



LTPDA Toolbox



Note LTPDA Repositories are external to MATLAB and need to be set up independently. See [Setting up an LTPDA Repository](#) for details.

Expected Background for Users

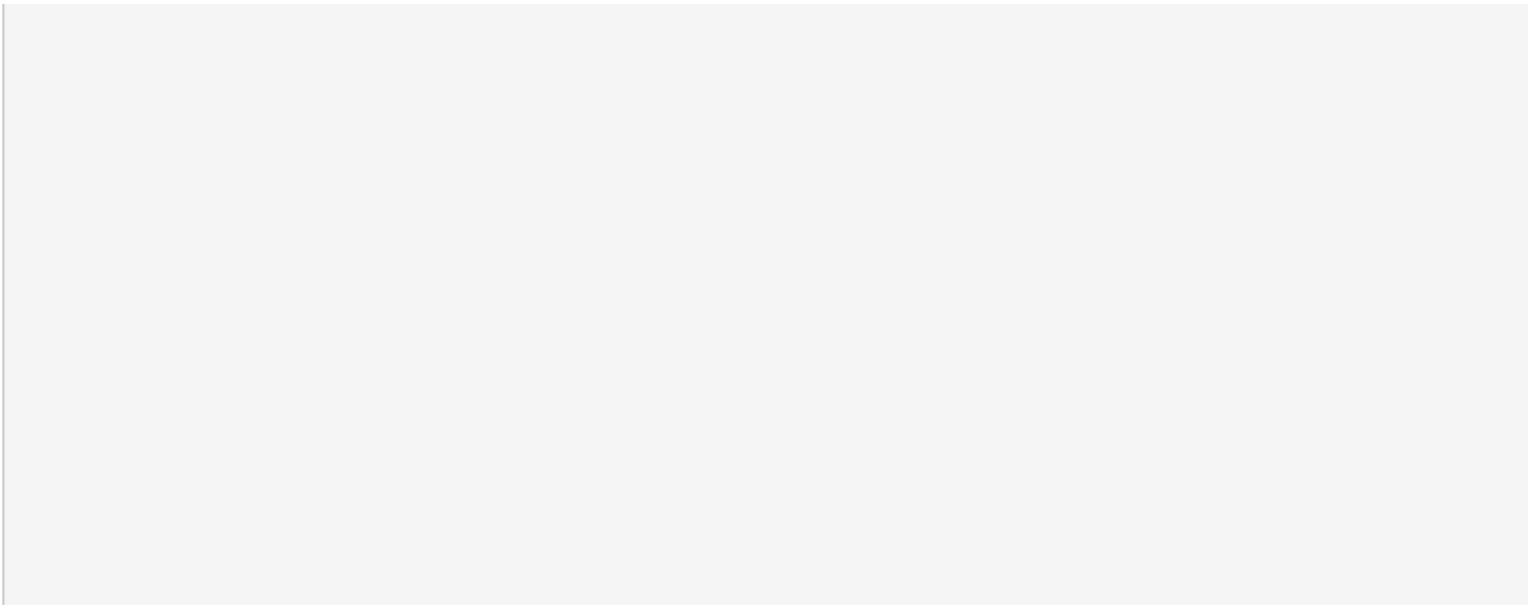
MATLAB

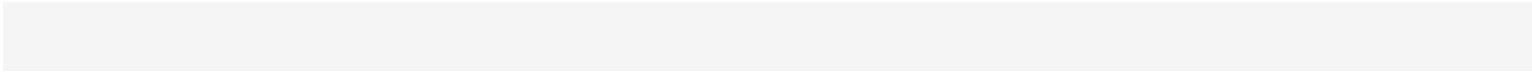
This documentation assumes you have a basic working understanding of MATLAB. You need to know basic MATLAB syntax for writing your own m-files, and you need to have basic familiarity





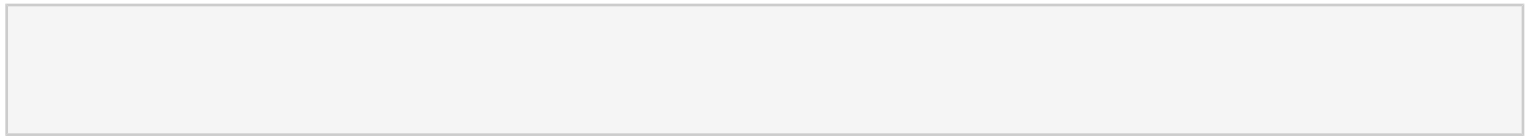


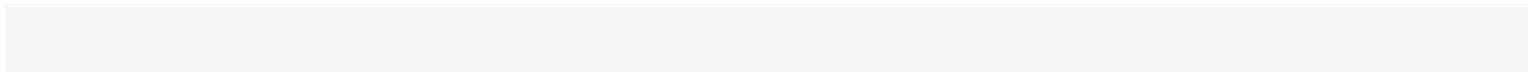
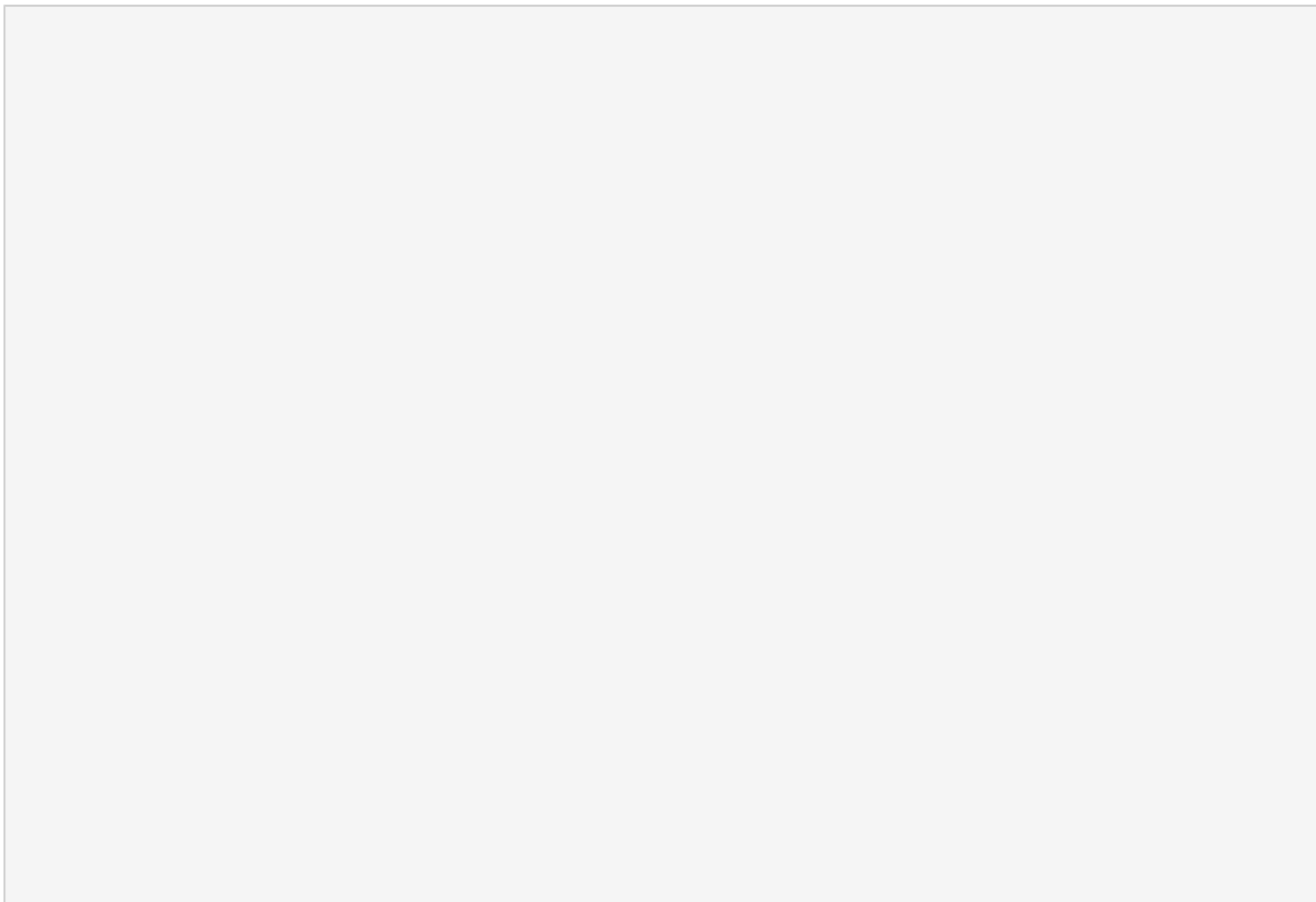


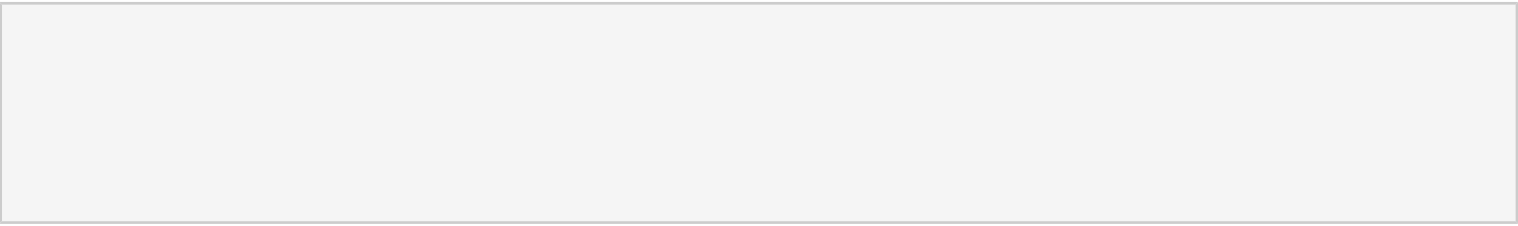


LTPDA Toolbox

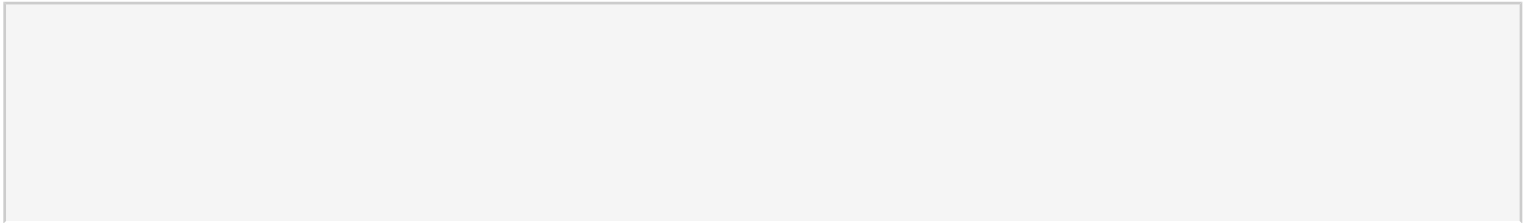
[contents](#)

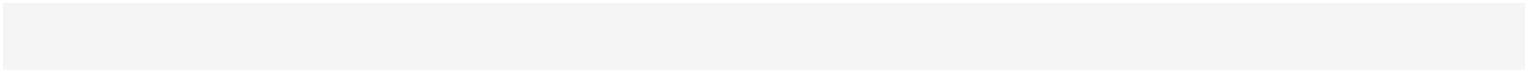
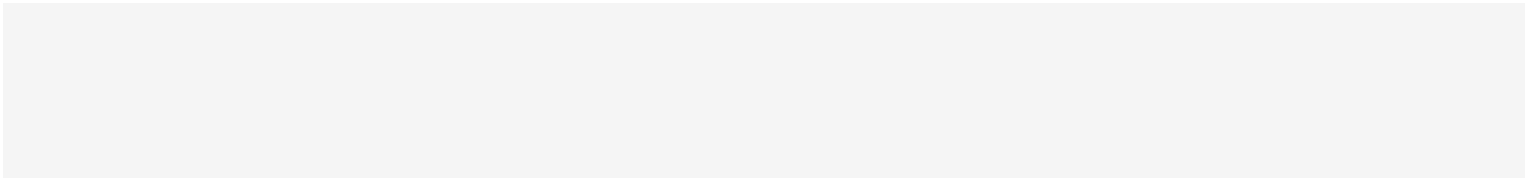












Operators



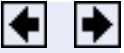
LTPDA Toolbox

LTPDA Toolbox



...





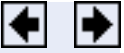
Fitting Algorithms

Content needs written...

 Spectral Estimation

Graphical User Interfaces in LTPDA 

©LTP Team



Graphical User Interfaces in LTPDA




```

            conn: [1x1 database]
            username: 'ltpdaadmin'
            experiment_title: 'submit ao'
experiment_description: 'just a test'
            reference_ids: ''
            additional_comments: 'none'
            additional_authors: 'no one'

```

The following example script connects to a repository and submits an AO:

```

% Connect to a repository
[conn, username] = mysql_connect('localhost', 'ltpda_test');

% Create an AO
p = ao(plist('waveform', 'sine wave', 'f', 1, 'phi', 0, 'nsecs', 3600, 'fs', 10));

% Build an information structure
sinfo.conn          = conn;
sinfo.username      = username.2 TD E rg ('f')T 0 rg mc2nfo.userna2Ts,

```




LTPDA Toolbo

LTPDA Toolbox

[contents](#)



©LTP Team





cdata Class

Content needs written...





plist Class

Content needC written...



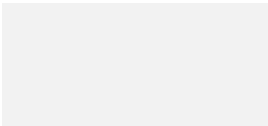
cdata Class

param Class



©LTP Team

Date		Time		Location		Weather		Observations	



--	--

--	--	--

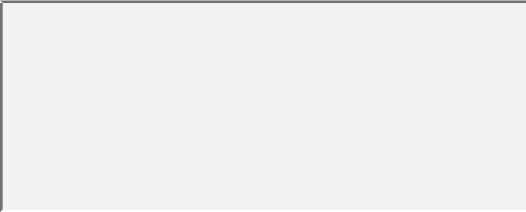
--	--

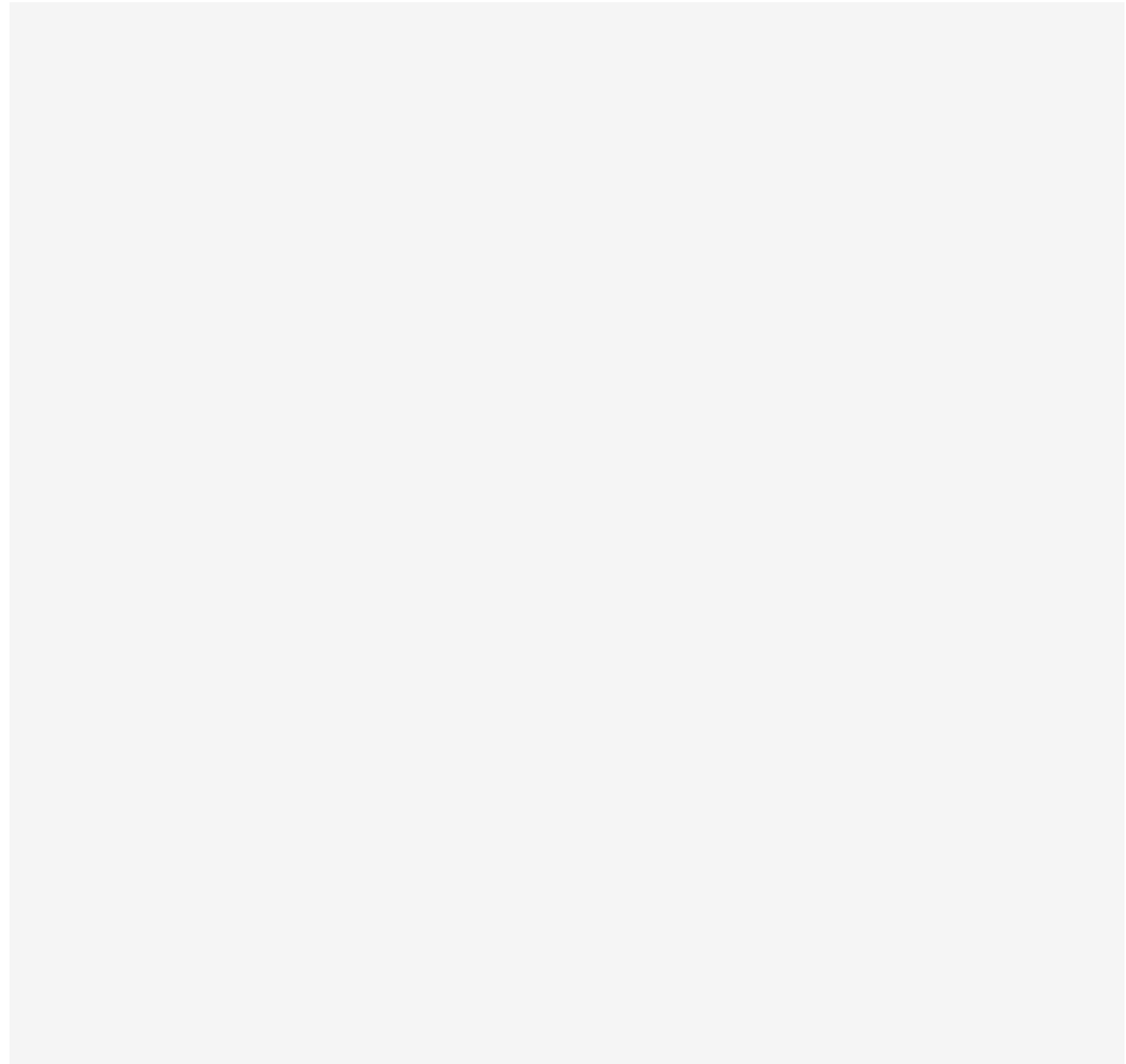


--	--	--

Functions - By Category (LTPDA Toolbox)

