just replaced all occurencies of the word

 Notes:
When the number of traces isn't known, we need to count
the traces for pscontour.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

If your system does make a disk file, consider altering
the code to remove the file on interrupt.  This could be
done either by trapping the interrupt with "signal"
or by using the "tmpnam" routine followed by an immediate
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUPSCUBECONTOUR - PostScript CUBE plot of a segy data set

supscubecontour <stdin [optional parameters] | ...

Optional parameters:

n2 is the number of traces per frame.  If not getparred then it
is the total number of traces in the data set.

n3 is the number of frames.  If not getparred then it
is the total number of frames in the data set measured by ntr/n2

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the pscubecontour selfdoc for the remaining parameters.

example:   supscubecontour < infile [optional parameters]  | gv -
```

```
 Credits:

CWP: Dave Hale and Zhiming Li (pscube)
     Jack K. Cohen  (suxmovie)
     John Stockwell (supscubecontour)

 Notes:
When n2 isn't getparred, we need to count the traces
for pscube. Although we compute ntr, we don't allocate a 2-d array
and content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUPSCUBE - PostScript CUBE plot of a segy data set

supscube <stdin [optional parameters] | ...

Optional parameters:

n2 is the number of traces per frame.  If not getparred then it
is the total number of traces in the data set.

n3 is the number of frames.  If not getparred then it
is the total number of frames in the data set measured by ntr/n2

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the pscube selfdoc for the remaining parameters.

On NeXT:     supscube < infile [optional parameters]  | open
```

```
 Credits:

CWP: Dave Hale and Zhiming Li (pscube)
     Jack K. Cohen  (suxmovie)
     John Stockwell (supscube)

 Notes:
When n2 isn't getparred, we need to count the traces
for pscube. Although we compute ntr, we don't allocate a 2-d array
and content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSGRAPH - PostScript GRAPH plot of a segy data set

 supsgraph <stdin [optional parameters] | ...

 Optional parameters:
 style=seismic seismic is default here, =normal is alternate
(see psgraph selfdoc for style definitions)

 nplot is the number of traces (ntr is an acceptable alias for nplot)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to over-ride the header values if
 not.

 See the psgraph selfdoc for the remaining parameters.

 On NeXT:     supsgraph < infile [optional parameters]  | open

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (pswigp, etc.)
   Jack Cohen and John Stockwell (supswigp, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigp.  You can make this value "known"
either by getparring nplot or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSIMAGE - PostScript IMAGE plot of a segy data set

 supsimage <stdin [optional parameters] | ...

 Optional parameters:

 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 See the psimage selfdoc for the remaining parameters.

 On NeXT: supsimage < infile [optional parameters]  | open

 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (psimage, etc.)
   Jack Cohen and John Stockwell (supsimage, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for psimage.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUPSMAX - PostScript of the MAX, min, or absolute max value on each trace
    of a SEGY (SU) data set

  supsmax <stdin >postscript file [optional parameters]

Optional parameters:
mode=max max value
 =min min value
 =abs absolute max value

n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the sumax selfdoc for additional parameter.
See the psgraph selfdoc for the remaining parameters.
```

```
 Credits:

CWP: John Stockwell, based on Jack Cohen's SU JACKet

 Notes:
When the number of traces isn't known, we need to count
the traces for psgraph.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.
```

```
SUPSMOVIE - PostScript MOVIE plot of a segy data set

supsmovie <stdin [optional parameters] | ...

Optional parameters:

n2 is the number of traces per frame.  If not getparred then it
is the total number of traces in the data set.

n3 is the number of frames.  If not getparred then it
is the total number of frames in the data set measured by ntr/n2

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set)

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Note that for seismic time domain data, the "fast dimension" is
time and the "slow dimension" is usually trace number or range.
Also note that "foreign" data tapes may have something unexpected
in the d2,f2 fields, use segyclean to clear these if you can afford
the processing time or use d2= f2= to over-ride the header values if
not.

See the psmovie selfdoc for the remaining parameters.

On NeXT:     supsmovie < infile [optional parameters]  | open
```

```
 Credits:

CWP: Dave Hale and Zhiming Li (psmovie)
     Jack K. Cohen  (suxmovie)
     John Stockwell (supsmovie)

 Notes:
When n2 isn't getparred, we need to count the traces
for psmovie.  In this case:
we are using tmpfile because on many machines it is
implemented as a memory area instead of a disk file.
Although we compute ntr, we don't allocate a 2-d array
and content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSWIGB - PostScript Bit-mapped WIGgle plot of a segy data set

 supswigb <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
 vsp (same as normal with axis 2 reversed)
 Note: vsp requires use of a keyword

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type   sukeyword -o   to see the complete list of SU keywords.

 This program is really just a wrapper for the plotting program: pswigb
 See the pswigb selfdoc for the remaining parameters.
```

```
 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2, keyword (if set)

 Credits:

CWP: Dave Hale and Zhiming Li (pswigb, etc.)
   Jack Cohen and John Stockwell (supswigb, etc.)
      Delphi: Alexander Koek, added support for irregularly spaced traces

Modified by Brian Zook, Southwest Research Institute, to honor
 scale factors, added vsp style

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigb.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUPSWIGP - PostScript Polygon-filled WIGgle plot of a segy data set

 supswigp <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
 vsp (same as normal with axis 2 reversed)
 Note: vsp requires use of a keyword

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type   sukeyword -o   to see the complete list of SU keywords.

 This program is really just a wrapper for the plotting program: pswigp
 See the pswigp selfdoc for the remaining parameters.
```

```
 On NeXT: supswigp < infile [optional parameters]  | open

 Trace header fields accessed: ns, ntr, tracr, tracl, delrt, trid,
dt, d1, d2, f1, f2, key specified by key

 Credits:

CWP: Dave Hale and Zhiming Li (pswigp, etc.)
   Jack Cohen and John Stockwell (supswigp, etc.)
Delphi: Alexander Koek, added support for irregularly spaced traces

Modified by Brian Zook, Southwest Research Institute, to honor
 scale factors, added vsp style

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigp.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

If your system does make a disk file, consider altering
the code to remove the file on interrupt.  This could be
done either by trapping the interrupt with "signal"
or by using the "tmpnam" routine followed by an immediate
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXCONTOUR - X CONTOUR plot of Seismic UNIX tracefile via vector plot call

 suxwigb <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type   sukeyword -o   to see the complete list of SU keywords.

 This program is really just a wrapper for the plotting program: xcontour
 See the xcontour selfdoc for the remaining parameters.



 Credits:
```

```
CWP: Dave Hale and Zhiming Li (xwigb, etc.)
   Jack Cohen and John Stockwell (suxwigb, etc.)
      Delphi: Alexander Koek, added support for irregularly spaced traces
      Aarhus University: Morten W. Pedersen copied everything from the xwigb
                         source and just replaced all occurencies of the word
                         xwigb with xcountour ;-)

 Notes:
When the number of traces isn't known, we need to count
the traces for xcontour.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXGRAPH - X-windows GRAPH plot of a segy data set

 suxgraph <stdin [optional parameters] | ...

 Optional parameters:
 style=seismic seismic is default here, =normal is alternate
(see xgraph selfdoc for style definitions)

 nplot= number of traces (ntr is an acceptable alias for nplot)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to over-ride the header values if
 not.

 See the xgraph selfdoc for the remaining parameters.

 On NeXT:     suxgraph < infile [optional parameters]  | open

 Credits:
```

```
CWP: Dave Hale and Zhiming Li (pswigp, etc.)
   Jack Cohen and John Stockwell (supswigp, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for pswigp.  You can make this value "known"
either by getparring nplot or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SUXIMAGE - X-windows IMAGE plot of a segy data set

suximage infile= [optional parameters] | ...  (direct I/O)
 or
suximage <stdin [optional parameters] | ... (sequential I/O)

Optional parameters:

infile=NULL SU data to be ploted, default stdin with sequential access
            if 'infile' provided, su data read by (fast) direct access

     with ftr,dtr and n2 suximage will pass a subset of data
     to the plotting program-ximage:
ftr=1       First trace to be plotted
dtr=1       Trace increment to be plotted
n2=tr.ntr   (Max) number of traces to be plotted (ntr is an alias for n2)
     Priority: first try to read from parameter line;
       if not provided, check trace header tr.ntr;
       if still not provided, figure it out using ftello

d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2 sampling interval in the slow dimension
  =1.0  (if not set or was set to 0)

key= key for annotating d2 (slow dimension)
 If annotation is not at proper increment, try
 setting d2; only first trace's key value is read

f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
  =1.0 for seismic     (if not set)
  =d2 for nonseismic     (if not set)

verbose=0             =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()
```

```
 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 See the ximage selfdoc for the remaining parameters.


 Credits:

CWP: Dave Hale and Zhiming Li (ximage, etc.)
   Jack Cohen and John Stockwell (suximage, etc.)
MTU: David Forel, June 2004, added key for annotating d2
      ConocoPhillips: Zhaobo Meng, Dec 2004, added direct I/O

 Notes:

      When provide ftr and dtr and infile, suximage can be used to plot
      multi-dimensional volumes efficiently.  For example, for a Offset-CDP
      dataset with 32 offsets, the command line
      suximage infile=volume3d.su ftr=1 dtr=32 ... &
      will display the zero-offset common offset data with ranrom access.
      It is highly recommend to use infile= to view large datasets, since
      using stdin only allows sequential access, which is very slow for
      large datasets.

When the number of traces isn't known, we need to count
the traces for ximage.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXMAX - X-windows graph of the MAX, min, or absolute max value on
each trace of a SEGY (SU) data set

   suxmax <stdin [optional parameters]

 Optional parameters:
 mode=max max value
  =min min value
  =abs absolute max value

 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information

 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to over-ride the header values if
 not.

 See the sumax selfdoc for additional parameter.
 See the xgraph selfdoc for the remaining parameters.
```

```
 Credits:

CWP: John Stockwell, based on Jack Cohen's SU JACKet

 Notes:
When the number of traces isn't known, we need to count
the traces for xgraph.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.
```

```
SUXMOVIE - X MOVIE plot of a 2D or 3D segy data set

suxmovie <stdin [optional parameters]

Optional parameters:

n1=tr.ns                number of samples per trace
ntr=tr.ntr            number of traces in the data set
n2=tr.shortpad or tr.ntr number of traces in inline direction
n3=ntr/n2            number of traces in crossline direction

d1=tr.d1 or tr.dt/10^6    sampling interval in the fast dimension
  =.004 for seismic  (if not set)
  =1.0 for nonseismic (if not set)

d2=tr.d2     sampling interval in the slow dimension
  =1.0  (if not set)

d3=1.0     sampling interval in the slowest dimension

f1=tr.f1 or 0.0       first sample in the z dimension
f2=tr.f2 or 1.0           first sample in the x dimension
f3=1.0

mode=0          0= x,z slice movie through y dimension (in line)
                1= y,z slice movie through x dimension (cross line)
                2= x,y slice movie through z dimension (time slice)

verbose=0              =1 to print some useful information

tmpdir=   if non-empty, use the value as a directory path
 prefix for storing temporary files; else if the
         the CWP_TMPDIR environment variable is set use
         its value for the path; else use tmpfile()

Notes:
For seismic data, the "fast dimension" is either time or
depth and the "slow dimension" is usually trace number.
The 3D data set is expected to have n3 sets of n2 traces representing
the horizontal coverage of n2*d2 in x  and n3*d3 in y direction.

The data is read to memory with and piped to xmovie with the
respective sampling parameters.
```

See the xmovie selfdoc for the remaining parameters and X functions.

```
 SUXPICKER - X-windows  WIGgle plot PICKER of a segy data set

 suxpicker <stdin [optional parameters] | ...

 Optional parameters:

 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)

 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic  (if not set)
   =1.0 for nonseismic (if not set)

 d2=tr.d2 sampling interval in the slow dimension
   =1.0  (if not set)

 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension

 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 verbose=0              =1 to print some useful information


 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type  sukeyword -o to see the complete list of SU keywords.
```

```
 See the xpicker selfdoc for the remaining parameters.


 Credits:

CWP: Dave Hale and Zhiming Li (xpicker, etc.)
   Jack Cohen and John Stockwell (suxpicker, etc.)

 Notes:
When the number of traces isn't known, we need to count
the traces for xpicker.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we do have to count the traces, we use the "tmpfile"
routine because on many machines it is implemented
as a memory area instead of a disk file.

If your system does make a disk file, consider altering
the code to remove the file on interrupt.  This could be
done either by trapping the interrupt with "signal"
or by using the "tmpnam" routine followed by an immediate
"remove" (aka "unlink" in old unix).

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
 SUXWIGB - X-windows Bit-mapped WIGgle plot of a segy data set
 This is a modified suxwigb that uses the depth or coordinate scaling
 when such values are used as keys.

 suxwigb <stdin [optional parameters] | ...

 Optional parameters:
 key=(keyword) if set, the values of x2 are set from header field
specified by keyword
 n2=tr.ntr or number of traces in the data set (ntr is an alias for n2)
 d1=tr.d1 or tr.dt/10^6 sampling interval in the fast dimension
   =.004 for seismic (if not set)
   =1.0 for nonseismic (if not set)
 d2=tr.d2 sampling interval in the slow dimension
   =1.0 (if not set)
 f1=tr.f1 or tr.delrt/10^3 or 0.0  first sample in the fast dimension
 f2=tr.f2 or tr.tracr or tr.tracl  first sample in the slow dimension
   =1.0 for seismic     (if not set)
   =d2 for nonseismic     (if not set)

 style=seismic  normal (axis 1 horizontal, axis 2 vertical) or
 vsp (same as normal with axis 2 reversed)
 Note: vsp requires use of a keyword
 verbose=0              =1 to print some useful information


 tmpdir=   if non-empty, use the value as a directory path
  prefix for storing temporary files; else if the
          the CWP_TMPDIR environment variable is set use
          its value for the path; else use tmpfile()

 Note that for seismic time domain data, the "fast dimension" is
 time and the "slow dimension" is usually trace number or range.
 Also note that "foreign" data tapes may have something unexpected
 in the d2,f2 fields, use segyclean to clear these if you can afford
 the processing time or use d2= f2= to override the header values if
 not.

 If key=keyword is set, then the values of x2 are taken from the header
 field represented by the keyword (for example key=offset, will show
 traces in true offset). This permit unequally spaced traces to be plotted.
 Type  sukeyword -o to see the complete list of SU keywords.
```

```
 This program is really just a wrapper for the plotting program: xwigb
 See the xwigb selfdoc for the remaining parameters.


 Credits:

CWP: Dave Hale and Zhiming Li (xwigb, etc.)
   Jack Cohen and John Stockwell (suxwigb, etc.)
Delphi: Alexander Koek, added support for irregularly spaced traces

Modified by Brian Zook, Southwest Research Institute, to honor
 scale factors, added vsp style

 Notes:
When the number of traces isn't known, we need to count
the traces for xwigb.  You can make this value "known"
either by getparring n2 or by having the ntr field set
in the trace header.  A getparred value takes precedence
over the value in the trace header.

When we must compute ntr, we don't allocate a 2-d array,
but just content ourselves with copying trace by trace from
the data "file" to the pipe into the plotting program.
Although we could use tr.data, we allocate a trace buffer
for code clarity.
```

```
SXPLOT - X Window plot a triangulated sloth function s(x1,x2)

sxplot <modelfile [optional parameters]

Optional Parameters:
edgecolor=cyan          color to draw fixed edges
tricolor=yellow         color to draw non-fixed edges of triangles
=none non-fixed edges of triangles are not shown
bclip=minimum sloth     sloth value corresponding to black
wclip=maximum sloth     sloth value corresponding to white
x1beg=x1min             value at which x1 axis begins
x1end=x1max             value at which x1 axis ends
x2beg=x2min             value at which x2 axis begins
x2end=x2max             value at which x2 axis ends
cmap=gray               gray, hue, or default colormaps may be specified

Optional resource parameters (defaults taken from resource database):
width=                  width in pixels of window
height=                 height in pixels of window
nTic1=                  number of tics per numbered tic on axis 1
grid1=                  grid lines on axis 1 - none, dot, dash, or solid
label1=                 label on axis 1
nTic2=                  number of tics per numbered tic on axis 2
grid2=                  grid lines on axis 2 - none, dot, dash, or solid
label2=                 label on axis 2
labelFont=              font name for axes labels
title=                  title of plot
titleFont=              font name for title
titleColor=             color for title
axesColor=              color for axes
gridColor=              color for grid lines
style=                  normal (axis 1 horizontal, axis 2 vertical) or
                       seismic (axis 1 vertical, axis 2 horizontal)



AUTHOR:  Dave Hale, Colorado School of Mines, 05/17/91
```

```
 GBBEAM - Gaussian beam synthetic seismograms for a sloth model

 gbbeam <rayends >syntraces xg= zg= [optional parameters]

 Required Parameters:
 xg=              x coordinates of receiver surface
 zg=              z coordinates of receiver surface

 Optional Parameters:
 ng=101           number of receivers (uniform distributed along surface)
 krecord=1        integer index of receiver surface (see notes below)
 ang=0.0          array of angles corresponding to amplitudes in amp
 amp=1.0          array of amplitudes corresponding to angles in ang
 bw=0             beamwidth at peak frequency
 nt=251           number of time samples
 dt=0.004         time sampling interval
 ft=0.0           first time sample
 reftrans=0       =1 complex refl/transm. coefficients considered
 prim             =1, only single-reflected rays are considered ",
                  =0, only direct hits are considered
 atten=0          =1 add noncausal attenuation
                  =2 add causal attenuation
 lscale=          if defined restricts range of extrapolation
 aperture=        maximum angle of receiver aperture
 fpeak=0.1/dt     peak frequency of ricker wavelet
 infofile         ASCII-file to store useful information
 NOTES:
 Only rays that terminate with index krecord will contribute to the
 synthetic seismograms at the receiver (xg,zg) locations.  The
 receiver locations are determined by cubic spline interpolation
 of the specified (xg,zg) coordinates.



 AUTHOR:  Dave Hale, Colorado School of Mines, 02/09/91
 MODIFIED:  Andreas Rueger, Colorado School of Mines, 08/18/93
Modifications include: 2.5-D amplitudes, computation of reflection/
transmission losses, attenuation,
timewindow, lscale, aperture, beam width, etc.
```

```
NORMRAY - dynamic ray tracing for normal incidence rays in a sloth model

   normray <modelfile >rayends [optional parameters]

Optional Parameters:
caustic  0: show all rays 1: show only caustic rays
nonsurface  0: show rays which reach surface 1: show all rays
surface  0: shot ray from subsurface 1: from surface
nrays   number of location to shoot rays
dangle   increment of ray angle for one location
nangle   number of rays shot from one location
ashift   shift first taking off angle
xs1           x of shooting location
zs1           z of shooting location
nangle=101     number of takeoff angles
fangle=-45     first takeoff angle (in degrees)
rayfile        file of ray x,z coordinates of ray-edge intersections
nxz=101        number of (x,z) in optional rayfile (see notes below)
wavefile       file of ray x,z coordinates uniformly sampled in time
nt=101         number of (x,z) in optional wavefile (see notes below)
infofile       ASCII-file to store useful information
fresnelfile    used if you want to plot the fresnel volumes.
               default is <fresnelfile.bin>
outparfile     contains parameters for the plotting software.
               default is <outpar>
krecord        if specified, only rays incident at interface with index
               krecord are displayed and stored
prim           =1, only single-reflected rays are plotted  ",
               =0, only direct hits are displayed
ffreq=-1       FresnelVolume frequency
refseq=1,0,0   index of reflector followed by sequence of reflection (1)
               transmission(0) or ray stops(-1).
               The default rayend is at the model boundary.
               NOTE:refseq must be defined for each reflector
NOTES:
The rayends file contains ray parameters for the locations at which
the rays terminate.

The rayfile is useful for making plots of ray paths.
nxz should be larger than twice the number of triangles intersected
by the rays.

The wavefile is useful for making plots of wavefronts.
```

```
 The time sampling interval in the wavefile is tmax/(nt-1),
 where tmax is the maximum time for all rays.

 The infofile is useful for collecting information along the
 individual rays. The fresnelfile contains data used to plot
 the Fresnel volumes. The outparfile stores information used
 for the plotting software.



 AUTHOR:  Dave Hale, Colorado School of Mines, 02/16/91
 MODIFIED:  Andreas Rueger, Colorado School of Mines, 08/12/93
Modifications include: functions writeFresnel, checkIfSourceIsOnEdge;
options refseq=, krecord=, prim=, infofile=;
computation of reflection/transmission losses, attenuation.
 MODIFIED: Boyi Ou, Colorado School of Mines, 4/14/95

 Notes:
 This code can shoot rays from specified interface by users, normally you
need to use gbmodel2 to generate interface parameters for this code, both
code have a parameter named nrays, it should be same. If you just want to
shoot rays from one specified location, you need to specify xs1,zs1,
otherwise, leave them alone. If you want to shoot rays from surface, you need
to define surface equal to 1. The rays from one location will be
approximately symetric with direction Normal_direction - ashift.(if nangle is
odd, it is symetric, even, almost symetric. The formula for the first take
off angle is: angle=normal_direction-nangle/2*dangle-ashift. If you only want to
see caustics, you specify caustic=1, if you want to see rays which does not
reach surface, you specify nonsurface=1.
/
```

```
TRI2UNI - convert a TRIangulated model to UNIformly sampled model

tri2uni <triangfile >uniformfile n2= n1= [optional parameters]

Required Parameters:
n1=                     number of samples in the first (fast) dimension
n2=                     number of samples in the second dimension

Optional Parameters:
d1=1.0                  sampling interval in first (fast) dimension
d2=1.0                  sampling interval in second dimension
f1=0.0                  first value in dimension 1 sampled
f2=0.0                  first value in dimension 2 sampled

Note:
The triangulated/uniformly-sampled quantity is assumed to be sloth=1/v^2



AUTHOR:  Dave Hale, Colorado School of Mines, 04/23/91
```

```
TRIMODEL - make a triangulated sloth (1/velocity^2) model

trimodel >modelfile [optional parameters]

Optional Parameters:
xmin=0.0                minimum horizontal coordinate (x)
xmax=1.0                maximum horizontal coordinate (x)
zmin=0.0                minimum vertical coordinate (z)
zmax=1.0                maximum vertical coordinate (z)
xedge=                  x coordinates of an edge
zedge=                  z coordinates of an edge
sedge=                  sloth along an edge
kedge=                  array of indices used to identify edges
normray                0:do not generate parameters 1: does it
normface               specify which interface to shoot rays
nrays                  number of locations to shoot rays
sfill=                  x, z, x0, z0, s00, dsdx, dsdz to fill a region
densfill=               x, z, dens to fill a region
qfill=                  x, z, Q-factor to fill a region
maxangle=5.0            maximum angle (deg) between adjacent edge segments

Notes:
More than set of xedge, zedge, and sedge parameters may be
specified, but the numbers of these parameters must be equal.

Within each set, vertices will be connected by fixed edges.

Edge indices in the k array are used to identify edges
specified by the x and z parameters.  The first k index
corresponds to the first set of x and z parameters, the
second k index corresponds to the second set, and so on.

After all vertices and their corresponding sloth values have
been inserted into the model, the optional sfill parameters
are used to fill closed regions bounded by fixed edges.
Let (x,z) denote any point inside a closed region.  Sloth inside
this region is determined by s(x,z) = s00+(x-x0)*dsdx+(z-z0)*dsdz.
The (x,z) component of the sfill parameter is used to identify a
closed region.
```

AUTHOR: Dave Hale, Colorado School of Mines, 02/12/91
MODIFIED: Andreas Rueger, Colorado School of Mines, 01/18/93
   Fill regions with attenuation Q-factors and density values.
MODIFIED: Craig Artley, Colorado School of Mines, 03/27/94
   Corrected bug in computing s00 in makeSlothForTri() function.
MODIFIED: Boyi Ou, Colorado School of Mines, 4/14/95
    Make code to generate interface parameters for shooting rays
    from specified interface

NOTE:
When you use normface to specify interface, the number of interface might not be the number of interface in the picture, for example, you build a one interface model, this interface is very long, so in the shell, you use three part of xedge,zedge,sedge to make this interface, so when you use normface to specify interface, this interface is just part of whole interface. If you want see the normal rays from entire interface, you need to maek normal ray picture few times, and merge them together.

```
 TRIRAY - dynamic RAY tracing for a TRIangulated sloth model

  triray <modelfile >rayends [optional parameters]

 Optional Parameters:
 xs=(max-min)/2 x coordinate of source (default is halfway across model)
 zs=min  z coordinate of source (default is at top of model)
 nangle=101 number of takeoff angles
 fangle=-45 first takeoff angle (in degrees)
 langle=45 last takeoff angle (in degrees)
 rayfile= file of ray x,z coordinates of ray-edge intersections
 nxz=101 number of (x,z) in optional rayfile (see notes below)
 wavefile= file of ray x,z coordinates uniformly sampled in time
 nt=101 number of (x,z) in optional wavefile (see notes below)
 infofile= ASCII-file to store useful information
 fresnelfile=  used if you want to plot the fresnel volumes.
default is <fresnelfile.bin>
 outparfile= contains parameters for the plotting software.
default is <outpar>
 krecord= if specified, only rays incident at interface with index
krecord are displayed and stored
 prim=    =1, only single-reflected rays are plotted
=0, only direct hits are displayed
 ffreq=-1 FresnelVolume frequency
 refseq=1,0,0  index of reflector followed by sequence of reflection (1)
transmission(0) or ray stops(-1).
The default rayend is at the model boundary.
NOTE:refseq must be defined for each reflector
 NOTES:
 The rayends file contains ray parameters for the locations at which
 the rays terminate.

 The rayfile is useful for making plots of ray paths.
 nxz should be larger than twice the number of triangles intersected
 by the rays.

 The wavefile is useful for making plots of wavefronts.
 The time sampling interval in the wavefile is tmax/(nt-1),
 where tmax is the maximum time for all rays.

 The infofile is useful for collecting information along the
 individual rays. The fresnelfile contains data used to plot
 the Fresnel volumes. The outparfile stores information used
```

```
 for the plotting software.


 AUTHOR:  Dave Hale, Colorado School of Mines, 02/16/91
 MODIFIED:  Andreas Rueger, Colorado School of Mines, 08/12/93
Modifications include: functions writeFresnel, checkIfSourceIsOnEdge;
options refseq=, krecord=, prim=, infofile=;
computation of reflection/transmission losses, attenuation.
```

```
TRISEIS - Gaussian beam synthetic seismograms for a sloth model

 triseis <modelfile >seisfile xs= zs= xg= zg= [optional parameters]

Required Parameters:
xs=             x coordinates of source surface
zs=             z coordinates of source surface
xg=             x coordinates of receiver surface
zg=             z coordinates of receiver surface

Optional Parameters:
ns=1            number of sources uniformly distributed along s surface
ds=             increment between source locations (see notes below)
fs=0.0          first source location (relative to start of s surface)
ng=101          number of receivers uniformly distributed along g surface
dg=             increment between receiver locations (see notes below)
fg=0.0          first receiver location (relative to start of g surface)
dgds=0.0        change in receiver location with respect to source location
krecord=1       integer index of receiver surface (see notes below)
kreflect=-1     integer index of reflecting surface (see notes below)
prim            =1, only single-reflected rays are considered  ",
                =0, only direct hits are considered
bw=0            beamwidth at peak frequency
nt=251          number of time samples
dt=0.004        time sampling interval
ft=0.0          first time sample
nangle=101      number of ray takeoff angles
fangle=-45      first ray takeoff angle (in degrees)
langle=45       last ray takeoff angle (in degrees)
reftrans=0      =1 complex refl/transm. coefficients considered
atten=0         =1 add noncausal attenuation
                =2 add causal attenuation
lscale=         if defined restricts range of extrapolation
fpeak=0.1/dt    peak frequency of ricker wavelet
aperture=       maximum angle of receiver aperture

NOTES:
Only rays that terminate with index krecord will contribute to the
synthetic seismograms at the receiver (xg,zg) locations.  The
source and receiver locations are determined by cubic spline
interpolation of the specified (xs,zs) and (xg,zg) coordinates.
The default source location increment (ds) is determined to span
the source surface defined by (xs,zs).  Likewise for dg.
```

```
 AUTHOR:  Dave Hale, Colorado School of Mines, 02/09/91
 MODIFIED:  Andreas Rueger, Colorado School of Mines, 08/18/93
Modifications include: 2.5-D amplitudes, correction for ref/transm,
timewindow, lscale, aperture, beam width, etc.
```

```
UNI2TRI - convert UNIformly sampled model to a TRIangulated model

uni2tri <slothfile >modelfile n2= n1= [optional parameters]

Required Parameters:
n1=                     number of samples in first (fast) dimension
n2=                     number of samples in second dimension

Optional Parameters:
d1=1.0                  sampling interval in dimension 1
d2=1.0                  sampling interval in dimension 2
f1=0.0                  first value in dimension 1
f2=0.0                  first value in dimesion 2
ifile=                  triangulated model file used as initial model
errmax=                 maximum sloth error (see notes below)
verbose=1               =0 for silence
                        =1 to report maximum error at each stage to stderr
                        =2 to also write the normalized error to efile
efile=emax.dat          filename for error file (for verbose=2)
mm=0 output every mm-th intermediate model (0 for none)
mfile=intmodel         intermediate models written to intmodel%d
method=3               =1 add 1 vertex at maximum error
                       =2 add vertex to every triangle that exceeds errmax
                       =3 method 2, but avoid closely spaced vertices
tol=10                 closeness criterion for (in samples)
sfill=                  x, z, x0, z0, s00, dsdx, dsdz to fill a region

Notes:
Triangles are constructed until the maximum error is
not greater than the user-specified errmax.  The default errmax
is 1% of the maximum value in the sampled input file.

After the uniform values have been triangulated, the optional sfill
parameters are used to fill closed regions bounded by fixed edges.
Let (x,z) denote any point inside a closed region.  Values inside
this region is determined by s(x,z) = s00+(x-x0)*dsdx+(z-z0)*dsdz.
The (x,z) component of the sfill parameter is used to identify a
closed region.

