GUISDAP Documentation

M. S. Lehtinen* and A. Huuskonen**

* Sodankylä Geophysical Observatory, Sodankylä, Finland
** Finnish Meteorological Institute, Helsinki, Finland

This is a documentation of all general-purpose utility routines

Generation of alternating code sequences. Input parameters are (a vector of) scan number(s) and (a vector of) Walsh index(es). The routine will then output vectors of alternating code signs.

/geo/gmt/askoh/guisdap/m152/altcod.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% altcod.m
%
% function to generate alternating codes
% both inputs may be vectors and the output will be a matrix with
% code sequences in its columns
%
% res=altcod(scan,index)
%
  function res=altcod(scan,index)
%
  x=(0:63)'/64; s=zeros(64,6);
  for i=1:6
    s(:,i)=(-1).^(floor(x*2^i));    % generate binary representations
  end                               % of integers
  s=(1-s)/2;
  res=zeros(length(index),length(scan));
  for i=1:length(scan)
    for j=1:length(index)
      res(j,i)=(-1)^sum(s(index(j)+1,:).*s(scan(i)+1,:));
    end
  end
```

Various Barker codes are give by

/geo/gmt/askoh/guisdap/m152/Barker.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% function to give various Barker codes
%
% function code=Barker(Nbits);
function code=Barker(Nbits);

if Nbits==3,
  code=[1 1 -1]';
elseif Nbits==4,
  code=[1 1 -1 1]';
elseif Nbits==5,
  code=[1 1 1 -1 1]';
elseif Nbits==7,
  code=[1 1 1 -1 -1 1 -1]';
elseif Nbits==13,
  code=[1 1 1 1 1 -1 -1 1 1 -1 1 -1 1]';
else
  fprintf('%.0f bit Barker code not available',Nbits)
  error(' Updata Barker')
end
```

The following gives the bitwise or of two integers when written in binary representation.

/geo/gmt/askoh/guisdap/m152/bitwiseor.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% function ans=bitwiseor(x,y,bits)
  function ans=bitwiseor(x,y,bits)

if x==y,
  ans=x;
```

```
8       else
9       ans=0;
10        for i=0:bits-1,
11          apu=2^i;
12          if (rem(floor(x/apu),2) | rem(floor(y/apu),2));
13            ans=ans+apu;
14          end
15        end
16      end
```

The routine `canon.m` transforms file path names given using any of the characters / (UNIX tree separator) \ (DOS) or : (Mac) to be useful in the system where MATLAB is currently running. It cannot handle disk names for DOS correctly yet.

/geo/gmt/askoh/guisdap/m152/canon.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % function to change directory characters in file names to
4       % be good for various file systems
5       % Does not work for disk names in DOS - somebody, please correct this
6       %
7       % Modified to handle DOS disk drives, 940613 CJH
8       %
9       function res=canon(fn,output);
10
11      if nargin==1, output=1; end
12      comp=computer;
13      comp(20)=' ';
14      if comp(1:2)=='PC', c='\';
15      elseif comp(1:3)=='MAC', c=':';
16      elseif comp(1:5)=='VAXVM', c='\';
17      else c='/';
18      end;
19      loc=find( (fn==':') | (fn=='/') | (fn=='\') );
20
21      % for a DOS machine, a ':' in the second character probably indicates
22      % a disk drive designator such as A: or C: - this must not be changed
23      % to a '\'
24      if length(loc) > 0
25        if comp(1:2)=='PC'
26          if loc(1)==2, loc(1)=[]; end
27        end;
28      end;
29
30      fn(loc)=setstr(ones(1,length(loc))*abs(c));
31      res=fn;
32
33      % Produce output in lowercase letters when PC-Matlab is used
34      if comp(1:2)=='PC' res=lower(fn); end;
35
36      if output, fprintf(['canon: ',res,'\n']); end
37
```

The following function is useful in checking, if all elements of its argument vector have the same value. If they have the same value, that common value is returned, otherwise an error is generated.