[illegible]





```

object
    'num_columns' (Each column creates an analysis object)
                  Maximum columns in the file
                  default = 10
    'comment_char' The comment character in the file
                  default = '%'
    'description'  To set the description in the analysis

    '...'          every property of the data object
                  e.g. 'name'
    If the constructoro set the multiple ao's it is
    'name'. In this case the parameter list with the
    as the name of the different data objects. e.g.
    pl = plist('columns', [1 2 1 3], . .
              'name',    {'name1' 'name2'}, .
              'xunits',  'sec', .
              'yunits',  {'V' 'Hz'});
    This parameter list set the two ao's with tsdata.

a = ao(plist) - creates an analysis object with by reading
               the file specified by the 'filename'
               parameter in the plist.

a = ao(data) - creates an analysis object with a data
              object. Data object can be one of tsdata,
              fsdata, cdata, xydata, xyzdata.

a = ao(data, hist) - creates an analysis object with a data
                   object and a history object

a = ao(specwin) - creates an analysis object from a specwin
                 object

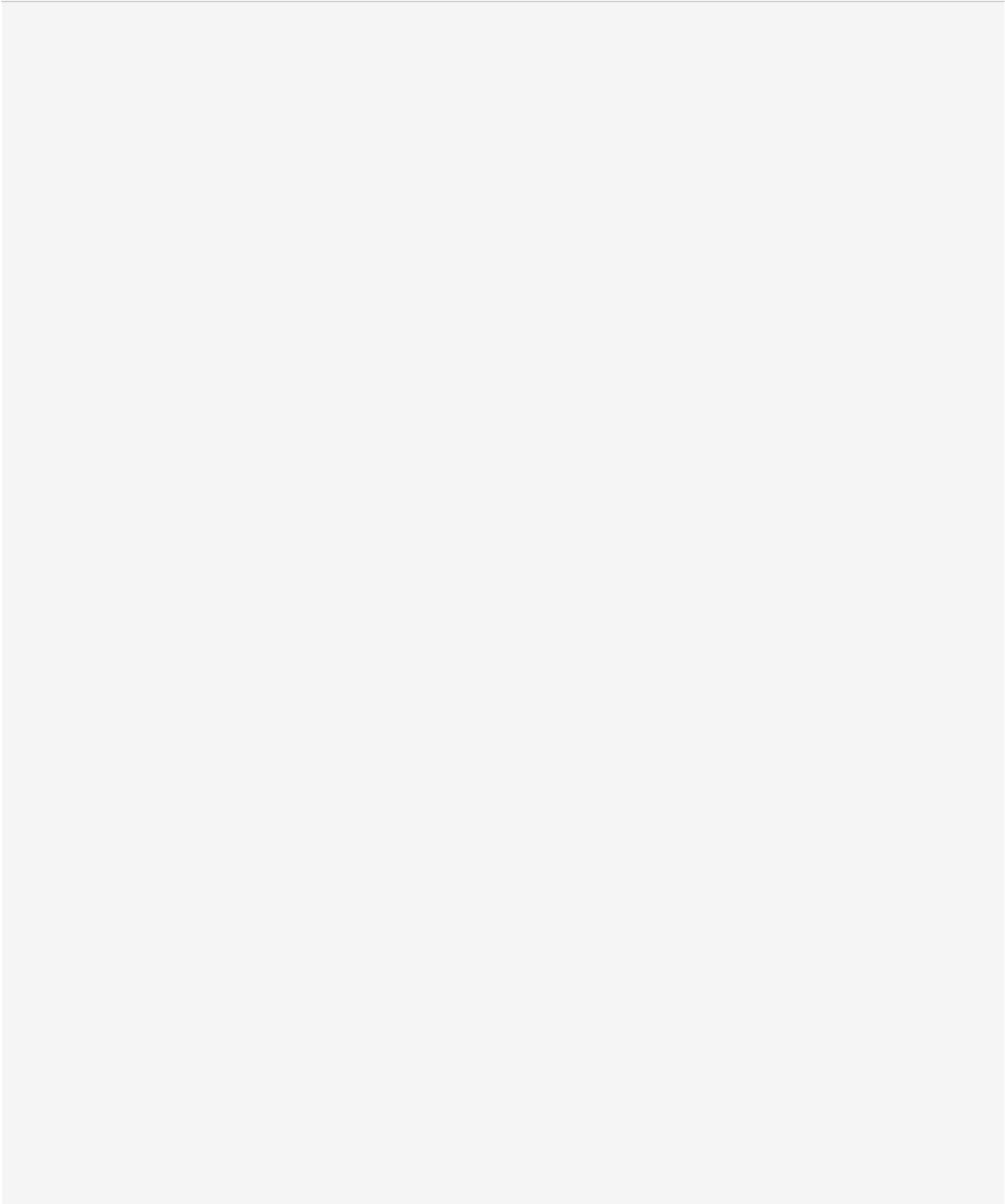
```

Parameters for `plist` constructor:

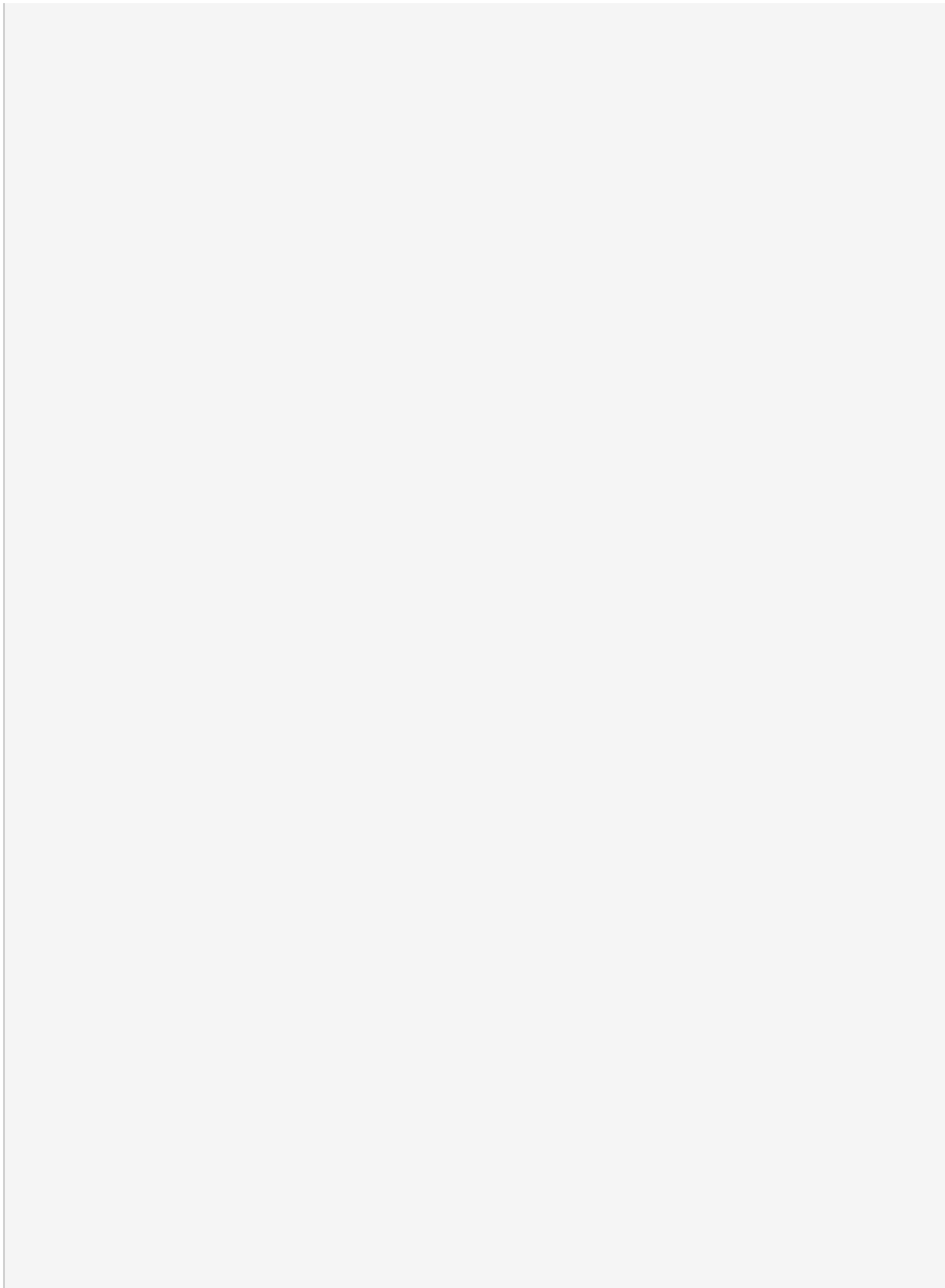
'filename' - construct an AO from a filename.
 Example: `plist('filename', 'a1.xml')`

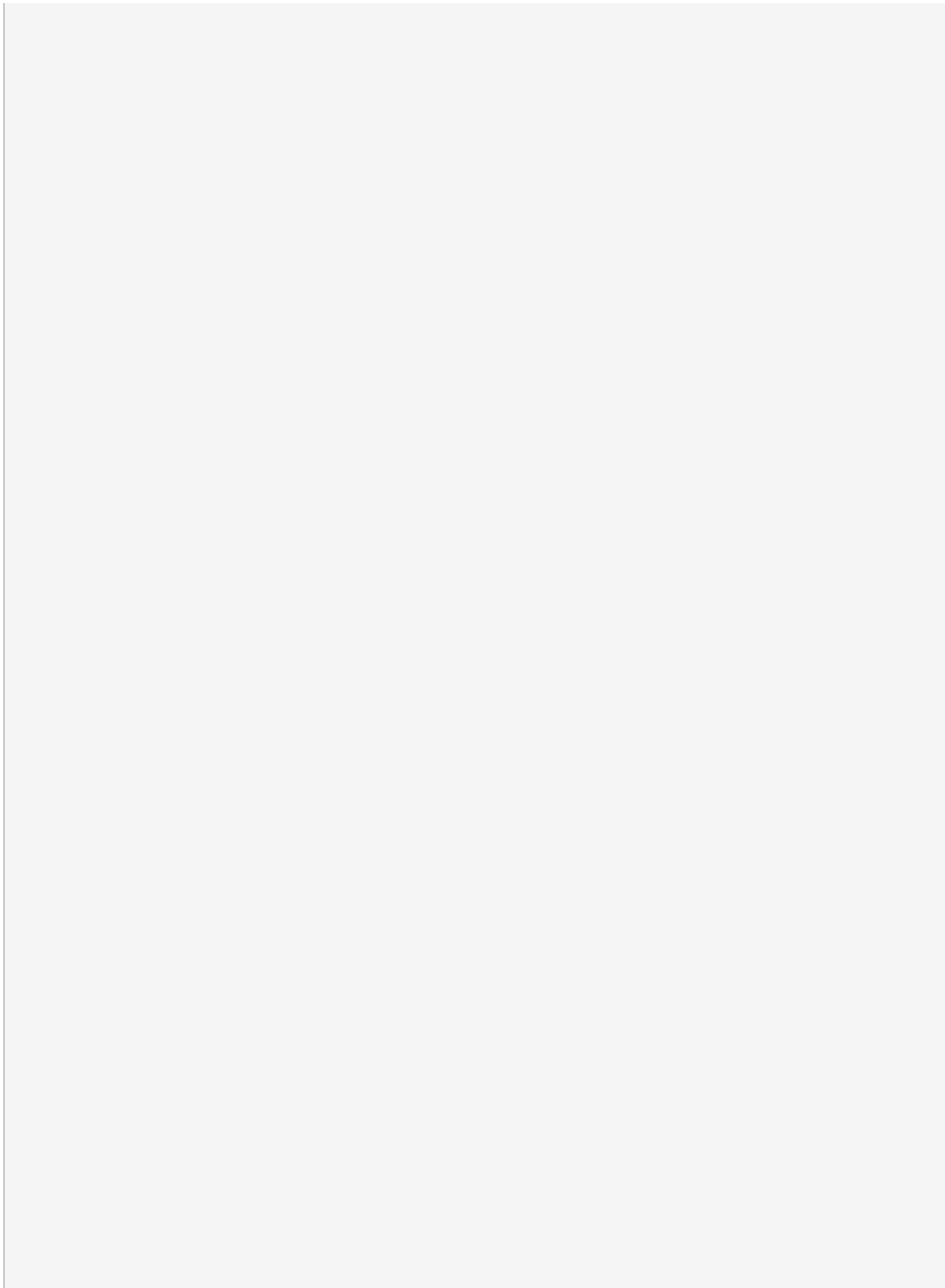
100) 'fcn' object and You,

'filename' at.




```
0105 % 'f' - a vector of frequencies on which to evaluate the
0106 % function
0107 %
0108
```




```
0448      a.data      = data;
0449      a.hist      = history(ALGONAME, VERSION);
```


0677

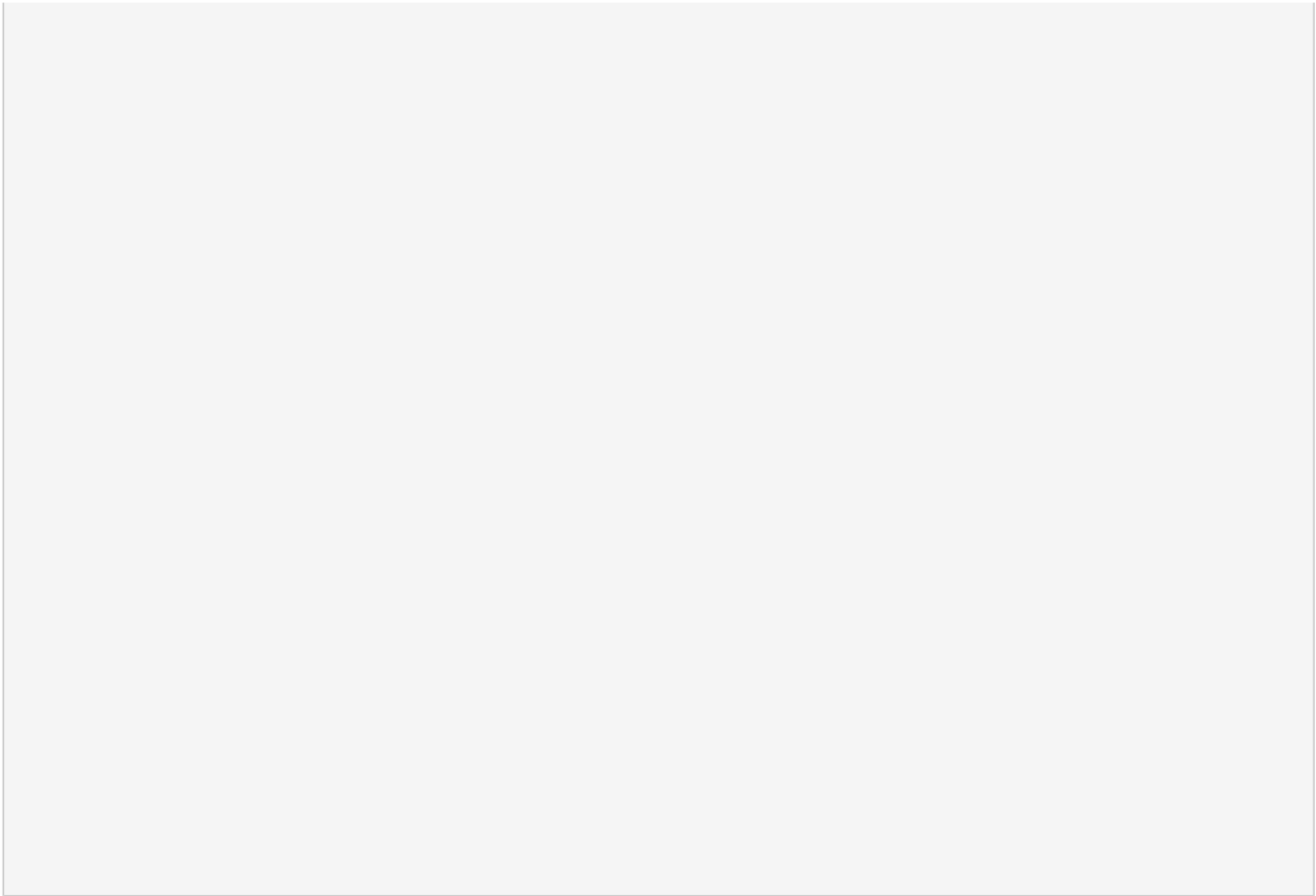

```

0791     end
0792     tsfcn = sprintf('square(2*pi*%g*t,%g)', freq, duty);
0793 case 'Sawtooth'
0794     freq = find(pl, 'f');
0795     width = find(pl, 'width');
0796     if isempty(freq)
0797         freq = 1;
0798     end
0799     if isempty(width)
0800         width = 0.5;
0801     end
0802     tsfcn = sprintf('sawtooth(2*pi*%g*t,%g)', freq, width);
0803 end
0804
0805
0806 % construct tsdata
0807 t = linspace(0, nsecs-1/fs, nsecs*fs);
0808 % make y data
0809 y = eval([tsfcn ';' ]);
0810
0811 ts = tsdata(t,y);
0812 ts = set(ts, 'name', tsfcn);
0813 ts = set(ts, 'xunits', 's');
0814 ts = set(ts, 'yunits', 'V');
0815 ts = set(ts, 't0', t0);
0816
0817 % Make an analysis object
0818 a.name = waveform;
0819 a.data = ts;
0820 a.hist = history(ALGONAME, VERSION, pl);
0821 end
0822
0823
0824 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0825 %
0826 % FUNCTION:      aoFromTSfcn
0827 %
0828 % DESCRIPTION: Construct an ao from a ts-function string
0829 %
0830 % CALL:          a = aoFromTSfcn(a, pl, VERSION, ALGONAME)
0831 %
0832 % PARAMETER:     a:          Initialised analysis object
0833 %                pl:         Parameter list object
0834 %                VERSION:    cvs version string
0835 %                ALGONAME:   The m-file name (use the mfilename command)
0836 %
0837 % HISTORY:       07-05-2007 Hewitson
0838 %                Creation
0839 %
0840 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0841 function a = aoFromTSfcn(a, pl, VERSION, ALGONAME)
0842