The uniformly sampled quantity is assumed to be sloth=1/v^2.
```

```
AUTHOR:  Craig Artley, Colorado School of Mines, 03/31/94
NOTE:  After a program outlined by Dave Hale, 12/27/90.
```

```
SPSPLOT - plot a triangulated sloth function s(x,z) via PostScript

spsplot <modelfile >postscriptfile [optional parameters]

Optional Parameters:
gedge=0.0             gray to draw fixed edges (in interval [0.0,1.0])
gtri=1.0              gray to draw non-fixed edges of triangles
gmin=0.0              min gray to shade triangles (in interval [0.0,1.0])
gmax=1.0              max gray to shade triangles (in interval [0.0,1.0])
sgmin=minimum s(x,z)  s(x,y) corresponding to gmin
sgmax=maximum s(x,z)  s(x,y) corresponding to gmax
xbox=1.5              offset in inches of left side of axes box
ybox=1.5              offset in inches of bottom side of axes box
wbox=6.0              width in inches of axes box
hbox=8.0              height in inches of axes box
xbeg=xmin             value at which x axis begins
xend=xmax             value at which x axis ends
dxnum=0.0             numbered tic interval on x axis (0.0 for automatic)
fxnum=xmin            first numbered tic on x axis (used if dxnum not 0.0)
nxtic=1               number of tics per numbered tic on x axis
gridx=none            grid lines on x axis - none, dot, dash, or solid
labelx=               label on x axis
zbeg=zmin             value at which z axis begins
zend=zmax             value at which z axis ends
dznum=0.0             numbered tic interval on z axis (0.0 for automatic)
fznum=zmin            first numbered tic on z axis (used if dynum not 0.0)
nztic=1               number of tics per numbered tic on z axis
gridz=none            grid lines on z axis - none, dot, dash, or solid
labelz=               label on z axis
labelfont=Helvetica   font name for axes labels
labelsize=12          font size for axes labels
title=                title of plot
titlefont=Helvetica-Bold  font name for title
titlesize=24          font size for title
titlecolor=black      color of title
axescolor=black       color of axes
gridcolor=black       color of grid
style=seismic         normal (z axis horizontal, x axis vertical) or
                      seismic (z axis vertical, x axis horizontal)

Note:  A value of gedge or gtri outside the interval [0.0,1.0]
results in that class of edge not being drawn.
```

```
AUTHOR:  Dave Hale, Colorado School of Mines, 10/18/90
MODIFIED: Craig Artley, Colorado School of Mines, 03/27/94
   Tweaks to improve PostScript header, add basic color support.

NOTE:  Have observed errors in output when compiled with optimization
   under NEXTSTEP 3.1.  Caveat Emptor.

Modified: Morten Wendell Pedersen, Aarhus University, 23/3-97
          Added ticwidth,axeswidth, gridwidth parameters
```

Shells:

```
------
 ARGV - give examples of dereferencing char **argv

 Usage: argv


------
 COPYRIGHT - insert CSM COPYRIGHT lines at top of files in working directory

 Usage: copyright file(s)


------
 CPALL , RCPALL - for local and remote directory tree/file transfer

 Usage: cpall sourcedir destinationdir
 Caveat: destinationdir must exist and be writeable by the user

 Usage: rcpall sourcedir remotemachine  destinationdir

 If user name is different on the remote machine, then second"
 entry is "remotemachine -l remoteusername"
 Caveats: rsh, copy, and write permissions required
         You must be on the source machine,
 destinationdir must exist and be writeable by the user.

 Notes: Both of these shell scripts use tar to do the transfer.


------
 CWPFIND - look for files with patterns in CWPROOT/src/cwp/lib

 Usage: cwpfind  pattern_fragment
        cwpfind -e exact_pattern


------
 DIRTREE - show DIRectory TREE

 Usage: dirtree
```

```
------
 FILETYPE - list all files of given type

 Usage: filetype string_from_file_output

 Examples:
filetype text      - list printable files
       filetype stripped  - list unstripped files

------
 Grep  - recursively call egrep in pwd

 Usage: Grep [-egrep_options] pattern

 Caution:  Do NOT redirect into file in pwd, either use something
like  >../Grep.out or perhaps pipe output into mail to yourself.

 Author: Jack, 7/95


------
 NEWCASE - Changes the case of all the filenames in a directory, dir

 Usage: newcase -l dir  change all filenames to lower case
                -u dir  change all filenames to upper case

 Notes: Useful for files downloaded from VAX.


------
 OVERWRITE - copy stdin to stdout after EOF

 This shell is called from the shell script:    replace


------
 PRECEDENCE - give table of C precedences from Kernighan and Ritchie

 Usage: precedence


------
 REPLACE - REPLACE string1 with string2  in files

 Usage:  replace string1 string2 files
```

```
------
 THIS_YEAR - print the current year

 Usage: this_year

 NOTES - useful for building dated filenames, etc.


------
 TIME_NOW - prints time in ZULU format with no spaces

 Usage: time_now

 Note: Useful for building dated filenames


------
 TODAYS_DATE - prints today's date in ZULU format with no spaces

 Usage: todays_date

 Note: Useful for building dated filenames


------
 USERNAMES - get list of all login names

 Usage: usernames


------
 VARLIST - list variables used in a Fortran program

 Usage: varlist file.f ...

 Output is in the file: vars.file


------
 WEEKDAY - prints today's WEEKDAY designation

 Usage: weekday

 Note: Useful for building dated filenames
```

```
------
 ZAP - kill processes by name

 Typical usages:
zap ximage
zap 'xmovie|xgraph'

 Zap accepts full pattern matching for the process names

 Caveat: zap assumes that the FIRST field produced by the
Unix "ps" command is the pid (process identifier) number.
If not, change the number in the awk print statement to
the appropriate field.

 Author: Jack, 6/95 -- after Kernighan and Pike's zap
```

```
------
 GENDOCS - generate complete list of selfdocs in latex form

 Usage: gendocs -o  output filename is:  selfdocs.tex


------
 STRIPTOTXT -  put files from $CWPROOT/src/doc/Stripped into a new
               directory in the form $CWPROOT/src/TXT/NAME.txt

 Usage: striptotxt

 Author: John Stockwell,  Sept 2001

------
 UPDATEDOCALL - put self-docs in ../doc/Stripped

 Usage: updatedocall

 Note: this shell uses updatedoc to update the  database used by
       suname and gendocs


------
 UPDATEDOC - put self-docs in ../doc/Stripped and ../doc/Headers

 Usage: updatedoc  path

 Notes:
 Paths include: cwp/main cwp/lib cwp/shell par/main par/lib par/shell
     xplot/main xplot/lib psplot/main psplot/lib psplot/shell
     Xtcwp/main Xtcwp/lib Sfio/main
      su/main/amplitudes su/main/attributes_parameter_estimation
      su/main/convolution_correlation /su/main/data_compression
      su/main/data_conversion su/main/datuming su/main/decon_shaping
      su/main/dip_moveout su/main/filters su/main/headers su/main/interp_extrap
      su/main/migration_inversion su/main/multicomponent su/main/noise
      su/main/operations su/main/picking su/main/stacking su/main/statics
      su/main/stretching_moveout_resamp su/main/supromax
      su/main/synthetics_waveforms_testpatterns su/main/tapering
      su/main/transforms su/main/velocity_analysis su/main/well_logs
      su/main/windowing_sorting_muting
     su/lib su/shell su/graphics/psplot
     su/graphics/xplot tri/main tri/lib xtri tri/graphics/psplot
```

```
    tetra/lib tetra/main
    comp/dct/lib comp/dct/main comp/dct/libutil comp/dwpt/1d/lib
    comp/dwpt/1d/main comp/dwpt/2d/lib comp/dwpt/2d/main

 Use: updatedocall to update full directory, use updatehead to
      to update the master header file.

 This shell builds the database used by  suname and gendocs

------
 UPDATEHEAD - update ../doc/Headers/Headers.all

 Usage: updatehead

 Notes:

 This file builds the database used by  suname
```

```
------
 LOOKPAR - show getpar lines in SU code with defines evaluated

 Usage: lookpar filename ...


------
 MAXDIFF - find absolute maximum difference in two segy data sets

 Usage: maxdiff file1 file2


------
 RECIP - sum opposing (reciprocal) offsets in cdp sorted data

 Usage: recip <stdin >stdout


------
 RMAXDIFF - find percentage maximum difference in two segy data sets

 Usage: rmaxdiff file1 file2


------
 SUAGC - perform agc on SU data

 Note: this is an interface to sugain for backward compatibility
 See selfdoc of:   sugain   for more information

------
 SUBAND - Trapezoid-like Sin squared tapered Bandpass filter via  SUFILTER

 Usage:   suband < stdin > stdout

 Note: this shell mimmics the old program SUBAND, supersceded by SUFILTER
 See selfdoc of:   sufilter   for more information

------
 SUDIFF, SUSUM, SUPROD, SUQUO, SUPTDIFF, SUPTSUM,
 SUPTPROD, SUPTQUO - difference, sum, product, quotient of two SU data
                     sets via suop2

 Usage:
```

```
 sudiff file1 file2 > stdout
 susum file1 file2 > stdout
 ...etc

 Note: uses   suop2  to perform the computation

------
 SUDOC - get DOC listing for code

 Usage: sudoc name

 Note: Use this shell script to get selfdoc information for
 codes labeled with and asterisk (*) or pound sign (#) in suname list

------
 SUENV - Instantaneous amplitude, frequency, and phase via: SUATTRIBUTES

 Usage:   suenv < stdin > stdout

 Note: this shell mimmics the old program SUENV, supersceded by SUATTRIBUTES
 See selfdoc of:   suattributes   for more information

------
 SUFIND - get info from self-docs

 Usage: sufind [-v -n] string

 sufind string    gives a brief synopsis
 sufind -v string  is a verbose hunt for relevant items
 sufind -n name_fragment      searches for command name


------
 SUFIND - get info from self-docs

 Usage: sufind [-v -n] string

 sufind string    gives a brief synopsis
 sufind -v string  is a verbose hunt for relevant items
 sufind -n name_fragment      searches for command name

 Author: CWP: Jack K. Cohen,  1992
 Modified by: CWP: S. Narahara, 04/11/1998.
```

```
------
 SUGENDOCS - generate complete list of selfdocs in latex form

 Usage: sugendocs -o  output filename is: selfdocs.tex
 Note: this shell simply calls    gendocs


------
 SUHELP - list the CWP/SU programs and shells

 Usage:   suhelp


------
 SUKEYWORD -- guide to SU keywords in segy.h

 Usage: sukeyword -o            to begin at the top of segy.h
        sukeyword [string]      to find [string]

 Note:  keyword=  occurs in many SU programs.

------
 SUNAME - get name line from self-docs

 Usage: suname [name]

 Note: dummy selfdocs have been included in all cwp and shell programs
       that don't have automatic selfdocs.

------
 UNGLITCH - zonk outliers in data

 Usage: unglitch < stdin

 Note: this shell just invokes:  sugain < stdin qclip=.99 > stdout
 See selfdoc of:   sugain   for further information
```

```
------
 MERGE2 - put 2 standard size PostScript figures on one page

 Usage: merge2 fig1 fig2

 Notes: Translation values are hard-coded numbers that work well for
standard size (8.5 x 11) figures.
 See selfdoc of:   psmerge for details


------
 MERGE4 - put 4 standard size PostScript plots on one page

 Usage: merge4 ulfig urfig llfig lrfig

 Note: Translation values are hard-coded numbers that work well for
standard size (8.5 x 11) figures.
 See selfdoc of:   psmerge   for further information
```

Libs:

```
BASIC - Basic C function interface to PostScript

beginps write PostScript prolog (including %%Pages comment)
endps write PostScript trailer (including %%Pages comment)
begineps write encapsulated PostScript prolog (no %%Pages comment)
endeps write encapsulated PostScript trailer (no %%Pages comment)
boundingbox set BoundingBox to llx lly urx ury
newpage print "%%%%Page: label ordinal" to stdout
showpage print "showpage" to stdout
gsave print "GS" to stdout
grestore print "GR" to stdout
newpath print "NP" to stdout
closepath print "CP" to stdout
clip print "clip" to stdout
translate print "tx ty TR" to stdout, tx,ty = translation in x,y
scale print "sx sy SC" to stdout, sx,sy = scaling in x,y
rotate print "angle RO" to stdout, angle = rotation angle
concat print "m[0] m[1] m[2] m[3] m[4] m[5] CAT" to stdout
setgray print "gray setgray" to stdout, gray is 0-255 gray level
setrgbcolor print "red green blue setrgbcolor" to stdout
red,green,blue = 0-255 red,green,blue levels
setcolor set color by name based on definition in color structure
setlinewidth print "width SLW" to stdout, width = desired line width
setlinejoin print "code setlinejoin"
setdash print "[ dash ] offset setdash" to stdout
dash = array defining dash, offset = dash offset
moveto print "x y M" to stdout,   move to x,y
rmoveto print "x y RM" to stdout,  move to x,y
lineto print "x y L" to stdout, draw a line to x,y
rlineto print "x y RL" to stdout, draw a line to x,y
arc print "x y r ang1 ang2 arc" to stdout, draw an arc
x,y = vertex  r = radius from ang1 to ang2
stroke print "S" to stdout
fill print "F" to stdout
show  print "str SH" to stdout, show a string
justshow justify and show a string
image write a sampled gray-scale image
rgbimage write sampled color (rgb) image
setfont execute findfont, scalefont, and setfont for specified font
 and size
fontbbox determine font bounding box for specified font and size
fontheight return maximum height for specified font and size
```

```
fontwidth return maximum width for specified font and size
fontcapheight return maximum capheight for specified font and size
fontxheight return maximum xheight for specified font and size
fontdescender return maximum descender for specified font and size
polyline draw a segmented line
markto draw a mark at specified location
rectclip set a rectangular clipping path
rectfill draw a filled rectangle
strokerect stroke a rectangle

Function Prototypes:
void beginps (void);
void endps (void);
void begineps (void);
void endeps (void);
void newpage (const char *label, int ordinal);
void boundingbox (int llx, int lly, int urx, int ury);
void showpage (void);
void gsave (void);
void grestore (void);
void newpath (void);
void closepath (void);
void clip(void);
void translate (float tx, float ty);
void scale (float sx, float sy);
void rotate (float angle);
void concat (float m[]);
void setgray (float gray);
void setrgbcolor (float red, float green, float blue);
void setcolor (const char *name);
void setlinewidth (float width);
void setlinejoin (int code);
void setdash (float dash[], int ndash, float offset);
void moveto (float x, float y);
void rmoveto (float x, float y);
void lineto (float x, float y);
void rlineto (float x, float y);
void arc (float x, float y, float r, float ang1, float ang2);
void stroke (void);
void fill (void);
void show (const char *str);
void justshow (float just, const char *str);
void image (int w, int h, int bps, float m[], unsigned char *samples);
```

```
void rgbimage (int w, int h, int bpc, float m[], unsigned char *samples);
void cymkimage (int w, int h, int bpc, float m[], unsigned char *samples);
void setfont (const char *fontname, float fontsize);
void fontbbox (const char *fontname, float fontsize, float bbox[]);
float fontheight (const char *fontname, float fontsize);
float fontwidth (const char *fontname, float fontsize);
float fontcapheight (const char *fontname, float fontsize);
float fontxheight (const char *fontname, float fontsize);
float fontdescender (const char *fontname, float fontsize);
float fontascender (const char *fontname, float fontsize);
void polyline (const float *x, const float *y, int n);
void markto (float x, float y, int index, float size);
void rectclip (float x, float y, float width, float height);
void rectfill (float x, float y, float width, float height);
void rectstroke (float x, float y, float width, float height);

justshow:
Input:
just justification factor
str string

image:
Input:
w width of image (in samples)
h height of image (in samples)
bps number of bits per sample
m array[6] containing image matrix
samples array[w*h] of sample values

rgbimage:
Input:
w width of image (in samples)
h height of image (in samples)
bpc number of bits per component
m array[6] containing image matrix
samples array[3*w*h] of sample values

cymkimage:
Input:
w width of image (in samples)
h height of image (in samples)
bpc number of bits per component
m array[6] containing image matrix
```

```
samples array[4*w*h] of sample values


polyline:
Input:
x array[n] of x-coordinates
y array[n] of y-coordinates
n number of points

markto:
Input:
x x-coordinate of mark
y y-coordinate of mark
index type of mark to draw
size size of mark

rectclip:
Input:
x x-coordinate of clipping path origin
y y-coordinate of clipping path origin
width width of clipping path
height height of clipping path

rectfill:
Input:
x x-coordinate of rectangle origin
y y-coordinate of rectangle origin
width width of rectangle
height height of rectangle

strokerect:
Input:
x x-coordinate of rectangle origin
y y-coordinate of rectangle origin
width width of rectangle
height height of rectangle

Notes:
The majority of these routines are self explanatory. They are just
C wrappers that echo PostScript graphics commands.

justshow:
The justification factor positions the string relative to the current point.
```

```
just" may assume any value, but the common uses are:
-1.0 right-justify the string
-0.5 center the string on the current point
 0.0 left-justify the string (like using "show")

image:
Level 1 PostScript implementations support 1, 2, 4, and 8 bits per
sample.  Level 2 adds support for 12 bits per sample.
Samples are hex-encoded, and output lines are limited to 78 characters.

rgbimage:
In general, Level 1 PostScript implementations do not support rgbimage.
Level 2 supports 1, 2, 4, 8, and 12 bits per color component.  The
samples array should contain three color components (in R,G,B... order)
for each sample value.  Samples are hex-encoded, and output lines are
limited to 78 characters.

polyline:
The path is stroked every 200 points.
References:

Author:  Dave Hale, Colorado School of Mines, 1989
with modifications by Craig Artley, Colorado School of Mines, 1991, and
additions by Dave Hale, Advance Geophysical, 1992.
```

```
PSAXESBOX3 -  Functions draw an axes box via PostScript, estimate bounding box
               these are versions of psAxesBox and psAxesBox3 for psmovie.

psAxesBox3 draw an axes box via PostScript
psAxesBBox3 estimate bounding box for an axes box drawn via psAxesBox3

Function Prototypes:
void psAxesBox3(
float x, float y, float width, float height,
float x1Beg, float x1End, float p1Beg, float p1End,
float d1Num, float f1Num, int n1Tic, int grid1, char *label1,
float x2Beg, float x2End, float p2Beg, float p2End,
float d2Num, float f2Num, int n2Tic, int grid2, char *label2,
char *labelFont, float labelSize,
char *title, char *titleFont, float titleSize,
int style, char *title2);
void psAxesBBox3(
float x, float y, float width, float height,
char *labelFont, float labelSize,
char *titleFont, float titleSize,
int style, int bbox[]);

Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
x1Beg   axis value at beginning of axis 1
x1End axis value at end of axis 1
p1Beg   pad value at beginning of axis 1
p1End pad value at end of axis 1
d1Num numbered tic increment for axis 1 (0.0 for automatic)
f1Num first numbered tic for axis 1
n1Tic number of horizontal tics per numbered tic for axis 1
grid1 grid code for axis 1:  NONE, DOT, DASH, or SOLID
label1 label for axis 1
x2Beg   axis value at beginning of axis 2
x2End axis value at end of axis 2
p2Beg   pad value at beginning of axis 2
p2End pad value at end of axis 2
d2Num vertical numbered tic increment (0.0 for automatic)
f2Num first numbered vertical tic
n2Tic number of vertical tics per numbered tic
```

```
grid2 grid code for vertical axis:  NONE, DOT, DASH, or SOLID
label2 vertical axis label
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
title axes box title
titleFont name of font to use for title
titleSize size of font to use for title
style NORMAL (axis 1 on bottom, axis 2 on left)
  SEISMIC (axis 1 on left, axis 2 on top)
title2 second title

psAxesBBox3:
Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
titleFont name of font to use for title
titleSize size of font to use for title
style NORMAL (axis 1 on bottom, axis 2 on left)
  SEISMIC (axis 1 on left, axis 2 on top)
Output:
bbox bounding box (bbox[0:3] = llx, lly, ulx, uly)

Notes:
psAxesBox3:
psAxesBox will determine the numbered tic increment and first
numbered tic automatically, if the specified increment is zero.

Pad values must be specified in the same units as the corresponding
axes values.  These pads are useful when the contents of the axes box
requires more space than implied by the axes values.  For example,
the first and last seismic wiggle traces plotted inside an axes box
will typically extend beyond the axes values corresponding to the
first and last traces.  However, all tics will lie with the limits
specified in the axes values (x1Beg, x1End, x2Beg, x2End).

psAxesBBox3:
psAxesBBox uses font sizes to estimate the bounding box for
an axes box drawn with psAxesBox.  To be on the safe side,
psAxesBBox overestimates.
```

```
psAxesBBox assumes that the axes labels and titles do not extend
beyond the corresponding edges of the axes box.

References:
(see references in basic.c)
Author:  Dave Hale, Colorado School of Mines, 06/27/89
 modified by Zhiming Li, CSM, 7/1/90
```

```
PSAXESBOX - Functions to draw PostScript axes and estimate bounding box

psAxesBox Draw an axes box via PostScript
psAxesBBox estimate bounding box for an axes box drawn via psAxesBox

Function Prototypes:
void psAxesBox(
float x, float y, float width, float height,
float x1Beg, float x1End, float p1Beg, float p1End,
float d1Num, float f1Num, int n1Tic, int grid1, char *label1,
float x2Beg, float x2End, float p2Beg, float p2End,
float d2Num, float f2Num, int n2Tic, int grid2, char *label2,
char *labelFont, float labelSize,
char *title, char *titleFont, float titleSize,
char *titleColor, char *axesColor, char *gridColor,
float ticwidth, float axeswidth, float gridwidth,
int style);

void psAxesBBox(
float x, float y, float width, float height,
char *labelFont, float labelSize,
char *titleFont, float titleSize,
int style, int bbox[]);

psAxesBox:
Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
x1Beg   axis value at beginning of axis 1
x1End axis value at end of axis 1
p1Beg   pad value at beginning of axis 1
p1End pad value at end of axis 1
d1Num numbered tic increment for axis 1 (0.0 for automatic)
f1Num first numbered tic for axis 1
n1Tic number of horizontal tics per numbered tic for axis 1
grid1 grid code for axis 1:  NONE, DOT, DASH, or SOLID
label1 label for axis 1
x2Beg   axis value at beginning of axis 2
x2End axis value at end of axis 2
p2Beg   pad value at beginning of axis 2
p2End pad value at end of axis 2
```

```
d2Num vertical numbered tic increment (0.0 for automatic)
f2Num first numbered vertical tic
n2Tic number of vertical tics per numbered tic
grid2 grid code for vertical axis:  NONE, DOT, DASH, or SOLID
label2 vertical axis label
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
title axes box title
titleFont name of font to use for title
titleSize size of font to use for title
titleColor color to use for title
axesColor color to use for axes and axes labels
gridColor color to use for grid lines
axeswidth   width (in points) of axes
ticwidth    width (in points) of tic marks
gridwidth   width (in points) of grid lines
style NORMAL (axis 1 on bottom, axis 2 on left)
SEISMIC (axis 1 on left, axis 2 on top)

psAxesBBox:
Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
titleFont name of font to use for title
titleSize size of font to use for title
style NORMAL (axis 1 on bottom, axis 2 on left)
  SEISMIC (axis 1 on left, axis 2 on top)
Output:
bbox bounding box (bbox[0:3] = llx, lly, ulx, uly)

Notes:
psAxesBox:
psAxesBox will determine the numbered tic increment and first numbered
tic automatically, if the specified increment is zero.  Axis numbering
is in scientific notation, if necessary and is plotted to four
significant digits.

Pad values must be specified in the same units as the corresponding
axes values.  These pads are useful when the contents of the axes box
```

```
requires more space than implied by the axes values.  For example, the
first and last seismic wiggle traces plotted inside an axes box will
typically extend beyond the axes values corresponding to the first and
last traces.  However, all tics will lie with the limits specified in
the axes values (x1Beg, x1End, x2Beg, x2End).

psAxesBBox:
psAxesBBox uses font sizes to estimate the bounding box for
an axes box drawn with psAxesBox.  To be on the safe side,
psAxesBBox overestimates.

psAxesBBox assumes that the axes labels and titles do not extend
beyond the corresponding edges of the axes box.

References:
(see References for basic.c)
Author:   Dave Hale,  Colorado School of Mines, 06/27/89
Modified: Ken Larner, Colorado School of Mines, 08/30/90
Modified: Dave Hale, Advance Geophysical, 10/18/92
Added color parameters for title, axes, and grid.
Modified: Morten Wendell Pedersen, Aarhus University, 23/3-97
Added ticwidth,axeswidth, gridwidth parameters
```

```
PSCAXESBOX - Draw an axes box for cube via PostScript

psCubeAxesBox Draw an axes box for cube via PostScript

Function Prototype:
void psCubeAxesBox(
float x, float y, float size1, float size2, float size3, float angle,
float x1Beg, float x1End, float p1Beg, float p1End,
float d1Num, float f1Num, int n1Tic, int grid1, char *label1,
float x2Beg, float x2End, float p2Beg, float p2End,
float d2Num, float f2Num, int n2Tic, int grid2, char *label2,
float x3Beg, float x3End, float p3Beg, float p3End,
float d3Num, float f3Num, int n3Tic, int grid3, char *label3,
char *labelFont, float labelSize,
char *title, char *titleFont, float titleSize,
char *titleColor, char *axesColor, char *gridColor);

Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
size1 size of 1st dimension of the input cube
size2 size of 2nd dimension of the input cube
size3 size of 3rd dimension of the input cube
angle projection angle of the cube
x1Beg   axis value at beginning of axis 1
x1End axis value at end of axis 1
p1Beg   pad value at beginning of axis 1
p1End pad value at end of axis 1
d1Num numbered tic increment for axis 1 (0.0 for automatic)
f1Num first numbered tic for axis 1
n1Tic number of tics for axis 1
grid1 grid code for axis 1:  NONE, DOT, DASH, or SOLID
label1 label for axis 1
x2Beg   axis value at beginning of axis 2
x2End axis value at end of axis 2
p2Beg   pad value at beginning of axis 2
p2End pad value at end of axis 2
d2Num numbered tic increment for axis 2 (0.0 for automatic)
f2Num first numbered tic for axis 2
n2Tic number of tics for axis 2
grid2 grid code for axis 2:  NONE, DOT, DASH, or SOLID
label2 label for axis 2
x3Beg   axis value at beginning of axis 3
```

```
x3End axis value at end of axis 3
p3Beg   pad value at beginning of axis 3
p3End pad value at end of axis 3
d3Num numbered tic increment for axis 3 (0.0 for automatic)
f3Num first numbered tic for axis 3
n3Tic number of tics for axis 3
grid3 grid code for axis 3:  NONE, DOT, DASH, or SOLID
label3 label for axis 3
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
title axes box title
titleFont name of font to use for title
titleSize size of font to use for title
titleColor color to use for title
axesColor color to use for axes and axes labels
gridColor color to use for grid lines
Authors:   Zhiming Li & Dave Hale,  Colorado School of Mines, 6/90
Modified: Craig Artley, Colorado School of Mines, 3/12/93
Changed name to psCubeAxesBox (from psAxes3), fixed minor bugs.
Modified: Craig Artley, Colorado School of Mines, 12/16/93
Added color parameters for title, axes, and grid.
```

PSCONTOUR - draw contour of a two-dimensional array via PostScript

psContour draw contour of a two-dimensional array via PostScript

Function Prototype:
void psContour (float c, int nx, float x[], int ny, float y[], float z[],
float lcs, char *lcf, char *lcc, float *w, int nplaces);

Input:
c contour value
nx number of x-coordinates
x array of x-coordinates (see notes below)
ny number of y-coordinates
y array of y-coordinates (see notes below)
lcs font size of contour label
lcf font name of contour label
lcc color of contour label
LSB flag arrays (see Notes):
z array of nx*ny z(x,y) values (see notes below)
w array of nx*ny z(x,y) values (see notes below)

Notes:
The two-dimensional array z is actually passed as a one-dimensional
array containing nx*ny values, stored with nx fast and ny slow.

The x and y arrays define a grid that is not necessarily
uniformly-sampled. Linear interpolation of z values on the
grid defined by the x and y arrays is used to determine z values
between the gridpoints.

The two least significant bits of z are used to mark intersections
of the contour with the x,y grid; therefore, the z values will
almost always be altered (slightly) by contour.

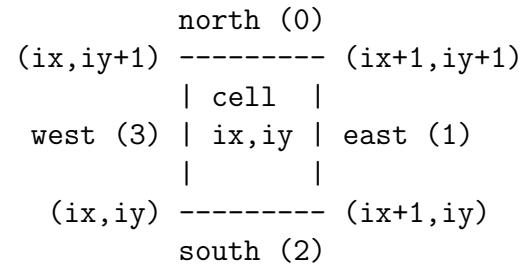
pscontour isolates the use of PostScript to four internal functions:

void coninit(void) - called before any contour drawing
void conmove(float x, float y) - moves current position to x,y
void condraw(float x, float y) - draws from current position to x,y
void condone(void) - called when contour drawing is done

These functions can usually be replaced with equivalent functions in
other graphics environments.

The w array is used to restrict the range of contour labeling that occurs only if w<1.

As suggested in the reference, the following scheme is used to refer to a cell of the two-dimensional array z:



Reference:
Cottafava, G. and Le Moli, G., 1969, Automatic contour map: Commuincations of the ACM, v. 12, n. 7, July, 1969.

Author: Dave Hale, Colorado School of Mines, 06/28/89
contour labeling added by: Zhenyue Liu, August 1993

```
PSDRAWCURVE - Functions to draw a curve from a set of points

psDrawCurve Draw a curve from a set of points via PostScript
Function Prototypes:
void psDrawCurve(
float x, float y, float width, float height,
float x1Beg, float x1End, float p1Beg, float p1End,
float x2Beg, float x2End, float p2Beg, float p2End,
float *x1curve, float *x2curve, int ncurve,
char *curveColor, float curvewidth, int curvedash, int style);
psDrawCurve:
Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
x1Beg   axis value at beginning of axis 1
x1End axis value at end of axis 1
p1Beg   pad value at beginning of axis 1
p1End pad value at end of axis 1
x2Beg   axis value at beginning of axis 2
x2End axis value at end of axis 2
p2Beg   pad value at beginning of axis 2
p2End pad value at end of axis 2
x1curve vector of x1 coordinates for points along curve
x2curve vector of x2 coordinates for points along curve
ncurve number of points along curve
curveColor color to use for curve
curvewidth width (in points) of curve
style NORMAL (axis 1 on bottom, axis 2 on left)
SEISMIC (axis 1 on left, axis 2 on top)

Author: Brian Macy, Phillips Petroleum Co., 11/20/98
(Adapted after Dave Hale and other's psAxesBox routine)
```

```
PSLEGENDBOX - Functions to draw PostScript axes and estimate bounding box

psLegendBox Draw an legend box via PostScript
psLegendBBox estimate bounding box for an legend box drawn via psLegendBox

Function Prototypes:
void psLegendBox(
float x, float y, float width, float height,
float x1Beg, float x1End, float p1Beg, float p1End,
float d1Num, float f1Num, int n1Tic, int grid1, char *label1,
char *labelFont, float labelSize,
char *axesColor, char *gridColor,
int style);

void psLegendBBox(
float x, float y, float width, float height,
char *labelFont, float labelSize,
int style, int bbox[]);


psLegendBox:
Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
x1Beg   axis value at beginning of axis 1
x1End axis value at end of axis 1
p1Beg   pad value at beginning of axis 1
p1End pad value at end of axis 1
d1Num numbered tic increment for axis 1 (0.0 for automatic)
f1Num first numbered tic for axis 1
n1Tic number of horizontal tics per numbered tic for axis 1
grid1 grid code for axis 1:  NONE, DOT, DASH, or SOLID
label1 label for axis 1
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
axesColor color to use for axes and axes labels
gridColor color to use for grid lines
style VERTLEFT (Vertical, axis label on left side)
VERTRIGHT (Vertical, axis label on right side)
HORIBOTTOM (Horizontal, axis label on bottom)
```

```
psLegendBBox:
Input:
x x coordinate of lower left corner of box
y y coordinate of lower left corner of box
width width of box
height height of box
labelFont name of font to use for axes labels
labelSize size of font to use for axes labels
style VERTLEFT (Vertical, axis label on left side)
VERTRIGHT (Vertical, axis label on right side)
HORIBOTTOM (Horizontal, axis label on bottom)
Output:
bbox bounding box (bbox[0:3] = llx, lly, ulx, uly)

Notes:
psLegendBox:
psLegendBox will determine the numbered tic increment and first numbered
tic automatically, if the specified increment is zero.  Axis numbering
is in scientific notation, if necessary and is plotted to four
significant digits.