/geo/gmt/askoh/guisdap/m152/cheq.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % function res=cheq(x)
4       function res=cheq(x)
5       maxim=max(x);minim=min(x);
6       if((maxim-minim)<=1000*eps*max(abs([1,maxim,minim]))),
7         res=x(1);
8       else
9         fprintf(' The input values to cheq are not all equal\n')
10        disp(x), dbstop error,
11        error(' ')
12      end
```

The following makes a column vector out of its argument.

/geo/gmt/askoh/guisdap/m152/col.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % col.m
4       %
5       % Makes a vector or a matrix into a column vector.
6       %
7       % res=col(A)
8       %
```

```
9         function res=col(A)
10      %
11        res=A(:);
```

The matlab size function is awkward, when one wants to know the number of columns of a matrix, because it returns both number of rows and columns in two arguments. We often need the following:

/geo/gmt/askoh/guisdap/m152/cols.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % cols.m
4       %
5       % Returns the number of columns in a matrix
6       %
7       % res=cols(A)
8       %
9         function res=cols(A)
10      %
11        [r,res]=size(A);
```

The following takes in a Fisher information matrix. It then calculates the error bars of the parameters and also correlation coefficients side diagonalwise. All these values are put to the output row vector.

/geo/gmt/askoh/guisdap/m152/covm2vec.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % creates a vector from the error covariance matrix. The vector contains
4       % - errors (sqrt of the diagonal elements, first 'len' elements)
5       % - correlations so that first side diagonal comes first etc
6       % Needed, as the matlab matrix elements cannot be matrices themselves
7       %
8       % See also: vec2covm
9       %function covvec=covm2vec(errmatr)
10       function covvec=covm2vec(errmatr)
11
12      er=sqrt(diag(errmatr));  % These are the errors of the parameters
13      covm=errmatr./(er*er');     % These are the covariances
14
15      len=length(er);
16      covvec=zeros(1,len*(len+1)/2);
17      covvec(1:len)=er';
18      ind=len;
19      for i=1:len-1
20        covvec(ind+(1:(len-i)))=diag(covm,i)';
21        ind=ind+len-i;
22      end
```

The following returns a set of unique values in a vector in ascending order.

/geo/gmt/askoh/guisdap/m152/diff_val.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       %
4       % Function returns differing elements of a vector in ascending order
5       % function val=diff_val(vec);
6
7       function val=diff_val(vec);
8
9       val=sort(vec);
10      val(find(diff(val)==0))=[];
```

`elem` give the (i,j) component of A, so that `elem(A,i,j)` is equal to `A(i,j)`, when A is a matrix. Useful when A is a function call itself.

/geo/gmt/askoh/guisdap/m152/elem.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % Gives the (i,j) element of A, i.e. equal to A(i,j)
4       % Useful when A is itself is function call
5       %
6       % function res=elem(A,i,j)
7         function res=elem(A,i,j)
8       %
9         if nargin==3,
10          res=A(i,j);
11        else
12          res=A(i);
13        end
```

The following routine gets two vectors `xi` and `yi` as a table for interpolation. Input values `x` are interpolated (or, if necessary, extrapolated) linearly to give corresponding `y`-values.

/geo/gmt/askoh/guisdap/m152/inter3.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Linear interpolation routine.
% When data (yi) is specified at points (xi) and a new set of points (x)
% is given, the routine interpolates or extrapolates the give the
% function values (y) at those points (x)
%
% function y=inter3(x,xi,yi);
function y=inter3(x,xi,yi);

[xi,ind]=sort(xi);
yi=yi(ind);

len=length(xi);
if len<2,
  error('No model exists')
elseif len>2,
  % linear interpolation and extrapolation
  k=diff(yi)./diff(xi);
  ind=find(x<=xi(2));
  if length(ind)>0, y(ind)=yi(1)+k(1)*(x(ind)-xi(1)); end
  for i=3:len-1,
    ind=find(xi(i-1)<x & x<=xi(i));
    if length(ind)>0, y(ind)=yi(i-1)+k(i-1)*(x(ind)-xi(i-1)); end
  end
  ind=find(x>xi(len-1));
  if length(ind)>0, y(ind)=yi(len-1)+k(len-1)*(x(ind)-xi(len-1)); end
end
```

The following natural constants are needed in GUISDAP.

/geo/gmt/askoh/guisdap/m152/nat_const.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% script defining useful natural constants:
% v_lightspeed=299792458;
% v_Boltzmann=1.380658e-23;
% v_electronmass=9.1093897e-31;
% v_amu=1.6605402e-27;
% v_electronradius=2.81794092e-15;
% v_epsilon0=8.854188E-12;
% v_elemcharge=1.602177E-19;

v_lightspeed=299792458;
v_Boltzmann=1.380658e-23;
v_electronmass=9.1093897e-31;
v_amu=1.6605402e-27;
v_electronradius=2.81794092e-15;
v_epsilon0=8.854188E-12;
v_elemcharge=1.602177E-19;

global v_lightspeed v_Boltzmann v_electronmass v_amu
global v_electronradius v_epsilon0 v_elemcharge
```

The following function returns the root-mean-square value of a matrix's columns.