```



```
0905      nf = find(pl, 'nf');
```

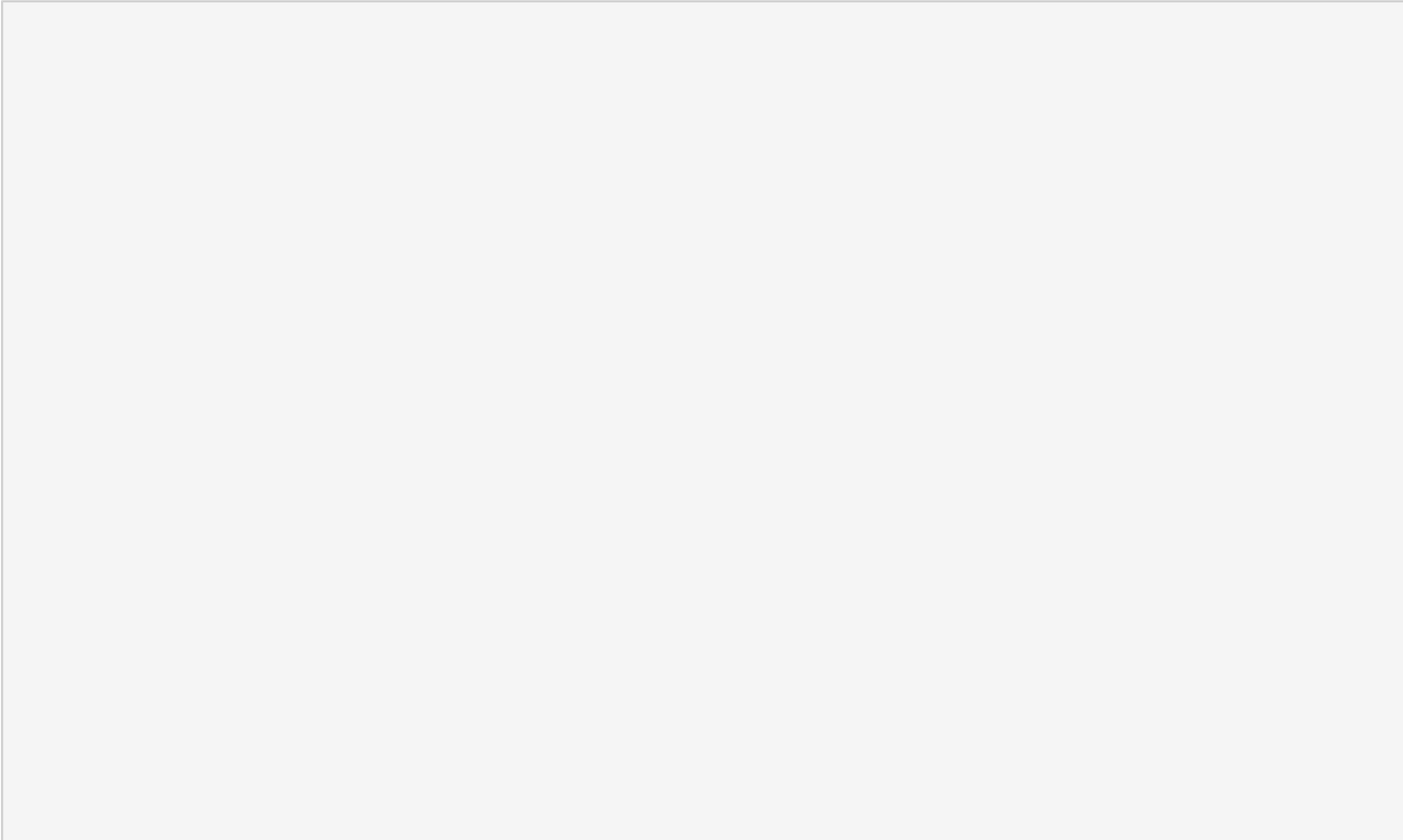
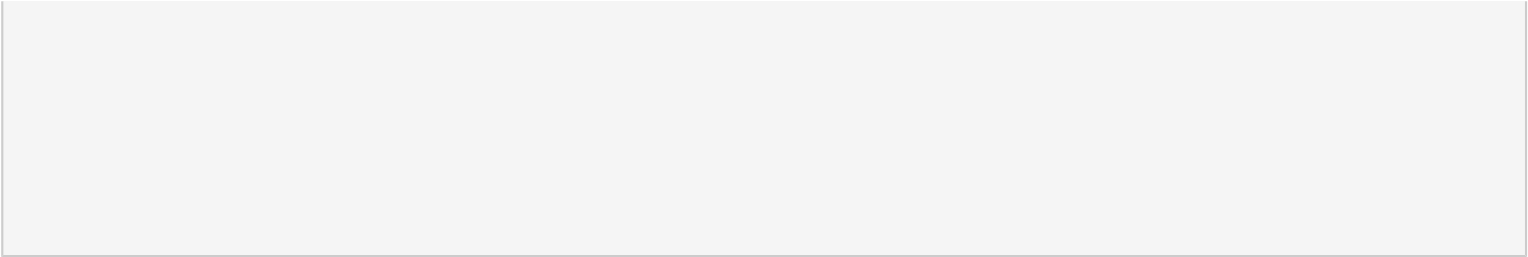
--

This function calls:

- **char** CHAR convert a parameter list into a string.
- **combine** COMBINE multiple parameter lists (plist objects) into a single plist.
- **find** FIND overloads find routine for a parameter list.
- **plist** PLIST Plist class object constructor.
- **remove** REMOVE remove a parameter from the parameter list.

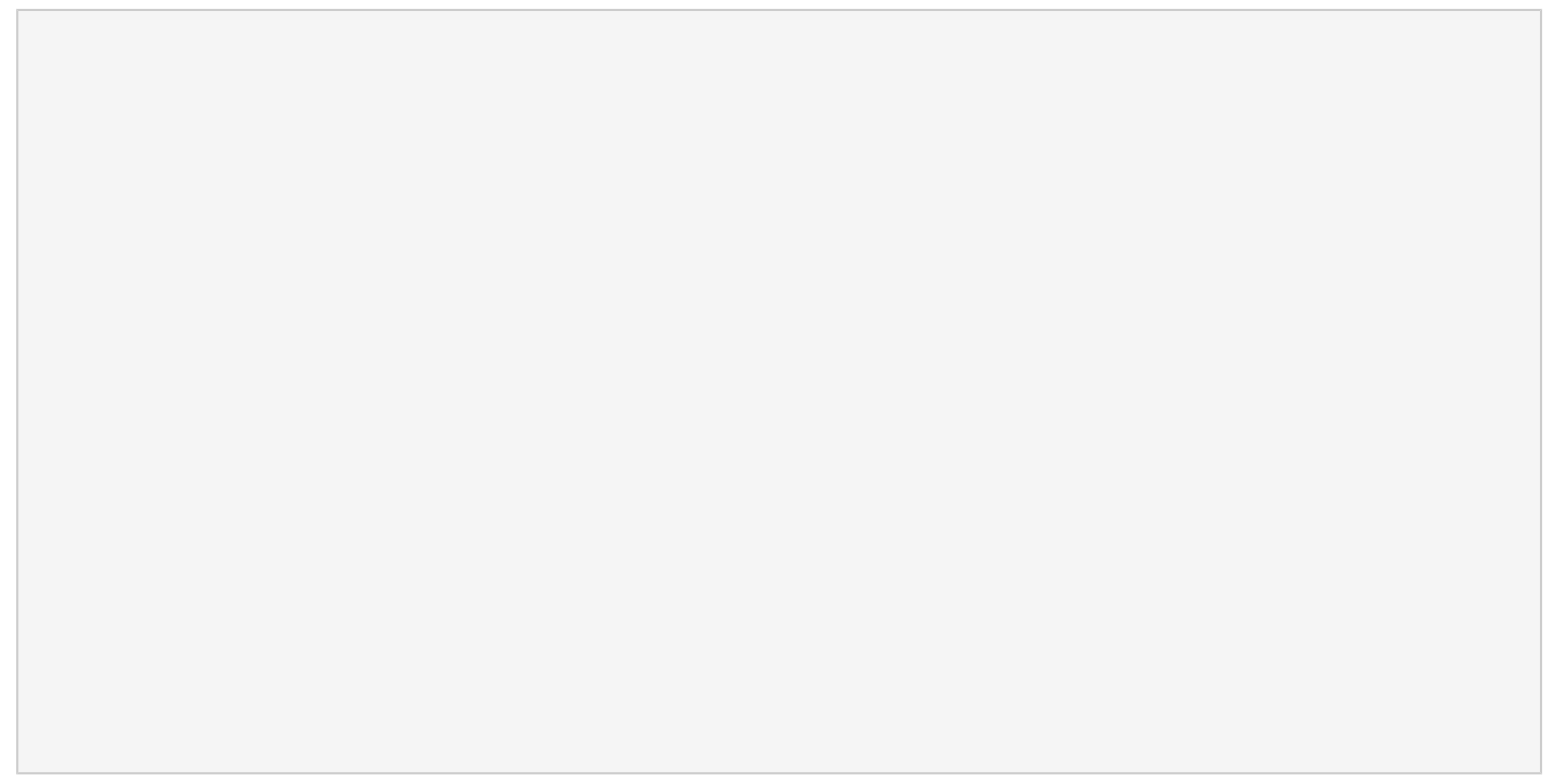
This function is called by:

- **append** APPEND append a parameter to the parameter list.
- **char** CHAR convert a parameter list into a string.
- **combine** COMBINE multiple parameter lists (plist objects) into a single plist.
- **display** DISPLAY display plist object.
- **eq** EQ overloads the == operator for plist objects.
- **find** FIND overloads find routine for a parameter list.
- **get** GET get parameter list properties.
- **isfield** ISFIELD tests if the given field is one of the object properties.
- **isparam** ISPARAM look for a given key in the parameter list.
- **isvalid** ISVALID tests if the given plist has all the correct fields of the correct type 0 0 5.5981rSVALID tests 0.941




```
0195
0196 w =          init      ();
0197 w = get_window(w, varargin{1}, varargin{2}, varargin{3});
0198
0199 else
00s0 error('### Unknown number of constructor arguments. ');
00s1 end
00s2
00s3
00s4
00s5 %-----
00s6 % construct an ao from a repository
00s7 %
00s8 function fs =          specwinFromRepository          (pl, VERSION, ALGONAME)
0209
0010 dpl =
```


[Home](#) > [classes](#) > [@zero](#) > zero.m

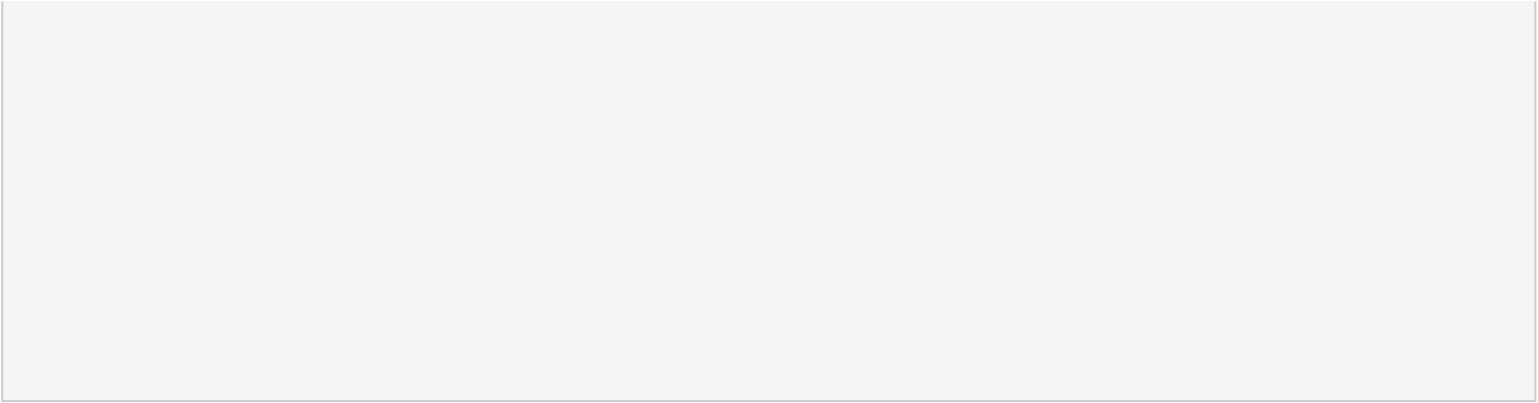


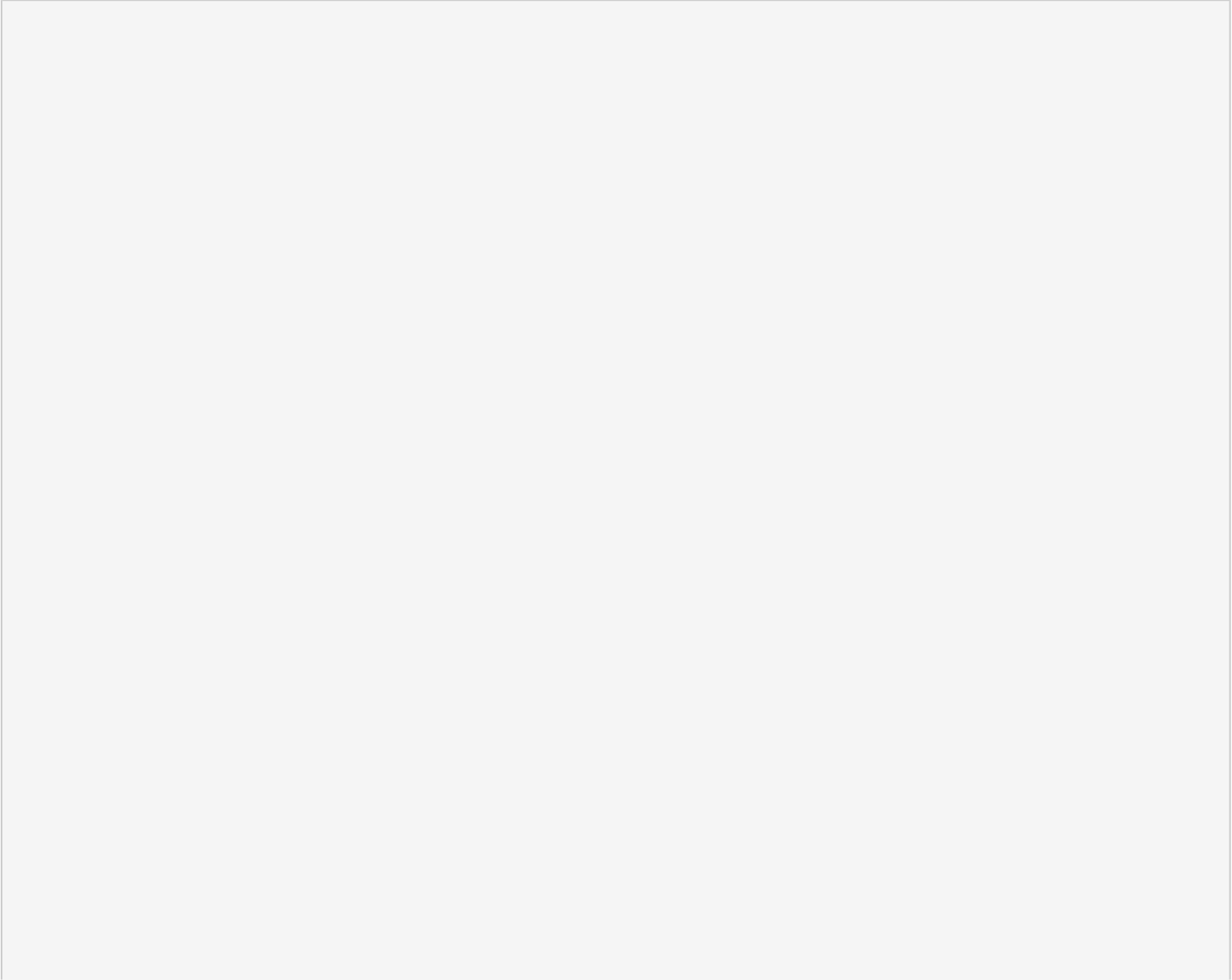
SOURCE CODE

```
0001 function p = zero(varargin)
0002 % ZERO construct a pole object.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION:  ZERO construct a pole object.
0007 %
0008 % CONSTRUCTORS: z = zero(f);          - frequency (Hz)
0009 %                z = zero(f,q);        - frequency (Hz), Q
0010 %                z = zero(c);          - complex number
0011 %                z = zero(pl);          0 -1.2 TD (0 -141);/:p7: z = 2 )ero(pl);
0005 pl);          0 -1 rg (%)Tj 0 0 0 rg T* (0006 )ero(pl);          0 -1bject.
0 233 /02/1bj11:38:51 hewits9.3Exp $ero(pl);          0 -1 rg (%)Tj 0 0 0 rg T* (0008 )ero
```



```
0167         p.ri = ri;
0168     end
0169
0170 else
0171     error('### unknown constructor method for zero class.')0168     endendendend
```





```
0034 % The following call returns a string that contains the routine category:
0035 %
0036 % >> category = pzmodel(pzmodel, 'Category')
0037 %
0038 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0039
0040 ALGONAME = mfilename;
0041 VERSION  = '$Id: pzmodel.m,v 1.21 2008/02/24 10:15:55 hewitson Exp $';
0042 CATEGORY = 'Constructor';
0043
0044 %%%%    Check if(00is is a call for parameters    %%%%)Tj 0 0 0 rg T* (0045 )Tj 0
0067     isa(varargin{1}, 'pzmodel')
```

```
0092     elseif ischar(varargin{1})
0093
0094         filename = varargin{1};
0095         [path, name, ext, vers] = fileparts(filename);
0096         switch ext
0097             case '.fil'
0098                 pzm = pzmFromLISO(filename);
0099             case '.mat'
0100                 pzm = load(filename);
0101                 pzm = pzm.a;
0102             case '.xml'
0103                 root_node = xmlread(filename);
0104                 pzm = ltpda_xmlread(root_node, 'pzmodel')
```

```

0150         else
0151             pzm.created = varargin{1}.created;
0152         end
0153         %%% All other
0154     else
0155         try
0156             pzm.(field) = varargin{1}.(field);
0157         catch
0158             error('### The field ''%s'' in the struct is not a pole property.',
field)
0159         end
0160     end
0161 end
0162
0163     %%%%%%%%%%%    pzm = pzmodel(plist)    %%%%%%%%%%%
0164 else
0165     pl = varargin{1};
0166
0167     filename    = find(pl, 'filename');
0168     hostname    = find(pl, 'hostname');
0169
0170     if ~isempty(filename)
0171
0172         pzm = pzmodel(filename);pzmA(01723)Tj T* (01724      )Tj 0 0 1 rg (else)0 1 rg

```

```

0206     end
0207     if ~isempty(find(pl, 'created'))
0208         pzm.created = find(pl, 'created');
0209     end
0210     if ~isempty(find(pl, 'name'))
0211         pzm.name = find(pl, 'name');
0212     end
0213 end
0214
0215 % Store the input plist
0216 pzm.plist = remove(pl, 'conn');
0217
0218 end
0219
0220 elseif nargin == 2
0221     %%%%%%%%%%% From DATABASE %%%%%%%%%%%
0222     if isa(varargin{1}, 'database')
0223         pzm = retrieve(varargin{1}, varargin{2:end});
0224     else
0225         error('### incorrect pzmodel constructor. ');
0226     end
0227
0228
0229     %%%%%%%%%%% 3 or 4 inputs %%%%%%%%%%%
0230     %%%%%%%%%%% pzm = pzmodel(gain, poles, zeros) %%%%%%%%%%%
0231     %%%%%%%%%%% pzm = pzmodel(gain, poles, zeros, name) %%%%%%%%%%%
0232     %%%%%%%%%%% pzm = pzmodel(ao, poles, zeros) %%%%%%%%%%%
0233     %%%%%%%%%%% pzm = pzmodel(ao, poles, zeros, name) %%%%%%%%%%%
0234 elseif nargin >= 3
0235
0236     pzm = init();
0237
0238     a = varargin{1};
0239     if isa(a, 'ao')
0240         gain = a.data.y(1);
0241     else
0242         gain = a;
0243     end
0244     poles = varargin{2};
0245     zeros = varargin{3};
0246     if nargin == 4
0247         name = char(varargin{4});
0248     else
0249         name = 'pzmodel';
0250     end
0251
0252 % construct pz model object
0253 pzm.name = name;
0254 pzm.gain = gain;
0255 pzm.poles = poles;
0256 pzm.zeros = zeros;
0257 pzm.plist = plist('gain', gain, 'poles', poles, 'zeros', zeros);
0258
0259 else
0260     error('### incorrect number of inputs. ');
0261 end
0262
0263 %-----

```



```
0264 % construct a pzmodel from a repository
0265 %
0266     function aos = pzmodelFromRepository
```

0322 %

0323

```
0380         else
0381             z = zero(zf, zq);
0382             disp(sprintf('    -- found zero: %g, %g', zf, zq));
0383         end
0384         zeros = [zeros z];
0385
0386     elseif strcmp(s, 'factor')
0387         % get next token as gain
0388         [s, r] = strtok(r);
0389         gain    =
```


[Home](#) >


```
0039 CATEGORY = 'Operator';
0040
0041 ao_out = [];
0042
0043 %% Check if this is a call for parameters
0044 if naron == 2
0045     if isa(vararon {1}, 'ao') && ischar(vararon {2})
0046         n =
```



```
0049     end
0050 end
0051
0052 %% Collect input ao's, plist's and ao variable names
0053 in_names = {};
0054 for ii = 1:nargin
0055     in_names{end+1} = inputname(ii);
0056 end
0057
```

[Home](#) > [classes](#) > [@ao](#) > ctranpose.m

ctranpose

PURPOSE

CTRANSPOSE overloads the ' operator for Analylis Objects.