Pad values must be specified in the same units as the corresponding
Legend values.  These pads are useful when the contents of the Legend box
requires more space than implied by the Legend values.  For example, the
first and last seismic wiggle traces plotted inside an Legend box will
typically extend beyond the Legend values corresponding to the first and
last traces.  However, all tics will lie with the limits specified in
the Legend values (x1Beg, x1End, x2Beg, x2End).

psLegendBBox:
psLegendBBox uses font sizes to estimate the bounding box for
an Legend box drawn with psLegendBox.  To be on the safe side,
psLegendBBox overestimates.

psLegendBBox assumes that the Legend labels and titles do not extend
beyond the corresponding edges of the Legend box.

References:
(see References for basic.c)
Author:   Dave Hale,  Colorado School of Mines, 06/27/89
Modified: Ken Larner, Colorado School of Mines, 08/30/90
Modified: Dave Hale, Advance Geophysical, 10/18/92
Added color parameters for title, axes, and grid.
```

```
Modified: Torsten Schoenfelder, Koeln, Germany, 07/06/97
        Display a legend for ps file, move axis from left to right
Modified: Torsten Schoenfelder, Koeln, Germany, 10/02/98
        Corrected width of bbox to include legend title
```

```
PSWIGGLE - draw wiggle-trace with (optional) area-fill via PostScript

psWiggle  draw wiggle-trace with (optional) area-fill via PostScript

Function Prototype:
void psWiggle (int n, float z[], float zmin, float zmax, float zbase,
float yzmin, float yzmax, float xfirst, float xlast, int fill);

Inputs:
n number of samples to draw
z array to draw
zmin z values below zmin will be clipped
zmax z values above zmax will be clipped
zbase z values between zbase and either zmin or zmax will be filled
yzmin y-coordinate corresponding to zmin
yzmax y-coordinate corresponding to zmax
xfirst x-coordinate corresponding to z[0]
xlast x-coordinate corresponding to z[n-1]
fill = 0 for no fill
  > 0 for fill between zbase and zmax
  < 0 for fill between zbase and zmin
+2 for fill solid between zbase and zmax grey between zbase and zmin
-2 for fill solid between zbase and zmin grey between zbase and zmax
SHADING: 2<= abs(fill) <=5   abs(fill)=2 light grey  abs(fill)=5 black

NOTES:
psWiggle reduces PostScript output by eliminating linetos when
z values are essentially constant.  The tolerance for detecting
constant" z values is ZEPS*(zmax-zmin), where ZEPS is a fraction
defined below.

A more complete optimization would eliminate all connected line
segments that are essentially colinear.

psWiggle breaks up the wiggle into segments that can be drawn
without exceeding the PostScript pathlimit.

Author:  Dave Hale, Colorado School of Mines, 07/03/89
Modified:  Craig Artley, Colorado School of Mines, 04/13/92
           Corrected dead trace bug.  Now the last point of each segment
           is guaranteed to be drawn.
MODIFIED: Paul Michaels, Boise State University, 29 December 2000
           added fill=+/-2 option of solid/grey color scheme
```

```
AXESBOX - Functions to draw axes in X-windows graphics

xDrawAxesBox draw a labeled axes box
xSizeAxesBox determine optimal origin and size for a labeled axes box

Function Prototypes:
void xDrawAxesBox (Display *dpy, Window win,
int x, int y, int width, int height,
float x1beg, float x1end, float p1beg, float p1end,
float d1num, float f1num, int n1tic, int grid1, char *label1,
float x2beg, float x2end, float p2beg, float p2end,
float d2num, float f2num, int n2tic, int grid2, char *label2,
char *labelfont, char *title, char *titlefont,
char *axescolor, char *titlecolor, char *gridcolor,
int style);
void xSizeAxesBox (Display *dpy, Window win,
char *labelfont, char *titlefont, int style,
int *x, int *y, int *width, int *height);

xDrawAxesBox:
Input:
dpy display pointer
win window
x x coordinate of upper left corner of box
y y coordinate of upper left corner of box
width width of box
height height of box
x1beg axis value at beginning of axis 1
x1end axis value at end of axis 1
p1beg pad value at beginning of axis 1
p1end pad value at end of axis 1
d1num numbered tic increment for axis 1 (0.0 for automatic)
f1num first numbered tic for axis 1
n1tic number of tics per numbered tic for axis 1
grid1 grid code for axis 1:  NONE, DOT, DASH, or SOLID
label1 label for axis 1
x2beg axis value at beginning of axis 2
x2end axis value at end of axis 2
p2beg pad value at beginning of axis 2
p2end pad value at end of axis 2
d2num numbered tic increment for axis 2 (0.0 for automatic)
f2num first numbered tic for axis 2
n2tic number of tics per numbered tic for axis 2
```

```
grid2 grid code for axis 2:  NONE, DOT, DASH, or SOLID
label2 label for axis 2
labelfont name of font to use for axes labels
title axes box title
titlefont name of font to use for title
axescolor name of color to use for axes
titlecolor name of color to use for title
gridcolor name of color to use for grid
int style NORMAL (axis 1 on bottom, axis 2 on left)
SEISMIC (axis 1 on left, axis 2 on top)

xSizeAxesBox:
Input:
dpy display pointer
win window
labelfont name of font to use for axes labels
titlefont name of font to use for title
int style NORMAL (axis 1 on bottom, axis 2 on left)
SEISMIC (axis 1 on left, axis 2 on top)

Output:
x x coordinate of upper left corner of box
y y coordinate of upper left corner of box
width width of box
height height of box
{
XFontStruct *fa,*ft;
Notes:
xDrawAxesBox:
will determine the numbered tic incremenet and first
numbered tic automatically, if the specified increment is zero.

Pad values must be specified in the same units as the corresponding
axes values.  These pads are useful when the contents of the axes box
requires more space than implied by the axes values.  For example,
the first and last seismic wiggle traces plotted inside an axes box
will typically extend beyond the axes values corresponding to the
first and last traces.  However, all tics will lie within the limits
specified in the axes values (x1beg, x1end, x2beg, x2end).

xSizeAxesBox:
is intended to be used prior to xDrawAxesBox.
```

```
An "optimal" axes box is one that more or less fills the window,
with little wasted space around the edges of the window.

Author: Dave Hale, Colorado School of Mines, 01/27/90
```

```
COLORMAP - Functions to manipulate X colormaps:

xCreateRGBDefaultMap create XA_RGB_DEFAULT_MAP property of root window if
it does not already exist
xGetFirstPixel return first pixel in range of contiguous pixels in
XA_RGB_DEFAULT_MAP
xGetLastPixel return last pixel in range of contiguous pixels in
XA_RGB_DEFAULT_MAP
xCreateHSVColormap create a 2 ramp colormap (HSV - Model)
xCreateRGBColormap create a 2 ramp colormap (RGB - Model)

Function Prototypes:
Status xCreateRGBDefaultMap (Display *dpy, XStandardColormap *scmap);
unsigned long xGetFirstPixel (Display *dpy);
unsigned long xGetLastPixel (Display *dpy);
Colormap xCreateRGBColormap (Display *dpy, Window win,
char *str_cmap, int verbose)
Colormap xCreateHSVColormap (Display *dpy, Window win,
char *str_cmap, int verbose)

xCreateRGBDefaultMap:
Input:
dpy display

Output:
scmap the standard colormap structure

xGetFirstPixel, xGetLastPixel:
Input:
dpy display

Notes:
PROBLEM
-------

Most mid-range display devices today support what X calls
the "PseudoColor visual".  Typically, only 256 colors (or gray
levels) may be displayed simultaneously.  Although these 256 colors
may be chosen from a much larger (4096 or more) set of available
colors, only 256 colors can appear on a display at one time.

These 256 colors are indexed by pixel values in a table called
the colormap.  Each window can have its own colormap, but only
```

one colormap can be installed in the display hardware at a time. (Again, only 256 colors may be displayed at one time.) The window manager is responsible for installing a window's colormap when that window becomes the key window.

Many of the applications we are likely to write require a large, contiguous range of pixels (entries in the colormap). In this range, we must be able to:
(1) given a color (or gray), determine the corresponding pixel.
(2) given a pixel, determine the corresponding color (or gray).
An example would be an imaging application that uses a gray scale to display images in shades of gray between black and white. Such applications are also likely to require a few additional colors for drawing axes, text, etc.

The problem is to coordinate the use of the limited number of 256 simultaneous colors so that windows for different applications appear reasonable, even when their particular colormaps are not installed in the display hardware. For example, we might expect an analog xclock's hands to be visible even when xclock's window is not the key window, when its colormap is not installed.

We should ensure that the range of contiguous pixels used by one application (perhaps for imaging) does not conflict with the pixels used by other applications to draw text, clock hands, etc.

SOLUTION

Applications that do not require special colormaps should simply use the default colormap inherited from the root window when new top-level windows are created.

Applications that do require a special colormap MUST create their own colormap. They must not assume that space will be available in the default colormap for a contiguous range of read/write pixels, because the server or window manager may have already allocated these pixels as read-only. Even if sufficient pixels are available in the default colormap, they should not be allocated by a single application. The default colormap should be used only for windows requiring a limited number of typical colors, such as red, yellow, etc.

Applications that require a contiguous range of read/write pixels should allocate these pixels in their window's private colormaps. They should determine which contiguous pixels to allocate from parameters in the standard colormap XA_RGB_DEFAULT_MAP. In particular, the first pixel in the range of contiguous pixels should be
base_pixel
and the last pixel in the range should be
base_pixel+red_max*red_mult+green_max*green_mult+blue_max*blue_mult,
where base_pixel, red_max, etc. are members in the XStandardColormap structure. On an 8-bit display, this range will typically provide 216 contiguous pixels, which may be set to a gray scale, color scale, or whatever. This leaves 40 colors for drawing text, axes, etc.

If the XA_RGB_DEFAULT_MAP does not exist, it should be created to consist of various colors composed of an equal number of reds, greens, and blues. For example, if 216 colors are to be allocated, then red_max=green_max=blue_max=5, red_mult=36, green_mult=6, and blue_mult=1. Because of the difficulty in forcing a particular pixel to correspond to a particular color in read-only color cells, these 216 colors will likely be read/write color cells unless created by the X server. In any case, these 216 colors should not be modified by any application. In creating custom colormaps, the only use of XA_RGB_DEFAULT_MAP should be in determining which 216 pixels to allocate for contiguous pixels.

In creating a custom colormap for a window, the application should initialize this colormap to the colors already contained in the window's colormap, which was inherited initially from its parent. This will ensure that typical colors already allocated by other applications will be consistent with pixels used by the application requiring the custom colormap. Ideally, windows might have different colormaps, but the only differences would be in the range of contiguous colors used for imaging, rendering, etc. Ideally, the pixels corresponding to colors used to draw text, axes, etc. would be consistent for all windows.

Unfortunately, it is impractical to maintain complete consistency among various private colormaps. For example, suppose a custom colormap is created for a window before other applications have had the opportunity to allocate their colors from the default colormap. Then, when the window with the custom colormap becomes the key window, the windows of the other applications may be displayed with false colors, since the colormap of the key window

```
may not contain the true colors.  The colors used by the other
applications did not exist when the custom colormap was created.
One solution to this problem might be to initially allocate a
set of "common" colors in the default colormap before launching
any applications.  This will increase the likelihood that typical
colors will be consistent among various colormaps.

Functions are provided below to
(1) create the standard colormap XA_RGB_DEFAULT_MAP, if it does not exist,
(2) determine the first and last pixels in the contiguous range of pixels,
(3) create some common private colormaps

xCreateRGBDefaultMap:
This function returns 0 if the XA_RGB_DEFAULT_MAP property does not exist
and cannot be created.  At least 8 contiguous color cells must be free
in the default colormap to create the XA_RGB_DEFAULT_MAP.  If created, the
red_max, green_max, and blue_max values returned in scmap will be equal.

xGetFirstPixel, xGetLastPixel:
If it does not already exist, XA_RGB_DEFAULT_MAP will be created.
If XA_RGB_DEFAULT_MAP does not exist and cannot be created, then
this function returns 0.

xCreateRGBColormap, xCreateHSVColormap:
The returned colormap is only created; the window's colormap attribute
is not changed, and the colormap is not installed by this function.
The returned colormap is a copy of the window's current colormap, but
with an RGB color scale allocated in the range of contiguous cells
determined by XA_RGB_DEFAULT_MAP.  If it does not already exist,
XA_RGB_DEFAULT_MAP will be created.

Author:  Dave Hale, Colorado School of Mines, 09/30/90
```

```
DRAWCURVE - Functions to draw a curve from a set of points

xDrawCurve draw a curve from a set of points
Function Prototypes:
void xDrawCurve(Display *dpy, Window win,
int x, int y, int width, int height,
float x1beg, float x1end, float p1beg, float p1end,
float x2beg, float x2end, float p2beg, float p2end,
float *x1curve, float *x2curve, int ncurve,
char *curvecolor, int style);
xDrawCurve:
Input:
dpy display pointer
win window
x x coordinate of upper left corner of box
y y coordinate of upper left corner of box
width width of box
height height of box
x1beg axis value at beginning of axis 1
x1end axis value at end of axis 1
p1beg pad value at beginning of axis 1
p1end pad value at end of axis 1
x2beg axis value at beginning of axis 2
x2end axis value at end of axis 2
p2beg pad value at beginning of axis 2
p2end pad value at end of axis 2
x1curve vector of x1 coordinates for points along curve
x2curve vector of x2 coordinates for points along curve
ncurve number of points along curve
curvecolor name of color to use for axes
int style NORMAL (axis 1 on bottom, axis 2 on left)
SEISMIC (axis 1 on left, axis 2 on top)

Author: Brian Macy, Phillips Petroleum Co., 11/14/98
(Adapted after Dave Hale's xDrawAxesBox routine)
```

```
IMAGE - Function for making the image in an X-windows image plot

xNewImage make a new image of pixels from bytes

Function Prototype:
XImage *xNewImage (Display *dpy, unsigned long pmin, unsigned long pmax,
int width, int height, float blank, unsigned char *bytes);

Input:
dpy display pointer
pmin minimum pixel value (corresponding to byte=0)
pmax maximum pixel value (corresponding to byte=255)
width number of bytes in x dimension
height number of bytes in y dimension
blank portion for blanking (0 to 1)
bytes unsigned bytes to be mapped to an image

Author:   Dave Hale, Colorado School of Mines, 06/08/90
Revision: Brian Zook, Southwest Research Institute, 6/27/96 added blank option
  This allows replacing the low end by the background.
```

LEGENDBOX - draw a labeled axes box for a legend (i.e. colorscale)

Function Prototype:
void xDrawLegendBox (Display *dpy, Window win,
int x, int y, int width, int height,
float bclip, float wclip, char *units, char *legendfont,
char *labelfont, char *title, char *titlefont,
char *axescolor, char *titlecolor, char *gridcolor,
int style);

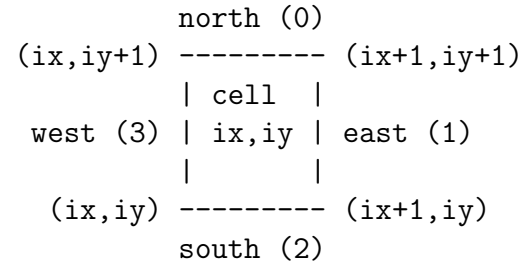
Input:
dpy display pointer
win window
x x coordinate of upper left corner of box
y y coordinate of upper left corner of box
width width of box
height height of box
units label for legend
legendfont name of font to use for legend labels
labelfont name of font to use for axes labels
title axes box title
titlefont name of font to use for title
axescolor name of color to use for axes
titlecolor name of color to use for title
gridcolor name of color to use for grid
int style NORMAL (axis 1 on bottom, axis 2 on left)
SEISMIC (axis 1 on left, axis 2 on top)
Notes:
xDrawLegendBox will determine the numbered tic incremenet and first
numbered tic automatically, if the specified increment is zero.

Pad values must be specified in the same units as the corresponding
axes values. These pads are useful when the contents of the axes box
requires more space than implied by the axes values. For example,
the first and last seismic wiggle traces plotted inside an axes box
will typically extend beyond the axes values corresponding to the
first and last traces. However, all tics will lie within the limits
specified in the axes values (x1beg, x1end, x2beg, x2end).
Author: Dave Hale, Colorado School of Mines, 01/27/90
Author: Berend Scheffers , TNO Delft, 06/11/92

```
RUBBERBOX -  Function to draw a rubberband box in X-windows plots

xRubberBox Track pointer with rubberband box

Function Prototype:
void xRubberBox (Display *dpy, Window win, XEvent event,
int *x, int *y, int *width, int *height);

Input:
dpy display pointer
win window ID
event event of type ButtonPress

Output:
x x of upper left hand corner of box in pixels
y y of upper left hand corner of box in pixels
width width of box in pixels
height height of box in pixels

Notes:
xRubberBox assumes that event is a ButtonPress event for the 1st button;
i.e., it tracks motion of the pointer while the 1st button is down, and
it sets x, y, w, and h and returns after a ButtonRelease event for the
1st button.

Before calling xRubberBox, both ButtonRelease and Button1Motion events
must be enabled.

This is the same rubberbox.c as in Xtcwp/lib, only difference is
that xRubberBox here is XtcwpRubberBox there, and a shift has been
added to make the rubberbox more visible.

Author: Dave Hale, Colorado School of Mines, 01/27/90
```

```
WINDOW - Function to create a window in X-windows graphics

xNewWindow Create a new window and return the window ID

Function Prototype:
Window xNewWindow (Display *dpy, int x, int y, int width, int height,
int border, int background, char *name);

Input:
dpy display pointer
x x in pixels of upper left corner
y y in pixels of upper left corner
width width in pixels
height height in pixels
border border pixel
background background pixel
name name of window (also used for icon)

Notes:
The parent window is the root window.
The border_width is 4 pixels.

Author: Dave Hale, Colorado School of Mines, 01/06/90
```

```
XCONTOUR - draw contour of a two-dimensional array via X vectorplot calls

xContour draw contour of a two-dimensional array via X vector plot calls

Function Prototype:
void xContour(Display *dpy, Window win,GC gcc, GC gcl,
       float *cp,int nx, float x[], int ny, float y[], float z[],
       char lcflag,char *lcf,char *lcc, float *w, int nplaces)
Input:
c contour value
nx number of x-coordinates
x array of x-coordinates (see notes below)
ny number of y-coordinates
y array of y-coordinates (see notes below)
lcf font name of contour label
lcc color of contour label
Least Significat Bits:
z array of nx*ny z(x,y) values (see notes below)
w array of nx*ny z(x,y) values (see notes below)
Notes:
The two-dimensional array z is actually passed as a one-dimensional
array containing nx*ny values, stored with nx fast and ny slow.

The x and y arrays define a grid that is not necessarily
uniformly-sampled.  Linear interpolation of z values on the
grid defined by the x and y arrays is used to determine z values
between the gridpoints.

The two least significant bits of z are used to mark intersections
of the contour with the x,y grid; therefore, the z values will
almost always be altered (slightly) by contour.

xContour is a modified version of psContour where the use of conmove
and condraw call have been changed to match X-Windows.

Since XDrawLine requires a start and end point, the use of a manually update
of the position variables x0 and y0 is used instead of conmove.

The w array is used to restrict the range of contour labeling that
occurs only if w<1.

As suggested in the reference, the following scheme is used to refer
to a cell of the two-dimensional array z:
```



Reference:
Cottafava, G. and Le Moli, G., 1969, Automatic contour map:
Commuincations of the ACM, v. 12, n. 7, July, 1969.

Author: Morten Wendell Pedersen Aarhus University 07/20/96
Heavily based on psContour by
Dave Hale, Colorado School of Mines, 06/28/89
and with contour labeling added by: Zhenyue Liu, June 1993
(actually most of the credit should go to these two guys)

AXES - the Axes Widget

XtcwpPointInAxesRectangle returns TRUE if point is inside axes
rectangle, otherwise FALSE
XtcwpSetAxesValues set axes values
XtcwpSetAxesPads set axes pads

Function Prototype:
Boolean XtcwpPointInAxesRectangle (Widget w, Position x, Position y);
void XtcwpSetAxesValues (Widget w, float x1beg, float x1end, float x2beg,
float x2end);

void XtcwpSetAxesPads (Widget w, float p1beg, float p1end, float p2beg,
float p2end);
XtcwpPointInAxesRectangle:
Input:
w axes widget
x x coordinate of point
y y coordinate of point

XtcwpSetAxesValues:
Input:
w axes widget
x1beg axis value at beginning of axis 1
x1end axis value at end of axis 1
x2beg axis value at beginning of axis 2
x2end axis value at end of axis 2

XtcwpSetAxesPads:
Input:
w axes widget
p1beg axis pad at beginning of axis 1
p1end axis pad at end of axis 1
p2beg axis pad at beginning of axis 2
p2end axis pad at end of axis 2

Notes:
XtcwpPointInAxesRectangle:
This function is useful for determining whether or not input events
occured with the pointer inside the axes rectangle. I.e., the input
callback function will typically call this function.

XtcwpSetAxesPads:

Pad values must be specified in the same units as the corresponding
axes values. These pads are useful when the contents of the axes box
require more space than implied by the axes values. For example, the
first and last seismic wiggle traces plotted inside an axes box
will typically extend beyond the axes values corresponding to the
first and last traces. However, all tics will lie within the limits
specified in the axes values (x1beg, x1end, x2beg, and x2end).

Author: Dave Hale, Colorado School of Mines, 08/28/90
Modified: Craig Artley, Colorado School of Mines, 06/03/93, Rotate label for
   vertical axis (Courtesy Dave Hale, Advance Geophysical).

COLORMAP - Functions to manipulate X colormaps:

XtcwpCreateRGBDefaultMap create XA_RGB_DEFAULT_MAP property of root
window if it does not already exist
XtcwpGetFirstPixel return first pixel in range of contiguous
pixels in XA_RGB_DEFAULT_MAP
XtcwpGetLastPixel return last pixel in range of contiguous
pixels in XA_RGB_DEFAULT_MAP
XtcwpCreateRGBColormap create a colormap with an RGB color scale in
contiguous cells
XtcwpCreateGrayColormap create a colormap with a gray scale in contiguous cells
XtcwpCreateHueColormap create a colormap with varying hues (blue to red)
in contiguous cells
XtcwpCreateSatColormap create a colormap with varying saturations in
contiguous cells

Function Prototypes:
Status XtcwpCreateRGBDefaultMap (Display *dpy, XStandardColormap *scmap);
unsigned long XtcwpGetFirstPixel (Display *dpy);
unsigned long XtcwpGetLastPixel (Display *dpy);
Colormap XtcwpCreateRGBColormap (Display *dpy, Window win);
Colormap XtcwpCreateGrayColormap (Display *dpy, Window win);
Colormap XtcwpCreateHueColormap (Display *dpy, Window win);
Colormap XtcwpCreateSatColormap (Display *dpy, Window win,
float fhue, float lhue, float wfrac, float bright)

XtcwpCreateRGBDefaultMap:
Input:
dpy display

Output:
scmap the standard colormap structure

XtcwpGetFirstPixel, XtcwpGetLastPixel:
Input:
dpy display

XtcwpCreateRGBColormap, XtcwpCreateGrayColormap, XtcwpCreateHueColormap:
Input:
dpy display
win window

XtcwpCreateSatColormap:

```
Input:
dpy display
win window
fhue first hue in colormap (saturation=1)
lhue last hue in colormap (saturation=1)
wfrac fractional position of white within the colormap (saturation=0)
bright brightness

Notes:
PROBLEM
-------

Most mid-range display devices today support what X calls
the "PseudoColor visual".  Typically, only 256 colors (or gray
levels) may be displayed simultaneously.  Although these 256 colors
may be chosen from a much larger (4096 or more) set of available
colors, only 256 colors can appear on a display at one time.

These 256 colors are indexed by pixel values in a table called
the colormap.  Each window can have its own colormap, but only
one colormap can be installed in the display hardware at a time.
(Again, only 256 colors may be displayed at one time.)  The window
manager is responsible for installing a window's colormap when that
window becomes the key window.

Many of the applications we are likely to write require a large,
contiguous range of pixels (entries in the colormap).  In this
range, we must be able to:
(1) given a color (or gray), determine the corresponding pixel.
(2) given a pixel, determine the corresponding color (or gray).
An example would be an imaging application that uses a gray scale
to display images in shades of gray between black and white.
Such applications are also likely to require a few additional colors
for drawing axes, text, etc.

The problem is to coordinate the use of the limited number of
256 simultaneous colors so that windows for different applications
appear reasonable, even when their particular colormaps are not
installed in the display hardware.  For example, we might expect
an analog xclock's hands to be visible even when xclock's window
is not the key window, when its colormap is not installed.

We should ensure that the range of contiguous pixels used by one
```

application (perhaps for imaging) does not conflict with the pixels used by other applications to draw text, clock hands, etc.

SOLUTION

Applications that do not require special colormaps should simply use the default colormap inherited from the root window when new top-level windows are created.

Applications that do require a special colormap MUST create their own colormap. They must not assume that space will be available in the default colormap for a contiguous range of read/write pixels, because the server or window manager may have already allocated these pixels as read-only. Even if sufficient pixels are available in the default colormap, they should not be allocated by a single application. The default colormap should be used only for windows requiring a limited number of typical colors, such as red, yellow, etc.

Applications that require a contiguous range of read/write pixels should allocate these pixels in their window's private colormaps. They should determine which contiguous pixels to allocate from parameters in the standard colormap XA_RGB_DEFAULT_MAP. In particular, the first pixel in the range of contiguous pixels should be
base_pixel
and the last pixel in the range should be
base_pixel+red_max*red_mult+green_max*green_mult+blue_max*blue_mult,
where base_pixel, red_max, etc. are members in the XStandardColormap structure. On an 8-bit display, this range will typically provide 216 contiguous pixels, which may be set to a gray scale, color scale, or whatever. This leaves 40 colors for drawing text, axes, etc.

If the XA_RGB_DEFAULT_MAP does not exist, it should be created to consist of various colors composed of an equal number of reds, greens, and blues. For example, if 216 colors are to be allocated, then red_max=green_max=blue_max=5, red_mult=36, green_mult=6, and blue_mult=1. Because of the difficulty in forcing a particular pixel to correspond to a particular color in read-only color cells, these 216 colors will likely be read/write color cells unless created by the X server. In any case, these 216 colors should not be modified by any application. In creating custom colormaps, the only use of XA_RGB_DEFAULT_MAP should be in determining which 216

pixels to allocate for contiguous pixels.

In creating a custom colormap for a window, the application should initialize this colormap to the colors already contained in the window's colormap, which was inherited initially from its parent. This will ensure that typical colors already allocated by other applications will be consistent with pixels used by the application requiring the custom colormap. Ideally, windows might have different colormaps, but the only differences would be in the range of contiguous colors used for imaging, rendering, etc. Ideally, the pixels corresponding to colors used to draw text, axes, etc. would be consistent for all windows.

Unfortunately, it is impractical to maintain complete consistency among various private colormaps. For example, suppose a custom colormap is created for a window before other applications have had the opportunity to allocate their colors from the default colormap. Then, when the window with the custom colormap becomes the key window, the windows of the other applications may be displayed with false colors, since the colormap of the key window may not contain the true colors. The colors used by the other applications did not exist when the custom colormap was created. One solution to this problem might be to initially allocate a set of "common" colors in the default colormap before launching any applications. This will increase the likelihood that typical colors will be consistent among various colormaps.

Functions are provided below to
(1) create the standard colormap XA_RGB_DEFAULT_MAP, if it does not exist,
(2) determine the first and last pixels in the contiguous range of pixels,
(3) create some common private colormaps - gray scale, hue scale, etc.

XtcwpCreateRGBDefaultMap:
This function returns 0 if the XA_RGB_DEFAULT_MAP property does not exist and cannot be created. At least 8 contiguous color cells must be free in the default colormap to create the XA_RGB_DEFAULT_MAP. If created, the red_max, green_max, and blue_max values returned in scmap will be equal.

XtcwpGetFirstPixel, XtcwpGetLastPixel:
If it does not already exist, XA_RGB_DEFAULT_MAP will be created. If XA_RGB_DEFAULT_MAP does not exist and cannot be created, then this function returns 0.

XtcwpCreateRGBColormap, XtcwpCreateGrayColormap, XtcwpCreateHueColormap,
XtcwpCreateSatColormap:
The returned colormap is only created; the window's colormap attribute
is not changed, and the colormap is not installed by this function.
The returned colormap is a copy of the window's current colormap, but
with an RGB color scale allocated in the range of contiguous cells
determined by XA_RGB_DEFAULT_MAP. If it does not already exist,
XA_RGB_DEFAULT_MAP will be created.

Author: Dave Hale, Colorado School of Mines, 09/30/90

```
FX - Functions to support floating point coordinates in X

FMapFX map float x to x
FMapFY map float y to y
FMapFWidth map float width to width
FMapFHeight map float height to height
FMapFAngle map float angle to angle
FMapFPoint map float x,y to x,y
FMapFPoints map float points to points
FMapX inverse map x to float x
FMapY inverse map y to float y
FMapWidth inverse map width to float width
FMapHeight inverse map height to float height
FMapAngle inverse map angle to float angle
FMapPoint map x,y to float x,y
FMapPoints map points to float points
FSetGC set graphics context
FSetMap set map (scales and shifts)
FSetClipRectangle set clip rectangle
FClipOn turn clip on
FClipOff turn clip off
FClipPoint clip point
FClipLine clip line
FClipRectangle clip rectangle
FXCreateFGC create float graphics context
FXFreeFGC free float graphic context
FXDrawPoint draw point at float x,y (with clipping)
FXDrawPoints draw float points (with clipping)
FXDrawLine draw line from float x1,y1 to float x2,y2
(with clipping)
FXDrawLines draw lines between float points (with clipping)
FXDrawRectangle draw rectangle with float x,y,width,height
(with clipping)
FXDrawArc draw arc with float x,y,width,height,angle1,angle2
FXDrawString draw string at float x,y
FXFillRectangle fill rectangle with float x,y,width,height (with
clipping)

Function Prototypes:
int FMapFX (FGC fgc, float fx);
int FMapFY (FGC fgc, float fy);
int FMapFWidth (FGC fgc, float fwidth);
int FMapFHeight (FGC fgc, float fheight);
```