/geo/gmt/askoh/guisdap/m152/rms.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% rms.m
%
% calculates the rms value of columns of a matrix
%
  function res=rms(A)
%
  res=sqrt(mean(A.^2));
```

The following routine makes a row vector of the values in a matrix. There was a famous error in the routine which made it to work only with real matrices.

/geo/gmt/askoh/guisdap/m152/row.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% row.m
%
% makes a vector or a matrix into a row vector
%
% res=row(A)
```

```
%
  function res=row(A)
%
  res=A(:).';
```

This is a simple way to get the number of rows of a matrix. (it gives the same result as calling x=size(y)).

/geo/gmt/askoh/guisdap/m152/rows.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% rows.m
%
% Returns the number of rows in a matrix.
%
% res=rows(A)
%
  function res=rows(A)
%
  [res,r]=size(A);
```

The following writes a standard GUP startup message on the screen, whenever Matlab is started (it is called from `startup.m`).

/geo/gmt/askoh/guisdap/m152/start_GUP.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% The startup file for GUISDAP.
% Defines the GUP version number and certain global variables
% namely:   GUP_ver path_GUP path_exps path_tmp name_expr name_site data_path result_path
% If you want this to executed every time you invoke matlab,
% add reference to start_GUP to your personal startup file

clear all, clear global

global GUP_ver path_GUP path_exps path_tmp name_expr name_site data_path result_path
GUP_ver=1.52;
fprintf('GUISDAP v. %4.2fbeta ',GUP_ver);
fprintf('by Markku Lehtinen and Asko Huuskonen\n');
% The Guisdap source directories are in directory
path_GUP='/geo/gmt/askoh/guisdap/';
% The GUISDAP experiment directories are in directory
path_exps=[path_GUP,'exps/'];
% The user can modify the latter e.g. when creating new experiments to say
% path_exps=['/usr/home/tony/exps'];
% define path to temporary storage space
path_tmp=[getenv('TMPDIR'),'/'];
% If environment variable was not found (especially in Macintosh and Windows), use
if length(path_tmp)==1
  path_tmp=[path_GUP,'garbage:'];
end
```

Matlab allways runs the file `startup.m` before doing anything else. This is a good place to specify the main path to GUP.

/geo/gmt/askoh/guisdap/m152/startup.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% This startup-file is provided for compatibility with the practise of
% version 1.50.  If you wish that start_GUP is not exectuted every time matlab is
% started, comment the next line or , preferably, make your own startup.m file
% Version 1.52 does not use this file anyway. The version 1.50 references to
% to startup have been replaced by start_GUP, whereever found.
  start_GUP
```

The following routines are useful in timing the execution speed of the GUISDAP system. Presently, the timing works correctly only on Macintosh ci – and anyway, only GUIZARDs can possibly understand, how to use them.

/geo/gmt/askoh/guisdap/m152/t_init.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Initialization for a set of timing routines.
% Not very user friendly or useful, but used in the analyis package
% The calls must be in the following order
% t_init     % Initialization
% t_start(1) % start of first timing block
% t_start(2) % start of second timing block within the first one
% t_start(3) % start of third timing block within the second one
% t_stop(3) % end of block 3
% t_stop(2) % end of block 2
```

```
12      % t_stop(1) % end of block 1
13      % t_result  % output of time used
14      % function t_init
15        function t_init
16
17      global ti_start ti_stop fl_start fl_stop ti_calls
18
19      ti_start=zeros(1,10);
20      ti_stop=zeros(1,10);
21      fl_start=zeros(1,10);
22      fl_stop=zeros(1,10);
23      ti_calls=zeros(1,10);
```

/geo/gmt/askoh/guisdap/m152/t_result.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % See also: t_init
4       % function t_result
5         function t_result
6
7       global ti_start ti_stop fl_start fl_stop ti_calls
8
9       ind=find(ti_start>0);
10      ajat=ti_stop(ind)-ti_start(ind)
11      cumcalls=fliplr(cumsum(fliplr(ti_calls(ind))));
12
13      %ajat=ajat-0.0186*cumcalls+0.0091*ti_calls(ind)
14      %ajat=ajat-0.0244*cumcalls+0.012*ti_calls(ind)  MacIIci
15
16      len=length(ti_stop);
17      NN=50;t=cputime;for i=1:NN;t_start(len);t_stop(len);end  %AH 94-6-14
18      time=(cputime-t)/NN  %AH 94-6-14
19      ajat=ajat-time*cumcalls+(time/2)*ti_calls(ind);
20
21      flo=(fl_stop(ind)-fl_start(ind))/1000;
22      for i=1:length(ind),
23        fprintf('%1.0f ',i);
24        fprintf('%9.3fs ',ajat(i));
25        fprintf('%10.3f kflops ',flo(i));
26        fprintf('%5.0f calls ',ti_calls(ind(i)));
27        fprintf('%7.3f kflops/s\n',flo(i)/ajat(i));
28      end;
```

/geo/gmt/askoh/guisdap/m152/t_start.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % See also: t_init
4       % t_start.m
5       function t_start(i),
6
7       global ti_start fl_start ti_calls
8
9       ti_calls(i)=ti_calls(i)+1;
10      ti_start(i)=ti_start(i)+cputime; %AH 94-6-14
11      fl_start(i)=fl_start(i)+flops;
```