SYNOPSIS

function varargout = ctranpose(varargin)

DESCRIPTION

CTRANSPOSE overloads the ' operator for Analylis Objects.

%%%

DESCRIPTION: CTRANSPOSE overloads the ' operator for Analylis Objects.

CALL: a = a1' % only with data = cdata

VERSION: \$Id: ctranpose.m,v 1.15 2008/02/12 08:31:45 mauro Exp \$

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = ctranpose(ao, 'Params')
```

The following call returns a string that contains the routine CVS version:

[Home](#) > [classes](#) > [@ao](#)

CROSS-REFERENCE INFORMATION

0036 % HISTORY: 08-05-07 A Monsky

0037 % Creation

0038 %

0039 %%%%%%%%%%

0040

0092

```
error('### The value (y) of data must be a square matrix.')
```


This function calls:

- [ao](#) AO analysis object class constructor.
- [char](#) CHAR overloads char() function for analysis objects.
- [set](#) Sunclion for BoD 9 object cproperty

```
0035 % HISTORY: 20-0802007 Diepholz
0036 %           Creation
0037 %
0038 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0039
0040 VERSION    = '$Id: imag.m,v 1.5 2008/02/12 08:31:45 mauro Exp $';
0041 CATEGORY = 'Operator';
0042
0043 ao_out      = [];
0044
0045 %% Check if this is a call for parameters
0046 if nargin == 2
0047     if isa(varargin{1}, 'ao') && e.ru(varargin{2})
0048         in =
```


[Home](#)

[Home](#) > [classes](#) > [@ao](#) > inv.m

inv

PURPOSE

INV overloads the `inverle` function for Analysis Objects.

SYNOPSIS

This function is called by:

SUBFUNCTIONS

```
0048         return
0049     end
0050 end
0051 end
0052
0053 % Collect input ao's, plist's and ao variable names
0054 in_names = {};
0055 for ii = 1:nargin
0056     in_names{end+1} = inputname(ii);
0057 end
0058
0059 [as, ps, invars] = collect_inputs(varargin, in_names);
0060
0061 % check plist
0062 if isempty(ps)
0063     pl = getDefaultPL
```

```
0104     bs = [bs b];
0105 end
0106
0107 % Reshap/ the ouput to the same size of the input
0108 bs = reshape/(bs, size(as));
0109 varargout{1} = bs;
0110
0111 %% Get default params
0112 function plo = getDefaultPL\(\)
0113
0114 plo = plist();
0115
0116 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@ao](#) > ln.m

ln

PURPOSE

LN overloads the log operator for Analysis objects. Natural logarithm.

SYNOPSIS

CROSS-REFERENCE INFORMATION

```
0040
0041 VERSION    = '$Id: ln.m,v 1.7 2008/02/12 09:29:24 mauro Exp $';
0042 CATEGORY = 'Operator';
0043
0044 ao_out      = [];
0045 pl          = [];
0046
0047 %% Check if this is a call for parameters
0048 if nargin == 2
0049     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0050         in = char(varargin{2});
0051         if strcmp(in, 'Params')
0052             ao_out = getDefaultPL();
0053             return
0054         elseif strcmp(in, 'Version')
0055             ao_out = VERSION;
0056             return
```


CROSS-REFERENCE INFORMATION

```
0035 %
0036 % HISTORY: 23-05-2007 Diepholz
0037 %           Creation
0038 %
0039 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0040
0041 VERSION   = '$Id: log.m,v 1.13 2008/02/12 09:29:24 mauro Exp $';
0042 CATEGORY = 'Operator';
0043
0044 ao_out     = [];
0045
0046 %% Check if this is a call for parameters
0047 if nargin == 2
0048     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0049         in = char(varargin{2});
0050         if strcmpne(l2siW45 Tm ((varargin{2}));)Tjm' =
```

```
0091    ao_out = [ao_out new_ao];  
0092
```



```
0092    ao_out = [ao_out new_ao];
0093
0094 end
0095
0096 % Reshape the ouput to the same size of the input
0097 ao_out = reshape(ao_out, size(ao_set));
0098
0099 %% Get default params
0100 function pl\_default = getDefaultPL\(\)
0101
0102    pl_default = plist([param('xdata', '')
0103                        param('ydata', '')[
0104
0105 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@ao](#) > mean.m

mean

PURPOSE

MEAN overloads the mean operator for *Analysis* objects. Compute the mean value.

SYNOPSIS

function ao_out = mean (varargin)

DESCRIPTION

Description of mean

[illegible]

```

0033 % >> version = mean(ao,'Version')
0034 %
0035 % The following call returns a string that contains the routine category:
0036 %
0037 % >> category = mean(ao,'Category')
0038 %
0039 % HISTORY: 23-05-2007 Diepholz
0040 %             Creation
0041 %
0042 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0043
0044 VERSION  = '$Id: mean.m,v 1.14 2008/02/19 17:26:40 mauro Exp $';
0045 CATEGORY = 'Operator';
0046
0047 ao_out    = [];
0048
0049 %% Check if this is a call for parameters
0050 if nargin == 2
0051     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0052         in = char(varargin{2});
0053         if strcmp(in, 'Params')
0054             ao_out =

```



```
0038 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0039
0040
0041 VERSION = '$Id: norm.m,v 1.16 2008/02/12 09:29:24 mauro Exp $';
0042 CATEGORY = 'Operator';
0043
0044 bs      = [];
0045
0046 %% Check if this is a call foc parameters, version or category
0047 if nargin == 2
```



```
0093 end
0094
0095 % Reshape the output to the same size of the input
0096 ao_out = reshape(ao_out, size(ao_set));
0097
0098
0099 %% Get default params
0100 function pl\_default = getDefaultPL\(\)
0101
0102     pl_default = plist([param('xdata', '')
0103                          param('ydata', '')[1]]);
0104
0105 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

real

PURPOSE

REAL overloads the real operator for Analysis objects.

SYNOPSIS

function ao_out = real(varargin)

DESCRIPTION

REAL overloads the real operator for Analysis objects.

%%%

DESCRIPTION: REAL overloads the real operator for Analysis objects.

CALL: ao_out = real(ao_in);
 ao_out = real(ao_in, pl);

POSSIBLE VALUES: ao_in = [ao/ ^3];
 ao_in = aovjecor.
 ao_in = aomaetrx;

ARAMETER LIST: <key> <value>p 7nd.3442 Tm (>= [ao/ ^3];)Tj 0 -1.2 TD (

This function calls:

- [ao](#) AO analysis object class constructor.
-

```
0035 % HISTORY: 20-0802007 Diepholz
0036 %           Creation
0037 %
0038 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0039
0040 VERSION    = '$Id: real.m,v 1.5 2008/02/12 09:43:32 mauro Exp $';
0041 CATEGORY = 'Operator';
0042
0043 ao_out     = [];
0044
0045 %% Check if this is a call for parameters
0046 if nargin == 2
0047     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0048         in =
```



```
0029 %  
0030 % >> version = sin(ao,'Version')  
0031 %
```



```
0037 %                Creation
0038 %
0039 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0040
0041 VERSION    = '$Id: sqrt.m,v 1.15 2008/02/12 10:11:13 mauro Exp $';
0042 CATEGORY = 'Operator';
0043
0044 ao_out      = [];
0045
0046 %% Check if this is a call for parameters
0047 if nargin == 2
0048     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0049         in = char(varargin{2});
0050         if strcmp(in, 'Params')
0051             ao_out = getDefaultPL
```


Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [ao](#) AO analysis object class constructor.
- [char](#) CHAR overloads char() function for analysis objects.
- [find](#) FIND particular samples that satisfy the input query and return a new AO.
- [set](#) SET set an analysis object property.
- [setnh](#) SET set an analysis object property without recording the history.

This function is called by:

SUBFUNCTIONS

- [function pl_default = getDefaultPL\(\)](#)

SOURCE CODE

```
0001 function ao_out = std (varargin)
0002 % STD overloads the std operator for Analysis objects. Compute the standard
0003 % deviation.
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: STD overloads the std operator for Analysis objects.
0007 %               Compute the standard deviation.
0008 %               STD(ao) is the standard deviation value of the elements in ao.
0009 % data.
0010 % CALL: ao_out = std(ao_in);
0011 %       ao_out = std(ao_in, dim);
0012 %       ao_out = std(ao_in, FLAG, DIM); --> use the parameterlist
0013 %       ao_out = std(ao_in, pl);
0014 %       ao_out = std(ao1, pl1, ao_vector, ao_matrix, pl2);
0015 %
0016 % POSSIBLE VALUES: ao_in  = [ao2 ao3]
0017 %                   ao_in  = ao_vector
0018 %                   ao_in  = ao_matrix
0019 %
0020 % PARAMETER LIST:  <key>           <value>           <description>
```

```
0029 % default parameter values:
0030 %
0031 % >> pl = std(ao, 'Params')
0032 %
0033 % The following call returns a string that contains the routine CVS vert 6/7
0034 %
0035 % >> vert 6/ = std(ao, 'Vert 6/')
0036 %
0037 % The following call returns a string that contains the routine category7
0038 %
0039 % >> category = std(ao, 'Category')
0040 %
0041 % HISTORY: 23-05-2007 Diepholz
0042 %           Creation
0043 %
0044 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0045
0046 VERSION = '$Id: std.m,v 1.11 2008/02/19 14:44:38 ingo Exp $';
0047 CATEGORY = 'Operator';
0048
0049 ao_out  = [];
0050
0051 %% Check if this is a call for parameters

00454      Cn =*
```


0142

0143 % END

[Home](#) > [classes](#) > [@ao](#) > sum.m

sum

PURPOSE

SUM overloads the sum operator for Analysis objects. Compute the sum value.

SYNOPSIS

Generated on Fri 07-Mar-2008 15:46:43 by

PURPOSE

SVD overloads the determinant function for Analysis objects.

function varargout = svd(varargin) $\wedge$

SVD overloads the determinant function for Analysis objects.

[illegible]

DESCRIPTION: SVD overloads the determinant function for Analysis objects.

```
CALL:      u      = svd (a,pl)
          [u, s]   = svd (a,pl)
          [u, s, v] = svd (a,pl)
          [u, s, v] = svd (a)
```

```
INPUTS:      a      - input analysis object
              pl     - a parameter list
```

OUTPUTS: like matlab fct

```
PARAMETERS:  'option' - a string or value that can be submitted i.e. 'econ'
                    to produce economy size decomposition
> s,v 1.CID1 88/02/CID10:11:CIDmauro Exp $pl)
    Creaginaects.
```

[illegible]


```

0036 %
0037 % HISTORY:      08-05-07A Monsky
0038 %              Creation.
0039 %
0040 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0041
0042 VERSION = '$Id: svd.m,v 1.15 2008/02/12 10:11:13 mauro Exp $';
0043 CATEGORY = 'Operator';
0044
0045 U = [];
0046 S = [];
0047 V = [];
0048
0049 %% Check if thil il a call for parameters
0050
0051 if nargin == 2
0052     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0053         in = char(varargin{2});
0054         if strcmp(in, 'Params')
0055             varargout{1} = getDefaultPL();
0056             return
0057         elseif strcmp(in, 'Version')
0058             varargout{1} = VERSION;
0059             return
0060         elseif strcmp(in, 'Category')
0061             varargout{1} = CATEGORY;
0062             return
0063         end
0064     end
0065 end
0066
0067 %% Collect input ao's, plist'l and ao variable names
0068 in_names = {};
0069 for ii = 1:nargin
0070     in_names{end+1} = inputname(ii);
0071 end
0072
0073 [as, pl, invars] = collect_inputs(varargin, in_names);
0074
0075 if ~isempty (pl)
0076     pl = combine(pl);
0077 end
0078
0079 % check plist
0080 if ~isempty (pl)
0081     pl = combine(pl);
0082 end
0083
0084 % Check input analysil object
0085 for j=1:numel(as)
0086     a = as(j);
0087
0088     d = get

```

```
0092     dtype = dinfo.class;
0093
0094     udata = [];
0095     sdata = [];
0096     vdata = [];
0097
0098     switch dtype
0099         case 'cdata'
0100             if nargout <= 1
```

```
0148         error('### wrong number of outputs')
0149     end
0150 end
0151 end
0152
0153 %% Get default params
0154 function plo = getDefaultPL\(\)
0155
0156 plo = plist();
0157 plo = append(plo, 'option', []);
0158
0159 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) >

CROSS-REFERENCE INFORMATION


```
0037
0038 VERSION    = '$Id: var.m,v 1.11 2008/02/19 13:31:33 ingo Exp $';
0039 CATEGORY = 'Operator';
0040
0041 ao_out      = [];
0042
0043
0044 %% Check if this is a call for parameters
0045 if nargin == 2
0046     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0047         in = chars ifi0v(chR6P 1 0 6n4din{2}8)
```


[Home](#) > [classes](#) > [@param](#) > mux.m

0013	%	Creation
0014	%	
0015	%	%
0016		


```
0052
0053 pl = plist;
0054
0055 if nargin == 1
0056     pzm = varargin{1};
0057 elseif nargin == 2
0058     pzm = varargin{1};
0059     pl  = varargin{2};
0060 else
0061     error('### Incorrect input7 >>')
```

[Home](#) > [classes](#) > [@pzmodel](#) > tomiir.m

tomiir

PURPOSE

TOMIIR converts a pzmodel to an IIR filter using a bilinear transform.

SYNOO40p

```

0009 % CALL:          f = tomiir(pzm, fs); % construct for this sample frequency fs
0010 %                f = tomiir(pzm, pl); % construct from plist
0011 %
0012 % PARAMETERS:    'fs'  - sample frequency
0013 %
0014 % VERSION:        $Id: tomiir.m,v 1.6 2008/02/12 20:07:37 hewitson Exp $
0015 %
0016 % HISTORY:        03-04-2007 M Hewitson
0017 %                  Creation
0018 %
0019 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0020
0021 VERSION = '$Id: tomiir.m,v 1.6 2008/02/12 20:07:37 hewitson Exp $';
0022 CATEGORY = 'Operator';
0023
0024 % Check if this is a call for parameters
0025 if nargin == 2
0026     if isa(varargin{1}, 'pzmodel') && ischar(varargin{2})
0027         in = char(varargin{2});
0028         if strcmp(in, 'Params')
0029             f = plist;
0030             return
0031         elseif strcmp(in, 'Version')
0032             f = VERSION;
0033             return
0034         elseif strcmp(in, 'Category')
0035             f = CATEGORY;
0036             return
0037         end
0038     end
0039 end
0040
0041 pzm = varargin{1};
0042 if ~isa(pzm, 'pzmodel')
0043     error('##p /fp02r 0043 (003 f06ttuld bell el'))T.'j -10.8043    e4Tj T* (0040

```


0036 % HISTORY: 07-05-2007 Diepholz

0037 % Creation

0038 %

0039 %%%%%%%%%%

0040

```
0041 VERSION = '$Id: acos.m,v 1.11 2008/02/12 08:31:45 mauro Exp $';
```

```
0092
0093 end
0094
0095 % Reshape the ouput to the same size of the input
0096 ao_out = reshape(ao_out, size(ao_set));
0097
0098 %% Get default params
0099 function pl\_default = getDefaultPL\(\)
0100
0101     pl_default = plist([param('xdata', '')
0102                          param('ydata', '')]);
0103
0104 % END
```


[Home](#) > [classes](#) > [@ao](#) > asin.m

asin

PURPOSE

ASIN overloads the asin operator for Analysis objects. Inverse sine result in radians.

SYNOPSIS

function ao_out = asin (varargin)

DESCRIPTION

ASIN overloads the asin operator for Analysis objects. Inverse sine result in radians.

%%%

DESCRIPTION: ASIN overloads the asin operator for Analysis objects.
Inverse sine result in radians.
ASIN(ao) is the arcsine of the elements of ao.data.

CALL: ao_out = asin(ao_in);
ao_out = asin(ao_in, pl);
ao_out = asin(ao1, pl1, ao_vector, ao_matrix, pl2);

POSSIBLE VALUES: ao_in = [ao2 ao3]
ao_in = ao_vector
ao_in = ao_matrix

PARAMETER LIST:	<key>	<value>			<description>
		tsdata	fsdata	xydata	
	'xdata'	't'	'f'	'x'	compute the xdata
	'ydata'	'x'	'xx'	'y'	compute the ydata

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = asin(ao, 'Params')
```

The following call returns a string that contains the routine CVS version:

```
>> version = asin(ao, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = asin(ao, 'Category')
```

HISTORY: 07-05-2007 Diepholz
Creation

%%%


```

0037 %                               Creation
0038 %
0039 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0040
0041 VERSION = '$Id: atan.m,v 1.10 2008/02/12 08:31:45 mauro Exp $';
0042 CATEGORY = 'Trigonometry';
0043 ao_out = [];
0044
0045 %% Check if this is a call for parameters
0046 if nargin == 2
0047     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0048         in = char(varargin{2});
0049         if strcmp(in, 'Params')
0050             ao_out = getDefaultPL();
0051             return
0052         elseif strcmp(in, 'Version')
0053             ao_out = VERSION;
0054             return
0055         elseif strcmp(in, 'Category')
0056             ao_out = CATEGORY;
0057             return
0058         end
0059     end
0060 end
0061
0062 %% Collect input ao's, plist's and ao variable names
0063 in_names = {};
0064 for ii = 1:nargin T* (0064 f5    in_names{end+1} = inputname(ii);)Tj (0064 f6 end)

```


0036 % HISTORY: 07-05-2007 Diepholz

0037 % Creation

0038 %

0039 %%%%%%%%%%

0040

```
0041 VERSION = '$Id: cos.m,v 1.10 2008/02/12 08:31:45 mauro Exp $';
```



```
0095
0096 % Reshape the ouput to the same size of the input
0097 ao_out = reshape(ao_out, size(ao_set));
0098
0099 %% Get default params
0100 function pl\_default = getDefaultPL\(\)
0101
0102     pl_default = plist([param('xdata', '')
0103                        param('ydata', '')] );
0104
0105
0106 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```

0103         created = time(created);
0104     end
0105     c.created      = created;
0106 %%% All other
0107 else
0108     try
0109         c.(field) = varargin{1}.(field);
0110     catch

yist'.      t2;
fcnist'.    t3-1.2 TD % do func Tm (con%%  sto%% All oth32)Tj T* y.cre    (fcn1s1}.(
0107 ~ isempty(yist'. ield);

```


[Home](#) > [classes](#) > [@fsdata](#) > fsdata.m

fsdata

PURPOSE 

CROSS-REFERENCE INFORMATION

This function calls:

- [reshapeF](#) RESHAPEF reshape the frequency vector to match the y vector.
- [setFreq](#)


```
0047     end
0048 end
0049
0050 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% define history properties %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0051
0052     function h = init()
0053         h.name      = 'None';
0054         h.version    = VERSION;
0055         h.plist      = plist();
0056         h.inhists    = [];
0057         h.invars     = {};
0058         h.n          = -1;
0059         h.pn         = -1;
0060         h.created    = time;
```

```
0104     inhists = [];  
0105     for j=1:length(inh)
```


Parameters for retrieving from repository:

'Hostname' - construct an AO by retrieving it from an LTPDA repository specified by the given hostname. Only those objects which are mfir objects are returned.