```
int FMapFAngle (FGC fgc, float fangle);
void FMapFPoint (FGC fgc, float fx, float fy, int *x_return, int *y_return);
void FMapFPoints (FGC fgc, FXPoint fpoints[], int npoints,
XPoint points_return[]);
float FMapX (FGC fgc, int x);
float FMapY (FGC fgc, int y);
float FMapWidth (FGC fgc, int width);
float FMapHeight (FGC fgc, int height);
float FMapAngle (FGC fgc, int angle);
void FMapPoint (FGC fgc, int x, int y, float *fx_return, float *fy_return);
void FMapPoints (FGC fgc, XPoint points[], int npoints,
FXPoint fpoints_return[]);
void FSetGC (FGC fgc, GC gc);
void FSetMap (FGC fgc, int x, int y, int width, int height,
float fx, float fy, float fwidth, float fheight);
void FSetClipRectangle(FGC fgc, float fxa, float fya, float fxb, float fyb);
void FClipOn (FGC fgc);
void FClipOff (FGC fgc);
int FClipPoint (FGC fgc, float fx, float fy);
int FClipLine (FGC fgc, float fx1, float fy1, float fx2, float fy2,
float *fx1c, float *fy1c, float *fx2c, float *fy2c);
int FClipRectangle (FGC fgc, float fx, float fy, float fwidth, float fheight,
float *fxc, float *fyc, float *fwidthc, float *fheightc);
FGC FXCreateFGC (GC gc, int x, int y, int width, int height,
float fx, float fy, float fwidth, float fheight);
void FXFreeFGC (FGC fgc);
void FXDrawPoint (Display *display, Drawable d, FGC fgc, float fx, float fy);
void FXDrawPoints (Display *display, Drawable d, FGC fgc,
FXPoint fpoints[], int npoints, int mode);
void FXDrawLine (Display *display, Drawable d, FGC fgc,
float fx1, float fy1, float fx2, float fy2);
void FXDrawLines (Display *display, Drawable d, FGC fgc,
FXPoint fpoints[], int npoints, int mode);
void FXDrawRectangle (Display *display, Drawable d, FGC fgc,
float fx, float fy, float fwidth, float fheight);
void FXDrawArc (Display *display, Drawable d, FGC fgc,
float fx, float fy, float fwidth, float fheight,
float fangle1, float fangle2);
void FXDrawString (Display *display, Drawable d, FGC fgc,
float fx, float fy, char *string, int length);
void FXFillRectangle (Display *display, Drawable d, FGC fgc,
float fx, float fy, float fwidth, float fheight);
```

Notes:
The functions defined below are designed to resemble the equivalent
X functions. For example, FXDrawLine() is analogous to XDrawLine.
Each of the FXDraw<xxx>() functions requires an FGC instead of a GC
(graphics context). An FGC contains a GC, along with the information
required to transform floating point coordinates to integer (pixel)
coordinates.

Additional functions are provided to transform floating point coordinates
to integer coordinates and vice versa. Where feasible, macros are also
provided to perform these coordinate transformations.

Clipping of floating point coordinates is supported, because clipping
after mapping to integer coordinates is not valid when the mapped
integer coordinates overflow the range of short integers. By clipping
the floating point coordinates before mapping to integers, this overflow
can be avoided. By default, clipping is turned off until a clip rectangle
is specified or until clipping is explicitly turned on. Clipping is
not currently supported for all FXDraw functions.

Author: Dave Hale, Colorado School of Mines, 07/24/90
Modified: Dave Hale, Colorado School of Mines, 05/18/91
Added floating point clipping capability to some FXDraw functions.

MISC - Miscellaneous X-Toolkit functions

XtcwpDrawString90 Draw a string rotated 90 degrees counter-clockwise

Function Prototype:
void XtcwpDrawString90 (Display *dpy, Drawable d, GC gc,
int x, int y, char *string, int count);

Input:
dpy X display
d X drawable
gc X graphics context
x,y coordinates of baseline starting position of the string
string array[count] of characters to be drawn
count number of characters in string

Author: Dave Hale, Advance Geophysical, 06/03/93

```
RESCONV - general purpose resource type converters

XtcwpStringToFloat convert  string to float in resource

Function Prototype:
void XtcwpStringToFloat (XrmValue *args, int *nargs,
XrmValue *fromVal, XrmValue *toVal);

Author:  Dave Hale, Colorado School of Mines, 08/28/90
```

```
RUBBERBOX -  Function to draw a rubberband box in X-windows plots

XtcwpRubberbox Track pointer with rubberband box

Function Prototype:
void XtcwpRubberbox (Display *dpy, Window win, XEvent event,
int *x, int *y, int *width, int *height);

Input:
dpy display pointer
win window ID
event event of type ButtonPress

Output:
x x of upper left hand corner of box in pixels
y y of upper left hand corner of box in pixels
width width of box in pixels
height height of box in pixels

Notes:
XtcwpRubberbox assumes that event is a ButtonPress event for the 1st button;
i.e., it tracks motion of the pointer while the 1st button is down, and
it sets x, y, w, and h and returns after a ButtonRelease event for the
1st button.

Before calling XtcwpRubberbox, both ButtonRelease and Button1Motion events
must be enabled.

Author: Dave Hale, Colorado School of Mines, 01/27/90
```

```
RADIOBUTTONS -  convenience functions creating and using radio buttons

XtcwpCreateStringRadioButtons create an XmFrame containing radio
buttons labeled with strings

Function Prototypes:
Widget XtcwpCreateStringRadioButtons (Widget parent, char *label,
int nstrings, char **strings, int first,
void (*callback)(int selected, void *clientdata), void *clientdata);

Input:
parent parent widget
label label for this collection of radio bottons
nstrings number of strings
strings array[nstrings] of character strings, one per button
first index of button to be initially selected
callback function called when radio buttons change state
clientdata pointer to client data to be passed to callback

Notes:
This code depends on the Motif Developer's Package.

An integer index of the selected button (along with the clientdata pointer)
is passed to the callback function.

The returned XmFrame is not managed.

Author:  Dave Hale, Colorado School of Mines, 08/28/90
```

```
SAMPLES - Motif-based Graphics Functions

samplesCreate
samplesDraw
samplesSetN
samplesSetData
samplesSetPlotValue
samplesSetEditMode
samplesSetOrigin

Function Prototypes:
Samples *samplesCreate (Widget parent, char *title,
void (*editDone)(Samples *s));
void samplesDraw (Samples *s);
void samplesSetN (Samples *s, int n);
void samplesSetData (Samples *s, float *d);
void samplesSetPlotValue (Samples *s, float pv);
void samplesSetEditMode (Samples *s, EditMode m);
void samplesSetOrigin (Samples *s, int i);

Notes:
Watch this space.

Author: Dave Hale, Colorado School of Mines
```

```
FGETGTHR - get gathers from SU datafiles

fget_gather - get a gather from a file
get_gather - get a gather from stdin


Function Prototypes:
segy **fget_gather(FILE *fp, cwp_String *key,cwp_String *type,Value *n_val,
int *nt,int *ntr, float *dt,int *first);
segy **get_gather(cwp_String *key,cwp_String *type,Value *n_val,
int *nt,int *ntr, float *dt,int *first)

fget_gather - get a gather from a file
Input:
fp file pointer of input file
key header key of ensemble
type header value type
value value of ensemble key word
nt number of time samples
ntr number of traces in ensemble
dt time sampling interval
first flag is this the first ensemble being read?
Notes:
The input seismic dataset must be sorted into ensembles defined by key
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.

get_gather - get a gather from stdin
Input:
key header key of ensemble
type header value type
value value of ensemble key word
nt number of time samples
ntr number of traces in ensemble
dt time sampling interval
first flag is this the first ensemble being read?
Notes:
The input seismic dataset must be sorted into ensembles defined by key
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.

segy tr;
```

```
segy **fget_gather(FILE *fp, cwp_String *key,cwp_String *type,Value *n_val,
int *nt,int *ntr, float *dt,int *first)
fget_gather - get a gather from a file
Input:
fp file pointer of input file
key header key of ensemble
type header value type
value value of ensemble key word
nt number of time samples
ntr number of traces in ensemble
dt time sampling interval
first flag is this the first ensemble being read?
Notes:
The input seismic dataset must be sorted into ensembles defined by key
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.
{
int nsegy;
FILE *tracefp=NULL;
FILE *headerfp=NULL;
static FILE *ntracefp=NULL; /* first different trace pointer
static FILE *nheaderfp=NULL;
segy **rec=NULL;
int ntrr=0;
int indx=0;
static Value val;


*type = hdtype(*key);
indx = getindex(*key);
        *ntr = 0;


if(*first==0) {
        /* get info from first trace
nsegy = fgettr(fp,&tr);
if (nsegy==0)  err("can't get first trace");
*nt = tr.ns;
         *dt = (float) tr.dt/1000000.0;
         ++ntrr;
gethval(&tr, indx, n_val);
ntracefp = etmpfile();
nheaderfp = etmpfile();
```

```
*first=1;
} else {
/* This is the first trace of the nex gather
erewind(nheaderfp);
         erewind(ntracefp);
                fread (&tr,HDRBYTES, 1, nheaderfp);
                fread (tr.data,FSIZE, *nt, ntracefp);
gethval(&tr, indx, n_val);
}



        /* Store traces in tmpfile while getting a count
tracefp = etmpfile();
headerfp = etmpfile();
        do {
                efwrite(&tr, 1, HDRBYTES, headerfp);
                efwrite(tr.data, FSIZE, *nt, tracefp);

/* read the next trace
*ntr+=1;
val=*n_val;
nsegy = fgettr(fp,&tr);
if(nsegy) ntrr++;
gethval(&tr, indx, n_val);
} while (nsegy && !valcmp(*type,val,*n_val));


/* If there are no more traces then return
if(nsegy==0 && ntrr==0 ) {
 *ntr=0;
 efclose(nheaderfp);
 efclose(ntracefp);
 return(rec=NULL);
} else {
/* store the first trace of the next gather
erewind(nheaderfp);
         erewind(ntracefp);
                efwrite(&tr, 1, HDRBYTES, nheaderfp);
                efwrite(tr.data, FSIZE, *nt, ntracefp);
}

/* allocate memory for the record
```

```
{ register int i;
rec = ealloc1(*ntr,sizeof(segy *));
for(i=0;i<*ntr;i++)
rec[i] = (segy *)ealloc1((*nt*FSIZE+HDRBYTES),sizeof(char));
}


/* load traces into an array and close temp file
erewind(headerfp);
        erewind(tracefp);
{ register int ix;
         for (ix=0; ix<*ntr; ix++)
                 fread (rec[ix],HDRBYTES, 1, headerfp);
         efclose (headerfp);
for(ix=0; ix<*ntr; ix++)
                 fread ((*rec[ix]).data,FSIZE, *nt, tracefp);
         efclose (tracefp);
}

return(rec);
}

segy tr;

segy **get_gather(cwp_String *key,cwp_String *type,Value *n_val,
int *nt,int *ntr, float *dt,int *first)
get_gather - get a gather from stdin
Input:
key header key of ensemble
type header value type
value value of ensemble key word
nt number of time samples
ntr number of traces in ensemble
dt time sampling interval
first flag is this the first ensemble being read?
Notes:
The input seismic dataset must be sorted into ensembles defined by key
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.
{
int nsegy;
FILE *tracefp=NULL;
FILE *headerfp=NULL;
```

```
static FILE *ntracefp=NULL; /* first different trace pointer
static FILE *nheaderfp=NULL;
segy **rec=NULL;
int ntrr=0;
int indx=0;
static Value val;


*type = hdtype(*key);
indx = getindex(*key);
        *ntr = 0;


if(*first==0) {
        /* get info from first trace
nsegy = gettr(&tr);
if (nsegy==0)  err("can't get first trace");
*nt = tr.ns;
         *dt = (float) tr.dt/1000000.0;
         ++ntrr;
gethval(&tr, indx, n_val);
ntracefp = etmpfile();
nheaderfp = etmpfile();
*first=1;
} else {
/* This is the first trace of the nex gather
erewind(nheaderfp);
         erewind(ntracefp);
                fread (&tr,HDRBYTES, 1, nheaderfp);
                fread (tr.data,FSIZE, *nt, ntracefp);
gethval(&tr, indx, n_val);
}



        /* Store traces in tmpfile while getting a count
tracefp = etmpfile();
headerfp = etmpfile();
        do {
                efwrite(&tr, 1, HDRBYTES, headerfp);
                efwrite(tr.data, FSIZE, *nt, tracefp);

/* read the next trace
```

```
*ntr+=1;
val=*n_val;
nsegy = gettr(&tr);
if(nsegy) ntrr++;
gethval(&tr, indx, n_val);
} while (nsegy && !valcmp(*type,val,*n_val));


/* If there are no more traces then return
if(nsegy==0 && ntrr==0 ) {
 *ntr=0;
 efclose(nheaderfp);
 efclose(ntracefp);
 return(rec=NULL);
} else {
/* store the first trace of the next gather
erewind(nheaderfp);
         erewind(ntracefp);
                efwrite(&tr, 1, HDRBYTES, nheaderfp);
                efwrite(tr.data, FSIZE, *nt, ntracefp);
}

/* allocate memory for the record
{ register int i;
rec = ealloc1(*ntr,sizeof(segy *));
for(i=0;i<*ntr;i++)
rec[i] = (segy *)ealloc1((*nt*FSIZE+HDRBYTES),sizeof(char));
}


/* load traces into an array and close temp file
erewind(headerfp);
        erewind(tracefp);
{ register int ix;
         for (ix=0; ix<*ntr; ix++)
                 fread (rec[ix],HDRBYTES, 1, headerfp);
         efclose (headerfp);
for(ix=0; ix<*ntr; ix++)
                 fread ((*rec[ix]).data,FSIZE, *nt, tracefp);
         efclose (tracefp);
}

return(rec);
```

```
}
```

```
fgethdr - get segy tape identification headers from the file by file pointer

Input:
fp         file pointer

Output:
chdr       3200 bytes of segy character header
bhdr        400 bytes of segy binary header
Authors:  zhiming li  and j. dulac ,   unocal
 modified for CWP/SU: R. Beardsley
```

```
FGETTR - Routines to get an SU trace from a file

fgettr get a fixed-length segy trace from a file by file pointer
fvgettr get a variable-length segy trace from a file by file pointer
fgettra get a fixed-length trace from disk file by trace number
gettr macro using fgettr to get a trace from stdin
vgettr macro using vfgettr to get a trace from stdin
gettra macro using fgettra to get a trace from stdin by trace number

Function Prototype:
int fgettr(FILE *fp, segy *tp);
int fvgettr(FILE *fp, segy *tp);
int fgettra(FILE *fp, segy *tp, int itr);

Returns:
fgettr, fvgettr:
int: number of bytes read on current trace (0 after last trace)

fgettra:
int: number of traces in disk file

Macros defined in segy.h
define gettr(x) fgettr(stdin, (x))
define vgettr(x) fgettr(stdin, (x))

Usage example:
  segy tr;
  ...
  while (gettr(&tr)) {
  tr.offset = abs(tr.offset);
  puttr(&tr);
  }
  ...

Authors: SEP: Einar Kjartansson, Stew Levin CWP: Shuki Ronen, Jack Cohen

 Revised: 7/2/95 Stewart A. Levin   Mobil
     Major rewrite:  Use xdr library for portable su output file
     format.   Merge fgettr and fgettra into same source file.
     Make input from multiple streams work (at long last!).
 Revised: 11/22/95 Stewart A. Levin  Mobil
     Always set ntr for DISK input.  This fixes susort failure.
 Revised: 1/9/96  jkc CWP
```

```
    Set lastfp on nread <=0 return, too.
Revised: 28 Mar, 2006 Stewart A. Levin   Landmark Graphics
    Reworked XDR to support random seeks on > 2GB files
    and to read big endian SHORTPACK data on little endian machines.
```

```
FPUTGTHR - put gathers to a file

fput_gather - put a gather to a file
put_gather - put a gather to stdout

Function Prototypes:
segy **fput_gather(FILE *fp, segy **rec,int *nt, int *ntr);
segy **put_gather(segy **rec,int *nt, int *ntr)

fput_gather - put a gather to a file
Input:
fp pointer to output file
rec array of segy traces
nt number of time samples per trace
ntr number of traces in ensemble
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.

put_gather - put a gather to stdout
Input:
rec array of segy traces
nt number of time samples per trace
ntr number of traces in ensemble
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.

include "su.h"
include "segy.h"
include "header.h"


segy tr;

segy **fput_gather(FILE *fp, segy **rec,int *nt, int *ntr)
fput_gather - put a gather to a file
Input:
fp pointer to output file
rec array of segy traces
nt number of time samples per trace
ntr number of traces in ensemble
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.
{
```

```
segy tr;

{ register int i;
for(i=0;i<*ntr;i++) {
memcpy( (void *) &tr, (const void *) rec[i],
*nt*FSIZE+HDRBYTES);
fputtr(fp,&tr);
free1((void *)rec[i]);
}
}
return(rec=NULL);
}


segy **put_gather(segy **rec,int *nt, int *ntr)
put_gather - put a gather to stdout
Input:
rec array of segy traces
nt number of time samples per trace
ntr number of traces in ensemble
Author: Potash Corporation Sascatchewan, Balasz Nemeth
given to CWP in 2008.
{
segy tr;

{ register int i;
for(i=0;i<*ntr;i++) {
memcpy( (void *) &tr, (const void *) rec[i],
*nt*FSIZE+HDRBYTES);
puttr(&tr);
free1((void *)rec[i]);
}
}
return(rec=NULL);
}
```

```
FPUTTR - Routines to put an SU trace to a file

fputtr put a segy trace to a file by file pointer
fvputtr put a segy trace to a file by file pointer (variable ns)
puttr macro using fputtr to put a trace to stdin
vputtr macro using fputtr to put a trace to stdin (variable ns)

Function Prototype:
void fputtr(FILE *fp, segy *tp);
void fvputtr(FILE *fp, segy *tp);

Returns:

void

Notes:

The functions puttr(x) vputtr(x) are macros defined in segy.h
define puttr(x) fputtr(stdin, (x))
define vputtr(x) fvputtr(stdin, (x))

Usage example:
  segy tr;
  ...
  while (gettr(&tr)) {
  tr.offset = abs(tr.offset);
  puttr(&tr);
  }
  ...

Authors: SEP: Einar Kjartansson, Stew Levin CWP: Shuki Ronen, Jack Cohen
```

```
HDRPKGE - routines to access the SEGY header via the hdr structure.

gethval get a trace header word by index
puthval put a trace header word by index
getbhval get a binary header word by index
putbhval put a binary header word by index
gethdval get a trace header word by name
puthdval put a trace header word by name


hdtype get the data type of a trace header word by name
getkey get the name of a trace header word from its index

getindex get the index of a trace header word from the name

swaphval swap the trace header words by index
swapbhval swap the binary header words by index
gettapehval get a tape trace header word by index
puttapehval put a tape trace header word by index
gettapebhval get a tape binary header word by index
puttapebhval put a tape binary header word by index
printheader display non-null header field values

Function Prototypes:
void gethval(const segy *tr, int index, Value *valp);
void puthval(segy *tr, int index, Value *valp);
void putbhval(bhed *bh, int index, Value *valp);
void getbhval(const bhed *bh, int index, Value *valp);
void gethdval(const segy *tr, char *key, Value *valp);
void puthdval(segy *tr, char *key, Value *valp);
char *hdtype(const char *key);
char *getkey(const int index);
int getindex(const char *key);
void swaphval(segy *tr, int index);
void swapbhval(bhed *bh, int index);
void gettapehval(tapesegy *tapetr, int index, Value *valp);
void puttapehval(tapesegy *tapetr, int index, Value *valp);
void gettapebhval(tapebhed *tapetr, int index, Value *valp);
void puttapebhval(tapebhed *tapetr, int index, Value *valp);
void printheader(const segy *tp);

Notes:
This package includes only those routines that directly access
```

the "hdr" or "bhdr" structures. It does not include routines
such as printfval, printftype, printfhead that use the routines
in this package to indirectly access these structures.

Note that while gethdval and puthdval are more convenient to use
than gethval and puthval, they incur an inefficiency in the
common case of iterating code over a set of traces with a fixed
key or keys. In such cases, it is advisable to set the index
or indices outside the loop using getindex.

swaphval:
Byte-swapping is needed for converting SU data from big-endian to little-
endian formats, and vice versa. The swap_.... subroutines are based
on subroutines provided by Jens Hartmann of the Institut fur Geophysik
in Hamburg. These are found in .../cwp/lib/swapbyte.c.
Authors: SEP: Einar Kjartansson CWP: Jack Cohen, Shuki Ronen
swaphval: CWP: John Stockwell

TABPLOT - TABPLOT selected sample points on selected trace

tabplot tabplot selected sample points on selected trace

Function Prototype:
void tabplot(segy *tp, int itmin, int itmax);

Input:
tp pointer to a segy
itmin minimum time sample printed
itmax maximum time sample printed

Authors: CWP: Brian Sumner, Jack K. Cohen

```
VALPKGE - routines to handle variables of type Value

vtoi cast Value variable as an int
vtol cast Value variable as a long
vtof cast Value variable as a float
vtod cast Value variable as a double
atoval convert ascii to Value
valtoabs take absolute value of a Value variable
valcmp compare Value variables
printfval printf a Value variable
fprintfval fprintf a Value variable
scanfval scanf a Value variable
printftype printf for the type of a segy header word

Function Prototypes:
int vtoi(register cwp_String type, Value val);
long vtol(register cwp_String type, Value val);
float vtof(register cwp_String type, Value val);
double vtod(register cwp_String type, Value val);
void atoval(cwp_String type, cwp_String keyval, Value *valp);
Value valtoabs(cwp_String type, Value val);
int valcmp(register cwp_String type, Value val1, Value val2);
void printfval(register cwp_String type, Value val);
void fprintfval(FILE *stream, register cwp_String type, Value val);
void scanfval(register cwp_String type, Value *valp);

Notes:
A Value is defined by the following in .../su/include/su.h:

typedef union { * storage for arbitrary type *
char s[8];
short h;
unsigned short u;
long l;
unsigned long v;
int i;
unsigned int p;
float f;
double d;
unsigned int U:16;
unsigned int P:32;
} Value;
```

The use of the valpkge routines, as well as the hdrpkge routines,
permits the user to change the definition of the types of the
various fields of the segy data type, without breaking codes that
look at part or all of these fields.

Authors: CWP: Jack K. Cohen, Shuki Ronen

```
ABEL - Functions to compute the discrete ABEL transform:

abelalloc allocate and return a pointer to an Abel transformer
abelfree  free an Abel transformer
abel compute the Abel transform

Function prototypes:
void *abelalloc (int n);
void abelfree (void *at);
void abel (void *at, float f[], float g[]);

Input:
ns number of samples in the data to be transformed
f[] array of floats, the function being transformed

Output:
at pointer to Abel transformer returned by abelalloc(int n)
g[] array of floats, the transformed data returned by
abel(*at,f[],g[])

Notes:
The Abel transform is defined by:

         Infinity
g(y) = 2 Integral dx f(x)/sqrt(1-(y/x)^2)
   |y|

Linear interpolation is used to define the continuous function f(x)
corresponding to the samples in f[].  The first sample f[0] corresponds
to f(x=0) and the sampling interval is assumed to be 1.  Therefore, the
input samples correspond to 0 <= x <= n-1.  Samples of f(x) for x > n-1
are assumed to be zero.  These conventions imply that

g[0] = f[0] + 2*f[1] + 2*f[2] + ... + 2*f[n-1]

References:
Hansen, E. W., 1985, Fast Hankel transform algorithm:  IEEE Trans. on
Acoustics, Speech and Signal Processing, v. ASSP-33, n. 3, p. 666-671.
(Beware of several errors in the equations in this paper!)

Authors:  Dave Hale and Lydia Deng, Colorado School of Mines, 06/01/90
```

```
AIRY - Approximate the Airy functions  Ai(x), Bi(x) and their respective
       derivatives  Ai'(x), Bi'(x)

airya return approximation of Ai(x)
airypa return approximation of Ai'(x)
airyb return approximation of Bi(x)
airybp return approximation of Bi'(x)

Function Prototypes:
float airya (float x);
float airyap (float x);
float airyb (float x);
float airybp (float x);

Input:
x value at which to evaluate Ai(x)

Returned:
airya Ai(x)
airypa Ai'(x)
airyb Bi(x)
airybp Bi'(x)

Reference:
The approximation is derived from tables and formulas in Abramowitz
and Stegun, p. 475-477.

Author:  Dave Hale, Colorado School of Mines, 06/06/89
```

```
ALLOC - Allocate and free multi-dimensional arrays

alloc1 allocate a 1-d array
realloc1 re-allocate a 1-d array
free1 free a 1-d array
alloc2 allocate a 2-d array
free2 free a 2-d array
alloc3 allocate a 3-d array
free3 free a 3-d array
alloc4 allocate a 4-d array
free4 free a 4-d array
alloc5 allocate a 5-d array
free5 free a 5-d array
alloc6 allocate a 6-d array
free6 free a 6-d arrayalloc1int
allocate a 1-d array of ints
realloc1int re-allocate a 1-d array of ints
free1int free a 1-d array of ints
alloc2int allocate a 2-d array of ints
free2int free a 2-d array of ints
alloc3int allocate a 3-d array of ints
free3int free a 3-d array of ints
alloc1float allocate a 1-d array of floats
realloc1float re-allocate a 1-d array of floats
free1float free a 1-d array of floats
alloc2float allocate a 2-d array of floats
free2float free a 2-d array of floats
alloc3float allocate a 3-d array of floats
free3float free a 3-d array of floats
alloc4float allocate a 4-d array of floats
free4float      free a 4-d array of floats
alloc5float     allocate a 5-d array of floats
free5float      free a 5-d array of floats
alloc6float     allocate a 6-d array of floats
free6float      free a 6-d array of floats
alloc4int       allocate a 4-d array of ints
free4int        free a 4-d array of ints
alloc5int       allocate a 5-d array of ints
free5int        free a 5-d array of ints
alloc5uchar allocate a 5-d array of unsigned chars
free5uchar free a 5-d array of unsiged chars
alloc5ushort    allocate a 5-d array of unsigned shorts
free5ushort     free a 5-d array of unsiged shorts
```

```
alloc6ushort    allocate a 6-d array of unsigned shorts
free6ushort     free a 6-d array of unsiged shorts
alloc1double allocate a 1-d array of doubles
realloc1double re-allocate a 1-d array of doubles
free1double free a 1-d array of doubles
alloc2double allocate a 2-d array of doubles
free2double free a 2-d array of doubles
alloc3double allocate a 3-d array of doubles
free3double free a 3-d array of doubles
alloc1complex allocate a 1-d array of complexs
realloc1complex re-allocate a 1-d array of complexs
free1complex free a 1-d array of complexs
alloc2complex allocate a 2-d array of complexs
free2complex free a 2-d array of complexs
alloc3complex allocate a 3-d array of complexs
free3complex free a 3-d array of complexs

alloc1dcomplex   allocate a 1-d array of complexs
realloc1dcomplex re-allocate a 1-d array of complexs
free1dcomplex    free a 1-d array of complexs
alloc2dcomplex   allocate a 2-d array of complexs
free2dcomplex    free a 2-d array of complexs
alloc3dcomplex   allocate a 3-d array of complexs
free3dcomplex    free a 3-d array of complexs

Function Prototypes:
void *alloc1 (size_t n1, size_t size);
void *realloc1 (void *v, size_t n1, size_t size);
void free1 (void *p);
void **alloc2 (size_t n1, size_t n2, size_t size);
void free2 (void **p);
void ***alloc3 (size_t n1, size_t n2, size_t n3, size_t size);
void free3 (void ***p);
void ****alloc4 (size_t n1, size_t n2, size_t n3, size_t n4, size_t size);
void free4 (void ****p);
                  size_t size);
int *alloc1int (size_t n1);
int *realloc1int (int *v, size_t n1);
void free1int (int *p);
int **alloc2int (size_t n1, size_t n2);
void free2int (int **p);
int ***alloc3int (size_t n1, size_t n2, size_t n3);
void free3int (int ***p);
```

```
float *alloc1float (size_t n1);
float *realloc1float (float *v, size_t n1);
void free1float (float *p);
float **alloc2float (size_t n1, size_t n2);
void free2float (float **p);
float ***alloc3float (size_t n1, size_t n2, size_t n3);
void free3float (float ***p);
float ****alloc4float (size_t n1, size_t n2, size_t n3, size_t n4);
void free4float (float ****p);
                         size_t n6);
int ****alloc4int (size_t n1, size_t n2, size_t n3, size_t n4);
void free4int (int ****p);
size_t n5);
        size_t n5);
        size_t n5,size_t n6);
double *alloc1double (size_t n1);
double *realloc1double (double *v, size_t n1);
void free1double (double *p);
double **alloc2double (size_t n1, size_t n2);
void free2double (double **p);
double ***alloc3double (size_t n1, size_t n2, size_t n3);
void free3double (double ***p);
complex *alloc1complex (size_t n1);
complex *realloc1complex (complex *v, size_t n1);
void free1complex (complex *p);
complex **alloc2complex (size_t n1, size_t n2);
void free2complex (complex **p);
complex ***alloc3complex (size_t n1, size_t n2, size_t n3);
void free3complex (complex ***p);

complex *alloc1dcomplex (size_t n1);
complex *realloc1dcomplex (dcomplex *v, size_t n1);
void free1dcomplex (dcomplex *p);
complex **alloc2dcomplex (size_t n1, size_t n2);
void free2dcomplex (dcomplex **p);
complex ***alloc3dcomplex (size_t n1, size_t n2, size_t n3);
void free3dcomplex (dcomplex ***p);


Notes:
The functions defined below are intended to simplify manipulation
of multi-dimensional arrays in scientific programming in C.  These
functions are useful only because true multi-dimensional arrays
```

```
in C cannot have variable dimensions (as in FORTRAN).  For example,
the following function IS NOT valid in C:
void badFunc(a,n1,n2)
float a[n2][n1];
{
a[n2-1][n1-1] = 1.0;
}
However, the following function IS valid in C:
void goodFunc(a,n1,n2)
float **a;
{
a[n2-1][n1-1] = 1.0;
}
Therefore, the functions defined below do not allocate true
multi-dimensional arrays, as described in the C specification.
Instead, they allocate and initialize pointers (and pointers to
pointers) so that, for example, a[i2][i1] behaves like a 2-D array.

The array dimensions are numbered, which makes it easy to add
functions for arrays of higher dimensions.  In particular,
the 1st dimension of length n1 is always the fastest dimension,
the 2nd dimension of length n2 is the next fastest dimension,
and so on.  Note that the 1st (fastest) dimension n1 is the
first argument to the allocation functions defined below, but
that the 1st dimension is the last subscript in a[i2][i1].
(This is another important difference between C and Fortran.)

The allocation of pointers to pointers implies that more storage
is required than is necessary to hold a true multi-dimensional array.
The fraction of the total storage allocated that is used to hold
pointers is approximately 1/(n1+1).  This extra storage is unlikely
to represent a significant waste for large n1.