/geo/gmt/askoh/guisdap/m152/t_stop.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % See also: t_init
4       function t_stop(i),
5
6       global fl_stop ti_stop
7
8       fl_stop(i)=fl_stop(i)+flops;
9       ti_stop(i)=ti_stop(i)+cputime;  %AH 94-6-14
```

The following routines transform dates/times to seconds calculated from the start of the year and vice versa.

/geo/gmt/askoh/guisdap/m152/tosecs.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % function to convert time in form
4       % [Year Month Day Hour Min Sec]
5       % or in form [YYMM DDHH MMSS]   (this is the EISCAT style)
6       % to seconds from the beginning of year
7       %
8       % See also: toYMDHMS
9       % function [secs,years]=tosecs(x)
10        function [secs,years]=tosecs(x)
11
12      [m,n]=size(x);
```

```
    if n~=2 & n~=3 & n~=6,
      if m==2 | m==3 | m==6,
         x=x'; [m,n]=size(x);
      else
        fprintf('Illegal argument:\n')
        disp(x)
        secs=NaN;
        years=NaN;
      end
    end

    if n==2,
      y=x;
      x(:,[3,6])=rem(y,100);
      x(:,[2,5])=rem(floor(y/100),100);
      x(:,[1,4])=floor(y/10000);
    elseif n==3,
      x(:,6)=rem(x(:,3),100); x(:,5)=floor(x(:,3)/100);
      x(:,4)=rem(x(:,2),100); x(:,3)=floor(x(:,2)/100);
      x(:,2)=rem(x(:,1),100); x(:,1)=floor(x(:,1)/100)+1900;
    elseif n==6,
      if x(1,1)<100, x(:,1)=1900+x(:,1); end
    end

    years=x(:,1);
    for year=diff_val(x(:,1))'
      daym=[0 31 28 31 30 31 30 31 31 30 31 30 31];
      if rem(year,4)==0, daym(3)=29; end  % works up to 2100
      days=cumsum(daym(1:12))';

      ind=find(years==year);
      hours(ind,1)=(days(x(ind,2))+(x(ind,3)-1))*24+x(ind,4);
      secs(ind,1)=(hours(ind)*60+x(ind,5))*60+x(ind,6);
    end
```

/geo/gmt/askoh/guisdap/m152/toYMDHMS.m

```
    % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
    % AH 94-4-15 Added the Uppsala formatting commands
    % toYMDHMS.m
    % function to convert time in second from the beginning of year
    % to form [Year Month Day Hour Min Sec]
    % If mask is given, those elements will be returned in a character
    % string, with separator 'sep' if specified
    %
    % See also: tosecs
    % function YMD=toYMDHMS(years,secs,mask,sep)
      function YMD=toYMDHMS(years,secs,mask,sep)

    secs=col(secs);
    if length(years)==1 years=years*ones(size(secs));
    else                years=col(years); end

      days=cumsum([0 31 28 31 30 31 30 31 31 30 31 30 31])';
      days=days(:,ones(1,length(years)));
      ind=find(rem(years,4)==0);
      if length(ind)>0,days(3:13,ind)=days(3:13,ind)+1;end

      YMD(:,6)=rem(secs,60); apu=floor(secs/60);
      YMD(:,5)=rem(apu,60); apu=floor(apu/60);
      YMD(:,4)=rem(apu,24); apu=floor(apu/24)+1;
      index=sum((apu(:,ones(1,13))'-days)>0);
      YMD(:,2)=index';
      YMD(:,3)=apu-days(YMD(:,2),1);
      YMD(:,1)=years;

    if nargin==2 return, end

    if nargin==3 sep=':';end

    YMD(:,1)=YMD(:,1)-100*floor(YMD(:,1)/100);
    fmt='%0.2d';
    for i=1:length(mask)-1 fmt=[fmt sep '%0.2d']; end
    for i=1:length(secs)
     YMD2(i,:)=sprintf(fmt,YMD(i,mask));
    end
    YMD=YMD2;

```

The following makes the inverse to what `covm2vec.m` does.