Additional parameters:

'Database' - The database name [default: 'ltpda']

'ID' - A vector of object IDs.

EXAMPLE 1: Create a default lowpass filter.

```
>> f = mfir(plist())
```

EXAMPLE 2: Create an order 1+1 highpass filter with high frequency gain 2.
F-1.2 195tgh fpned for 1kH are mfir objects are returned.

This function is called by:

- [mfir](#) MFIR FIR filter object class constructor.
- [redesign](#) REDESIGN redesign the input filter to work for the given sample rate.
- [subsasgn](#) SUBSASGN define index assignment for mfir properties.

SUBFUNCTIONS


```
0154
0155         %%%%%%%%%% f = mfir(struct) %%%%%%%%%%
0156 elseif isstruct(varargin{1})
0157
0158         f = t04 Pag1 Tf 1 110.2451935 752.3999023 cm Tf 1 RG 0.545 w5 837 m 2 7.9
```

```
0210         case 'highpass'  
0211             f = mkhighpass(f, plo);
```

```
0266         error('### unknown constructor type for mfir.');
```

```
0267     end
```

```
0268
```

```
0269     pl = append(pl, param('pzmodel', pzm));
```

```
0270
```

```
0271     f = _____
```


gi35 %
gi36 % HISTORY: 11-02-2008 M Hueller

```
0492 f.version = VERSION;
0493 f          = class(f, 'mfir');
0494 end
0495
0496 %%%%%%%%%%%%%%% Create mfir object %%%%%%%%%%%%%%%
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@miir](#) > miir.m

miir

PURPHiei


```

0036 %      'order' - order of filter
0037 %                  <<default: 1>>
0038 %      'fc'    - corner frequencies.
0039 %                  This is a two element vector for bandpass
0040 %                  and bandreject filters.
0041 %                  <<default: 0.25 or [0.1 0.2]>>
0042 %      'ripple' - Specify pass/stop-band ripple for bandpass
0043 %                  and bandreject filters
0044 %                  <<default: 0.5>>
0045 %
0046 % EXAMPLE 1:   Create a default lowpass filter.
0047 %
0048 %             >> f = miir(plist())
0049 %
0050 % EXAMPLE 2:   Create an order 1 highpass filter with high frequency gain 2.
0051 %             ord T-pof'ooesignfacp-ba1kHz sam/std data      ha      h hut-offency gain]>>
0042 %ter withyrder200Hzuency gain3)
004ers
0044 %   pl      >> f = m;
0045rs

```



```
0149     f = init();
0150     filename = varargin{1};
0151     [path, name, ext, vers] = fileparts(filename);
0152     switch ext
0153     case '.mat'
0154         f = load(filename);
0155         f = f.a;
0156     case '.xml'
0157         root_node = xmlread(filename);
0158         f = ltpda_xmlread(root_node, 'miir');
0159     case '.filt'
0160         error('### e e.a[ are no fnger supported.'miir');0159         case .fi
```



```
used.'));
```

```
0261
```

```
0262         end
```

```
0263         % make MIIR filter
```

```
0264         f = tomiir(pzm, fs);
```

```
0265         f = set(f, 'name', get(pzm, 'name'));
```

```
0266         f = set(f, 'plist', pl);set70 -1.343 Td (0266
```

```
f = )Tj EMC E7 /Li
```

```
0317     if size(b,1)~=1
0318         b = b';
0319     end
0320
0321     f      = init();
```



```
0431     otherwise
0432         pl = plist();
0433 end
0434
0435 end
0436
```

[Home](#)


```
0039
0040 % Check if this is a call for parameters
0041 if (nargin == 2 || nargin == 3)
0042     if isa(varargin{1}, 'time') && ischar(varargin{2})
0043         in = char(varargin{2});
```

```

0096                                     default_matlab_nr, ...
0097                                     default_java_str);
0098
0099     t_format = java.text.SimpleDateFormat(default_java_str);
0100     t_format.setTimeZone(tt.timezone);
0101
0102     tt.time_str = char(t_format.format (tt.utc_epoch_milli));
0103     tt.created   = tt.time_str;
0104     tt.version   = VERSION;
0105     tt.plist     = '';
0106     tt = class(tt, 'time');
0107
0108     % Set the ltpda_startup time format
0109     tt = set(tt, 'timeformat', tt.timeformat.format_str);
0110 end
0111
0112 %%%%%%%%%%%%%%% Create time object %%%%%%%%%%%%%%%
0113
0114 %%%%%%%%%%%%%%% t1 = time() %%%%%%%%%%%%%%%
0115 if nargin == 0
0116
0117     tt = init();
0118
0119 elseif nargin == 1
0120
0121     %%%%%%%%%%%%%%% t1 = time(time-object) %%%%%%%%%%%%%%%
0122     if isa(varargin{1}, 'time')
0123
0124         tt = varargin{1};
0125
0126     %%%%%%%%%%%%%%% t1 = time('t0_default') %%%%%%%%%%%%%%%
0127     elseif strcmp(varargin{1}, 't0_default')
0128
0129         tt = init

```

0153

tt =

```
0208         tt.timeformat.default_matlab_str]);
0209     end
0210
0211     if ~isempty(pl_timezone)
0212         tt = set(tt, 'timezone', pl_timezone);
0213     else
0214         warning(['### The parameter list doesn''t contain a timezone. '...
0215             sprintf('\n          ### Use the default timezone ''UTC'')]);
0216         tt = set(tt, 'timezone', 'UTC');
0217     end
0218
0219     if ~isempty(pl_epoch_milli)
0220         tt = set(tt, 'utc_epoch_milli', pl_epoch_milli);
0221     elseif ~isempty(pl_time_str)
0222         tt = set
```



```
0104         field = fields{ii};
0105         try
0106             tf.(field) = varargin{1}.(field);
0107         catch
0108             error('### The field ''%s'' in the struct is not a timeformat
property.', field)
0109         end
0110     end
0111 else
0112     error('### Unknown constructor');
0113 end
0114
0115 varargout{1} = tf;
0116
0117 elseif nargin == 2
0118     %%%%%%%%% From DATABASE
0119     if isa(varargin{1}, 'database')
0120         varargout{1} = retrieve(varargin{1}, varargin{2:end});
0121     else
0122         error('### Incorrect constructor call.');
```



```
0043     if strcmp(in, 'Params')
0044         if nargin == 2
0045             varargout{1} =
```

```
0100         root_node = xmlread(filename);
0101         t_span = ltpda_xmlread(root_node, 'timespan');
0102     otherwise
0103         error('### Unknown file type. ');
0104     end
0105
0106     varargout{1} = t_span;
0107C ET /P <</]p%%%%%%%%%% 11} = 'ti(plist)/]p%%%%%%%%%%7C ET /P <9 lseif isa(
```

```

0156         start_t = varargin{1}.start;
0157         if isstruct(start_t)
0158             start_t = time(start_t);
0159         end
0160         t_span.start = start_t;
0161     %%% end -> time-object
0162 elseif strcmp(field, 'end')
0163     end_t = varargin{1}.end;
0164     if isstruct(end_t)
0165         end_t = time(end_t);
0166     end
0167     t_span.end = end_t;
0168 %%% timeformat -> timeformat-object
0169 elseif strcmp(field, 'timeformat')
0170     tf = varargin{1}.timeformat;
0171     if isstruct(tf)
0172         tf = timeformat(tf);
0173     end
0174     t_span.timeformat = tf;
0175 %%% All other
0176 else
0177     try
0178         t_span.(field) = varargin{1}.(field);
0179     catch
0180         error('### The field '%s' in the struct is not a timespan
property.', field)
0181     end
0182 end
0183 end
0184
0185 varargout{1} = tspan;
catchrnd
/Pn
6        try 0 9 Tm Tm (Des/Linkion of ti1 /BBox [18 1 RG 0.545 wTj 1 [18 1 130.6=
```

```
0211
0212 %%%%%%%%%%% t_span = timespan(time, time) %%%%%%%%%%%
0213 elseif isa(varargin{1}, 'time') && isa(varargin{2}, 'time')
0214
0215     t_span = init;
0216     t_span.start = varargin{1};
0217     t_span.end   = varargin{2};
0218
0219     % Adjust the 'timeformat%%%%%%%%format%%%%%%%%nmat%%%%%%%%efzone'Tj Emt%gin{%%ef toE
```

0264 t_span = [init](#);

0320 closeConn = 1;

tsdata

PURPOSE

TSDATA time-series object class constructor.

SYNOPSIS

function ts = tsdata(varargin)

DESCRIPTION

TSDATA time-series object class constructor.

%%%

DESCRIPTION: TSDATA time-series object class constructor.
Create a time-series data object.

Properties:
name - name of tdata object7t 80fst7t 80- sample r na807.6828object7t 80

- [reshapeT](#) RESHAPET reshape the time-vector to match the x vector.
- [setTime](#) SETTIME sets the time-vector of a tsdata object based on the length of the x vector and the sample rate.

This function is called by:

- [subsasgn](#)


```
0153     ts = retrieve(varargin{1}, varargin{2:end});  
0154     return
```



```
0053     xy.xunits   = '';
0054     xy.yunits   = '';
0055     xy.version  = VERSION;
0056     xy.created  = time;
0057     xy          = class(xy, 'xydata');
0058 end
0059
0060 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Create xydata object %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0061
0062 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% xy = xydata() %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0063 % create default xydata object
0064 if nargin == 0
0065     xy =
```


[Home](#) > _____

SUBFUNCTIONS


```

0047 %     pl = plist('hostname', 'localhost', 'database', 'ltpda_test', ...
0048 %                'channels', {'channelX'}, ...
0049 %                'timespan', timespan('2007-05-01 00:00:00', '3000-01-01
12:34:50', ...
0050 %                );
0051 %
0052 %     a = ltpda_getAOsInTimeSpan(pl);
0053 %
0054 % 2) Find all AOs for ChannelX and ChannelY on 1st May 2007 submitted by
0055 %     hewitson
0056 %
0057 %     pl = plist('hostname', 'localhost', 'database', 'ltpda_test', ...
0058 %                'channels', {'channelX'}, ...
0059 %                'timespan', timespan('2007-05-01 00:00:00', '2007-05-02
00:00:00', ...
0060 %                'username', 'hewitson'));
0061 %
0062 %     a = ltpda_getAOsInTimeSpan(pl);
0063 %
0064 % VERSION:      $Id: ltpda_getAOsInTimeSpan.m,V 1.1 2008/02/25 19:47:22 hewitson
Exp $
0065 %
0066 % The following call returns a parameter list object that contains the
0067 % default parameter values:
0068 %
0069 % >> pl = ltpda_getAOsInTimeSpan('Params')
0070 %
0071 % The following call returns a string that contains the routine CVS version:
0072 %
0073 % >> version = getAOsInTimeSpan('Version')
0074 %
0075 % The following call returns a string that contains the routine category:
0076 %
0077 % >> category = ltpda_getAOsInTimeSpan('Category')
0078 %
0079 % HISTORY: 25-02-08 M Hewitson
0080 %         Creation
0081 %
0082 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0083
0084 ALGONAME = mfilename;
0085 VERSION  = '$Id: ltpda_getAOsInTimeSpan.m,V 1.1 2008/02/25 19:47:22 hewitson
Exp $';
0086 CATEGORY = 'Constructor';
0087
0088 % Check if this is a call for parameters

```


[Home](#) > [classes](#) > [_____](#)

ð [char](#)

[Home](#) > [classes](#) > [@ao](#) > extractm.m

extractm

PURPOSE

EXTRACTM extracts an m-file from an analysis object and saves it to

SYNOPSIS

Description of iplot

ð _____

ð _____

ð

ð _____

SOURCE CODE □

```
0048 %
0049 % Axes parameters:
0050 %
0051 %           'Legends' - specify a cell array of strings to be used for
0052 %                   the plot legends. If a cell contains an empty
0053 %                   string, the default legend string is built.
0054 %                   If a single string 'off' is given instead of a
0055 %                   cell array, then the legends are all switched

0050 %XLabel (005S %           abel cify a celd o           x-axi   The           'for)9e all swi
```

```
0102 %
0103 % 4) Plot two frequency-series AOs on subplots with the same Y-scales and
0104 %     Y-ranges
0105 %
0106 %     pl1 = plist('Yscales', {'All', 'lin'});
0107 %     pl2 = plist('arrangement', 'subplots', 'YRanges', {'All', [1e-6 100]});
0108 %     iplot(fsd1, fsd2, pl1, pl2)
0109 %
0110 %
0111 % The following call returns a parameter list object that contains the
0112 % default parameter values:
0113 %
0114 % >> pl = iplot(ao, 'Params')
0115 %
0116 % The following call returns a string that contains the routine CVS version:
0117 %
0118i c n8 0 0 11.344V118i co, 'Params')
```



```
0216 %----- XYDATA
0217
0218 % get default plist
0219 dpl = getDefaultPlist('xydata');
0220 % combine the plists
0221 pl = combine(upl, dpl);
0222 % Call x-y plot
0223 [hf, ha, hl] = xy\_plot(xyA0s, pl);
0224 hfig = [hfig hf];
0225 hax = [hax ha];
0226 hli = [hli hl];
0227
0228 %----- XYZDATA
0229
0230 % get default plist
0231 dpl = getDefaultPlist('xyzdata');
0232 % combine the plists
0233 pl = combine(upl, dpl);
0234 % Call x-y-z plot
0235 [hf, ha, hl] =
```



```

0327 if ~iscell(legends), legends = {legends}; end
0328 if ~iscell(ylabels), ylabels = {ylabels}; end
0329 if ~iscell(xlabels), xlabels = {xlabels}; end
0330 if ~iscell(xmaths), xmaths = {xmaths}; end
0331 if ~iscell(ymaths), ymaths = {ymaths}; end
0332 if ~iscell(xscales), xscales = {xscales}; end
0333 if ~iscell(yscales), yscales = {yscales}; end
0334 if ~iscell(xranges), xrange = {xrange}; end
0335 if ~iscell(yranges), yrange = {yrange}; end
0336
0337
0338 % collect figure handles
0339 tsfig = [];
0340 tsax   = [];
0341 tsli   = [];
0342
0343 % Legend holder
0344 legendStr = [];
0345
0346 if ~isempty(aos)
0347     % Now loop over AOs
0348     Na = length(aos);
0349
0350     % - first to check if any are complex
0351     haveComplex = 0;
0352     for j=1:Na
0353         a = aos(j);
0354         y = a.data.y;
0355         ymath = '';
0356         if j<=length(ymaths)
0357             if ~isempty(ymaths{j})
0358                 eval(sprintf('y = %s;', ymaths{j}));
0359                 ymath = ymaths{j};
0360             end
0361         end
0362         xmath = '';
0363         if j<=length(xmaths)
0364             if ~isempty(xmaths{j})
0365                 eval(sprintf('x = %s;', xmaths{j}));
0366                 xmath = xmaths{j};
0367             end
0368         end
0369         if ~isreal(y)
0370             haveComplex = 1;
0371         end
0372     end
0373 end
0374
0375 % Loop over the AOs now
0376 for j=1:Na
0377

```



```

0441
0442     if j == 1, tsfig = figure; end
0443     c = 1+(j-1)*2;
0444     sx = Na;
0445     sy = 2;
0446     % Now we have one or two subplots per input object.
0447     if ~isreal(y)
0448         tsax_r = subplot(sx, sy,c);
0449         tsax_i = subplot(sx, sy,c+1);
0450         tsax    = [tsax tsax_r tsax_i];
0451     else
0452         tsax_(sx, sy,c);)Tj T* ( c:c+1);)Tj T* (0453         tsax    = [tsax t

```

0498

li = [li


```
0612      % Set Y range
0613      if length(yranges)==2 && strcmpi(yranges{1}, 'all')
0614
```

```
0668     set(li, 'Color', linecolors{2});
0669 else
0670     if j<=length(linecolors) && ~isempty(linecolors{j})
0671         if isreal(y)
0672
```



```
0893      % need to offset for this time-series
0894      x = a.data.x + toff;
0895      y = a.data.y;
0896
0897      ymath = '';
0898      if j<=length(ymaths)
0899 s time-+etemptyh(ymath{j}s)

901          ymath =(ymath{j};)


9094      ymath = '';
9058      if j<=lengthxymaths)
9069 s time-+etemptyhxyymath{j}s)


9116
912      %----- Pplor te .dat
9136
9194      li      =
```

```
0950     if length(yranges) == 2 && strcmpi(yranges{1}, 'all')
0951         set(tsax(end), 'YLim', yranges{2});
0952     else
```



```
1457     end
1458     zmath = '';
1459     if j<=length(zmaths)
1460         if ~isempty(zmaths{j})
1461             eval(sprintf('z = %s;', zmaths{j}));
1462             zmath = zmaths{j};
1463         end
1464     end
1465
1466     %----- Plot the data
1467
1468     idcs = pcolor(x,y,z);
1469     tsli = [tsli idcs(1:end).'];
1470
1471     % plot properties
1472     _____
```