The functions defined below are significantly different from similar
functions described by Press et al, 1988, NR in C.
In particular, the functions defined below:
(1) Allocate arrays of arbitrary size elements.
(2) Allocate contiguous storage for arrays.
(3) Return NULL if allocation fails (just like malloc).
(4) Do not provide arbitrary lower and upper bounds for arrays.

Contiguous storage enables an allocated multi-dimensional array to
be passed to a C function that expects a one-dimensional array.
```

```
For example, to allocate and zero an n1 by n2 two-dimensional array
of floats, one could use
a = alloc2(n1,n2,sizeof(float));
zeroFloatArray(n1*n2,a[0]);
where zeroFloatArray is a function defined as
void zeroFloatArray(int n, float *a)
{
int i;
for (i=0; i<n; i++)
a[i] = 0.0;
}

Internal error handling and arbitrary array bounds, if desired,
should be implemented in functions that call the functions defined
below, with the understanding that these enhancements may limit
portability.

Author:     Dave Hale, Colorado School of Mines, 12/31/89
                Zhaobo Meng, added 4D, 5D and 6D functions, 1996
```

```
ANTIALIAS - Butterworth anti-aliasing filter

antialias use before increasing the sampling interval of data
 i.e. subsampling

Function Prototype:
void antialias (float frac, int phase, int n, float p[], float q[]);

Input:
frac current sampling interval / future interval (should be <= 1)
phase =0 for zero-phase filter; =1 for minimum-phase filter
n number of samples
p array[n] of input samples

Output:
q array[n] of output (anti-alias filtered) samples

Notes:
The anti-alias filter is a recursive (Butterworth) filter.  For zero-phase
anti-alias filtering, the recursive filter is applied forwards and backwards.

Author:  Dave Hale, Colorado School of Mines, 06/06/90
```

```
AXB - Functions to solve a linear system of equations Ax=b by LU
decomposition, invert a square matrix or directly multiply an
inverse matrix by another matrix (without explicitely computing
the inverse).

LU_decomposition Decompose a matrix (A) into a lower triangular (L)
and an upper triangular (U) such that A=LU

backward_substitution Apply backward substitution to an LU decomposed
matrix to solve the linear system of equations Ax=b

inverse_matrix compute the inverse of a square non-singular matrix

inverse_matrix_multiply computes the product A^(-1)*B without explicitely
computing the inverse matrix

Function prototypes:
void LU_decomposition (int nrows, float **matrix, int *idx, float *d);
void backward_substitution (int nrows, float **matrix, int *idx, float *b);
void inverse_matrix (int nrows, float **matrix);
void inverse_matrix_multiply (int nrows1, float **matrix1, int ncols2,
        int nrows2, float **matrix2, float **out_matrix);

LU_decomposition:
Input:
nrows number of rows of matrix to invert
matrix matrix of coefficients in linear system Ax=b

Output:
matrix matrix containing LU decomposition (original matrix destroyed)
idx vector recording the row permutations effected by partial
pivoting
d +/- 1 depending on whether the number of row interchanges
was even or odd
backward_substitution
Input:
nrows number of rows (and columns) of input matrix
matrix matrix of coefficients (after LU decomposition)
idx permutation vector obtained from routine LU_decomposition
b right hand side vector in equation Ax=b

Output:
b vector with the solution
```

```
inverse_matrix
Input:
nrows number of rows (and columns) of input matrix
matrix matrix to invert

Output:
matrix inverse of input matrix
inverse_matrix_multiply
nrows1          number of rows (and columns) of matrix to invert
matrix1         square matrix to invert
ncols2          number of coulmns of second matrix
nrows2 number of rows of second matrix
matrix          second matrix (multiplicator)

Output Parameters:
out_matrix      matrix containing the product of the inverse of the first
                matrix by the second one.
Note:
matrix1 and matrix2 are not destroyed, (not clobbered)
Notes:
To solve the set of linear equations Ax=b, first do the LU decomposition of
A (which will clobber A with its LU decomposition) and then do the backward
substitution with this new matrix and the right-hand side vector b. The vector
b will be clobbered with the solution. Both, the original matrix and vector B,
will have been destroyed.

The LU decomposition is carried out with the Crout's method with implicit
partial pivoting that guaratees that the maximum pivot is used in every
step of the algorithm.

The operation count to solve a linear system of equations via LU decomposition
is 1/3N^3 and is a factor of 3 better than the standard Gauss-Jordan algorithm
To invert a matrix the count is the same with both algorithms: N^3.

Once a linear system Ax=b has been solved, to solve another linear system
with the same matrix A but with different vetor b, ONLY the back substitution
has to be repeated with the new b (remember that the matrix in backsubstitution
is not the original matrix but its LU decomposition)

If you want to compute A^(-1)*B from matrices A and B, it is better to
use the subroutine inverse_matrix_multiply rather than explicitely computing
the inverse. This saves a whole martix multiplication and is also more accurate.
```

Refferences:
Press, Teukolsky, Vettering and Flannery, Numerical Recipes in C:
The art of scientific computing. Cambridge University Press.
second edition. (1992).
Golub and Van Loan, Matrix Computations. John Hopkins University Press.
Second Edition. (1989).
Horn and Johnson, Matrix Analysis. Cambridge University Press. (1985).
Credits:
Adapted from discussions in Numerical Recipes, by Gabriel Alvarez (1995)

```
BIGMATRIX - Functions to manipulate 2-dimensional matrices that are too big
    to fit in real memory, but that are small enough to fit in
virtual memory:

bmalloc allocate a big matrix
bmfree free a big matrix
bmread read a vector from a big matrix
bmwrite write a vector to a big matrix

Function Prototypes:
void *bmalloc (int nbpe, int n1, int n2);
void bmfree (void *bm);
void bmread (void *bm, int dir, int k1, int k2, int n, void *v);
void bmwrite (void *bm, int dir, int k1, int k2, int n, void *v);

bmalloc:
Input:
nbpe number of bytes per matrix element
n1 number of elements in 1st (fastest) dimension
n2 number of elements in 2nd (slowest) dimension

Returned:
bm pointer to big matrix

bmfree:
Input:
bm pointer to big matrix state (returned by bmalloc)

bmread:
Input:
bm     pointer to big matrix state (returned by bmalloc)
d   = 1 or 2:  direction in which to read matrix elements
k1 1st dimension index of first matrix element to read
k2 2nd dimension index of first matrix element to read
n number of elements to read

Output:
v array[n] to contain matrix elements read

bmwrite:
Input:
bm     pointer to big matrix state (returned by bmalloc)
d   = 1 or 2:  direction in which to write matrix elements
```

```
k1 1st dimension index of first matrix element to write
k2 2nd dimension index of first matrix element to write
n number of elements to write
v array[n] containing matrix elements to write

Notes:
The bm functions provide access to a big 2-dimensional matrix along
either the 1st or 2nd dimensions.  Although, the matrix must be small
enough to fit in virtual memory, it may be too large to fit in real memory.
These functions provide equally efficient (or equally inefficient) access
to vectors in a big matrix along either the 1st or 2nd dimensions.

For example, the following algorithm will efficiently transpose an
n1 by n2 array of (n1*n2) floats stored in a file:

void *bm;
float *v;
bm = bmalloc(sizeof(float),n1,n2);
for (i2=0; i2<n2; i2++) {
(read n1 floats from input file into array v);
bmwrite(bm,1,0,i2,n1,(char*)v);
}
for (i1=0; i1<n1; i1++) {
bmread(bm,2,i1,0,n2,(char*)v);
(write n2 floats in array v to output file);
}
bmfree(bm);

Author:  Dave Hale, Colorado School of Mines, 05/17/89
```

```
BUTTERWORTH - Functions to design and apply Butterworth filters:

bfdesign design a Butterworth filter
bfhighpass apply a high-pass Butterworth filter
bflowpass apply a low-pass Butterworth filter

Function Prototypes:
void bfhighpass (int npoles, float f3db, int n, float p[], float q[]);
void bflowpass (int npoles, float f3db, int n, float p[], float q[]);
void bfdesign (float fpass, float apass, float fstop, float astop,
int *npoles, float *f3db);

bfdesign:
Input:
fpass frequency in pass band at which amplitude is >= apass
apass amplitude in pass band corresponding to frequency fpass
fstop  frequency in stop band at which amplitude is <= astop
astop amplitude in stop band corresponding to frequency fstop

Output:
npoles number of poles
f3db frequency at which amplitude is sqrt(0.5) (-3 db)

bfhighpass and bflowpass:
Input:
npoles number of poles (and zeros); npoles>=0 is required
f3db 3 db frequency; nyquist = 0.5; 0.0<=f3db<=0.5 is required
n length of p and q
p array[n] to be filtered

Output:
q filtered array[n] (may be equivalent to p)

Notes:
(1) Nyquist frequency equals 0.5

(2) The following conditions must be true:
(0.0<fpass && fpass<0.5) &&
(0.0<fstop && fstop<0.5) &&
(fpass!=fstop) &&
(0.0<astop && astop<apass && apass<1.0)

(3) if (fpass<fstop)
```

```
bfdesign:
Butterworth filter:  compute number of poles and -3 db frequency
for a low-pass or high-pass filter, given a frequency response
constrained at two frequencies.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
COMPLEX - Functions to manipulate complex numbers

cadd add two complex numbers
csub subtract two complex numbers
cmul multiply two complex numbers
cdiv divide two complex numbers
cmplx make a complex number from two real numbers
conjg complex conjugate of a complex number
cneg negate a complex number
cinv invert a complex number
csqrt complex square root of a complex number
cexp complex exponential of a complex number
crmul multiply a complex number by a real number
rcabs real magnitude of a complex number

Structure:
typedef struct _complexStruct {  complex number
float r,i;
} complex;

Function Prototypes:
complex cadd (complex a, complex b);
complex csub (complex a, complex b);
complex cmul (complex a, complex b);
complex cdiv (complex a, complex b);
float rcabs (complex z);
complex cmplx (float re, float im);
complex conjg (complex z);
complex cneg (complex z);
complex cinv (complex z);
complex csqrt (complex z);
complex cexp (complex z);
complex crmul (complex a, float x);

Notes:
The function "rcabs" was originally called "fcabs". This produced
a collision on some systems so a new name was chosen.

Reference:
Adapted from Press et al, 1988, Numerical Recipes in C (Appendix E).

Author:  Dave Hale, Colorado School of Mines, 06/02/89
Modified:  Dave Hale, Colorado School of Mines, 04/26/90
```

```
Added function cinv().
```

```
COMPLEXD - Functions to manipulate double-precision complex numbers

dcadd add two dcomplex numbers
dcsub subtract two dcomplex numbers
dcmul multiply two dcomplex numbers
dcdiv divide two dcomplex numbers
dcmplx make a dcomplex number from two real numbers
dconjg dcomplex conjugate of a dcomplex number
dcneg negate a dcomplex number
dcinv invert a dcomplex number
dcsqrt dcomplex square root of a dcomplex number
dcexp dcomplex exponential of a dcomplex number
dcrmul multiply a dcomplex number by a real number
drcabs real magnitude of a dcomplex number

Structure:
typedef struct _dcomplexStruct {  dcomplex number
double r,i;
} dcomplex;

Function Prototypes:
dcomplex dcadd (dcomplex a, dcomplex b);
dcomplex dcsub (dcomplex a, dcomplex b);
dcomplex dcmul (dcomplex a, dcomplex b);
dcomplex dcdiv (dcomplex a, dcomplex b);
double drcabs (dcomplex z);
dcomplex dcmplx (double re, double im);
dcomplex dconjg (dcomplex z);
dcomplex dcneg (dcomplex z);
dcomplex dcinv (dcomplex z);
dcomplex dcsqrt (dcomplex z);
dcomplex dcexp (dcomplex z);
dcomplex dcrmul (dcomplex a, double x);

Notes:
The function "drcabs" was originally called "fcabs". This produced
a collision on some systems so a new name was chosen.

Reference:
Adapted from Press et al, 1988, Numerical Recipes in C (Appendix E).

Author:  Dave Hale, Colorado School of Mines, 06/02/89
Modified:  Dave Hale, Colorado School of Mines, 04/26/90
```

```
Added function dcinv().
```

```
COMPLEXF  - Subroutines to perform operations on complex numbers.
This set of functions complement the one in complex.c
of the CWP library

cipow raise a complex number to an integer power
crpow raise a complex number to a real power
rcpow raise a real number to a complex power
ccpow raise a complex number to a complex power
ccos compute the complex cosine of a complex angle
csin compute the complex sine of a complex angle
ccosh compute the complex hyperbolic cosine of a complex angle
csinh compute the complex hyperbolic sine of a complex angle
cexp1 compute the complex exponential of a complex number
clog compute the complex logarithm of a complex number
Function Prototypes:
complex cipow(complex a, int p);
complex crpow(complex a, float p);
complex rcpow(float a, complex p);
complex ccpow (complex a, complex p)
complex ccos(complex a);
complex csin(complex a);
complex ccosh(complex a);
complex csinh(complex a);
complex cexp1(complex a);
complex clog(complex a);
Credits:
Dave Hale, original version in C++
Gabriel Alvarez, translation to C
```

```
COMPLEXFD  - Subroutines to perform operations on double complex numbers.
This set of functions complement the one in complexd.c
of the CWP library

dcipow raise a double complex number to an integer power
dcrpow raise a double complex number to a real power
rdcpow raise a real number to a double complex power
dcdcpow raise a double complex number to a double complex power
dccos compute the double complex cosine of a double complex angle
dcsin compute the double complex sine of a double complex angle
dccosh compute the double complex hyperbolic cosine of a double complex angle
dcsinh compute the double complex hyperbolic sine of a double complex angle
dcexp1 compute the double complex exponential of a double complex number
dclog compute the double complex logarithm of a double complex number
Function Prototypes:
dcomplex dcipow(dcomplex a, int p);
dcomplex dcrpow(dcomplex a, float p);
dcomplex rdcpow(float a, dcomplex p);
dcomplex dcdcpow (dcomplex a, dcomplex p)
dcomplex dccos(dcomplex a);
dcomplex dcsin(dcomplex a);
dcomplex dccosh(dcomplex a);
dcomplex dcsinh(dcomplex a);
dcomplex dcexp1(dcomplex a);
dcomplex dclog(dcomplex a);
Credits:
Dave Hale, original version in C++
Gabriel Alvarez, translation to C
```

```
CONVOLUTION - Compute z = x convolved with y

conv compute the convolution of two input vector arrays

Input:
lx length of x array
ifx sample index of first x
x array[lx] to be convolved with y
ly length of y array
ify sample index of first y
y array[ly] with which x is to be convolved
lz length of z array
ifz sample index of first z

Output:
z array[lz] containing x convolved with y

Function Prototype:
void conv (int lx, int ifx, float *x, int ly, int ify, float *y,
int lz, int ifz, float *z);

Notes:
The operation z = x convolved with y is defined to be
           ifx+lx-1
    z[i] =   sum    x[j]*y[i-j]  ;  i = ifz,...,ifz+lz-1
            j=ifx
The x samples are contained in x[0], x[1], ..., x[lx-1]; likewise for
the y and z samples.  The sample indices of the first x, y, and z values
determine the location of the origin for each array.  For example, if
z is to be a weighted average of the nearest 5 samples of y, one might
use
...
x[0] = x[1] = x[2] = x[3] = x[4] = 1.0/5.0;
conv(5,-2,x,lx,0,y,ly,0,z);
...
In this example, the filter x is symmetric, with index of first sample = -2.

This function is optimized for architectures that can simultaneously perform
a multiply, add, and one load from memory; e.g., the IBM RISC System/6000.
Because, for each value of i, it accumulates the convolution sum z[i] in a
scalar, this function is not likely to be optimal for vector architectures.

Author:  Dave Hale, Colorado School of Mines, 11/23/91
```

```
CUBICSPLINE - Functions to compute CUBIC SPLINE interpolation coefficients

cakima compute cubic spline coefficients via Akima's method
  (continuous 1st derivatives, only)
cmonot compute cubic spline coefficients via the Fritsch-Carlson method
  (continuous 1st derivatives, only)
csplin compute cubic spline coefficients for interpolation
  (continuous 1st and 2nd derivatives)

chermite compute cubic spline coefficients via Hermite Polynomial
  (continuous 1st derivatives only)
Function Prototypes:
void cakima   (int n, float x[], float y[], float yd[][4]);
void cmonot   (int n, float x[], float y[], float yd[][4]);
void csplin   (int n, float x[], float y[], float yd[][4]);
void chermite (int n, float x[], float y[], float yd[][4]);

Input:
n number of samples
x   array[n] of monotonically increasing or decreasing abscissae
y array[n] of ordinates

Output:
yd array[n][4] of cubic interpolation coefficients (see notes)

Notes:
The computed cubic spline coefficients are as follows:
yd[i][0] = y(x[i])    (the value of y at x = x[i])
yd[i][1] = y'(x[i])   (the 1st derivative of y at x = x[i])
yd[i][2] = y''(x[i])  (the 2nd derivative of y at x = x[i])
yd[i][3] = y'''(x[i]) (the 3rd derivative of y at x = x[i])

To evaluate y(x) for x between x[i] and x[i+1] and h = x-x[i],
use the computed coefficients as follows:
y(x) = yd[i][0]+h*(yd[i][1]+h*(yd[i][2]/2.0+h*yd[i][3]/6.0))

Akima's method provides continuous 1st derivatives, but 2nd and
3rd derivatives are discontinuous.  Akima's method is not linear,
in that the interpolation of the sum of two functions is not the
same as the sum of the interpolations.

The Fritsch-Carlson method yields continuous 1st derivatives, but 2nd
and 3rd derivatives are discontinuous.  The method will yield a
```

monotonic interpolant for monotonic data. 1st derivatives are set
to zero wherever first divided differences change sign.

The method used by "csplin" yields continuous 1st and 2nd derivatives.

References:
See Akima, H., 1970, A new method for
interpolation and smooth curve fitting based on local procedures,
Journal of the ACM, v. 17, n. 4, p. 589-602.

For more information, see Fritsch, F. N., and Carlson, R. E., 1980,
Monotone piecewise cubic interpolation: SIAM J. Numer. Anal., v. 17,
n. 2, p. 238-246.
Also, see the book by Kahaner, D., Moler, C., and Nash, S., 1989,
Numerical Methods and Software, Prentice Hall. This function was
derived from SUBROUTINE PCHEZ contained on the diskette that comes
with the book.

For more general information on spline functions of all types see the book by:
Greville, T.N.E, 1969, Theory and Applications of Spline Functions,
Academic Press.

Author: Dave Hale, Colorado School of Mines c. 1989, 1990, 1991

```
DBLAS - Double precision Basic Linear Algebra subroutines
(adapted from LINPACK FORTRAN):

idamax return index of element with maximum absolute value
dasum return sum of absolute values
daxpy compute y[i] = a*x[i]+y[i]
dcopy copy x[i] to y[i] (i.e., set y[i] = x[i])
ddot return sum of x[i]*y[i] (i.e., return the dot product of x and y)
dnrm2 return square root of sum of squares of x[i]
dscal compute x[i] = a*x[i]
dswap swap x[i] and y[i]

Function Prototypes:
int idamax (int n, double *sx, int incx);
double dasum (int n, double *sx, int incx);
void daxpy (int n, double sa, double *sx, int incx, double *sy, int incy);
void dcopy (int n, double *sx, int incx, double *sy, int incy);
double ddot (int n, double *sx, int incx, double *sy, int incy);
double dnrm2 (int n, double *sx, int incx);
void dscal (int n, double sa, double *sx, int incx);
void dswap (int n, double *sx, int incx, double *sy, int incy);

idmax:
Input:
n number of elements in array
sx array[n] of elements
incx increment between elements

Returned:
index of element with maximum absolute value (idamax)

dasum:
Input:
n number of elements in array
sx array[n] of elements
incx increment between elements

Returned:
sum of absolute values (dasum)

daxpy:
Input:
n number of elements in arrays
```

```
sa the scalar multiplier
sx array[n] of elements to be scaled and added
incx increment between elements of sx
sy array[n] of elements to be added
incy increment between elements of sy

Output:
sy array[n] of accumulated elements

dcopy:
Input:
n number of elements in arrays
sx array[n] of elements to be copied
incx increment between elements of sx
incy increment between elements of sy

Output:
sy array[n] of copied elements

ddot:
Input:
n number of elements in arrays
sx array[n] of elements
incx increment between elements of sx
sy array[n] of elements
incy increment between elements of sy

Returned: dot product of the two arrays

dnrm2:
Input:
n number of elements in array
sx array[n] of elements
incx increment between elements

Returned: square root of sum of squares of x[i]

dscal:
Input:
n number of elements in array
sa the scalar multiplier
sx array[n] of elements
incx increment between elements
```

```
Output:
sx array[n] of scaled elements

Author:  Dave Hale, Colorado School of Mines, 10/01/89
```

```
DGE - Double precision Gaussian Elimination matrix subroutines  adapted
from LINPACK FORTRAN:

dgefa Gaussian elimination to obtain the LU factorization of a matrix.
dgeco Gaussian elimination to obtain the LU factorization and condition
number of a matrix.
dgesl Solve linear system Ax = b or A'x = b after LU factorization.

Function Prototypes:
void dgefa (double **a, int n, int *ipvt, int *info);
void dgeco (double **a, int n, int *ipvt, double *rcond, double *z);
void dgesl (double **a, int n, int *ipvt, double *b, int job);

dgfa:
Input:
a matrix[n][n] to be factored (see notes below)
n dimension of a

Output:
a matrix[n][n] factored (see notes below)
ipvt indices of pivot permutations (see notes below)
info index of last zero pivot (or -1 if no zero pivots)

dgeco:
Input:
a matrix[n][n] to be factored (see notes below)
n dimension of a

Output:
a matrix[n][n] factored (see notes below)
ipvt indices of pivot permutations (see notes below)
rcond reciprocal of condition number (see notes below)

Workspace:
z array[n]

dgesl:
Input:
a matrix[n][n] that has been LU factored (see notes below)
n dimension of a
ipvt indices of pivot permutations (see notes below)
b right-hand-side vector[n]
job =0 to solve Ax = b
```

```
=1 to solve A'x = b

Output:
b solution vector[n]

Notes:
These functions were adapted from LINPACK FORTRAN.  Because two-dimensional
arrays cannot be declared with variable dimensions in C, the matrix a
is actually a pointer to an array of pointers to floats, as declared
above and used below.

Elements of a are stored as follows:
a[0][0]    a[1][0]    a[2][0]   ... a[n-1][0]
a[0][1]    a[1][1]    a[2][1]   ... a[n-1][1]
a[0][2]    a[1][2]    a[2][2]   ... a[n-1][2]
.                                     .
.              .                      .
.                         .           .
.                                     .
a[0][n-1]  a[1][n-1]  a[2][n-1] ... a[n-1][n-1]

Both the factored matrix a and the pivot indices ipvt are required
to solve linear systems of equations via dgesl.


dgeco:
Given the reciprocal of the condition number, rcond, and the double
epsilon, DBL_EPSILON, the number of significant decimal digits, nsdd,
in the solution of a linear system of equations may be estimated by:
nsdd = (int)log10(rcond/DBL_EPSILON)

Author:  Dave Hale, Colorado School of Mines, 1989
```

```
PFAFFT - Functions to perform Prime Factor (PFA) FFT's, in place

npfa_d return valid n for complex-to-complex PFA
npfar_d return valid n for real-to-complex/complex-to-real PFA
npfao_d return optimal n for complex-to-complex PFA
npfaro_d return optimal n for real-to-complex/complex-to-real PFA
pfacc_d 1D PFA complex to complex
pfacr_d 1D PFA complex to real
pfarc_d 1D PFA real to complex
pfamcc_d multiple PFA complex to real
pfa2cc_d 2D PFA complex to complex
pfa2cr_d 2D PFA complex to real
pfa2rc_d 2D PFA real to complex

Function Prototypes:
int npfa_d (int nmin);
int npfao_d (int nmin, int nmax);
int npfar_d (int nmin);
int npfaro_d (int nmin, int nmax);
void pfacc_d (int isign, int n, real_complex z[]);
void pfacr_d (int isign, int n, real_complex cz[], real rz[]);
void pfarc_d (int isign, int n, real rz[], real_complex cz[]);
void pfamcc_d (int isign, int n, int nt, int k, int kt, real_complex z[]);
void pfa2cc_d (int isign, int idim, int n1, int n2, real_complex z[]);
void pfa2cr_d (int isign, int idim, int n1, int n2, real_complex cz[], real rz[]);
void pfa2rc_d (int isign, int idim, int n1, int n2, real rz[], real_complex cz[]);

npfa_d:
Input:
nmin lower bound on returned value (see notes below)

Returned: valid n for prime factor fft

npfao_d
Input:
nmin lower bound on returned value (see notes below)
nmax desired (but not guaranteed) upper bound on returned value

Returned: valid n for prime factor fft

npfar_d
Input:
nmin lower bound on returned value
```

```
Returned: valid n for real-to-real_complex/real_complex-to-real prime factor fft

npfaro_d:
Input:
nmin lower bound on returned value
nmax desired (but not guaranteed) upper bound on returned value

Returned: valid n for real-to-real_complex/real_complex-to-real prime factor fft

pfacc_d:
Input:
isign sign of isign is the sign of exponent in fourier kernel
n length of transform (see notes below)
z array[n] of real_complex numbers to be transformed in place

Output:
z array[n] of real_complex numbers transformed

pfacr_d:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
n           length of transform (see notes below)
cz          array[n/2+1] of real_complex values (may be equivalenced to rz)

Output:
rz          array[n] of real values (may be equivalenced to cz)

pfarc_d:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
n           length of transform; must be even (see notes below)
rz          array[n] of real values (may be equivalenced to cz)

Output:
cz          array[n/2+1] of real_complex values (may be equivalenced to rz)

pfamcc_d:
Input:
isign        sign of isign is the sign of exponent in fourier kernel
n            number of real_complex elements per transform (see notes below)
nt           number of transforms
k            stride in real_complex elements within transforms
```

```
kt           stride in real_complex elements between transforms
z            array of real_complex elements to be transformed in place

Output:
z array of real_complex elements transformed

pfa2cc_d:
Input:
isign        sign of isign is the sign of exponent in fourier kernel
idim         dimension to transform, either 1 or 2 (see notes)
n1           1st (fast) dimension of array to be transformed (see notes)
n2           2nd (slow) dimension of array to be transformed (see notes)
z            array[n2][n1] of real_complex elements to be transformed in place

Output:
z array[n2][n1] of real_complex elements transformed

pfa2cr_d:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
idim        dimension to transform, which must be either 1 or 2 (see notes)
n1          1st (fast) dimension of array to be transformed (see notes)
n2          2nd (slow) dimension of array to be transformed (see notes)
cz          array of real_complex values (may be equivalenced to rz)

Output:
rz          array of real values (may be equivalenced to cz)

pfa2rc_d:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
idim        dimension to transform, which must be either 1 or 2 (see notes)
n1          1st (fast) dimension of array to be transformed (see notes)
n2          2nd (slow) dimension of array to be transformed (see notes)
rz          array of real values (may be equivalenced to cz)

Output:
cz          array of real_complex values (may be equivalenced to rz)

Notes:
Table of valid n and cost for prime factor fft.  For each n, cost
was estimated to be the inverse of the number of ffts done in 1 sec
on an IBM RISC System/6000 Model 320H, by Dave Hale, 08/04/91.
```

```
(Redone by Jack Cohen for 15 sec to rebuild NTAB table on advice of
David and Gregory Chudnovsky, 05/03/94).
Cost estimates are least accurate for very small n.  An alternative method
for estimating cost would be to count multiplies and adds, but this
method fails to account for the overlapping of multiplies and adds
that is possible on some computers, such as the IBM RS/6000 family.

npfa_d:
The returned n will be composed of mutually prime factors from
the set {2,3,4,5,7,8,9,11,13,16}.  Because n cannot exceed
720720 = 5*7*9*11*13*16, 720720 is returned if nmin exceeds 720720.

npfao_d:
The returned n will be composed of mutually prime factors from
the set {2,3,4,5,7,8,9,11,13,16}.  Because n cannot exceed
720720 = 5*7*9*11*13*16, 720720 is returned if nmin exceeds 720720.
If nmin does not exceed 720720, then the returned n will not be
less than nmin.  The optimal n is chosen to minimize the estimated
cost of performing the fft, while satisfying the constraint, if
possible, that n not exceed nmax.

npfar and npfaro:
Current implemenations of real-to-real_complex and real_complex-to-real prime
factor ffts require that the transform length n be even and that n/2
be a valid length for a real_complex-to-real_complex prime factor fft.  The
value returned by npfar satisfies these conditions.  Also, see notes
for npfa.

pfacc:
n must be factorable into mutually prime factors taken
from the set {2,3,4,5,7,8,9,11,13,16}.  in other words,
n = 2**p * 3**q * 5**r * 7**s * 11**t * 13**u
where
0 <= p <= 4,  0 <= q <= 2,  0 <= r,s,t,u <= 1
is required for pfa to yield meaningful results.  this
restriction implies that n is restricted to the range
1 <= n <= 720720 (= 5*7*9*11*13*16)

pfacr:
Because pfacr uses pfacc to do most of the work, n must be even
and n/2 must be a valid length for pfacc.  The simplest way to
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.
```

```
pfarc:
Because pfarc uses pfacc to do most of the work, n must be even
and n/2 must be a valid length for pfacc.  The simplest way to
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.

pfamcc:
To perform a two-dimensional transform of an n1 by n2 real_complex array
(assuming that both n1 and n2 are valid "n"), stored with n1 fast
and n2 slow:
    pfamcc(isign,n1,n2,1,n1,z); (to transform 1st dimension)
    pfamcc(isign,n2,n1,n1,1,z); (to transform 2nd dimension)

pfa2cc:
Only one (either the 1st or 2nd) dimension of the 2-D array is transformed.

If idim equals 1, then n2 transforms of n1 real_complex elements are performed;
else, if idim equals 2, then n1 transforms of n2 real_complex elements are
performed.

Although z appears in the argument list as a one-dimensional array,
z may be viewed as an n1 by n2 two-dimensional array:  z[n2][n1].

Valid n is computed via the "np" subroutines.

To perform a two-dimensional transform of an n1 by n2 real_complex array
(assuming that both n1 and n2 are valid "n"), stored with n1 fast
and n2 slow:  pfa2cc(isign,1,n1,n2,z);  pfa2cc(isign,2,n1,n2,z);

pfa2cr:
If idim equals 1, then n2 transforms of n1/2+1 real_complex elements to n1 real
elements are performed; else, if idim equals 2, then n1 transforms of n2/2+1
real_complex elements to n2 real elements are performed.

Although rz appears in the argument list as a one-dimensional array,
rz may be viewed as an n1 by n2 two-dimensional array:  rz[n2][n1].
Likewise, depending on idim, cz may be viewed as either an n1/2+1 by
n2 or an n1 by n2/2+1 two-dimensional array of real_complex elements.