/geo/gmt/askoh/guisdap/m152/vec2covm.m

```
    % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
    %
    % creates the covariance matrix back from the storage vector form
```

```
4       % Needed, as the matlab matrix elements cannot be matrices themselves
5       %
6       % See also: covm2vec
7       %function errmatr=vec2covm(covvec)
8        function errmatr=vec2covm(covvec)
9
10       len=floor(sqrt(2*length(covvec)));
11       errm=zeros(len,len);
12       er=covvec(1:len)';
13       ind=len;
14       errm=errm+diag(ones(len,1),0);
15       for i=1:len-1
16         errm=errm+diag(covvec(ind+(1:(len-i))),i);
17         errm=errm+diag(covvec(ind+(1:(len-i))),-i);
18         ind=ind+len-i;
19       end
20
21       errmatr=errm.*(er*er');
22
```

Some of the ambiguity functions in GUISDAP are specified by giving two vectors – the first one gives all the ranges, where the ambiguity is non-zero and the second one gives the corresponding ambiguity weights. The range vector (=coordinate column vector) consists of integer values, as the range is calculated in units of `p_dtau`. It is however, not a good idea to plot the ambiguity function by asking matlab to plot the ambiguity weights as a function of the range vector, as the ambiguity will then not appear to be zero, where it vanishes. The following routine pads the range vector correctly by ranges with vanishing ambiguity mass around every ambiguity component.

/geo/gmt/askoh/guisdap/m152/w1fill.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % w1fill.m
4       %
5       % fills the coordinate column vector zin with an appropriate number of
6       % zero boundary coordinates.
7       %
8       % See also: w2fill w2uni
9       % function zo=w1fill(zi)
10        function zo=w1fill(zi)
11      %
12        z1=min(zi)-1;z2=max(zi)+1;
13        dz=diff(zi);
14        w=find(dz>1);
15        if isempty(w), zo=[z1;zi;z2]; return; end
16        zr=zi(w)+1;zl=zi(w+1)-1;
17        z=w2uni(zr,zl);
18        zo=[z1;w2uni(z,zi);z2];
```

The following routine gives the union of two range vectors (so that each range can appear only once).

/geo/gmt/askoh/guisdap/m152/w2uni.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % The sorted union of two vectors x and y of the same type.
4       % The vectors x and y are assumed to be sorted in ascending
5       % order.
6       % The output is a vector of the same type as the input.
7       %
8       % z=w2uni(x,y)
9       %
10        function z=w2uni(x,y)
11      %
12        if isempty(x), z=y;
13        elseif isempty(y), z=x;
14        return;
15        end;
16      %
17        nx=diff(size(x));ny=diff(size(y));
18      %
19        if (nx<0 | ny<0) | (nx==0 & ny==0)
20          m=1;                        % make row vectors
21          xx=x';yy=y';
22        else
23      %   m=0;
24          xx=x;yy=y;
25        end
26        w=sort([xx yy]');             % column vector
27        dw=diff(w);
28        wi=find(dw==0);
29        w(wi)=[];
30
```

```
31        if m
32          z=w;
33        else
34          z=w';
35        end
```

The following routine returns the smalles possible range of indices to a specified column of the input matrix that contain all non-zero values appearing in the column.

/geo/gmt/askoh/guisdap/m152/wnz.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % wnz.m
4       %
5       % The interval between uppper and lower boundaries of the non-zero
6       % elements of a column 'ch' in a matrix 'A'
7       %
8       %  function resi=wnz(A,ch)
9         function resi=wnz(A,ch)
10      %
11        B=A(:,ch);
12        ivector=find(B~=0);
13        resi=(min(ivector):max(ivector))';
```

The following routine is useful in plotting columns of a matrix so that they do no overlap.

/geo/gmt/askoh/guisdap/m152/zadj.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % Function to separate columns from a waveform
4       % matrix so that they do not overlap when plotted.
5       % Useful in plotting many of GUP waveforms.
6       % function res=zadj(curv);
7       function res=zadj(curv);
8       maxs=max(curv); mins=min(curv);
9       [m,n]=size(curv);
10      res=curv-ones(m,1)*mins+...
11          ones(m,1)*[0 cumsum( maxs(1:(n-1))-mins(1:(n-1)) )]*1.1;
```