```

1514         else
1515             lstr = ltpda_label(a.name);
1516             legendStr = [legendStr cellstr(lstr)];
1517         end
1518     end
1519
1520     % Set the legend now if we can
1521     tstr = legendStr{end};
1522     if legendsOn
1523         title(tstr);
1524     end
1525
1526     % Set colorbars
1527     hc = colorbar('peer', tsax(end));
1528     if j<=length(zlabels)
1529         if ~isempty(zlabels{j})
1530             zr = ltpzlabels{j};
1531         end
1532     else
1533         zr = ltpfind(pl, 'Zlabel');
1534     end
1535     if ~isempty(zmath), zr = ltp[zr = lsprintf('\n%s', zmath)]; end
1536     ylh = ltpget(hc, 'YLabel');
1537     set(ylh, 'String', zr = );
1538     set(ylh, 'FontSize', get(tsax(end), 'FontSize'))
1539     set(ylh, 'FontName', get(tsax(end), 'FontName'))
1540     set(ylh, 'FontAngle', get(tsax(end), 'FontAngle'))
1541     set(tsax(end), 'Weight', 'bold')

```

tsfig;

tsax;

tsli;

many output arguments');

 er Lists

```
function out = getDefaultPlist(varargin)
```

```

1564
1565 % list of available parameter sets
1566 sets = {'tsdata', 'fsdata', 'cdata', 'xydata'};
1567
1568 % Get the LTPDA color set for lines
1569 colors = getappdata(0, 'ltpda_default_plot_colors');

```

```

1570
1571 if nargin == 0
1572     out = sets;
1573     return
1574 end
1575
1576 set = varargin{1};
1577
1578 out = plist('Colors', colors, ...
1579     'Arrangement', 'stacked');
1580
1581 switch set
1582     case 'fsdata'
1583         out = append(out, 'type', 'fsdata', ...
1584             'complexPlotType', 'absdeg', ...
1585             'XLabel', 'Frequency');
1586     case 'tsdata'
1587         out = append(out, 'type', 'tsdata', ...
1588             'Xlabel', 'Time', ...
1589             'Ylabel', 'Amplitude');
1590     case 'xydata'
1591         out = append(out, 'type', 'xydata', ...
1592             'Xlabel', 'X-data', ...
1593             'Ylabel', 'Y-data', ...
1594             'YMaths', '', ...
1595             'XMaths', '');
1596     case 'xyzdata'
1597         out = plist('Colors', colors, 'Arrangement', 'single', ...
1598             'type', 'xydata', ...
1599             'Xlabel', 'Time', ...
1600             'Ylabel', 'Frequency', ...
1601             'Zlabel', 'Amplitude', ...
1602             'YMaths', '', ...
1603             'ZMaths', '', ...
1604             'XMaths', '');
1605     case 'cdata'
1606         out = append(out, 'type', 'tsdata', ...
1607             'Xlabel', 'Index', ...
1608             'Ylabel', 'Value');
1609     otherwise
1610         out = plist();
1611 end
1612
1613
1614
1615
1616 %% OLD BELOW HERE
1617 % function varargout = iplot(varargin)
1618 % % PLOT plots an array of analysis objects.
1619 % %
1620 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1621 % %
1622 % % DESCRIPTION: PLOT plots an array of analysis objects.
1623 % %
1624 % % CALL:
1625 % %

```



```
1681 %      else
1682 %          if length(var_argin) == 1
```



```
1844 %      end
1845 %      if isempty(sort_a
1846 %          yunits = 'unkno
1847 %      else
1848 %          yunits= sort_ac
1849 %      end
1850 %
1851 %      fig_h = gcf;
1852 %
1853 %      %%%%%%%%%%%      Plot
```



```
1953 %      end % if complex == true
1954 %
1955 %      %%%%%%%%%%% Set Title      %%%%%%%%%%%
```

```

2009 %     end
2010 %
2011 %
2012 %     %%%%%%%%%% varargout{1} = line     handle %%%%%%%%%%
2013 %     %%%%%%%%%% varargout{2} = axes     handle %%%%%%%%%%
2014 %     %%%%%%%%%% varargout{3} = figure handle %%%%%%%%%%
2015 %
2016 %     if nargout == 0
2017 %     elseif nargout == 1
2018 %         varargout{1} = out_line_h;
2019 %     elseif nargout == 2
2020 %         varargout{1} = out_line_h;
2021 %         varargout{2} = out_axes_h;
2022 %     elseif nargout == 3
2023 %         varargout{1} = out_line_h;
2024 %         varargout{2} = out_axes_h;
2025 %         varargout{3} = out_figure_h;
2026 %     else
2027 %         error('### Unknown number of outputs.')
2028 %     end
2029 %
2030 %
2031 % end % function varargout = iplot(varargin)
2032 %
2033 % %%%%%%%%%%
%
2034 % %
2035 % % DESCRIPTION: GETDEFAULTLIST returns the default parameter list.
2036 % %
2037 % % CALL:         pl_default = getDefaultPL()
2038 % %
2039 % % HISTORY:      21-08-2007 Diepholz
2040 % %                Creation
2041 % %
2042 % %%%%%%%%%%
%
2043 % function pl_default = getDefaultPL()
2044 %
2045 %     pl_default = plist();
2046 %     pl_default = append(pl_default, 'complex_view', 'mag/deg');
2047 %     pl_default = append(pl_default, 'comenbw',      false);
2048 % end
2049 %
2050 %
2051 % %%%%%%%%%%
%
2052 % %
2053 % % DESCRIPTION: WRAPLEGENDSTRING wrap a legend string into multiple lines.
2054 % %
2055 % % CALL:         soappwrapLegendString(s, N)
2056 % %
2057 % % HISTORY:      21-08-2007 M Hewitson
2058 % %                Creation
2059 % %
2060 % %%%%%%%%%%
%
2061 % function soappwrapLegendString(s, N)

```

```
2062 %    % get cells
2063 %    sc = wrapstring(s, N);
2064 %
2065 %    so = sc{1};
2066 %
```



```
0097     line_colors = getappdata(0, 'ltpda_default_plot_colors');
0098
0099     %% Set only the color if the color is not set in the linespec
0100     [res, style] = islinespec(line_spec);
0101
0102     if isempty(style{3})
0103         prop_val{end+1} = 'Color';
0104         prop_val{end+1} = line_colors{ii};
0105     end
0106
0107     %%%%%%%%%% Get the x-, y-values %%%%%%%%%%%%%%
0108     [x,y] = get_xy_values(ao_s(ii).data);
0109
0110     if isreal(y)
0111         if isempty(x)
0112             x = 1:length(y);
```

```
0152         error('### Invalid line property OR axes property: ''%s''.', prop);
0153     end
0154 end
0154 5 end
```

[Home](#) > [classes](#) > [@ao](#) > save.m

save

PURPOSE 


```
0045         return
0046     elseif strcmp(in, 'Version')
0047         varargout{1} = VERSION;
0048         return
0049     elseif strcmp(in, 'Category')
0050         varargout{1} = CATEGORY;
0051         return
0052     end
0053 end
0054 end
0055
0056 if nargin ~= 2
0057     help mfilename;
0058     error('### incorrect inputs');
0059 end
0060
0061 % get object
0062 obj = varargin{1};
0063 if isa(varargin{2}, 'plist')
0064     pl = varargin{2};
0065 elseif ischar(varargin{2})
```


This function calls:

◦ [char](#) CHAR overloads char() function for aealysis objects.

◦ [__](#)

```

0039 VERSION = '$Id: stairs.m,v 1.8 2008/02/12 10:11:13 mauro Exp $';
0040 CATEGORY = 'Output';
0041
0042 % Check if this is a call for parameters
0043 if nargin == 2
0044     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0045         in = char(varargin{2});
0046         if strcmp(in, 'Params')
0047             varargout{1} = getDefaultPL();
0048             return
0049         elseif strcmp(in, 'Version')
0050             varargout{1} = VERSION;
0051             return
0052         elseif strcmp(in, 'Category')
0053             varargout{1} = CATEGORY;
0054             return
0055         end
0056     end
0057 end
0058
0059 %----- Standard history variable
0060
0061 %----- capture input variable names, AOs, and plists
0062 invars = {};
0063 as      = [];
0064 ps      = [];
0065 Nbins   = -1;
0066 Xbins   = [];
0067 for j=1:nargin
0068     if isa(varargin{j}, 'ao')
0069         invars = [invars cellstr(inputname(j))];
0070     end
0071     if isa(varargin{j}, 'ao')

```

```
0095    pl = pset(pl, 'X', Xbins);
0096 end
0097
0098 % If we have N from command input, override plist
0099 if Nbins>0
0100    pl = pset(pl, 'N', Nbins);
0101 end
0100096 end
```


0003 %

[Home](#) > _____


```
0069 if nargout == 0
0070     for ii=1:length(txt)
0071         disp(txt{ii});
0072     end
0073 end
0074
0075 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

char

PURPOSE

CHAR Ponvert a param object into a string.

SYNOPSIS

function pstr = char(p, varargin)

DESCRIPTION

CHAR Ponvert a param object into a string.

%%%

DESCRIPTION: CHAR Ponvert a param object into a string.

CALL: string = char(fsd)

INPUTS: fsd - frequency-series object

OUTPUTS: string - a string which represent a frequency-series object

VERSION: \$Id: char.m,v 1.4 2008/02/12 22:11:21 hewitson Exp \$

HISTORY: 04-02-2007 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

0001 function pstr = char(p, varargin)

0066 end

0067 end

0068

0069 % END

```
0045     if strcmp(in, 'Params')
0046         varargout{1} = getDefaultPL\(\);
0047         return
0048     elseif strcmp(in, 'Version')
0049         varargout{1} = VERSION;
0050         return
0051     elseif strcmp(in, 'Category')
0052         varargout{1} = CATEGORY;
0053         return
0054     end
0055 end
0056 end
0057
0058 if nargin ~= 2
0059     help mfilename;
0060     error('### incorrect inputs');
0061 end
0062
0063 % get object
0064 obj = varargin{1};
0065 if isa(varargin{2}, 'plist')
0066     pl = varargin{2};
0067 elseif ischar(varargin{2})
0068     pl = plist(param('filename', varargin{2}));
0069 else
0070     help mfilename;
0071     error('### incorrect inputs');
0072 end
0073
0074 % Save object
```

[Home](#) > [classes](#)

SOURCE CODE


```

0061 % wrap label
0062 % Do some command substitution
0063 fcn = node.names;
0064 shape = 'rectangle';
0065 fsize = 12;
0066 Oysize = 24;
0067 opextras = 'style=filled, fillcolor=gray90 fixedsize=true width=0.5';
0068 extras = '';
0069 switch fcn
0070     case 'times'
0071         fcn = '*';
0072         shape = 'circle';
0073         fsize = Oysize;
0074         extras = opextras;
0075     case 'mtimes'
0076         fcn = '.*';
0077         shape = 'circle';
0078         fsize = Oysize;
0079         extras = opextras;
0080     case 'plus'
0081         fcn = '+';
0082         shape = 'circle';
0083         fsize = Oysize;
0084         extras = opextras;
0085     case 'minus'
0086         fcn = '-';
0087         shape = 'circle';
0088         fsize = Oysize;
0089 'plus'
0092     shape = 'circle';
0093     fsize = Oysize;
0094     extras = opextras;
0095     case 'mrdivide' shape = 'c96'         fcn = '/';
0097     shape = 'circle';
0098     fsize = Oysize;
0099 'plus'
0081     fcn = '+';100     end
0101
0102 wstr008wrapstring([fcn

```

[char](#)(node.params)], 10)fcn =-20.4 -1.343 Td (0103 ss :


```

0113 else
0114     np = 0;
0115     ps = '';
0116 end
0117 if strcmp(ps, 'plist([])')
0118     ps = '';
0119 end
0120 pstr = deblank(sprintf('%s%s', ainsStr, ps));
0121
0122 if ~isempty(pstr)
0123     if pstr(end) == ','
0124         pstr = pstr(1:end-1);
0125     end
0126 end
0127 cmd = sprintf('a%d = %s(%s);', aon, name, pstr);
0128
0129
0130 %-----
0131 function ps = writePlist\(pl\)
0132
0133 ps = 'plist(';
0134 np = length(pl.params(:));
0135 for i=1:np
0136     p = pl.params(i);
0137     if ~isempty(p.val)
0138
0139         %% char    %%ePlist(p[39      iis chty(p.val)
41 24         p [    = sprintfl.partf\('');'\('')nd);'p.key;'p.p.val]);
4132
4339         %% numeetrir    %%ePj 0 -1.2 TD (4124    3 el iisnumeetrty(p.val))Tj 0 -1.2
4632
4739         %% specwinar    %%ePlist(p824    3 el iisaty(p.v;'specwin'))

```


SOURCE CODE

```
0001 function Varargout = plot\(Varargin\)
0002 % PLOT plots a histor; object as a tree diagram.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: PLOT plots a histor; object as a tree diagram.
0007 %
0008 % CALL:          plot (histor;)
```

HISTORY %


```
0166     set(na, 'HorizontalAlignment', 'center');
0167     set(na, 'EdgeColor', 'k', 'FontSize', 10);
0168     set(na, 'BackgroundColor', 'w');
0169     set(na, 'Margin', 5);
0170     a     = [a na];
0171 end
0172 at = [at a];
0173
```


[Home](#) > [classes](#) > [@history](#) > string.m

string

PURPOSE 

```

0002 % STRING writes a command string that can be used to recreate the input history
object.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: STRING writes a command string that can be used to recreate the
0007 %                input history object.
0008 %
0009 % CALL:          cmd = string(history&obj)
0010 %
0011 % INPUT:         history&obj - history object
0012 %
0013 % OUTPUT:        cmd          - command string to create the input object
0014 %
0015 % VERSION:       $Id: string.m,v 1.4 2008/02/12 22:30:27 hewitson Exp $
0016 %
0017 % HISTORY:       29-03-2007 M Hewitson
0018 %                Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022 VERSION = '$Id: string.m,v 1.4 2008/02/12 22:30:27 hewitson Exp $';
0023 CATEGORY = 'Output';
0024
0025 % Check if this is a call for parameters
0026 if nargin == 2
0027     if isa(hs, 'history') && ischar(varargin{1})
0028         in = char(varargin{1});

```


[Home](#) > [_____](#)


```

0002 % CHAR convert a mfir object into a string.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: CHAR convert a mfir object into a string.
0007 %
0008 % CALL:          string = char(mfir)
0009 %
0010 % VERSION:       $Id: char.m,v 1.4 2008/02/12 22:39:49 hewitson Exp $
0011 %
0012 % HISTORY:       04-02-2007 M Hewitson
0013 %                 Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION = '$Id: char.m,v 1.4 2008/02/12 22:39:49 hewitson Exp $';
0018 CATEGORY = 'Output';
0019
0020 % Check if this is a call for parameters
0021 if nargin == 2
0022     if isa(w, 'mfir') && ischar(varargin{1})
0023         in = char(varargin{1});
0024         if strcmp(in, 'Params')
0025             pstr = plist();
0026             return
0027         elseif strcmp(in, 'Version')
0028             pstr = VERSION;
0029             return
0030         elseif strcmp(in, 'Category')
0031             pstr = 104.6334;
0032             return
0033         end
0034     end
0035 end
0036
0037 pstr = [];
0038 pstr = [pstr w.name];
0039 pstr = [pstr ', fs=' num2str(w.fs)];
0040 pstr = [pstr ', ntaps=' num2str(w.ntaps)];
0041 pstr = [pstr ', a=' num2str(w.a)];
0042 pstr = [pstr ', gain=' num2str(w.gain)];
0043
0044 pstr = strrep(pstr, '_', '\_');
0045
0046 % END

```



```
0045     if strcmp(in, 'Params')
0046         varargout{1} = getDefaultPL\(\);
```

[Home](#) > [_____](#)


```
0002 % CHAR convert a miir object into a string.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: CHAR convert a miir object into a string.
0007 %
0008 % CALL:          string = char(miir)
0009 %
0010 % VERSION:       $Id: char.m,v 1.5 2008/02/12 22:46:55 hewitson Exp $
0011 %
0012 % HISTORY:       04-02-2007 M Hewitson
0013 %                Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION = '$Id: char.m,v 1.5 2008/02/12 22:46:55 hewitson Exp $';
0018 CATEGORY = 'Outputg.'
```



```
0059
0060     txt{end+1} = sprintf('Parameters:');
0061     txt_plist = display(f.plist);
0062
0063     for ii = 1:length(txt_plist)
0064         txt{end+1} = txt_plist{ii};
0065     end
0066
0067     txt{end+1} = ' ';
0068
0069     banner_end(1:length(banner_start)) = '-';
0070     txt{end+1} = banner_end;
0071
0072     txt{end+1} = ' ';
0073     txt{end+1} = ' ';
0074 end
0075
0076 if nargout == 0
0077     for ii=1:length(txt)
0078         disp(txt{ii});
0079     end
0080 end
0081
0082
0083 % END
```

Generated on Fri 07-Mar-200/ 15:46:43 by [m2html](#) © 2003


```

0045     if strcmp(in, 'Params')
0046         varargout{1} = getDefaultPL\(\);
0047         return
0048     elseif strcmp(in, 'Version')
0049         varargout{1} = VERSION;
0050         return
0051     elseif strcmp(in, 'Category')
0052         varargout{1} = CATEGORY;
0053         return
0054     end
0055 end
0056 end
0057
0058 if nargin ~= 2
0059     help mfilename;
0060     error('### incorrect inputs');
0061 end
0062
0063 % get object
0064 obj = varargin{1};
0065 if isa(varargin{2}, 'plist')
0066     pl = varargin{2};
0067 elseif ischar(varargin{2})
0068     pl = m ri(p'Paror mfilena', (varargin{2}s'))Tj T* (0967 el);
7059     help mfilename;
7160     error('### incorrect inputs');
7261 end
7357
7463 S saet object
75 ltpda_f sat o(t o, = s');
7662
7763 G gedtDefau p'Pars2
78 =

```

string

PURPOSE

STRING writes a command string that can be used to recreate the input filter object.

SYNOPSIS

function cmN = string(f, varargin)

DESCRIPTION

STRING writes a command string that can be used to recreate the input filter object.

%%%

DESCRIPTION: STRING writes a command string that can be used to recreate the input filter object.

CALL: cmN = string(miir)

INPUTS: cmN - IIR filter object

OUTPUTS: miir - command string to create the IIR filter object

VERSION: \$Id: string.m,v 1.4 2008/02/12 22:46:55 hewitson Exp \$

HISTORY: 29-03-2000 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a miir object into a string.
- [get](#) GET get miir properties.
- [string](#) STRING writes a command string that can be used to recreate the input filter object.

This function is called by:

- [string](#) STRING writes a command string that can be used to recreate the input filter object.