Let n denote the transform length, either n1 or n2, depending on idim.
Because pfa2rc uses pfa2cc to do most of the work, n must be even
and n/2 must be a valid length for pfa2cc.  The simplest way to
```

```
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.

pfa2rc:
If idim equals 1, then n2 transforms of n1 real elements to n1/2+1 real_complex
elements are performed; else, if idim equals 2, then n1 transforms of n2
real elements to n2/2+1 real_complex elements are performed.

Although rz appears in the argument list as a one-dimensional array,
rz may be viewed as an n1 by n2 two-dimensional array:  rz[n2][n1].
Likewise, depending on idim, cz may be viewed as either an n1/2+1 by
n2 or an n1 by n2/2+1 two-dimensional array of real_complex elements.

Let n denote the transform length, either n1 or n2, depending on idim.
Because pfa2rc uses pfa2cc to do most of the work, n must be even
and n/2 must be a valid length for pfa2cc.  The simplest way to
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.

References:
Temperton, C., 1985, Implementation of a self-sorting
in-place prime factor fft algorithm:  Journal of
Computational Physics, v. 58, p. 283-299.

Temperton, C., 1988, A new set of minimum-add rotated
rotated dft modules: Journal of Computational Physics,
v. 75, p. 190-198.

Press et al, 1988, Numerical Recipes in C, p. 417.

Author:  Dave Hale, Colorado School of Mines, 04/27/89
Revised by Baoniu Han to handle double precision. 12/14/98
```

CWP_Exit - exit subroutine for CWP/SU codes

```
FRANNOR - functions to generate a pseudo-random float normally distributed
with N(0,1); i.e., with zero mean and unit variance.

frannor return a normally distributed random float
srannor seed random number generator for normal distribution

Function Prototypes:
float frannor (void);
void srannor (int seed);

frannor:
Input: (none)
Returned: normally distributed random float

srannor:
Input:
seed different seeds yield different sequences of random numbers.

Notes:
Adapted from subroutine rnor in Kahaner,  Moler and Nash (1988)
which in turn was based on  an algorithm by
Marsaglia and Tsang (1984).

References:
Numerical Methods and Software", D.
Kahaner, C. Moler, S. Nash, Prentice Hall, 1988.

Marsaglia G. and Tsang, W. W., 1984,
A fast, easily implemented method for sampling from decreasing or symmetric
unimodal density functions:  SIAM J. Sci. Stat. Comput., v. 5, no. 2,
p. 349-359.

Author:  Dave Hale, Colorado School of Mines, 01/21/89
```

```
FRANUNI - Functions to generate a pseudo-random float uniformly distributed
on [0,1); i.e., between 0.0 (inclusive) and 1.0 (exclusive)

franuni return a random float
sranuni seed random number generator

Function Prototypes:
float franuni (void);
void sranuni (int seed);

franuni:
Input: (none)
Returned: pseudo-random float

sranuni:
seed different seeds yield different sequences of random numbers.

Notes:
Adapted from subroutine uni in Kahaner, Moler, and Nash (1988).
This book references a set of unpublished notes by
Marsaglia.

According to the reference, this random
number generator "passes all known tests and has a period that is ...
approximately 10^19".

References:
Numerical Methods and Software", D. Kahaner, C. Moler, S. Nash,
Prentice Hall, 1988.

Marsaglia G., "Comments on the perfect uniform random number generator
Unpublished notes, Wash S. U.

Author:  Dave Hale, Colorado School of Mines, 12/30/89
```

```
HANKEL - Functions to compute discrete Hankel transforms

hankelalloc allocate and return a pointer to a Hankel transformer
hankelfree  free a Hankel transformer
hankel0 compute the zeroth-order Hankel transform
hankel1 compute the first-order Hankel transform

Function Prototypes:
void *hankelalloc (int nfft);
void hankelfree (void *ht);
void hankel0 (void *ht, float f[], float h[]);
void hankel1 (void *ht, float f[], float h[]);

hankelalloc:
Input:
nfft valid length for real to complex fft  (see notes below)

Returned:
pointer to Hankel transformer

hankelfree:
Input:
ht pointer to Hankel transformer (as returned by hankelalloc)

hankel0:
Input:
ht pointer to Hankel transformer (as returned by hankelalloc)
f array[nfft/2+1] to be transformed

Output:
h array[nfft/2+1] transformed

hankel1:
Input:
ht pointer to Hankel transformer (as returned by hankelalloc)
f array[nfft/2+1] to be transformed

Output:
h array[nfft/2+1] transformed

Notes:
The zeroth-order Hankel transform is defined by:
```

```
        Infinity
h0(k) = Integral dr r j0(k*r) f(r)
   0
```

where j0 denotes the zeroth-order Bessel function.

The first-order Hankel transform is defined by:

```
        Infinity
h1(k) = Integral dr r j1(k*r) f(r)
   0
```

where j1 denotes the first-order Bessel function.

The Hankel transform is its own inverse.

The Hankel transform is computed by an Abel transform followed by
a Fourier transform.

References:
Hansen, E. W., 1985, Fast Hankel transform algorithm: IEEE Trans. on
Acoustics, Speech and Signal Processing, v. ASSP-33, n. 3, p. 666-671.
(Beware of several errors in the equations in this paper!)

Authors: Dave Hale, Colorado School of Mines, 06/04/90

```
Hartley - routines for fast Hartley transform

srfht -  in-place FHT using the split-radix algorithm
dsrfht - in-place FHT using the split-radix algorithm (double precision)
r4fht - in-place FHT using the radix-4 algorithm
nextpow2 - find m such that n=2^m via lookup table
nextpow4 - find m such that n=4^m via lookup table
srfht - split-radix in-place FHT

Function Prototypes:

void srfht(int *n, int *m, float *f);
void dsrfht(int *n, int *m, double *f);
void r4fht(int n, int m, float *f);
int nextpow2(int n);
int nextpow4(int n);



Notes:
srfht -  in-place FHT using the split-radix algorithm
(single precision)
does not include division by n
n and m are not modified
usage:  srfht(int *n, int *m, float *f)
n  length of input sequence n=2^m
m  exponent such that n=2^m
   f  input sequence to FHT

dsrfht - in-place FHT using the split-radix algorithm
(double precision)
does not include division by n
n and m are not modified
usage:  srfht(int *n, int *m, double *f)
n  length of input sequence n=2^m
m  exponent such that n=2^m
   f  input sequence to FHT

r4fht - in-place FHT using the radix-4 algorithm
does not include division by n
usage:  r4fht(int n, int m, float *f)
n  length of input sequence n=4^m
m  exponent such that n=4^m
```

```
   f  input sequence to FHT

nextpow2 - find m such that n=2^m via lookup table
max(n)=2^24

nextpow4 - find m such that n=4^m via lookup table
max(n)=4^12

srfht - split-radix in-place FHT
  n length of input sequence n=2^m
  m exponent such that n=2^m
    f input sequence to FHT
   Reference:
       Sorensen, H. V., Jones, D. L., Burrus, C. S, & Heideman, M. T.,
       1985, On computing the Hartley transform: IEEE Trans. Acoust.,
       Speech, and Signal Proc., v. ASSP-33, no. 4, p. 1231-1238.

possible improvements:
find optimum length for FHT (nfhto)
use bit shift operators in r4fht
  Credits: CENPET: Werner M. Heigl, April 2006
```

```
HILBERT - Compute Hilbert transform y of x

hilbert compute the Hilbert transform

Function Prototype:
void hilbert (int n, float x[], float y[]);

Input:
n length of x and y
x array[n] to be Hilbert transformed

Output:
y array[n] containing Hilbert transform of x

Notes:
The Hilbert transform is computed by convolving x with a
windowed (approximate) version of the ideal Hilbert transformer.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
HOLBERG1D - Compute coefficients of Holberg's 1st derivative filter

holberg1d comput the coefficients of Holberg's 1st derivative filter

Function Prototype:
void holbergd1 (float e, int n, float d[]);

Input:
e maximum relative error in group velocity
n number of coefficients in filter (must be 2, 4, 6, or 8)

Output:
d array[n] of coefficients

Notes:
Coefficients are output in a form suitable for convolution.  The
derivative is centered halfway between coefficients d[n/2-1] and d[n/2].

Coefficients are computed via the power series method of Kindelan et al.,
1990, On the construction and efficiency of staggered numerical
differentiators for the wave equation:  Geophysics 55, 107-110.
See also, Holberg, 1987, Computational aspects of the choice of
operator and sampling interval for numerical differentiation in
large-scale simulation of wave phenomena:  Geophys. Prosp., 35, 629-655

Reference:
Kindelan et al., 1990,
On the construction and efficiency of staggered numerical
differentiators for the wave equation:  Geophysics 55, 107-110.

See also, Holberg, 1987, Computational aspects of the choice of
operator and sampling interval for numerical differentiation in
large-scale simulation of wave phenomena:  Geophys. Prosp., 35, 629-655

Author:  Dave Hale, Colorado School of Mines, 06/06/91
```

```
INTCUB - evaluate y(x), y'(x), y''(x), ... via piecewise cubic interpolation

intcub evaluate y(x), y'(x), y''(x), ... via piecewise spline
interpolation

Function Prototype:
void intcub (int ideriv, int nin, float xin[], float ydin[][4],

Input:
ideriv =0 if y(x) desired; =1 if y'(x) desired, ...
nin length of xin and ydin arrays
xin array[nin] of monotonically increasing or decreasing x values
ydin array[nin][4] of y(x), y'(x), y''(x), and y'''(x)
nout length of xout and yout arrays
xout array[nout] of x values at which to evaluate y(x), y'(x), ...

Output:
yout array[nout] of y(x), y'(x), ... values

Notes:
xin values must be monotonically increasing or decreasing.

Extrapolation of the function y(x) for xout values outside the range
spanned by the xin values is performed using the derivatives in
ydin[0][0:3] or ydin[nin-1][0:3], depending on whether xout is closest
to xin[0] or xin[nin-1], respectively.

Reference:
See: Greville, T. N., Theory and Application of Spline Functions, Academic Press
for a general discussion of spline functions.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

INTL2B - bilinear interpolation of a 2-D array of bytes

intl2b bilinear interpolation of a 2-D array of bytes

Function Prototype:
void intl2b (int nxin, float dxin, float fxin,
int nyin, float dyin, float fyin, unsigned char *zin,
int nxout, float dxout, float fxout,
int nyout, float dyout, float fyout, unsigned char *zout);

Input:
nxin number of x samples input (fast dimension of zin)
dxin x sampling interval input
fxin first x sample input
nyin number of y samples input (slow dimension of zin)
dyin y sampling interval input
fyin first y sample input
zin array[nyin][nxin] of input samples (see notes)
nxout number of x samples output (fast dimension of zout)
dxout x sampling interval output
fxout first x sample output
nyout number of y samples output (slow dimension of zout)
dyout y sampling interval output
fyout first y sample output

Output:
zout array[nyout][nxout] of output samples (see notes)

Notes:
The arrays zin and zout must passed as pointers to the first element of
a two-dimensional contiguous array of unsigned char values.

Constant extrapolation of zin is used to compute zout for
output x and y outside the range of input x and y.

For efficiency, this function builds a table of interpolation
coefficents pre-multiplied by byte values. To keep the table
reasonably small, the interpolation does not distinguish
between x and y values that differ by less than dxin/ICMAX
and dyin/ICMAX, respectively, where ICMAX is a parameter
defined above.

Author: Dave Hale, Colorado School of Mines, c. 1989-1991.

```
INTLINC - evaluate complex y(x) via linear interpolation of y(x[0]), y(x[1]), ...

Function Prototype:
void intlinc (int nin, float xin[], complex yin[], complex yinl, complex yinr,
int nout, float xout[], complex yout[]);

Input:
nin length of xin and yin arrays
xin array[nin] of monotonically increasing or decreasing x values
yin array[nin] of input y(x) values
yinl value used to extraplate y(x) to left of input yin values
yinr value used to extraplate y(x) to right of input yin values
nout length of xout and yout arrays
xout array[nout] of x values at which to evaluate y(x)

Output:
yout array[nout] of linearly interpolated y(x) values

Notes:
xin values must be monotonically increasing or decreasing.

Extrapolation of the function y(x) for xout values outside the range
spanned by the xin values in performed as follows:

For monotonically increasing xin values,
yout=yinl if xout<xin[0], and yout=yinr if xout>xin[nin-1].

For monotonically decreasing xin values,
yout=yinl if xout>xin[0], and yout=yinr if xout<xin[nin-1].

If nin==1, then the monotonically increasing case is used.
/
```

```
INTLIN - evaluate y(x) via linear interpolation of y(x[0]), y(x[1]), ...

intlin evaluate y(x) via linear interpolation of y(x[0]), y(x[1]), ...

Function Prototype:
void intlin (int nin, float xin[], float yin[], float yinl, float yinr,
int nout, float xout[], float yout[]);

Input:
nin length of xin and yin arrays
xin array[nin] of monotonically increasing or decreasing x values
yin array[nin] of input y(x) values
yinl value used to extraplate y(x) to left of input yin values
yinr value used to extraplate y(x) to right of input yin values
nout length of xout and yout arrays
xout array[nout] of x values at which to evaluate y(x)

Output:
yout array[nout] of linearly interpolated y(x) values

Notes:
xin values must be monotonically increasing or decreasing.

Extrapolation of the function y(x) for xout values outside the range
spanned by the xin values in performed as follows:

For monotonically increasing xin values,
yout=yinl if xout<xin[0], and yout=yinr if xout>xin[nin-1].

For monotonically decreasing xin values,
yout=yinl if xout>xin[0], and yout=yinr if xout<xin[nin-1].

If nin==1, then the monotonically increasing case is used.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
INTLIRR2B - bilinear interpolation of a 2-D array of bytes

intlirr2b bilinear interpolation of a 2-D array of bytes
                where x spacings are irregular

Function Prototype:
void intlirr2b (int nxin, float *xin,
int nyin, float dyin, float fyin, unsigned char *zin,
int nxout, float dxout, float fxout,
int nyout, float dyout, float fyout, unsigned char *zout);

Input:
nxin number of x samples input (fast dimension of zin)
xin array of (irregular) input x-coordinates
nyin number of y samples input (slow dimension of zin)
dyin y sampling interval input
fyin first y sample input
zin array[nyin][nxin] of input samples (see notes)
nxout number of x samples output (fast dimension of zout)
dxout x sampling interval output
fxout first x sample output
nyout number of y samples output (slow dimension of zout)
dyout y sampling interval output
fyout first y sample output

Output:
zout array[nyout][nxout] of output samples (see notes)

Notes:
The arrays zin and zout must passed as pointers to the first element of
a two-dimensional contiguous array of unsigned char values.

Constant extrapolation of zin is used to compute zout for
output x and y outside the range of input x and y.

For efficiency, this function builds a table of interpolation
coefficents pre-multiplied by byte values.  To keep the table
reasonably small, the interpolation does not distinguish
between x and y values that differ by less than dxin/ICMAX
and dyin/ICMAX, respectively, where ICMAX is a parameter
defined above.

Author:  Dave Hale, Colorado School of Mines, c. 1989-1991.
```

```
INTSINC8 - Functions to interpolate uniformly-sampled data via 8-coeff. sinc
approximations:

ints8c interpolation of a uniformly-sampled complex function y(x) via an
 8-coefficient sinc approximation.
ints8r Interpolation of a uniformly-sampled real function y(x) via a
table of 8-coefficient sinc approximations

Function Prototypes:
void ints8c (int nxin, float dxin, float fxin, complex yin[],
complex yinl, complex yinr, int nxout, float xout[], complex yout[]);
void ints8r (int nxin, float dxin, float fxin, float yin[],
float yinl, float yinr, int nxout, float xout[], float yout[]);

Input:
nxin number of x values at which y(x) is input
dxin x sampling interval for input y(x)
fxin x value of first sample input
yin array[nxin] of input y(x) values:  yin[0] = y(fxin), etc.
yinl value used to extrapolate yin values to left of yin[0]
yinr value used to extrapolate yin values to right of yin[nxin-1]
nxout number of x values a which y(x) is output
xout array[nxout] of x values at which y(x) is output

Output:
yout array[nxout] of output y(x):  yout[0] = y(xout[0]), etc.

Notes:
Because extrapolation of the input function y(x) is defined by the
left and right values yinl and yinr, the xout values are not restricted
to lie within the range of sample locations defined by nxin, dxin, and
fxin.

The maximum error for frequiencies less than 0.6 nyquist is less than
one percent.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
INTTABLE8 -  Interpolation of a uniformly-sampled complex function y(x)
via a table of 8-coefficient interpolators

intt8c interpolation of a uniformly-sampled complex function y(x)
via a table of 8-coefficient interpolators
intt8r interpolation of a uniformly-sampled real function y(x) via a
table of 8-coefficient interpolators

Function Prototype:
void intt8c (int ntable, float table[][8],
int nxin, float dxin, float fxin, complex yin[],
complex yinl, complex yinr, int nxout, float xout[], complex yout[]);
void intt8r (int ntable, float table[][8],
int nxin, float dxin, float fxin, float yin[],
float yinl, float yinr, int nxout, float xout[], float yout[]);

Input:
ntable number of tabulated interpolation operators; ntable>=2
table array of tabulated 8-point interpolation operators
nxin number of x values at which y(x) is input
dxin x sampling interval for input y(x)
fxin x value of first sample input
yin array of input y(x) values:  yin[0] = y(fxin), etc.
yinl value used to extrapolate yin values to left of yin[0]
yinr value used to extrapolate yin values to right of yin[nxin-1]
nxout number of x values a which y(x) is output
xout array of x values at which y(x) is output

Output:
yout array of output y(x) values:  yout[0] = y(xout[0]), etc.

NOTES:
ntable must not be less than 2.

The table of interpolation operators must be as follows:

Let d be the distance, expressed as a fraction of dxin, from a particular
xout value to the sampled location xin just to the left of xout.  Then,
for d = 0.0,

table[0][0:7] = 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0

are the weights applied to the 8 input samples nearest xout.
```

Likewise, for d = 1.0,

table[ntable-1][0:7] = 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0

are the weights applied to the 8 input samples nearest xout. In general, for d = (float)itable/(float)(ntable-1), table[itable][0:7] are the weights applied to the 8 input samples nearest xout. If the actual sample distance d does not exactly equal one of the values for which interpolators are tabulated, then the interpolator corresponding to the nearest value of d is used.

Because extrapolation of the input function y(x) is defined by the left and right values yinl and yinr, the xout values are not restricted to lie within the range of sample locations defined by nxin, dxin, and fxin.

AUTHOR: Dave Hale, Colorado School of Mines, 06/02/89

```
LINEAR_REGRESSION - Compute linear regression of (y1,y2,...,yn) against
(x1,x2,...,xn)
Function Prototype:
void linear_regression(float *y, float *x, int n, float coeff[4]);
Input:
y array of y values
x array of x values
n length of y and x arrays
Output:
coeff[4] where:

coeff[0] slope of best fit line
coeff[1] intercept of best fit line
coeff[2] correlation coefficient of fit (1 = perfect) [dimensionless]
coeff[3] standard error of fit (0 = perfect) [dimensions of y]
Notes:

y(x)
    |      *  .    fit is  y(x) = a x + b
    |       .
    |     .  *
    | * .
    | . *
     ------------------- x

         n Sum[x*y] - Sum[x]*Sum[y]
     a = --------------------------
         n Sum[x*x] - Sum[x]*Sum[x]

         Sum[y] - a*Sum[x]
     b = -----------------
                 n

     cc = std definition

     se = std definition

Author:  Chris Liner, UTulsa, 11/16/03
```

```
maxmin - subroutines that pertain to maximum and minimum values
min_index - find the value of the index where an array is a minimum

function prototypes:
int max_index(int n, float *a,int inc);
int min_index(int n, float *a,int inc);

Author: Balasz Nemeth: Potash Corporation, given to CWP in 2008
```

```
MKDIFF - Make an n-th order DIFFerentiator via Taylor's series method.

mkdiff make discrete Taylor series approximation to n'th derivative.

Function Prototype:
void mkdiff (int n, float a, float h, int l, int m, float d[]);

Input:
n order of desired derivative (n>=0 && n<=m-l is required)
a fractional distance from integer sampling index (see notes)
h sampling interval
l sampling index of first coefficient (see notes below)
m sampling index of last coefficient (see notes below)

Output:
d array[m-l+1] of coefficients for n'th order differentiator

Notes:
The abscissae x of a sampled function f(x) can always be expressed as
x = (j+a)*h, where j is an integer, a is a fraction, and h is the
sampling interval.  To approximate the n'th order derivative fn(x)
of the sampled function f(x) at x = (j+a)*h, use the m-l+1 coefficients
in the output array d[] as follows:

fn(x) = d[0]*f(j-l) + d[1]*f(j-l-1) +...+ d[m-l]*f(j-m)

i.e., convolve the coefficients in d with the samples in f.

m-l+1 (the number of coefficients) must not be greater than the
NCMAX parameter specified below.

For best approximations,
when n is even, use a = 0.0, l = -m
when n is odd, use a = 0.5, l = -m-1

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
MKHDIFF - Compute filter approximating the bandlimited HalF-DIFFerentiator.

mkhdiff - Compute filter approximating the bandlimited half-differentiator.

Function Prototype:
void mkhdiff (float h, int l, float d[]);

Input:
h sampling interval
l half-length of half-differentiator (length = 1+2*l is odd)

Output:
d array[1+2*l] of coefficients for half-differentiator

Notes:
The half-differentiator is defined by

  pi
    d[l+j] = sqrt(1/h)/(2pi) * integral dw sqrt(-iw)*exp(-iwj)
 -pi

  pi
           = sqrt(2/h)/(2pi) * integral dw sqrt(w)*(cos(wj)-sin(wj))
  0

    for j = -l, -l+1, ... , l.

An alternative definition is that f'(j) = d(j)*d(j)*f(j), where
f'(j) denotes the derivative of a sampled function f(j) and *
denotes a convolution sum.

The half-derivative g(j) of f(j) may be computed by the following sum:

g(j) = d[0]*f(j+l) + d[1]*f(j+l-1) + ... + d[2*l]*f(j-l)

The integral over frequency is evaluated numerically using Simpson's
method.  Although the Filon method of numerical integration is more
appropriate for this integral, the truncation of d[l+j] for |j| > l
is probably the greatest source of error.  In any case, d[l+j] is
cosine-tapered to reduce these truncation errors.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

MKSINC - Compute least-squares optimal sinc interpolation coefficients.

mksinc Compute least-squares optimal sinc interpolation coefficients.

Function Prototype:
void mksinc (float d, int lsinc, float sinc[]);

Input:
d fractional distance to interpolation point; 0.0<=d<=1.0
lsinc length of sinc approximation; lsinc%2==0 and lsinc<=20

Output:
sinc array[lsinc] containing interpolation coefficients

Notes:
The coefficients are a least-squares-best approximation to the ideal
sinc function for frequencies from zero up to a computed maximum
frequency. For a given interpolator length, lsinc, mksinc computes
the maximum frequency, fmax (expressed as a fraction of the nyquist
frequency), using the following empirically derived relation (from
a Western Geophysical Technical Memorandum by Ken Larner):

fmax = min(0.066+0.265*log(lsinc),1.0)

Note that fmax increases as lsinc increases, up to a maximum of 1.0.
Use the coefficients to interpolate a uniformly-sampled function y(i)
as follows:

```
            lsinc-1
   y(i+d) =  sum  sinc[j]*y(i+j+1-lsinc/2)
             j=0
```

Interpolation error is greatest for d=0.5, but for frequencies less
than fmax, the error should be less than 1.0 percent.

Author: Dave Hale, Colorado School of Mines, 06/02/89

```
MNEWT - Solve non-linear system of equations f(x) = 0 via Newton's method

mnewt Solve non-linear system of equations f(x) = 0 via Newton's method

Function Prototype:
int mnewt (int maxiter, float ftol, float dxtol, int n, float *x, void *aux,
void (*fdfdx)(int n, float *x, float *f, float **dfdx, void *aux));

Input:
maxiter maximum number of iterations
ftol converged when sum of absolute values of f less than ftol
dxtol converged when sum of absolute values of dx less than dxtol
n number of equations
x array[n] containing initial guess of solution
aux pointer to auxiliary parameters to be passed to fdfdx
fdfdx pointer to function to evaluate f(x) and f'(x)

Output:
x array[n] containing solution

Returned: number of iterations; -1 if failed to converge in maxiter

Input to the user-supplied function fdfdx:
n number of equations
x array[n] of x0, x1, ...
aux pointer to auxiliary variables required by fdfdx.

Output from the user-supplied function fdfdx:
f array[n] of f0(x), f1(x), ...
dfdx array[n][n] of f'(x);  dfdx[j][i] = dfi/dxj

Author:  Dave Hale, Colorado School of Mines, 06/06/91
```

```
ORTHPOLYNOMIALS - compute ORTHogonal POLYNOMIALS

hermite_n_polynomial:  Compute n-th order generalized Hermite polynomial.

hermite_n_polynomial:

Input:
h0 array of size nt holding H_{n-1}
h1 array of size nt holding H_{n}
t array of size nt holding time vector
nt size of arrays, no. of samples
n order of polynomial
sigma variance
Output:
h array of size nt holding H_{n+1}
Notes: Note that n in the function call is the order of the derivative and
        j in the code below is the n in the recurrence relation
Copyright (c) 2007 by the Society of Exploration Geophysicists.
For more information, go to http://software.seg.org/2007/0004 .
You must read and accept usage terms at:
http://software.seg.org/disclaimer.txt before use.
Author: Werner M. Heigl, Apache Corporation, E&P Technology, December 2006



include "cwp.h"



void
hermite_n_polynomial(double *h, double *h0, double *h1, double *t, int nt, int n, do
  hermite_n_polynomial:  Compute n-th order generalized Hermite polynomial.
Input:
h0 array of size nt holding H_{n-1}
h1 array of size nt holding H_{n}
t array of size nt holding time vector
nt size of arrays, no. of samples
n order of polynomial
sigma variance
Output:
h array of size nt holding H_{n+1}
Note: Note that n in the function call is the order of the derivative and
        j in the code below is the n in the recurrence relation
```

```
Copyright (c) 2007 by the Society of Exploration Geophysicists.
For more information, go to http://software.seg.org/2007/0004 .
You must read and accept usage terms at:
http://software.seg.org/disclaimer.txt before use.
Author: Werner M. Heigl, Apache Corporation, E&P Technology, December 2006
{
int i; /* loop variable
int j=1; /* recurrence counter

/* as long as necessary use recurrence relation
do {
/* current instance of recurrence relation
for (i = 0; i < nt; ++i)
h[i] = ( t[i] * h1[i] - j * h0[i] ) / sigma;

/* update inputs to recurrence relation
memcpy((void *) h0, (const void *) h1, DSIZE * nt);
memcpy((void *) h1, (const void *) h , DSIZE * nt);

/* update counter
++j;

} while (j < n);

}
```

```
PFAFFT - Functions to perform Prime Factor (PFA) FFT's, in place

npfa return valid n for complex-to-complex PFA
npfar return valid n for real-to-complex/complex-to-real PFA
npfao return optimal n for complex-to-complex PFA
npfaro return optimal n for real-to-complex/complex-to-real PFA
pfacc 1D PFA complex to complex
pfacr 1D PFA complex to real
pfarc 1D PFA real to complex
pfamcc multiple PFA complex to real
pfa2cc 2D PFA complex to complex
pfa2cr 2D PFA complex to real
pfa2rc 2D PFA real to complex

Function Prototypes:
int npfa (int nmin);
int npfao (int nmin, int nmax);
int npfar (int nmin);
int npfaro (int nmin, int nmax);
void pfacc (int isign, int n, complex z[]);
void pfacr (int isign, int n, complex cz[], float rz[]);
void pfarc (int isign, int n, float rz[], complex cz[]);
void pfamcc (int isign, int n, int nt, int k, int kt, complex z[]);
void pfa2cc (int isign, int idim, int n1, int n2, complex z[]);
void pfa2cr (int isign, int idim, int n1, int n2, complex cz[], float rz[]);
void pfa2rc (int isign, int idim, int n1, int n2, float rz[], complex cz[]);

npfa:
Input:
nmin lower bound on returned value (see notes below)

Returned: valid n for prime factor fft

npfao
Input:
nmin lower bound on returned value (see notes below)
nmax desired (but not guaranteed) upper bound on returned value

Returned: valid n for prime factor fft

npfar
Input:
nmin lower bound on returned value
```

```
Returned: valid n for real-to-complex/complex-to-real prime factor fft

npfaro:
Input:
nmin lower bound on returned value
nmax desired (but not guaranteed) upper bound on returned value

Returned: valid n for real-to-complex/complex-to-real prime factor fft

pfacc:
Input:
isign sign of isign is the sign of exponent in fourier kernel
n length of transform (see notes below)
z array[n] of complex numbers to be transformed in place

Output:
z array[n] of complex numbers transformed

pfacr:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
n           length of transform (see notes below)
cz          array[n/2+1] of complex values (may be equivalenced to rz)

Output:
rz          array[n] of real values (may be equivalenced to cz)

pfarc:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
n           length of transform; must be even (see notes below)
rz          array[n] of real values (may be equivalenced to cz)

Output:
cz          array[n/2+1] of complex values (may be equivalenced to rz)

pfamcc:
Input:
isign        sign of isign is the sign of exponent in fourier kernel
n            number of complex elements per transform (see notes below)
nt           number of transforms
k            stride in complex elements within transforms
```

```
kt           stride in complex elements between transforms
z            array of complex elements to be transformed in place

Output:
z array of complex elements transformed

pfa2cc:
Input:
isign        sign of isign is the sign of exponent in fourier kernel
idim         dimension to transform, either 1 or 2 (see notes)
n1           1st (fast) dimension of array to be transformed (see notes)
n2           2nd (slow) dimension of array to be transformed (see notes)
z            array[n2][n1] of complex elements to be transformed in place

Output:
z array[n2][n1] of complex elements transformed

pfa2cr:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
idim        dimension to transform, which must be either 1 or 2 (see notes)
n1          1st (fast) dimension of array to be transformed (see notes)
n2          2nd (slow) dimension of array to be transformed (see notes)
cz          array of complex values (may be equivalenced to rz)

Output:
rz          array of real values (may be equivalenced to cz)

pfa2rc:
Input:
isign       sign of isign is the sign of exponent in fourier kernel
idim        dimension to transform, which must be either 1 or 2 (see notes)
n1          1st (fast) dimension of array to be transformed (see notes)
n2          2nd (slow) dimension of array to be transformed (see notes)
rz          array of real values (may be equivalenced to cz)

Output:
cz          array of complex values (may be equivalenced to rz)

Notes:
Table of valid n and cost for prime factor fft.  For each n, cost
was estimated to be the inverse of the number of ffts done in 1 sec
on an IBM RISC System/6000 Model 320H, by Dave Hale, 08/04/91.
```

```
(Redone by Jack Cohen for 15 sec to rebuild NTAB table on advice of
David and Gregory Chudnovsky, 05/03/94).
Cost estimates are least accurate for very small n.  An alternative method
for estimating cost would be to count multiplies and adds, but this
method fails to account for the overlapping of multiplies and adds
that is possible on some computers, such as the IBM RS/6000 family.

npfa:
The returned n will be composed of mutually prime factors from
the set {2,3,4,5,7,8,9,11,13,16}.  Because n cannot exceed
720720 = 5*7*9*11*13*16, 720720 is returned if nmin exceeds 720720.

npfao:
The returned n will be composed of mutually prime factors from
the set {2,3,4,5,7,8,9,11,13,16}.  Because n cannot exceed
720720 = 5*7*9*11*13*16, 720720 is returned if nmin exceeds 720720.
If nmin does not exceed 720720, then the returned n will not be
less than nmin.  The optimal n is chosen to minimize the estimated
cost of performing the fft, while satisfying the constraint, if
possible, that n not exceed nmax.

npfar and npfaro:
Current implemenations of real-to-complex and complex-to-real prime
factor ffts require that the transform length n be even and that n/2
be a valid length for a complex-to-complex prime factor fft.  The
value returned by npfar satisfies these conditions.  Also, see notes
for npfa.

pfacc:
n must be factorable into mutually prime factors taken
from the set {2,3,4,5,7,8,9,11,13,16}.  in other words,
n = 2**p * 3**q * 5**r * 7**s * 11**t * 13**u
where
0 <= p <= 4,  0 <= q <= 2,  0 <= r,s,t,u <= 1
is required for pfa to yield meaningful results.  this
restriction implies that n is restricted to the range
1 <= n <= 720720 (= 5*7*9*11*13*16)

pfacr:
Because pfacr uses pfacc to do most of the work, n must be even
and n/2 must be a valid length for pfacc.  The simplest way to
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.
```

```
pfarc:
Because pfarc uses pfacc to do most of the work, n must be even
and n/2 must be a valid length for pfacc.  The simplest way to
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.

pfamcc:
To perform a two-dimensional transform of an n1 by n2 complex array
(assuming that both n1 and n2 are valid "n"), stored with n1 fast
and n2 slow:
    pfamcc(isign,n1,n2,1,n1,z); (to transform 1st dimension)
    pfamcc(isign,n2,n1,n1,1,z); (to transform 2nd dimension)

pfa2cc:
Only one (either the 1st or 2nd) dimension of the 2-D array is transformed.

If idim equals 1, then n2 transforms of n1 complex elements are performed;
else, if idim equals 2, then n1 transforms of n2 complex elements are
performed.

Although z appears in the argument list as a one-dimensional array,
z may be viewed as an n1 by n2 two-dimensional array:  z[n2][n1].

Valid n is computed via the "np" subroutines.

To perform a two-dimensional transform of an n1 by n2 complex array
(assuming that both n1 and n2 are valid "n"), stored with n1 fast
and n2 slow:  pfa2cc(isign,1,n1,n2,z);  pfa2cc(isign,2,n1,n2,z);

pfa2cr:
If idim equals 1, then n2 transforms of n1/2+1 complex elements to n1 real
elements are performed; else, if idim equals 2, then n1 transforms of n2/2+1
complex elements to n2 real elements are performed.

Although rz appears in the argument list as a one-dimensional array,
rz may be viewed as an n1 by n2 two-dimensional array:  rz[n2][n1].
Likewise, depending on idim, cz may be viewed as either an n1/2+1 by
n2 or an n1 by n2/2+1 two-dimensional array of complex elements.

Let n denote the transform length, either n1 or n2, depending on idim.
Because pfa2rc uses pfa2cc to do most of the work, n must be even
and n/2 must be a valid length for pfa2cc.  The simplest way to
```

```
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.

pfa2rc:
If idim equals 1, then n2 transforms of n1 real elements to n1/2+1 complex
elements are performed; else, if idim equals 2, then n1 transforms of n2
real elements to n2/2+1 complex elements are performed.

Although rz appears in the argument list as a one-dimensional array,
rz may be viewed as an n1 by n2 two-dimensional array:  rz[n2][n1].
Likewise, depending on idim, cz may be viewed as either an n1/2+1 by
n2 or an n1 by n2/2+1 two-dimensional array of complex elements.

Let n denote the transform length, either n1 or n2, depending on idim.
Because pfa2rc uses pfa2cc to do most of the work, n must be even
and n/2 must be a valid length for pfa2cc.  The simplest way to
obtain a valid n is via n = npfar(nmin).  A more optimal n can be
obtained with npfaro.

References:
Temperton, C., 1985, Implementation of a self-sorting
in-place prime factor fft algorithm:  Journal of
Computational Physics, v. 58, p. 283-299.

Temperton, C., 1988, A new set of minimum-add rotated
rotated dft modules: Journal of Computational Physics,
v. 75, p. 190-198.

Differ significantly from Press et al, 1988, Numerical Recipes in C, p. 417.

Author:  Dave Hale, Colorado School of Mines, 04/27/89
```

```
POLAR - Functions to map data in rectangular coordinates to polar and vise versa

recttopolar convert a function p(x,y) to a function q(a,r)
polartorect convert a function q(a,r) to a function p(x,y)

Function Prototypes:
void recttopolar ( int nx, float dx, float fx, int ny, float dy, float fy,
float **p, int na, float da, float fa, int nr,
float dr, float fr, float **q);
void polartorect ( int na, float da, float fa, int nr, float dr, float fr,
float **q, int nx, float dx, float fx, int ny,
float dy, float fy, float **p)

recttopolar:
Input:
nx number of x samples
dx x sampling interval
fx first x sample
ny number of y samples
dy y sampling interval
fy first y sample
p array[ny][nx] containing samples of p(x,y)
na number of a samples
da a sampling interval
fa first a sample
nr number of r samples
dr r sampling interval
fr first r sample

Output:
q array[nr][na] containing samples of q(a,r)

polartorect:
Input:
na number of a samples
da a sampling interval
fa first a sample
nr number of r samples
dr r sampling interval
fr first r sample
nx number of x samples
dx x sampling interval
fx first x sample
```

```
ny number of y samples
dy y sampling interval
fy first y sample
q array[nr][na] containing samples of q(a,r)

Output:
p array[ny][nx] containing samples of p(x,y)

Notes:
The polar angle a is measured in radians,
x = r*cos(a) and y = r*sin(a).

recttopolar:
Linear extrapolation is used to determine the value of p(x,y) for
x and y coordinates not in the range corresponding to nx, dx, ....

polartorect:
Linear extrapolation is used to determine the value of q(a,r) for
a and r coordinates not in the range corresponding to na, da, ....

Author:  Dave Hale, Colorado School of Mines, 06/15/90
```

```
PRINTERPLOT - Functions to make a printer plot of a 1-dimensional array

pp1d printer plot of 1d array
pplot1 printer plot of 1d array

Function Prototypes:
void pp1d (FILE *fp, char *title, int lx, int ifx, float x[]);
void pplot1 (FILE *fp, char *title, int nx, float ax[]);

pp1d:
Input:
fp file pointer for output (e.g., stdout, stderr, etc.)
title title of plot
lx length of x
ifx index of first x
x array[lx] to be plotted

pplot1:
Input:
fp file pointer for printed output (e.g., stdout, stderr, etc.)
title title of plot
nx number of x values to be plotted
ax array[nx] of x values

Notes:
Are two subroutines to do this really necessary?

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
QUEST - Functions to ESTimate Quantiles:

quest returns estimate - use when entire array is in memory
questalloc returns quantile estimator - use before questupdate
questupdate updates and returns current estimate - use for large
numbers of floats, too big to fit in memory at one time
questfree frees quantile estimator

quest:
Input:
p quantile to be estimated (0.0<=p<=1.0 is required)
n number of samples in array x (n>=5 is required)
x array[n] of floats

Returned: the estimate of a specified quantile

questalloc:
Input:
p quantile to be estimated (0.0<=p<=1.0 is required)
n number of samples in array x (n>=5 is required)
x array[n] of floats

Returned: pointer to a quantile estimator

questupdate:
Input:
q pointer to quantile estimator (as returned by questalloc)
n number of samples in array x
x array[n] of floats

Returned: quantile estimate

questfree:
q pointer to quantile estimator (as returned by questalloc)

Notes:
quest:
The estimate should approach the sample quantile in the limit of large n.

The estimate is most accurate for cumulative distribution functions
that are smooth in the neighborhood of the quantile specified by p.

This function is an implementation of the algorithm published by
```

Jain and Chlamtac (1985).

questalloc:
This function must be called before calling function questupdate.
See also notes in questupdate.

questupdate:
The estimate should approach the sample quantile in the limit of large n.

The estimate is most accurate for cumulative distribution functions
that are smooth in the neighborhood of the quantile specified by p.

This function is an implementation of the algorithm published by
Jain, R. and Chlamtac, I., 1985, The PP algorithm for dynamic
calculation of quantiles and histograms without storing observations:
Comm. ACM, v. 28, n. 10.

Reference:
Jain, R. and Chlamtac, I., 1985, The PP algorithm for dynamic
calculation of quantiles and histograms without storing observations:
Comm. ACM, v. 28, n. 10.

Author: Dave Hale, Colorado School of Mines, 05/07/89

```
RESSINC8 - Functions to resample uniformly-sampled data  via 8-coefficient sinc
 approximations:

ress8c resample a uniformly-sampled complex function via 8-coeff. sinc approx.
ress8r resample a uniformly-sampled real function via 8-coeff. sinc approx.

Function Prototypes:
void ress8r (int nxin, float dxin, float fxin, float yin[],
float yinl, float yinr,
int nxout, float dxout, float fxout, float yout[]);
void ress8c (int nxin, float dxin, float fxin, complex yin[],
complex yinl, complex yinr,
int nxout, float dxout, float fxout, complex yout[]);

Input:
nxin number of x values at which y(x) is input
dxin x sampling interval for input y(x)
fxin x value of first sample input
yin array[nxin] of input y(x) values:  yin[0] = y(fxin), etc.
yinl value used to extrapolate yin values to left of yin[0]
yinr value used to extrapolate yin values to right of yin[nxin-1]
nxout number of x values at which y(x) is output
dxout x sampling interval for output y(x)
fxout x value of first sample output

Output:
yout array[nxout] of output y(x) values:  yout[0] = y(fxout), etc.

Notes:
Because extrapolation of the input function y(x) is defined by the
left and right values yinl and yinr, the output x values defined
by nxout, dxout, and fxout are not restricted to lie within the range
of input x values defined by nxin, dxin, and fxin.

The maximum error for frequiencies less than 0.6 nyquist is
less than one percent.

Author:  Dave Hale, Colorado School of Mines, 06/06/90
```

```
RFWTVA - Rasterize a Float array as Wiggle-Trace-Variable-Area.

rfwtva rasterize a float array as wiggle-trace-variable-area.

Function Prototype:
void rfwtva (int n, float z[], float zmin, float zmax, float zbase,
int yzmin, int yzmax, int xfirst, int xlast,
int wiggle, int nbpr, unsigned char *bits, int endian);

Input:
n number of samples in array to rasterize
z array[n] to rasterize
zmin z values below zmin will be clipped
zmax z values above zmax will be clipped
zbase z values between zbase and zmax will be filled (see notes)
yzmin horizontal raster coordinate corresponding to zmin
yzmax horizontal raster coordinate corresponding to zmax
xfirst vertical raster coordinate of z[0] (see notes)
xlast vertical raster coordinate of z[n-1] (see notes)
wiggle =0 for no wiggle (VA only); =1 for wiggle (with VA)
wiggle 2<=wiggle<=5 for solid/grey coloring of VA option
                shade of grey: wiggle=2 light grey, wiggle=5 black
nbpr number of bytes per row of bits
bits pointer to first (top,left) byte in image
endian byte order  =1 big endian  =0 little endian

Output:
bits pointer to first (top,left) byte in image

Notes:
The raster coordinate of the (top,left) bit in the image is (0,0).
In other words, x increases downward and y increases to the right.
Raster scan lines run from left to right, and from top to bottom.
Therefore, xfirst, xlast, yzmin, and yzmax should not be less than 0.
Likewise, yzmin and yzmax should not be greater than nbpr*8-1, and
care should be taken to ensure that xfirst and xlast do not cause bits
to be set outside (off the bottom) of the image.

Variable area fill is performed on the right-hand (increasing y) side
of the wiggle.  If yzmin is greater than yzmax, then z values between
zmin will be plotted to the right of zmax, and z values between zbase
and zmin are filled.  Swapping yzmin and yzmax is an easy way to
reverse the polarity of a wiggle.
```

```
The variable "endian" must have a value of 1 or 0. If this is
not a case an error is returned.

Author:  Dave Hale, Colorado School of Mines, 07/01/89
MODIFIED:  Paul Michaels, Boise State University, 29 December 2000
           Added solid/grey shading scheme, wiggle>=2 option for peaks/troughs
```

```
RFWTVAINT - Rasterize a Float array as Wiggle-Trace-Variable-Area, with
    8 point sinc INTerpolation.

rfwtvaint rasterize a float array as wiggle-trace-variable-area, and
apply sinc interploation for display purposes.

Function Prototype:
void rfwtvaint (int n, float z[], float zmin, float zmax, float zbase,
int yzmin, int yzmax, int xfirst, int xlast,
int wiggle, int nbpr, unsigned char *bits, int endian);

Input:
n number of samples in array to rasterize
z array[n] to rasterize
zmin z values below zmin will be clipped
zmax z values above zmax will be clipped
zbase z values between zbase and zmax will be filled (see notes)
yzmin horizontal raster coordinate corresponding to zmin
yzmax horizontal raster coordinate corresponding to zmax
xfirst vertical raster coordinate of z[0] (see notes)
xlast vertical raster coordinate of z[n-1] (see notes)
wiggle =0 for no wiggle (VA only); =1 for wiggle (with VA)
                wiggle 2<=wiggle<=5 for solid/grey coloring of VA option
                shade of grey: wiggle=2 light grey, wiggle=5 black
nbpr number of bytes per row of bits
bits pointer to first (top,left) byte in image
endian byte order  =1 big endian  =0 little endian

Output:
bits pointer to first (top,left) byte in image

Notes:
The raster coordinate of the (top,left) bit in the image is (0,0).
In other words, x increases downward and y increases to the right.
Raster scan lines run from left to right, and from top to bottom.
Therefore, xfirst, xlast, yzmin, and yzmax should not be less than 0.
Likewise, yzmin and yzmax should not be greater than nbpr*8-1, and
care should be taken to ensure that xfirst and xlast do not cause bits
to be set outside (off the bottom) of the image.

Variable area fill is performed on the right-hand (increasing y) side
of the wiggle.  If yzmin is greater than yzmax, then z values between
zmin will be plotted to the right of zmax, and z values between zbase
```

```
and zmin are filled.  Swapping yzmin and yzmax is an easy way to
reverse the polarity of a wiggle.

The variable "endian" must have a value of 1 or 0. If this is
not a case an error is returned.

The interpolation is by the 8 point sinc interpolation routine s8r.

Author:  Dave Hale, Colorado School of Mines, 07/01/89
Memorial University of Newfoundland: Tony Kocurko, Sept 1995.
 Added sinc interpolation.
MODIFIED: Paul Michaels, Boise State University, 29 December 2000
          added solid/grey color scheme for peaks/troughs  wiggle=2 option
```

```
SBLAS - Single precision Basic Linear Algebra Subroutines
(adapted from LINPACK FORTRAN):

isamax return index of element with maximum absolute value
sasum return sum of absolute values
saxpy compute y[i] = a*x[i]+y[i]
scopy copy x[i] to y[i] (i.e., set y[i] = x[i])
sdot return sum of x[i]*y[i] (i.e., return the dot product of x and y)
snrm2 return square root of sum of squares of x[i]
sscal compute x[i] = a*x[i]
sswap swap x[i] and y[i]

Function Prototypes:
int isamax (int n, float *sx, int incx);
float sasum (int n, float *sx, int incx);
void saxpy (int n, float sa, float *sx, int incx, float *sy, int incy);
void scopy (int n, float *sx, int incx, float *sy, int incy);
float sdot (int n, float *sx, int incx, float *sy, int incy);
float snrm2 (int n, float *sx, int incx);
void sscal (int n, float sa, float *sx, int incx);
void sswap (int n, float *sx, int incx, float *sy, int incy);

isamax:
Input:
n number of elements in array
sx array[n] of elements
incx increment between elements

Returned: index of element with maximum absolute value

sasum:
Input:
n number of elements in array
sx array[n] of elements
incx increment between elements

Returned: sum of absolute values

saxpy:
Input:
n number of elements in arrays
sa the scalar multiplier
sx array[n] of elements to be scaled and added
```

```
incx increment between elements of sx
sy array[n] of elements to be added
incy increment between elements of sy

Output:
sy array[n] of accumulated elements

scopy:
Input:
n number of elements in arrays
sx array[n] of elements to be copied
incx increment between elements of sx
incy increment between elements of sy

Output:
sy array[n] of copied elements

sdot:
Input:
n number of elements in arrays
sx array[n] of elements
incx increment between elements of sx
sy array[n] of elements
incy increment between elements of sy

Returned: dot product of the two arrays

snrm2
Input:
n number of elements in array
sx array[n] of elements
incx increment between elements

Returned: square root of sum of squares of x[i]

sscal:
Input:
n number of elements in array
sa the scalar multiplier
sx array[n] of elements
incx increment between elements

Output:
```

```
sx array[n] of scaled elements

sswap:
Input:
n number of elements in arrays
sx array[n] of elements
incx increment between elements of sx
sy array[n] of elements
incy increment between elements of sy

Output:
sx array[n] of swapped elements
sy array[n] of swapped elements

Notes:
Adapted from Linpack Fortran.

snrm2:
This simple version may cause overflow or underflow!

Author:  Dave Hale, Colorado School of Mines, 10/01/89
```

```
SCAXIS - compute a readable scale for use in plotting axes

scaxis compute a readable scale for use in plotting axes

Function Prototype:
void scaxis (float x1, float x2, int *nxnum, float *dxnum, float *fxnum);

Input:
x1 first x value
x2 second x value
nxnum desired number of numbered values

Output:
nxnum number of numbered values
dxnum increment between numbered values (dxnum>0.0)
fxnum first numbered value

Notes:
scaxis attempts to honor the user-specified nxnum.  However, nxnum
will be modified if necessary for readability.  Also, fxnum and nxnum
will be adjusted to compensate for roundoff error; in particular,
fxnum will not be less than xmin-eps, and fxnum+(nxnum-1)*dxnum
will not be greater than xmax+eps, where eps = 0.0001*(xmax-xmin).
xmin is the minimum of x1 and x2.  xmax is the maximum of x1 and x2.

Author:  Dave Hale, Colorado School of Mines, 01/13/89
```

```
SGA - Single precision general matrix functions adapted from LINPACK FORTRAN:

sgefa Gaussian elimination to obtain the LU factorization of a matrix.
sgeco Gaussian elimination to obtain the LU factorization and
condition number of a matrix.
sgesl Solve linear system Ax = b or A'x = b after LU factorization.

Function Prototypes:
void sgefa (float **a, int n, int *ipvt, int *info);
void sgeco (float **a, int n, int *ipvt, float *rcond, float *z);
void sgesl (float **a, int n, int *ipvt, float *b, int job);

sgefa:
Input:
a matrix[n][n] to be factored (see notes below)
n dimension of a

Output:
a matrix[n][n] factored (see notes below)
ipvt indices of pivot permutations (see notes below)
info index of last zero pivot (or -1 if no zero pivots)

sgeco:
Input:
a matrix[n][n] to be factored (see notes below)
n dimension of a

Output:
a matrix[n][n] factored (see notes below)
ipvt indices of pivot permutations (see notes below)
rcond reciprocal condition number (see notes below)

Workspace:
z array[n]

sgesl
Input:
a matrix[n][n] that has been LU factored (see notes below)
n dimension of a
ipvt indices of pivot permutations (see notes below)
b right-hand-side vector[n]
job =0 to solve Ax = b
=1 to solve A'x = b
```

```
Output:
b solution vector[n]

Notes:
These functions were adapted from LINPACK FORTRAN.  Because two-dimensional
arrays cannot be declared with variable dimensions in C, the matrix a
is actually a pointer to an array of pointers to floats, as declared
above and used below.

Elements of a are stored as follows:
a[0][0]    a[1][0]    a[2][0]   ... a[n-1][0]
a[0][1]    a[1][1]    a[2][1]   ... a[n-1][1]
a[0][2]    a[1][2]    a[2][2]   ... a[n-1][2]
.                                     .
.             .                       .
.                        .            .
.                                     .
a[0][n-1]  a[1][n-1]  a[2][n-1] ... a[n-1][n-1]

Both the factored matrix a and the pivot indices ipvt are required
to solve linear systems of equations via sgesl.

sgeco:
Given the reciprocal of the condition number, rcond, and the float
epsilon, FLT_EPSILON, the number of significant decimal digits, nsdd,
in the solution of a linear system of equations may be estimated by:
nsdd = (int)log10(rcond/FLT_EPSILON)

This function was adapted from LINPACK FORTRAN.  Because two-dimensional
arrays cannot be declared with variable dimensions in C, the matrix a
is actually a pointer to an array of pointers to floats, as declared
above and used below.

Author:  Dave Hale, Colorado School of Mines, 10/01/89
```

```
SHFS8R - Shift a uniformly-sampled real-valued function y(x) via
a table of 8-coefficient sinc approximations.

shfs8r shift a uniformly-sampled real function via a table of 8-coeff. sinc
approximations.

Function Prototypes:
void shfs8r (float dx, int nxin, float fxin, float yin[],
float yinl, float yinr, int nxout, float fxout, float yout[]);

Input:
dx x sampling interval for both input and output y(x)
nxin number of x values at which y(x) is input
fxin x value of first sample input
yin array[nxin] of input y(x) values:  yin[0] = y(fxin), etc.
yinl value used to extrapolate yin values to left of yin[0]
yinr value used to extrapolate yin values to right of yin[nxin-1]
nxout number of x values a which y(x) is output
fxout x value of first sample output

Output:
yout array[nxout] of output y(x) values:  yout[0] = y(fxout), etc.

Notes:
Because extrapolation of the input function y(x) is defined by the
left and right values yinl and yinr, the output samples defined by
dx, nxout, and fxout are not restricted to lie within the range of
input sample locations defined by dx, nxin, and fxin.

The maximum error for frequencies less than 0.6*nyquist is less
than one percent.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
SINC - Return SINC(x) for as floats or as doubles

fsinc return float value of sinc(x) for x input as a float
dsinc return double precision sinc(x) for double precision x

Function Prototype:
float fsinc (float x);
double dsinc (double x);

Input:
x value at which to evaluate sinc(x)

Returned:  sinc(x)

Notes:
    sinc(x) = sin(PI*x)/(PI*x)

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
SORT - Functions to sort arrays of data or arrays of indices

hpsort sort an array of floats by the heap sort method
qkisort sort an array of indices i[] so that
a[i[0]] <= a[i[1]] <= ... <= a[i[n-1]]
uses the "quick sort" method
qkifind partially sort an array of indices i[] so that the
index i[m] has the value it would have if the entire
array of indices were sorted such that
a[i[0]] <= a[i[1]] <= ... <= a[i[n-1]]
uses the "quick sort" method
qksort sort an array of floats such that a[0] <= a[1] <= ... <= a[n-1]
uses the "quick sort" method
qkfind partially sort an array of floats  so that the element a[m] has
the value it would have if the entire array were sorted
such that a[0] <= a[1] <= ... <= a[n-1]
uses the "quick sort" method

Function Prototypes:
void hpsort (int n, float a[]);
void qkisort (int n, float a[], int i[]);
void qkifind (int m, int n, float a[], int i[]);
void qksort (int n, float a[]);
void qkfind (int m, int n, float a[]);

hpsort:
Input:
n number of elements in a
a array[n] to be sorted

Output:
a array[n] sorted

qkisort:
Input:
n number of elements in array a
a array[n] elements
i array[n] indices to be sorted

Output:
i array[n] indices sorted

qkifind:
```

```
Input:
m index of element to be found
n number of elements in array a
a array[n] elements
i array[n] indices to be partially sorted

Output:
i array[n] indices partially sorted sorted

qksort:
Input:
n number of elements in array a
a array[n] containing elements to be sorted

Output:
a array[n] containing sorted elements

qkfind:
Input:
m index of element to be found
n number of elements in array a
a array[n] to be partially sorted

Output:
a array[n] partially sorted


Notes:
hpsort:
The heap sort algorithm is, at worst, N log_2 N, and in most cases
is 20% faster.  Adapted from Standish.

qkisort, qkifind, qksort, qkfind:
n must be less than 2^NSTACK, where NSTACK is defined above.

qkisort:
This function is adapted from procedure quicksort by
Hoare, C.A.R., 1961, Communications of the ACM, v. 4, p. 321;
the main difference is that recursion is accomplished
explicitly via a stack array for efficiency; also, a simple
insertion sort is used to sort subarrays too small to be
partitioned efficiently.
```

```
qkifind:
This function is adapted from procedure find by
Hoare, C.A.R., 1961, Communications of the ACM, v. 4, p. 321.

qksort:
This function is adapted from procedure quicksort by
Hoare, C.A.R., 1961, Communications of the ACM, v. 4, p. 321;
the main difference is that recursion is accomplished
explicitly via a stack array for efficiency; also, a simple
insertion sort is used to sort subarrays too small to be
partitioned efficiently.

qkfind:
This function is adapted from procedure find by Hoare 1961.

Reference:
hpsort:
Standish, T. A., Data Structure Techniques, p. 91.
See also Press, W. A., et al., Numerical Recipes in C.

quick sort:
Hoare, C.A.R., 1961, Communications of the ACM, v. 4, p. 321.

Author:  Dave Hale, Colorado School of Mines, 12/26/89
```

```
SQR - Single precision QR decomposition functions adapted from LINPACK FORTRAN:

sqrdc Compute QR decomposition of a matrix.
sqrsl Use QR decomposition to solve for coordinate transformations,
projections, and least squares solutions.
sqrst Solve under-determined or over-determined least squares problems,
with a user-specified tolerance.

Function Prototypes:
void sqrdc (float **x, int n, int p, float *qraux, int *jpvt,
float *work, int job);
void sqrsl (float **x, int n, int k, float *qraux,
float *y, float *qy, float *qty,
float *b, float *rsd, float *xb, int job, int *info);
void sqrst (float **x, int n, int p, float *y, float tol,
float *b, float *rsd, int *k,
int *jpvt, float *qraux, float *work);

sqrdc:
Input:
x matrix[p][n] to decompose (see notes below)
n number of rows in the matrix x
p number of columns in the matrix x
jpvt array[p] controlling the pivot columns (see notes below)
job =0 for no pivoting;
=1 for pivoting

Output:
x matrix[p][n] decomposed (see notes below)
qraux array[p] containing information required to recover the
orthogonal part of the decomposition
jpvt array[p] with jpvt[k] containing the index of the original
matrix that has been interchanged into the k-th column, if
pivoting is requested.

Workspace:
work array[p] of workspace

sqrsl:
Input:
x matrix[p][n] containing output of sqrdc.
n number of rows in the matrix xk; same as in sqrdc.
k number of columns in the matrix xk; k must not be greater
```

```
than MIN(n,p), where p is the same as in sqrdc.
qraux array[p] containing auxiliary output from sqrdc.
y array[n] to be manipulated by sqrsl.
job specifies what is to be computed.  job has the decimal
expansion ABCDE, with the following meaning:
if A != 0, compute qy.
if B, C, D, or E != 0, compute qty.
if C != 0, compute b.
if D != 0, compute rsd.
if E != 0, compute xb.
Note that a request to compute b, rsd, or xb automatically
triggers the computation of qty, for which an array must
be provided.

Output:
qy array[n] containing Qy, if its computation has been
requested.
qty array[n] containing Q'y, if its computation has
been requested.  Here Q' denotes the transpose of Q.
b array[k] containing solution of the least squares problem:
minimize norm2(y - xk*b),
if its computation has been requested.  (Note that if
pivoting was requested in sqrdc, the j-th component of
b will be associated with column jpvt[j] of the original
matrix x that was input into sqrdc.)
rsd array[n] containing the least squares residual y - xk*b,
if its computation has been requested.  rsd is also the
orthogonal projection of y onto the orthogonal complement
of the column space of xk.
xb array[n] containing the least squares approximation xk*b,
if its computation has been requested.  xb is also the
orthogonal projection of y onto the column space of x.
info =0 unless the computation of b has been requested and R
is exactly singular.  In this case, info is the index of
the first zero diagonal element of R and b is left
unaltered.

sqrst:
Input:
x matrix[p][n] of coefficients (x is destroyed by sqrst.)
n number of rows in the matrix x (number of observations)
p number of columns in the matrix x (number of parameters)
y array[n] containing right-hand-side (observations)
```

```
tol relative tolerance used to determine the subset of
columns of x included in the solution.  If tol is zero,
a full complement of the MIN(n,p) columns is used.
If tol is non-zero, the problem should be scaled so that
all the elements of X have roughly the same absolute
accuracy eps.  Then a reasonable value for tol is roughly
eps divided by the magnitude of the largest element.

Output:
x matrix[p][n] containing output of sqrdc
b array[p] containing the solution (parameters); components
corresponding to columns not used are set to zero.
rsd array[n] of residuals y - Xb
k number of columns of x used in the solution
jpvt array[p] containing pivot information from sqrdc.
qraux array[p] containing auxiliary information from sqrdc.

Workspace:
work array[p] of workspace.

Notes:
!!! WARNING !!!
These functions have many options, not all of which have been tested!
(Dave Hale, 12/31/89)


This function was adapted from LINPACK FORTRAN.  Because two-dimensional
arrays cannot be declared with variable dimensions in C, the matrix x
is actually a pointer to an array of pointers to floats, as declared
above and used below.

Elements of x are stored as follows:
x[0][0]    x[1][0]    x[2][0]   ... x[p-1][0]
x[0][1]    x[1][1]    x[2][1]   ... x[p-1][1]
x[0][2]    x[1][2]    x[2][2]   ... x[p-1][2]
.                                     .
.             .                       .
.                        .            .
.                                     .
x[0][n-1]  x[1][n-1]  x[2][n-1] ... x[p-1][n-1]

sqrdc:
Uses Householder transformations to compute the QR decomposition of an n by p
```

```
matrix x.  Column pivoting based on the 2-norms of the reduced columns may be
performed at the user's option.

After decomposition, x contains in its upper triangular matrix R of the QR
decomposition.  Below its diagonal x contains information from which the
orthogonal part of the decomposition can be recovered.  Note that if
pivoting has been requested, the decomposition is not that of the original
matrix x but that of x with its columns permuted as described by jpvt.