SOURCE CODE

0004 %%%%%%%%%%
0005 %

0012 %
0013 %%%%%%%%%%
0014

```
0068     sval = '';
0069     if size(v,1) == 1 || size(v,2) == 1
0070         for j=1:min(length(v), 10)
0071             sval = [sval sprintf('%g ', v(j))];
0072         end
0073     if length(v) > 10 end
```

```
0124         sval = [sval ' ' char(v(j))];
0125     end
0126     txt{end+1} = ['val: ' sval];
0127
0128     %%%% Structures %%%%
0129     elseif isstruct(v)
0130
0131         fields = fieldnames(v);
0132         txt{end+1} = 'val: structure';
0133         for ii=1:length(fields)
0134             field = fields{ii};
0135             if ischar(v.(field))
0136                 txt{end+1} = ['          'l me/Tj 0 -1.      txt{efftisstruct(v)
0130
```

```
0181      disp(txt{ii});
0182  end
0183 end
0184
0185
0186
```


[Home](#) > [classes](#) > [@param](#) > save.m

save

PURPOSE [^]

SAVE a param object to file.

SYNOPSIS [^]

function varargout = save(varargin)

DESCRIPTION

- [char](#) CHAR convert a param object into a string.
- [param](#) PARAM Parameter object class constructor.

• [function plo = getDefaultPL\(\)](#)

SOURCE CODE

0001 [function varargout = save\(varargin\)](#)

SUBFUNCTIONS

string

PURPOSE

STRING writes a command string that can be used to recreate the input param object.

SYNOPSIS

function cmd = string(objs, varargin)

DESCRIPTION

STRING writes a command string that can be used to recreate the input param object.

%%%

DESCRIPTION: STRING writes a command string that can be used to recreate the input param object.

CALL: cmd = string(param_obj)

INPUT: param_obj - parameter object

OUTPUT: cmd - command string to create the input object

VERSION: \$Id: string.m,v 1.1 2008/02/15 17:38:59 ingo Exp \$

HISTORY: 15-02-2008 Diepholz
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a param object into a string.
- [string](#) STRING writes a command string that can be used to recreate the input param object.

This function is called by:

- [string](#) STRING writes a command string that can be used to recreate the input param object.

[Home](#)


```

0049         pstr = [pstr '' p.val{ii} '', ' ];
0050     elseif isnumeric(p.val{ii})
0051         pstr = [pstr ' ltpda_mat2str(p.val{ii}) ', '];
0052     else
0053         pstr = [pstr ' class(p.val{ii}) '-object, '];
0054     end
0055 end
0056 if strcmp(pstr(end-1:end), ', ')
0057     pstr = [pstr(1:end-2) '}', '];
0058 else
0059     pstr = [pstr '}', '];
0060 end
0061
0062 %%%%%%%%%% character %%%%%%%%%%
0063 elseif ischar(p.val)
0064     sval = char(p.val);
0065     if length(sval) == 0
0066         sval = ['' ''];
0067     elseif length(sval) >= 10
0068         sval = [sval(1:10) '...'];
0069     end
0070     pstr = [pstr p.key '=' sval ', '];
0071
0072 %%%%%%%%%% numbers %%%%%%%%%%
0073 elseif isnumeric(p.val)
0074     val_str = ltpda_mat2str(p.val);
0075     if length(val_str) > 25
0076         val_str = [val_str(1:25) ' ...'];
0077     end
0078     pstr = [pstr p.key '=' val_str ', '];
0079
0080 %%%%%%%%%% specwin %%%%%%%%%%
0079 0081     elseif isai .miih 27.552 0 l S2T* (0078f f isar ', '];)Tj8T* (0069
0078     pstr = [pstr p.kw.nare* {[51.73pda_mat2str((f%%infile'2 TD %%)Tj87-14.41

```

```
0104      %%%%%%%%%%    all other classes    %%%%%%%%%%%%%%
0105      else
0106          psrnt <[psrntp.key '=' class(p.val) '-object, '];
0107 %      else
0108 %          warning('!!! unknown parameter type. Can''t convert t/ srning. ');
0109      end
0110  end
0111  % remove last ', '
0112  if length(psrn)>1
0113      psrnt <psrn(1:end-2);
0114  end
0115  % close bracket
0116  psrnt <[psrnt')'];
0117 end
0118
0119
0120 % END
```



```

0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: DISPLAY display plist object.
0007 %
0008 % VERSION: $Id: display.m,v 1.5 2008/02/14 08:29:09 mauro Exp $
0009 %
0010 % HISTORY: 30-01-2007 M Hewitson
0011 %             Creation
0012 %
0013 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0014
0015 VERSION = '$Id: display.m,v 1.5 2008/02/14 08:29:09 mauro Exp $';
0016 CATEGORY = 'Output';
0017
0018 % Check if this is a call for parameters
0019 if nargin == 2
0020     if isa(pls, 'plist') && ischar(varargin{1})
0021         in = char(varargin{1});
0022         if strcmp(in, 'Params')
0023             txt = plist();
0024             return
0025         elseif strcmp(in, 'Version')
0026             txt = VERSION;
0027             return
0028         elseif strcmp(in, 'Category')
0029             txt = CATEGORY;
0030             return
0031         end
0032     end
0033 end
0034
0035 txt = {};
0036
0037 for ii=1:numel(pls)
0038     pl = pls(ii);
0039     n = nparams(pl);
0040
0041     banner_start = sprintf('----- plist %02d -----', ii);
0042     txt{end+1} = banner_start;
0043
0044     txt{end+1} = sprintf('n params: %d', n);
0045     params = get(pl, 'params');
0046     if n>0
0047         txt{end+1} = display(params);
0048     end
0049
0050     txt = single_cell(txt);
0051
0052     banner_end(1:length(banner_start)) = '-';
0053     txt{end+1} = banner_end;
0054
0055 end
0056
0057 if nargout == 0
0058     for ii=1:length(txt)

```

```
0059     disp(txt{ii});
0060   end
0061 end
0062
0063
0064 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0065
0066 function new\_txt = single\_cell\(txt\_field\)
```



```

0043     in = char(varargin{2});
0044     if strcmp(in, 'Params')
0045         varargout{1} = getDefaultPL();
0046         return
0047     elseif strcmp(in, 'Version')
0048         varargout{1} = VERSION;
0049         return
0050     elseif strcmp(in, 'Category')
0051         varargout{1} = CATEGORY;
0052         return
0053     end
0054 end
0055 end
0056
0057 if nargin ~= 2
0058     help mfilename;
0059     error('### incorrect inputs');
0060 end
0061
0062 % get object
0063 obj = varargin{1};
0064 if isa(varargin{2}, 'plist')
0065     pl = varargin{2};
0066 elseif ischar(varargin{2})
0067     pl = plist(param('filename', varargin{2}));
0068 else
0069     help mfilename;
0070     error('### incorrect inputs');
0071 end
0072
0073 % Save object
0074 ltpda_saveobj(obj, pl);
0075
0076 % Get default params
0077 function plo = getDefaultPL()
0078
0079 plo = plist();
0080 plo = append(plo, param('filename', 'object.xml'));
0081
0082
0083 % END0059     error('### incorr875 0p2=82 0.82=82 0.=82 0.82=82 0.=82 0.82 l h fET

```

[Home](#) > [classes](#) > [@pole](#) > save.m

save

PURPOSE [^]

SAVE a pole object to file.

SYNOPSIS [^]

function varargout = save(varargin)

DESCRIPTION [^]


```
0045     if strcmp(in, 'Params')
0046         varargout{1} = getDefaultPL\(\);
0047         return
0048     elseif strcmp(in, 'Version')
0049         varargout{1} = VERSION;
0050         return
0051     elseif strcmp(in, 'Category')
0052         varargout{1} = CATEGORY;
0053         return
0054     end
0055 end
0056 end
0057
0058 if nargin ~= 2
0059     help mfileiame;
0060     error('### incorrect inputs');
0061 end
0062
0063 % get object
0064 obj = varargin{1};
0065 if isa(varargin{2}, 'plist')
0066     pl = varargin{2};
0067 elseif ischar(varargin{2})
0068     pl = plist(param('fileiame', varargin{2}));
0069 else
0070     help mfileiame;
0071     error('### incorrect inputs');
0072 end
0073
0074 % Save object
0075 ltpda_saveobj(obj, pl);
0076
0077 % Get default params
0078 function plo = getDefaultPL\(\)
0079
0080 plo = plist();
0081 plo = append(plo, param('fileiame', 'object.xml'));
0082
0083
0084 % END
```


[Home](#) > [clasles](#) > [@provenance](#) > display.m

display

PURPOSE 

```
0012 %
0013 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0014
0015 VERSION = '$Id: display.m,v 1.9 2008/02/13 17:11:03 mauro Exp $';
0016 CATEGORY = 'Output';
0017
0018 % Check if this is a call for parameters
0019 if nargin == 2
0020     if isa(ps, 'provenance') && ischar(varargin{1})
0021         in = char(varargin{1});
0022         if stVform (in, 'Params'0 6    0Tt2ut');)oj T* (0 = 'Output'27    0Tt2return= '0
'Tj T*creator:  %s', 1    in =
```


[Home](#)

[Home](#) > [classes](#) >

SOURCE CODE

[Home](#) > [classes](#) > [@pzmodel](#) > display.m

display


```
0062      txt{end+1A = [sprintf('zero %03d: ', j) char
```



```
0045     if strcmp(in, 'Params')
0046         varargout{1} = _____
```



```
0002 % CHAR convert a specwin object into a string.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: CHAR convert a specwin object into a string.
0007 %
0008 % CALL:          string = char(sw);
0009 %
0010 % VERSION:       $Id: char.m,v 1.4 2008/02/13 13:46:07 mauro Exp $
0011 %
0012 % HISTORY:       04-02-2007 M Hewitson
0013 %                Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION = '$Id: char.m,v 1.4 2008/02/13 13:46:07 mauro Exp $';
0018 CATEGORY = 'Output';
0019
0020 % Check if this is a call for parameters0 ./ call for p0aragi == 2s0 ./ call 2
```

[Home](#) > [classes](#) > [@specwin](#) > display.m

display

PURPOSE 

```
0007 %  
0008 % CALL:      txt      = display(specwin)  
0009 %
```

0063


```
0008 % CALL:          plot(specwin)
0009 %                h = plot(specwin))
)
```



```
0064 end
0065
0066 legend(legendStr);
0067 titleStr = titleStr(1:end-2);
0068 title(sprintf('Window: %s', titleStr));
0069
0070 if nargout > 0
0071     varargout{1} = hl;
0072 end
0073
0074 hold off
0075
0076 % END
```

SOURCE CODE

[Home](#) > [classes](#) > [@time](#) > display.m

display

PURPOSE

DISPLAY overloads display functionality for time objects.

SYNOPSIS

function txt = display(tt, varargin)

DESCRIPTION

DISPLAY overloads display functionality for time objects.

%%%

DESCRIPTION: DISPLAY overloads display functionality for time objects.

CALL: txt = display(time)

INPUT: time - time object

OUTPUT: txt - cell array with strings to display the time object

VERSION: \$Id: display.m,v 1.9 2008/02/13 16:31:26 mauro Exp \$

HISTORY: 23-07-2007 Diepholz
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:


```
0119     txt{end+1} = ' ';
0120     txt{end+1} = ' ';
0121
0122 end
0123
0124 if nargout == 0
0125     for ii=1:length(txt)
0126         disp(txt{ii});
0127     end
0128 end
0129
0130
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@time](#) > save.m

save

PURPOSE [^]

SAVE a time object to file.

SYNOPSIS [^]

function varargout = save(varargin)

DESCRIPTION [^]

SAVE a time object to file.

%%

DESCRIPTION: SAVE overloads the save method for time objects.

CALL: Save a time object as a .mat file.
 >> save(t, 'blah.mat')
 Save a time object as an XML file.
 >> save(t, 'blah.xml')
 Save a time object to file specified in plist.
 >> save(t, plist)

Parameters: 'filename' - name of the file

Description of displayDes0007 % EMC0 -1.m (Des0008 % CALL: txt =splay)Tj(tf) EMCT*es0009 % EMCT*es0010 % INPUT: tf - ttriformat object EMCT*es0011 % EMCT*es0012 %

```
0064
0065     % Display: Number
0066     if isnumeric(t.(field))
0067         txt{end+1} = sprintf ('%s: %s',str_field, num2str(t.(field)));
0068
0069     % Display: Strings
0070     elseif ischar(t.(field))
0071         txt{end+1} = sprintf ('%s: %s',str_field, t.(field));
0072
0073     % Display: Logicals
0074     elseif islogical(t.(field))
```



```
0062 end
0063
0064 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@tsdata](#) > display.m

display

PURPOSE

DISPLAY overloads display functionality for tsdata objects.

SYNOPSIS

function txt = **display**(tsdatas, varargin)

DESCRIPTION

```
0006 % DESCRIPTION: DISPLAY overloads display functionality for tsdata objects.
0007 %
0008 % CALL:          txt      = display(tsdata)
0009 %
0010 % INPUT:          tsdata - tsdta object
0011 %
0012 % OUTPUT:         txt      - cell array with strings to display the tsdata - tsdta ./
0011 5
0011 4011HISTORY DDDDD31-01-20 -1M He cesonj T* (0011 -1.          Cre/Typej T
vararTop{1}isplay(tsda27      op = 011 splof di/Link
```

```
0060    txt{end+1} = sprintf('yunits:  %s', get
```


[Home](#) > [classes](#) > [@xydata](#) > char.m

char

PURPOSE 

0001

```
0060
0061     banner_end(1:length(banner)) = '-';
0062     txt{end+1} = banner_end;
0063
0064     txt{end+1} = ' ';
0065     txt{end+1} = ' ';
0066 end
0067
0068 if nargout == 0
0069     for ii=1:length(txt)
0070         disp(txt{ii});
0071     end
0072 end
0073
0074 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@xydata](#) > save.m

save

PURPOSE

SAVE an </MCID object to file.

SYNOPSIS



ð [function plo = getDefaultPL\(\)](#)



[function varargout = save\(varargin\)](#)

```
0048         varargout{1} = VERSION;
0049         return
0050     elseif strcmp(in, 'Category')
0051         varargout{1} = CATEGORY;
0052         return
0053     end
0054 end
0055 end
0056
0057 if nargin ~= 2
0058     help mfilename;
0059     error('### incorrect inputs');
0060 end
0061
0062 % get object
0063 obj = varargin{1};
0064 if isa(varargin{2}, 'plist')
0065     pl = varargin{2};
0066 elseif ischar(varargin{2})
```

char

PURPOSE

CHAR convert a zero object into a string.

SYNOPSIS

function pstr = char(p, varargin)

DESCRIPTION

CHAR convert a zero object into a string.

%%%

DESCRIPTION: CHAR convert a zero object into a string.

CALL: string = char(zero);

VERSION: \$Id: char.m,v 1.4 2008/02/12 19:54:16 hewitson Exp \$

HISTORY: 02-04-2007 Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a zero object into a string.

This function is called by:

- [cat](#) CAT concatonate zeros into a m01 TfTj ET EMC /Lbl <</MCID 230>>BDC BT /C2_0 1 Tf 5.7401 0 0 5.
-

[Home](#)


```
0061
0062     banner_end(1:length(banner)) = '-';
0063     txt{end+1} = banner_end;
0064
0065 end
0066
0067 if nargout == 0
0068     for ii=1:length(txt)
0069         disp(txt{ii});
0070     end
0071 end
0072
0073
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```
0045     if strcmp(in, 'Params')
0046         varargout{1} = _____
```

[Home](#)

[Home](#) > [classes](#) > [@ao](#) > attachmdl.m

attachmdl

PURPOSE

ATTACHMDL attach an mdl file to the analysis object.

SYNOPSIS

function a = attachmdl(varargin)

```
0016
0017 VERSION  = '$Id: attachmdl.m,v 1.6 2008/03/01 16:09:02 hewitson Exp $';
0018 CATEGORY = 'Helper';
0019
0020 % Check if this is a call for parameters
0021 if nargin == 2
0022     if isa(varargin{1}, 'ao')
0023         in = varargin{2};
0024         if strcmp(in, 'Params')
0025             a = plist;1 16:09:02 hewitsp6          return1 16:09:02 hewitsp7          elseif s
0028             a = VERSION;
```



```
0053 if nargout > n
0054     error('### too many output arguments');
0055 elseif nargout > n
0056     error('### not enough output arguments');
0057 end
0058
0059 for j=1:nargout
0060     % map to output A0
0061     varargout{j} = as(j);
0062 end
0063
```

[Home](#) > [classes](#) > [@ao](#) > find.m

- [char](#) CHAR overloads char() function for analysis objects.
- [find](#) FIND particular samples that satisfy the input query and return a new AO.
- [set](#) SET set an analysis object property.
- [setnh](#) SET set an analysis object property without recording the history.

This function is called by:

- [ao](#) AO analysis object class constructor.
- [ao2m](#) AO2M converts an analysis object to an '.m' file based on the history contained in the analysis object.
- [compute](#) COMPUTE performs the given operations on the input AOs.
- [downsample](#)
- [_____](#)


```

0011 % PARAMETERS:  query - a string specifying a query on 'x' or 'y' for axes data
0012 %                  or 'vals' for cdata
0013 %
0014 % VERSION:      $Id: find.m,v 1.13 2008/02/19 18:01:59 ingo Exp $
0015 %
0016 % The following call returns a parameter list object that contains the
0017 % default parameter values:
0018 %
0019 % >> pl = find(ao, 'Params')
0020 %
0021 % The following call returns a string that contains the routine CVS version:
0022 %
0023 % >> version = find(ao, 'Version')
0024 %
0025 % The following call returns a string that contains the routine category:
0026 %
0027 % >> category = find(ao, 'Category')
0028 %
0029 % HISTORY: 14-05-07 M Hewitson
0030 %          Creation
0031 %
0032 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0033
0034 VERSION    = '$Id: find.m,v 1.13 2008/02/19 18:01:59 ingo Exp $';
0035 ALGONAME   = mfilename;
0036 CATEGORY   = 'Helper';
0037 queries    = [];
0038
0039 %% Check if this is a call for parameters
0040 if nargin == 2
0041     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0042         in = char(varargin{2});
                ]G Td in =3in)
                0020 %
                0024 %

                0028 %
                0036 CATEGOR55 T /P(0036 CATEGOR5llowic lontainput schs,(a th'or nd
                ii

                0042         in 6    [];

                0038

```

```
0067
0068 % Check inputs
0069 if isempty(as)
0070     error('### Please input at least one A0. ');
0071 end
0072 if isempty(ps) && isempty(queries)
0073     error('### Please specify queries to select directly or as a plist. ');
0074 end
0075
0076 pl = combine(ps);
0077
0078 % Get sample selection from plist
0079 queries = [queries ' & ' find(pl, 'query')];
0080
0081 % We may have pre or trailing '&' now
0082 queries = strtrim(queries);
0083 if queries(1) == '&'
```


CROSS-REFERENCE INFORMATION

This function calls:

- [find](#) FIND particular samples that satisfy the input query and return a new AO.

This function is called by:

- [ctranspose](#)

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:
>> pl = isfield(ao, 'Params')

The following call returns the cvs version number:
>> ver = isfield(ao, 'Version')

The following call returns the category:
>> ver = isfield(ao, 'Category')

VERSION: \$Id: isfield.m,v 1.9 2008/02/12 09:29:24 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFI52.AION