The selection of pivot columns is controlled by jpvt as follows.
The k-th column x[k] of x is placed in one of three classes according to
the value of jpvt[k].
if jpvt[k] >  0, then x[k] is an initial column.
if jpvt[k] == 0, then x[k] is a free column.
if jpvt[k] <  0, then x[k] is a final column.
Before the decomposition is computed, initial columns are moved to the
beginning of the array x and final columns to the end.  Both initial and
final columns are frozen in place during the computation and only free
columns are moved.  At the k-th stage of the reduction, if x[k] is occupied
by a free column it is interchanged with the free column of largest reduced
norm.  jpvt is not referenced if job == 0.

sqrsl:
Uses the output of sqrdc to compute coordinate transformations, projections,
and least squares solutions.  For k <= MIN(n,p), let xk be the matrix
xk = (x[jpvt[0]], x[jpvt[1]], ..., x[jpvt[k-1]])
formed from columns jpvt[0], jpvt[1], ..., jpvt[k-1] of the original
n by p matrix x that was input to sqrdc.  (If no pivoting was done, xk
consists of the first k columns of x in their original order.)  sqrdc
produces a factored orthogonal matrix Q and an upper triangular matrix R
such that
xk = Q * (R)
         (0)
This information is contained in coded form in the arrays x and qraux.

The parameters qy, qty, b, rsd, and xb are not referenced if their
computation is not requested and in this case can be replaced by NULL
pointers in the calling program.  To save storage, the user may in
some cases use the same array for different parameters in the calling
sequence.  A frequently occuring example is when one wishes to compute
any of b, rsd, or xb and does not need y or qty.  In this case one may
equivalence y, qty, and one of b, rsd, or xb, while providing separate
arrays for anything else that is to be computed.  Thus the calling
```

```
sequence
sqrsl(x,n,k,qraux,y,NULL,y,b,y,NULL,110,&info)
will result in the computation of b and rsd, with rsd overwriting y.
More generally, each item in the following list contains groups of
permissible equivalences for a single calling sequence.
1. (y,qty,b) (rsd) (xb) (qy)
2. (y,qty,rsd) (b) (xb) (qy)
3. (y,qty,xb) (b) (rsd) (qy)
4. (y,qy) (qty,b) (rsd) (xb)
5. (y,qy) (qty,rsd) (b) (xb)
6. (y,qy) (qty,xb) (b) (rsd)
In any group the value returned in the array allocated to the group
corresponds to the last member of the group.

sqrst:
Computes least squares solutions to the system
Xb = y
which may be either under-determined or over-determined.  The user may
supply a tolerance to limit the columns of X used in computing the solution.
In effect, a set of columns with a condition number approximately bounded
by 1/tol is used, the other components of b being set to zero.

On return, the arrays x, jpvt, and qraux contain the usual output from
sqrdc, so that the QR decomposition of x with pivoting is fully available
to the user.  In particular, columns jpvt[0], jpvt[1], ..., jpvt[k-1]
were used in the solution, and the condition number associated with
those columns is estimated by ABS(x[0][0]/x[k-1][k-1]).

Author:  Dave Hale, Colorado School of Mines, 12/29/89
```

```
STOEP - Functions to solve a symmetric Toeplitz linear system of equations
 Rf=g for f

stoepd solve a symmetric Toeplitz system - doubles
stoepf solve a symmetric Toeplitz system - floats

Function Prototypes:
void stoepd (int n, double r[], double g[], double f[], double a[]);
void stoepf (int n, float r[], float g[], float f[], float a[]);

Input:
n dimension of system
r array[n] of top row of Toeplitz matrix
g array[n] of right-hand-side column vector

Output:
f array[n] of solution (left-hand-side) column vector
a array[n] of solution to Ra=v (Claerbout, FGDP, p. 57)

Notes:
These routines do NOT solve the case when the main diagonal is zero, it
just silently returns.

The left column of the Toeplitz matrix is assumed to be equal to the top
row (as specified in r); i.e., the Toeplitz matrix is assumed symmetric.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
STRSTUFF -- STRing manuplation subs

cwp_strdup -  duplicate a string
strchop - chop off the tail end of a string "s" after a "," returning
  the front part of "s" as "t".
cwp_strrev - reverse a string



Input:
char *str input string

Output:
none

Returns:
char *   duplicated string

Notes:
This local definition of strdup is necessary because some systems
do not have it.

Author: John Stockwell, Spring 2000.
```

SWAPBYTE - Functions to SWAP the BYTE order of binary data

swap_short_2 swap a short integer
swap_u_short_2 swap an unsigned short integer
swap_int_4 swap a 4 byte integer
swap_u_int_4 swap an unsigned integer
swap_long_4 swap a long integer
swap_u_long_4 swap an unsigned long integer
swap_float_4 swap a float
swap_double_8 swap a double

Function Prototypes:
```
void swap_short_2(short *tni2);
void swap_u_short_2(unsigned short *tni2);
void swap_int_4(int *tni4);
void swap_u_int_4(unsigned int *tni4);
void swap_long_4(long *tni4);
void swap_u_long_4(unsigned long *tni4);
void swap_float_4(float *tnf4);
void swap_double_8(double *tndd8);
```

Notes:
These routines are necessary for reversing the byte order of binary data for transportation between big-endian and little-endian machines. Examples of big-endian machines are IBM RS6000, SUN, NeXT. Examples of little endian machines are PC's and DEC.

These routines have been tested with PC data and run on PC's running several PC versions of UNIX, but have not been tested on DEC.

Also, the number appended to the name of the routine refers to the number of bytes that the item is assumed to be.

Authors: Jens Hartmann, Institut fur Geophysik, Hamburg, Jun 1993
John Stockwell, CWP, Colorado School of Mines, Jan 1994

```
SYMMEIGEN - Functions solving the eigenvalue problem for symmetric matrices
eig_jacobi - find eigenvalues and corresponding eigenvectors via
         the jacobi algorithm for symmetric matrices
sort_eigenvalues -  sort eigenvalues and corresponding eigenvectors
in descending order

Function Prototypes:
void eig_jacobi(float **a, float d[], float **v, int n);
void sort_eigenvalues(float d[], float **v, int n);
 (inspired by Press et. al., 1996)

 Macro used internally
define ROTATE(a,i,j,k,l) g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);\
    a[k][l]=h+s*(g-h*tau);

void eig_jacobi(float **a, float d[], float **v, int n)
eig_jacobi - find eigenvalues and corresponding eigenvectors via
         the jacobi algorithm for symmetric matrices

Function Prototype:
void eig_jacobi(float **a, float d[], float **v, int n);
 (inspired by Press et. al., 1996)
{
    int j,iq,ip,i;
    float tresh,theta,tau,t,sm,s,h,g,c,*b,*z;

    /* allocate space temporarily
    b=alloc1float(n); b-=1;
    z=alloc1float(n); z-=1;

    /* initialize v to the identity matrix
    for (ip=1;ip<=n;ip++) {
        for (iq=1;iq<=n;iq++) v[ip][iq]=0.0;
        v[ip][ip]=1.0;
    }
    /* initialilize to the diagonal on matrix a
    for (ip=1;ip<=n;ip++) {
        b[ip]=d[ip]=a[ip][ip];
        z[ip]=0.0;
    }

    /* main iteration loop
    for (i=1;i<=50;i++) {
```

```
sm=0.0;
for (ip=1;ip<=n-1;ip++) {
    for (iq=ip+1;iq<=n;iq++)
        sm += fabs(a[ip][iq]);
}
/* normal return
if (sm == 0.0) {
    z+=1; free1float(z);
    b+=1; free1float(b);
    return;
}
/* tresh values for first 3 sweeps and therafter
if (i < 4)
    tresh=0.2*sm/(n*n);
else
    tresh=0.0;
for (ip=1;ip<=n-1;ip++) {
    for (iq=ip+1;iq<=n;iq++) {
        g=100.0*fabs(a[ip][iq]);
        if (i > 4 && (float)(fabs(d[ip])+g) == (float)fabs(d[ip])
            && (float)(fabs(d[iq])+g) == (float)fabs(d[iq]))
            a[ip][iq]=0.0;
        else if (fabs(a[ip][iq]) > tresh) {
            h=d[iq]-d[ip];
            if ((float)(fabs(h)+g) == (float)fabs(h))
                t=(a[ip][iq])/h;
            else {
                theta=0.5*h/(a[ip][iq]);
                t=1.0/(fabs(theta)+sqrt(1.0+theta*theta));
                if (theta < 0.0) t = -t;
            }
            c=1.0/sqrt(1+t*t);
            s=t*c;
            tau=s/(1.0+c);
            h=t*a[ip][iq];
            z[ip] -= h;
            z[iq] += h;
            d[ip] -= h;
            d[iq] += h;
            a[ip][iq]=0.0;

            /* Jacobi rotations
```

```
                    for (j=1;j<=ip-1;j++) {
                        ROTATE(a,j,ip,j,iq)
                    }
                    for (j=ip+1;j<=iq-1;j++) {
                        ROTATE(a,ip,j,j,iq)
                    }
                    for (j=iq+1;j<=n;j++) {
                        ROTATE(a,ip,j,iq,j)
                    }
                    for (j=1;j<=n;j++) {
                        ROTATE(v,j,ip,j,iq)
                    }
                }
            }
        }
        for (ip=1;ip<=n;ip++) {
            b[ip] += z[ip];
            d[ip]=b[ip];
            z[ip]=0.0;
        }
    }
    /* this will not happen, hopefully
    fprintf(stderr,"jacobi iteration does not converge\n");
}


void sort_eigenvalues(float d[], float **v, int n)
sort_eigenvalues -  sort eigenvalues and corresponding eigenvectors
in descending order

Function Prototypes:
void sort_eigenvalues(float d[], float **v, int n);
 (inspired by Press et. al., 1996)
{
    int k,j,i;
    float p;

    for (i=1;i<n;i++) {
        p=d[k=i];
        for (j=i+1;j<=n;j++)
            if (d[j] >= p) p=d[k=j];
        if (k != i) {
            d[k]=d[i];
```

```
            d[i]=p;
            for (j=1;j<=n;j++) {
                p=v[j][i];
                v[j][i]=v[j][k];
                v[j][k]=p;
            }
        }
    }
}
```

TEMPORARY_FILENAME - Creates a file name in a user-specified directory.

Function prototype:
char *temporary_filename(char *tempfile);

Input:
tempfile pointer to directory prefix string (eg. /usr/tmp/)

Output:
tempfile pointer to filename string (eg. /usr/tmp/1206aaa)

Notes:
temporary_filename creates a file name by appending a sequence of numbers and letters (which is created by tmpnam) to the prefix string passed as its argument. On return the input argument points to the (now augmented) prefix string.

It is duty of the calling program to provide room for the augmented string. The resulting string is typically used as a name for a temporary file; in this case it is the calling program's job to make sure that the supplied prefix ends with a slash.

This routine was written to supplement the ANSI C function tmpnam which also creates a temporary filename, but within a fixed directory, usually the /tmp directory. Unfortunately, some /tmp directories are too small to hold typical seismic data sets, so this routine allows the user to specify a directory with sufficient capacity. Also note that on many systems, the tmpfile() call avoids this problem by simulating a temporary file with a memory buffer. However, this is not a panacea as the file size might exceed available memory and on some systems this call does actually create a file (again, usually in tmp).
Author: Jack K. Cohen, Colorado School of Mines, 12/12/95

```
TRIDIAGONAL - Functions to solve tridiagonal systems of equations Tu=r for u.

tridif Solve a tridiagonal system of equations (float version)
tridid Solve a tridiagonal system of equations (double version)
tripd Solve a positive definite, symmetric tridiagonal system
tripp Solve an unsymmetric tridiagonal system that uses
Gaussian elimination with partial pivoting

Function Prototypes:
void tridif (int n, float a[], float b[], float c[], float r[], float u[]);
void tridid (int n, double a[], double b[], double c[], double r[], double u[]);
void tripd (float *d, float *e, float *b, int n);
void tripp(int n, float *d, float *e, float *c, float *b);

tridif, tridid:
Input:
n dimension of system
a array[n] of lower sub-diagonal of T (a[0] ignored)
b array[n] of diagonal of T
c array[n] of upper super-diagonal of T (c[n-1] ignored)
r array[n] of right-hand-side column vector

Output:
u array[n] of solution (left-hand-side) column vector

tripd:
Input:
d array[n], the diagonal of A
e array[n], the superdiagonal of A
b array[n], the rhs column vector of Ax=b

Output:
b b is overwritten with the solution to Ax=b
tripp:
Input:
d diagonal vector of matrix
e       upper-diagonal vector of matrix
c       lower-diagonal vector of matrix
b       right-hand vector
n       dimension of matrix

Output:
b       solution vector
```

```
Notes:
For example, a tridiagonal system of dimension 4 is specified as:

    |b[0]    c[0]     0       0  | |u[0]|     |r[0]|
    |a[1]    b[1]    c[1]     0  | |u[1]|  =  |r[1]|
    | 0      a[2]    b[2]    c[2]| |u[2]|     |r[2]|
    | 0       0      a[3]    b[3]| |u[3]|     |r[3]|

The tridiagonal matrix is assumed to be non-singular.

tripd:
Given an n-by-n symmetric, tridiagonal, positive definite matrix A and
 n-vector b, following algorithm overwrites b with the solution to Ax = b

Authors:  tridif, tridid: Dave Hale, Colorado School of Mines, 10/03/89
tripd, tripp: Zhenyue Liu, Colorado School of Mines,  1992-1993
```

```
VANDERMONDE - Functions to solve Vandermonde system of equations Vx=b

vanded solve Vandermonde system of doubles
vandef solve Vandermonde system of floats

Function Prototypes:
void vanded (int n, double v[], double b[], double x[]);
void vandef (int n, float v[], float b[], float x[]);

Input:
n dimension of system
v array[n] of 2nd row of Vandermonde matrix (1st row is all ones)
b array[n] of right-hand-side column vector

Output:
x array[n] of solution column vector

Notes:
The arrays b and x may be equivalenced.

Reference:
Adapted from Algorithm 5.6-2 in Golub, G. H., and Van Loan, C. F., 1983,
Matrix Computations, John-Hopkins University Press.

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
WAVEFORMS   Subroutines to define some wavelets for modeling of seismic
            data

ricker1_wavelet     Compute the time response of a source function as
            a Ricker wavelet with peak frequency "fpeak" Hz.

ricker2_wavelet     Compute a Ricke wavelet with a given period, amplitude
            and distorsion factor

akb_wavelet         Compute the time response of a source function as
            a wavelet based on a wavelet used by Alford, Kelly,
            and Boore.

spike_wavelet       Compute the time response of a source function as
            a spike.

unit_wavelet        Compute the time response of a source function as
            a constant unit shift.

zero_wavelet        Compute the time response of a source function as
            zero everywhere.

berlage_wavelet     Compute the time response of a source function as a
            Berlage wavelet with peak frequency "fpeak" Hz,
            exponential decay factor "decay", time exponent
            "tn", and initial phase angle "ipa".

gaussian_wavelet    Compute the time response of a source function as a
            Gaussian wavelet with peak frequency "fpeak" in Hz.

gaussderiv_wavelet  Compute the time response of a source function as a
            Gaussian first derivative wavelet with peak frequency "fpeak"
            in Hz.
deriv_n_gauss  Compute the n-th derivative of a gaussian in double precision

Function Prototypes:
void ricker1_wavelet (int nt, float dt, float fpeak, float *wavelet);
void ricker2_wavelet (int hlw, float dt, float period, float ampl,
    float distort, float *wavelet);
void akb_wavelet (int nt, float dt, float fpeak, float *wavelet);
void spike_wavelet (int nt, int tindex, float *wavelet);
void unit_wavelet (int nt, float *wavelet);
void zero_wavelet (int nt, float *wavelet);
```

```
void berlage_wavelet (int nt, float dt, float fpeak, float ampl, float tn,
                      float decay, float ipa, float *wavelet);
void gaussian_wavelet (int nt, float dt, float fpeak, float *wavelet);
void gaussderiv_wavelet (int nt, float dt, float fpeak, float *wavelet);

Authors: Tong Fei, Ken Larner
Author: Nils Maercklin, February 2007
```

```
WINDOW - windowing routines

hanningnWindow - returns an n element long hanning window

Function prototypes:
void hanningnWindow(int n,float *w);
Author: Potash Corporation, Sascatchewan: Balasz Nemeth given to CWP 2008
```

wrapArray - wrap an array

Function prototype:
void wrapArray(void *base,size_t nmemb,size_t size,int f)
Author: Potash Corporation: Balasz Nemeth, given to CWP 2008

```
XCOR - Compute z = x cross-correlated with y

xcor compute z= x cross-correlated with y

Function Prototype:
void xcor (int lx, int ifx, float *x, int ly, int ify, float *y ,
int lz, int ifz, float *z);

Input:
lx length of x array
ifx sample index of first x
x array[lx] to be cross-correlated with y
ly length of y array
ify sample index of first y
y array[ly] with which x is to be cross-correlated
lz length of z array
ifz sample index of first z

Output:
z array[lz] containing x cross-correlated with y

Notes:
See notes for convolution function conv().

The operation "x cross correlated with y"  is defined to be:

            ifx+lx-1
    z[i] =   sum    x[j]*y[i+j]  ;  i = ifz,...,ifz+lz-1
             j=ifx

This function performs cross-correlation by
(1) reversing the samples in the x array while copying
    them to a temporary array, and
(2) calling function conv() with ifx set to 1-ifx-lx.
Assuming that the overhead of reversing the samples in x is negligible,
this method enables cross-correlation to be performed as efficiently as
convolution, while reducing the amount of code that must be optimized
and maintained.

Author:  Dave Hale, Colorado School of Mines, 11/23/91
```

```
XINDEX - determine index of x with respect to an array of x values

xindex determine index of x with respect to an array of x values

Input:
nx number of x values in array ax
ax array[nx] of monotonically increasing or decreasing x values
x the value for which index is to be determined
index index determined previously (used to begin search)

Output:
index for monotonically increasing ax values, the largest index
for which ax[index]<=x, except index=0 if ax[0]>x;
for monotonically decreasing ax values, the largest index
for which ax[index]>=x, except index=0 if ax[0]<x

Notes:
This function is designed to be particularly efficient when called
repeatedly for slightly changing x values; in such cases, the index
returned from one call should be used in the next.

Author:  Dave Hale, Colorado School of Mines, 12/25/89
```

```
YCLIP - Clip a function y(x) defined by linear interpolation of the
uniformly sampled values:  y(fx), y(fx+dx), ..., y(fx+(nx-1)*dx).
Returns the number of samples in the clipped function.

yclip clip a function y(x) defined by linear interplolation of
uniformly sampled values

Function Prototype:
int yclip (int nx, float dx, float fx, float y[], float ymin, float ymax,
float xc[], float yc[]);

Input:
nx number of x (and y) values
dx x sampling interval
fx first x
y array[nx] of uniformly sampled y(x) values
ymin minimum y value; must not be greater than ymax
ymax maximum y value; must not be less than ymin

Output:
xc array[?] of x values for clipped y(x)
yc array[?] of y values for clipped y(x)

Returned: number of samples in output arrays xc and yc

Notes:
The output arrays xc and yc should contain space 2*nx values, which
is the maximum possible number (nc) of xc and yc returned.

Author:  Dave Hale, Colorado School of Mines, 07/03/89
```

```
YXTOXY - Compute a regularly-sampled, monotonically increasing function x(y)
from a regularly-sampled, monotonically increasing function y(x) by
inverse linear interpolation.

yxtoxy compute a regularly sampled function x(y) from a regularly
sampled, monotonically increasing function y(x)

Function Prototype:
void yxtoxy (int nx, float dx, float fx, float y[],
int ny, float dy, float fy, float xylo, float xyhi, float x[]);

Input:
nx number of samples of y(x)
dx x sampling interval; dx>0.0 is required
fx first x
y array[nx] of y(x) values; y[0] < y[1] < ... < y[nx-1] required
ny number of samples of x(y)
dy y sampling interval; dy>0.0 is required
fy first y
xylo x value assigned to x(y) when y is less than smallest y(x)
xyhi x value assigned to x(y) when y is greater than largest y(x)

Output:
x array[ny] of x(y) values

Notes:
User must ensure that:
(1) dx>0.0 && dy>0.0
(2) y[0] < y[1] < ... < y[nx-1]

Author:  Dave Hale, Colorado School of Mines, 06/02/89
```

```
ZASC - routine to translate ncharacters from ebcdic to ascii

zasc - convert n characters from ebcdic to ascii format

Input:
nchar number of characters to be translated
ainput pointer to input characters

Output:
aoutput pointer to output characters

Function Prototype:
int zasc(char *ainput, char *aoutput, integer nchar);
Notes:
 translated by f2c.  Horribly inefficient, but little used

Author: Stew Levin of Mobil, 1997
```

```
ZEBC - routine to translate ncharacters from ascii to ebcdic

zebc - convert n characters from ascii to ebcdic format

Input:
nchar number of characters to be translated
ainput pointer to input characters

Output:
aoutput pointer to output characters

Function Prototype:
int zebc(char *ainput, char *aoutput, integer nchar);
Notes:
 translated by f2c.  Horribly inefficient, but little used

Author: Stew Levin of Mobil, 1997
```

```
CHECK - CHECK triangulated models


badModel bad Model flag
checkVertexUse check Vertex Use
checkEdgeUse check Edge Use
checkFace check Face
checkModel check Model

Function Prototypes:

void badModel(void);
void checkVertexUse (VertexUse *vu);
void checkEdgeUse (EdgeUse *eu);
void checkFace (Face *f);
void checkModel (Model *m);

checkVertexUse:
Input:
vu Pointer to VertexUse

checkEdgeUse:
Input:
eu pointer to EdgeUse

checkFace:
Input:
f pointer to Face

checkModel:
Input:
m pointer to Model

Notes: Routines for checking triangulated models.

Author:  Dave Hale, Colorado School of Mines, 07/09/90
```

```
CIRCUM - define CIRCUMcircles for Delaunay triangulation

circum - compute center and radius-squared of circumcircle of 3 (x,y)
          locations
circumTri - compute center and radius-squared of circumcircle of
            triangular face

Function Prototypes:
void circum (float x1, float y1, float x2, float y2, float x3, float y3,
float *xc, float *yc, float *rs);
void circumTri (Tri *t);
circum:
Input:
x1 x-coordinate of first point
y1 y-coordinate of first point
x2 x-coordinate of second point
y2 y-coordinate of second point
x3 x-coordinate of third point
y3 y-coordinate of third point

Output:
xc pointer to x-coordinate of center of circumcircle
yc pointer to y-coordinate of center of circumcircle
rs pointer radius^2 of circumcircle

circumTri:
Input:
t Pointer to Tri

Returns:
xc x-coordinate of circumcircle
yc      y-coordinate of circumcircle
rs      radius^2 of circumcircle

Author:  Dave Hale, Colorado School of Mines, Fall 1990.
```

```
COLINEAR - determine if edges or vertecies are COLINEAR in triangulated
           model
edgesColinear see whether or not two edges are colinear
vertexBetweenVertices determine whether or not a vertex is on a line
                          between two other vertices
Function Prototypes:
int edgesColinear (Edge *e1, Edge *e2);
int vertexBetweenVertices (Vertex *v, Vertex *v1, Vertex *v2);

edgesColinear:
Input:
e1 pointer to first Edge
e2 pointer to second Edge

Returns: (int)
1 if colinear

vertexBetweenVertices:
Input:
v pointer to first Vertex in question
v1 pointer to first reference Vertex
v2 pointer to second reference Vertex

Returns: integer
1 if v=v1 or v=v2 or if v is between v1 and v2
0 otherwise

Author:  Dave Hale, Colorado School of Mines, Fall 1990.
```

```
CREATE - create model, boundary edge triangles, edge face, edge vertex, add
         a vertex

makeModel Make and return a pointer to a new model
makeBoundaryEdgeTri Create a boundary edge and triangle
makeEdgeFace Create an edge by connecting two vertices
makeEdgeVertex Create an edge connecting an existing vertex (v1) to a
                        new vertex
addVertexToModel Add a vertex to model, and return pointer to new vertex
insideTriInModel return pointer to triangle in model containing
                    specified (x,y) coordinates

Function Prototypes:
Model *makeModel (float xmin, float ymin, float xmax, float ymax);
void makeBoundaryEdgeTri (Vertex *v, Edge **enew, Tri **tnew);
void makeEdgeFace (Vertex *v1, Vertex *v2, Edge **enew, Face **fnew);
Vertex* addVertexToModel (Model *m, float x, float y);
Tri* insideTriInModel (Model *m, Tri *start, float x, float y);

makeModel:
Input:
xmin minimum x-coordinate
ymin minimum y-coordinate
xmax maximum x-coordinate
ymax maximum y-coordinate

Returns: pointer to a new Model

makeBoundaryEdgeTri:
Input:
v specified boundary Vertex

Output:
enew new boundary Edge
tnew new boundary triangle

Notes:
The specified vertex and the adjacent vertices on the boundary
are assumed to be colinear.  Therefore, the resulting
boundary triangle has zero area, and is intended to enable
deletion of the specified vertex from the boundary.

makeEdgeFace:
```

```
Input:
v1 First Vertex
v2 second Vertex

Output:
enew new Edge
fnew new Face

Notes:
The vertices must be adjacent to a single common face.
This face is closed off by the new edge, and a new edge and
a new face are made and returned.

addVertexToModel:
Input:
m model
x x-coordinate of new vertex
y y-coordinate of new vertex

Notes:
If the new vertex is close to an existing vertex, this function returns NULL.

insideTriInModel:
Input:
m Model
start triangle to look at first (NULL to begin looking anywhere)
x x-coordinate
y y-coordinate

Notes:
Points on an edge of a triangle are assumed to be inside that triangle.
An edge may be used by two triangles, so two triangles may "contain"
a point that lies on an edge.  The first triangle found to contain
the specified point is returned.

Author:  Dave Hale, Colorado School of Mines, Fall 1990.
```

```
DELETE - DELETE vertex, model, edge, or boundary edge from triangulated model

deleteVertexFromModel Delete a vertex from model
killModel Delete a model along with everything in it
killEdge Delete an edge
killBoundaryEdge Kill a boundary edge

Function Prototypes:
void deleteVertexFromModel (Model *m, Vertex *v);
void killModel (Model *m);
void killEdge (Edge *e, Face **fs);
void killBoundaryEdge (Edge *e);

deleteVertexFromModel:
Input:
m   pointer to Model
v pointer to Vertex to be deleted

killModel:
Input:
m pointer to Model

killEdge:
Input:
e Edge to delete

Output:
fs surviving Face

killBoundaryEdge:
Input:
e  boundary Edge

Notes:
Killing a boundary edge is typically done after a new boundary vertex
is inserted on an existing boundary edge.

Author:  Dave Hale, Colorado School of Mines, Fall 1990.
```

```
DTE - Distance to Edge

distanceToEdge - return distance to edge from specified (x,y) coordinates

Function Prototype:
float distanceToEdge (Edge *e, float x, float y);
distanceToEdge:
Input:
e edge to which distance is to be computed
x x-coordinate
y y-coordinate

Author:  Dave Hale, Colorado School of Mines, 09/11/90
```

```
FIXEDGES - FIX or unFIX EDGES between verticies

fixEdgeBetweenVertices Fix edge(s) between vertices, creating new
                              colinear edges as necessary.
unfixEdge unfix edge

fixEdgesBetweenVertices:
Input:
v1 pointer to first Vertex
v2 pointer to second Vertex

Returns:
0 if unable to fix edges
1 otherwise

unfixEdge:
Input:
e edge to be unfixed

Returns:
0 if unable to unfix edge
1 otherwise
Author:  Dave Hale, Colorado School of Mines, 06/04/91
```

```
INSIDE -  Is a vertex or point inside a circum circle, etc. of a triangulated
          model

inCircum determine whether or not a vertex is inside a circum circle
inCircumTri determine whether or not a vertex is inside a circum circle of
                 a triangle
in3Vertices determine whether or not a vertex is inside triangle (v1,v2,v3)
inTri determine whether or not a vertex is inside a triangle

Function Prototypes:
int inCircum (float x, float y, float xc, float yc, float rs);
int inCircumTri (float x, float y, Tri *t);
int in3Vertices (float x, float y, Vertex *v1, Vertex *v2, Vertex *v3);
int inTri (float x, float y, Tri *t);
inCircum:
Input:
x x-coordinate of vertex
y y-coordinate of vertex
xc x-coordinate of center of circumcircle
yc y-coordinate of center of circumcircle
rs radius^2 of circumcircle

Returns:
1 if x,y inside of circumcircle
0 otherwise

Notes:
A vertex exactly on the edge of a circumcircle is taken as being outside

inCircumTri:
Input:
x x-coordinate of vertex
y y-coordinate of vertex
t pointer to Tri

Returns:
1 if x,y inside of circumcircle of a triangle
0 otherwise

Notes:
A vertex exactly on the edge of a circumcircle is taken as being outside

in3Vertices:
```

```
Input:
x x-coordinate of vertex
y y-coordinate of vertex
v1 pointer to first Vertex
v2 pointer to second Vertex
v3 pointer to third Vertex

Returns:
1 if x,y inside of v1,v2,v3
0 otherwise

Notes:
A vertex exactly on an edge of the triangle is taken as being inside

inTri:
Input:
x x-coordinate of vertex
y y-coordinate of vertex
t pointer to Tri

Returns:
1 if x,y inside a triangle
0 otherwise
Notes:
A vertex exactly on the edge of a triangle is inside

Author:  Dave Hale, Colorado School of Mines, 06/04/91
```

```
NEAREST - NEAREST edge or vertex in triangulated model

nearestEdgeInModel Return pointer to edge in model nearest to
                     specified (x,y) coordinates

nearestVertexInModel Return pointer to vertex in model nearest
                    to specified (x,y) coordinates

Function Prototypes:
Vertex* nearestVertexInModel (Model *m, Vertex *start, float x, float y);
Edge* nearestEdgeInModel (Model *m, Edge *start, float x, float y);

nearestEdgeInModel:
Input:
m model
start edge to look at first (NULL to begin looking anywhere)
x x-coordinate
y y-coordinate

Returns: pointer to nearest Edge

nearestVertexInModel:
Input:
m model
start vertex to look at first (NULL to begin looking anywhere)
x x-coordinate
y y-coordinate

Returns: pointer to nearest Vertex

Author:  Dave Hale, Colorado School of Mines, Fall 1990
```

```
PROJECT - project to edge in triangulated model

projectToEdge - Project point with specified (x,y) coordinates to specified
                edge
Function Prototype:
void projectToEdge (Edge *e, float *x, float *y)

Input:
e edge to which point is to be projected
x x-coordinate before projection
y y-coordinate before projection

Output:
x x-coordinate after projection
y y-coordinate after projection

Author:  Dave Hale, Colorado School of Mines, 09/11/90
```

```
READWRITE - READ or WRITE a triangulated model

readModel Read a model in the form produced by writeModel
writeModel Write a model to a file

Function Prototypes:
Model *readModel (FILE *fp);
void writeModel (Model *m, FILE *fp);

readModel:
Input:
fp file pointer to file containing model

writeModel:
Input:
m pointer to Model
fp file pointer

Author:  Jack K. Cohen, Center for Wave Phenomena, 09/21/90
Modified:  Dave Hale, Center for Wave Phenomena, 11/30/90
Converted representation of model from ascii to binary for speed.
Added code to read attributes.
Modified (writeModel):  Craig Artley, Center for Wave Phenomena, 04/08/94
Corrected bug;  previously the edgeuses and vertextuses of the
exterior face were not written.
```