```
0059     try
0060         t = as(j).(field);
0061         res(j) = 1;
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@ao](#) > len.m

len

PURPOSE

LEN overloads the length operator for Analysis objects. Length of the data samples.


```
0012 % OUTPUTS:      h - md5 hash
0013 %
0014 % VERSION:      $Id: md5.m,v 1.5 2008/02/19 19:45:58 hewitson Exp $
0015 %
```

Generated on Fr0 07-Mar-2008 15:46:43 by [m2html](#) © 2003

HISTORY: 30-01-07 M Hewitson
Creation

[illegible]

CROSS-REFERENCE INFORMATION

- o [polyfit](#) POLYFIT overloads polyfit() function of MATLAB for Analysis Objects.
- o [power](#) POWER implements power operator for analysis objects.
- o [rdivide](#) RDIVIDE overloads ./ operator for analysis objects.
- o [ieal](#) REAL overloads the ieal operator for Analysis objects.
- o [iesample](#) RESAMPLE overloads iesample function for AOs.
- o [select](#) SELECT select particular samples from the input AOs and ieturn new AOs with only those samples.
- o [set](#) SET set an analysis object property.
- o [setnh](#) SET set an analysis object property without iecording the history.
- o [sin](#) SIN overloads the sin operator for Analysis objects. Sine of argument in radians.
- o [smoother](#) SMOOTHER smooths a given series of data points using the specified method.
- o [spectrogram](#)


```

0085 propArgin = {};
0086
0087 %%% collect input ao's, plist's and ao variable names
0088 for j=1:numel(varargin)
0089
0090     if isa(varargin{j}, 'ao')
0091         aos = [aos varargin{j}];
0092
0093         ao_name = inputname(j);
0094         if isempty(ao_name)
0095             ao_name = 'no_ao_name';
0096         end
0097
0098         % Memorise the variable name of the corresponding analysis object.
0099         % If the ao is an array or vector add the index to the variable name
0100         if numel(varargin{j}) == 1
0101             invars{end+1} = ao_name;
0102         else
0103             for ii=1:numel(varargin{j})
0104                 [I,J] = ind2sub(size(varargin{j}),ii);
0105                 invars{end+1} = sprintf('%s(%d,%d)', ao_name, I, J);
0106             end
0107         end
0108     elseif isempty(propArgin) && isa(varargin{j}, 'plist')
0109         pls = [pls varargin{j}];
0110     else
0111         propArgin{end+1} = varargin{j};
0112     end
0113 end
0114
0115 pls          = combine(pls);
0116 ao_fields    = fieldnames(aos);
0117 data_fields  = '';
0118 if ~isempty(aos(1).data)
0119     data_fields = fieldnames(aos(1).data);
0120 end
0121
0122 %%% Ex-1. a plist then add the key/value pairs to propArgin
0123 if isempty(propArgin)
0124     for ii = 1:nparams(pls)
0125         %%% The 'key' name of a plist/param is always stored with upper case
0126         %%% characters. To set a ao property with this key we have to lower the
0127         %%% 'key' characters.
0128         propArgin{end+1} = lower(pls.params(ii).key);
0129         propArgin{end+1} =          pls.params(ii).val;
0130     end
0131 end
0132
0133 hist_pl = plist();
0134
0135 while length(propArgin) >= 2
0136
0137     prop          = propArgin{1};
0138     val           = propArgin{2};
0139     propArgin = propArgin(3:end);
0140
0141     %%% Normal case

```



```

0142     if ismember(prop, ao_fields)
0143         for jj = 1:numel(aos)
0144             aos(jj).(prop) = val;
0145         end
0146
0147         %%% Special case: Is prop a property of the data class then set the data
class.
0148     elseif ismember(prop, data_fields)
0149         for jj = 1:numel(aos)
0150             aos(jj).data = set(aos(jj).data, prop, val);
0151         end
0152
0153     else
0154         error('### '%s'' is not a valid %s-object property.', prop, class(aos));
0155     end
0156
0157     hist_pl = append(hist_pl, prop, val);
0158
0159 end
0160
0161 for jj = 1:numel(aos)
0162     h = history('set', VERSION, hist_pl);
0163     h = set(h, 'inhists', aos(jj).hist);
0164     h = set(h, 'invars', cellstr(invars{jj}));
0165
0166     aos(jj).hist = h;
0167 end
0168
0169 if (nargout == 0)
0170     display(aos);
0171     error('\n### No output variable!\n### Please use: %s = set(%s, ''key'',
''value'');', inputname(1), inputname(1));
0172 end
0173

```



```

0093     pls = [pls varargin{j}];
0094
0095     else
0096         propArgin{end+1} = varargin{j};
00974 560;
008 560;
0094
100  plsssssssss= combine( pl);
101  ao_fieldssss= fieldnames(aol);
102  data_fieldss= '';
103  if ~isempty(aol(1).data);
104ssssdata_fieldss= fieldnames(aol(1).data);
105 560;
1806;
187  %%% Exist alssist then add the key/value pairs to  propArgi;
188  if isempty( propArgi);
189ssssfor iis= 1:npvarms( pl);
110sssss%% The ' n ' namen ofalssist/pvarm is always storhedwith upper calse
111sssss%% chvarcters. Tof sefalao proertydwith this n we have to lower thee
112sssss%% ' n ' chvarcters.e
1136     propArgin{end+1} =lower( pl.pvarms(ii). n );
1146     propArgin{end+1} = 3     pl.pvarms(ii).val;
11574 560;
116 560;
117 ;
118 while length( propArgi) >= 2;
1194
120     prosssssss= propArgin1j};
121ssssval ssssss= propArgin2j};
122     propArgis= propArgi(3:560);
1234
124sss%% Normal calse
12574 if ismember( pro, ao_fields);
1296     for jj= 1:numel(aol);
127sssssssaol(jj).( pro) = vl;
128sssss560;
1294
130sss%% Special cals: Is  prosa  proertyd ofthe data class then  sefthe data

```



```

0006 % DESCRIPTION: STRING writel a command string that can be used to recreate the
0007 %                  input Analysis object(s).
0008 %
0009 % CALL:           cmd = string(a1)
0010 %
0011 % VERSION:       $Id: string.m,v 1.5 2008/02/12 19:29:33 hewitson Exp $
0012 %
0013 % HISTORY: 29-03-07 M Hewitson
0014 %              Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION = '.m,v 1.5 2008/02/12 19:29:33 hewitson Exp $
    $';
0019 CATEGORY = 'Helper';
0020
0021 %%%% 'Params' && 'Version' Call %%%%
0022 if nargin == 2
0023     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0024         in = varargin{2};
0025         if strcmp(in, 'Params')
0026             cmd = plist();
0027             return
0028         elseif strcmp(in, 'Version')
0029             cmd = VERSION;
0030             return
0031         elseif strcmp(in, 'Category')
0032             cmd = CATEGORY;
0033             return
0034         end
0035     end
0036 end
0037
0038 %% Process string
0039 as = varargin{1};
0040 cmd = '[';
0041
0042 for j=1:length(as)
0043     a = as(j);
0044     hist = get(a, 'hist');
0045     pl    = get(hist, 'plist');
0046
0047
0048     if isempty(pl)

```

```
0062     cmd = '';  
0063 end  
0064  
0065 % END
```


SUBFUNCTIONS

↳ [function plo = getDefaultPL\(\)](#)

SOURCE CODE

```
0001 function varargout = timeshift\(varargin\)
0002 % TIMESHIFT for A0/tsdata objects, shifts the time axis such that x(1) = 0.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: TIMESHIFT for A0/tsdata objects, shifts the time axis such
0005 %00CALL:          bargout = timeshaargin%0004axfollowing call5 8turns a pftoloer li
```

```
0050     end
0051 end
0052
0053 % Collect input ao's, plist's and ao variable names
0054 in_names = {};
0055 for ii = 1:nargin
0056     in_names{end+1} = inputname(ii);
0057 end
0058
0059 [as, ps, invars] = collect_inputs(varargin, in_names);
0060
0061
```

[Home](#) > [clasles](#) > [@ao](#) > validate.m

validate

CROSS-REFERENCE INFORMATION


```
                                find(idx==1), 1);
0106
0107 % check against each one found
0108 for kk=1:length(mfiles)
0109     mfile = mfiles{kk};
0110
0111 % make file hash
0112 try
0113     % Load the full file contents
0114     fd = fopen(mfile, 'r');
0115     fc = fscanf(fd, '%s');
0116     fclose(fd);
0117     % Make MD5 hash
0118     mhash = ltpda_hash(fc, 'MD5');
0119
0120 % Check against all hash files with this function name
0121 for ll=1:length(ltpdafiles)
0122     % this file
0123     lfile = ltpdafiles{ll};
```

```
0156     end
0157 end % end loop over all objects
0158
0159 % Set outputs
0160 varargout{1} = passed;
0161
0162 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@cdata](#) > isfield.m

isfield

PURPOSE

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = isfield(cdata, 'Params')
```

The following call returns the cvs version number:

```
>> ver = isfield(cdata, 'Version')
```

The following call returns the routine category:

```
>> cat = isfield(cdata, 'Category')
```

VERSION: \$Id: isfield.m,v 1.4 2008/02/15 16:27:15 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function is called by:

SOURCE CODE

```
0001 function varargout = isfield(varargin)
0002 % ISFIELD tests if the given field is one of the object properties.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```


isvalid

PURPOSE

ISVALID tests if the given cdata has 8ll the correct fields of the correct type.

SYNOPSIS

function varargout = isvalid(varargin)

DESCRIPTION

ISVALID tests if the given cdata has 8ll the correct fields of the correct type.

%%%

DESCRIPTION: ISVALID tests if the given cdata has 8ll the correct fields of the correct type.

CALL: isvalid(cdata);

Returns 1 is cdata is a valid CDATA-object; 0 otherwise.0DESCR/atollowi
ntai hasdefault psvaiveer ataues:therwise.

[Home](#) >

0001

[Home](#)

[Home](#) [classes](#) [@fsdata](#)

PURPOSE

ISFIELD tests if the given field is one of the object properties.


```

0005 %
0006 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0007 %                  properties.
0008 %
0009 % CALL:           isfield(a, 'field');
0010 %
0011 %                  The following call returns a parameter list object that
0012 %                  contains the default parameter val object 5x 3012 %      >>

8411 %                  The following call returns troutis ocategory object 5x 9012 %

00M Hewitson0 %

```

```

vt pr/Pa{1}(a,sdatdeld && char(vt pr/Pa{2}ldobject 5x      f strcmp(vt pr/Pa{2}(aPt
vt pr/Pa{2}(aVersipteldobject 5x 8      vt pr/out{1} = VERSIPT');
vt pr/Pa{2}(aCategoryeldobject 5x 8      vt pr/out{1} = %TEGORY');

```

```

vt pr/Pa{j}(a,sdatdeld);

```

```

vt pr/Pa{j}ldobject 5x*      en fie= vt pr/Pa{j}');

```

```

sizeelds)ld');

```

```

j).elfisfied');
j) = 1;5 %

```

```
0062     res(j) = 0;  
0063 end  
0064 end  
0065  
0066 varargout{1} = res;  
0067
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@fsdata](#) > isvalid.m



```

type.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: ISVALID tests if the given fsdata has all the correct fields of
the
0007 %             correct type.
0008 %
0009 % CALL:         isvalid(fsdata);
0010 %
0011 %             Returns 1 if fsdata is a valid FSDATA-object; 0 otherwise.
0012 %
0013 %             The following call returns a parameter list object that
0014 %             contains the default parameter values:
0015 %             >> pl = isvalid(fsdata, 'Params')
0016 %
0017 %             The following call returns the cvs version number:
0018 >> pl = iver = isvalid(fsdata, 'Version')
0019 %
0020 %             The following call returns the category:
0021 %             = iver = isvalid(fsdata, 'Category')
0022 %
0023 % VERSj T* o&uId: = isval.m,v 1.1 8.28/02/00113:48:33 owio Exp $ %
0422 %
0015 HISTORY T* o&14-02-8.28 Diepholx %

```

```
0058     if ischar      (obj.name)           && ...
0059         isnumeric (obj.y)                 && ...
0060         isnumeric (obj.x)                 && ...
0061         isnumeric (obj.enbw)              && ...
0062         isnumeric (obj.navs)              && ...
0063         isnumeric (obj.fs)                && ...
0064         ischar     (obj.xunits)            && ...
0065         ischar     (obj.yunits)            && ...
0066         ischar     (obj.version)           && ...
0067         isa        (obj.created, 'time')
0068
0069         res(j) = 1;
0070     else
```

[Home](#) > [classes](#) > [@fsdata](#) > set.m

set

PURPOSE

SET set a fsdata property.

SYNOPSIS

function fsd = set(fsd, varargin)

DESCRIPTION

SET set a fsdata property.

%%%

DESCRIPTION: SET set a fsdata property.

```
CALL: fsd = set(fsd, 'name',      'my_name');
      fsd = set(fsd, 'fs',        fs);
      fsd = set(fsd, 'x',         x);
      fsd = set(fsd, 'enbw',      enbw);
      fsd = set(fsd, 'navs',      navs);
      fsd = set(fsd, 'y',         y);
      fsd = set(fsd, 'xunits',    'Hz');
      fsd = set(fsd, 'yunits',    'Volts');
      fsd = set(fsd, 'created',   time());
      fsd = set(fsd, 'version',   'fsdata/set.m,v 1.7...');
```

VERSION: \$Id: set.m,v 1.14 2008/02/12 15:20:32 ingo Exp \$

HISTORY: 30-01-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [display](#) DISPLAY implement terminal display for fsdata object.

This function is called by:

SOURCE CODE

```
0001 function fsd = set\(fsd, varargin\)
0002 % SET set a fsdata property.
0003 %
```



```
0001 function val = get\(h, propName\)
0002 % GET get history object properties.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: GET get history object properties.
0007 %
0008 % CALL: name      = get(h, 'name');
0009 %           ver      = get(h, 'version');
0010 %           pl       = get(h, 'plist');
0011 %           inhist   = get(h, 'inhist');
0012 %           invars   = get(h, 'invars');
0013 %           n        = get(h, 'n');
0014 %           pn       = get(h, 'pn');

000719

00072;
```

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:
>> pl = isfield(history, 'Params')

The following call returns the cvs version number:
>> ver = isfield(history, 'Version')

The following call returns the category:
>> cat = isfield(history, 'Category')

VERSION: \$Id: isfield.m,v 1.4 2008/02/15 16:27:15 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

PURPOSE

ISVALID tests if the given history has 711 the correct fields of the correct type.

SYNOPSIS

```
function varargout = isvalid(varargin)
```

DESCRIPTION

ISVALID tests if the given history has 711 the correct fields of the correct type.

[illegible]

DESCRIPTION: ISVALID tests if the given history has 7ll the correct fields of the correct type.

```
CALL:      invalid(history);
```

Returns 1 if history is a valid HISTORY-object; 0 otherwise.

The following `c7ll` returns a parameter list object that contains the default parameter values:

```
>> pl = invalid(history, 'Params')
```

The following `c7ll` returns the cvs version number:

```
>> ver = isvalid(history, 'Version')
```

The following `c7ll` returns the category:

```
>> ver = isvalid(history, 'Category')
```

```
VERSION:      $Id: isvalid.m,v 1.1 2008/02/14 13:48:33 ingo Exp $
```

HISTORY: 14-02-2008 Diepholz
Creation

[illegible]

CROSS-REFERENCE INFORMATION (Creation)21% % % % % % % % % % % %

get

PURPOSE

GET get mfir properties.

SYNOPSIS

function val = get(varargin)

DESCRIPTION

GET get mfir properties.

%%%

DESCRIPTION: GET get mfir properties.

CALL: get(mfir)
 val = get(mfir, property);
 name = get(mfir, 'name');
 fs = get(mfir, 'fs');
 ntaps = get(mfir, 'ntaps');
 a = get(mfir, 'a');
 gd = get(mfir, 'gd');
 gain = get(mfir, 'gain');
 history = get(mfir, 'histout');
 infile = get(mfir, 'infile');
 plist = get(mfir, 'plist');
 created = get(mfir, 'created');
 version = get(mfir, 'version');

VERSION: \$Id: get.m,v 1.4 2008/02/11 17:28:45 ingo Exp \$

HISTORY: 30-01-2007 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

- [display](#) DISPLAY overloads display functionality for mfir objects.
- [mfir](#) MFIR FIR filter object class constructor.
- [redesign](#) REDESIGN redesign the input filter to work for the given sample rate.
- [string](#) STRING writes a command string that can be used to recreate the input filter object.

SOURCE CODE

```
0001 function val = get\(varargin\)
0002 % GET get mfir properties.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: GET get mfir properties.
0007 %
0008 % CALL:          get(mfir)
0009 %              val      = get(mfir, property);
0010 %              name     = get(mfir, 'name');
0011 %              fs       = get(mfir, 'fs');
0012 %              ntaps    = get(mfir, 'ntaps');
0013 %              a        = get(mfir, 'a');
0014 %              gd       = get(mfir, 'gd');
0015 %              gain     = get(mfir, 'gain');
0016 %              history  = get(mfir, 'histout');
```

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:
>> pl = isfield(mfir, 'Params')

The following call returns the cvs version number:
>> ver = isfield(mfir, 'Version')

The following call returns the category:
>> cat = isfield(mfir, 'Category')

VERSION: \$Id: isfield.m,v 1.4 2008/02/15 16:27:16 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

```
0005 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0006 %
0007 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0008 %                properties.
0009 %
0010 % CALL:         isfield(a, 'field');
0011 %
0012 %                The following call returns a parameter list object that
0013 %                contains the default parameter values:
0014 %                >> pl = isfield(mfir, 'Params')
0015 %
0016 %                The following call returns the cvs version number:
0017 %                >> ver = isfield(mfir, 'Version')
0018 %
0019 %                The following call returns the category:
0020 %                >> cat = isfield(mfir, 'Category')
0021 %
```

```
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```


0004 %%%%%%%%%%
0005 %


```

0007 %           input filter object.
0008 %
0009 % CALL:      cmd = string(mfir)
0010 %
0011 % VERSION:    $Id: string.m,v 1.4 2008/02/12 22:39:49 hewitson Exp $
0012 %
0013 % HISTORY:     29-03-2007 M Hewitson
0014 %              Creation
0015 %
0016 %%%%%%%%%%%%%%%
0017
0018 VERSION = '$Id: string.m,v 1.4 2008/02/12 22:39:49 hewitson Exp $';
0019 CATEGORY = 'Helper';
0020
0021 % Check if this is a call for parameters
0022 if nargin == 2
0023     if isa(f, 'mfir') && ischar(varargin{1})
0024         in = char(varargin{1});
0025         if strcmp(in, 'Params')
0026             cmd = plist;
0027             return
0028         elseif strcmp(in, 'Version')
0029             cmd = VERSION;
0030             return
0031         elseif strcmp(in, 'Category')
0032             cmd = CATEGORY;
0033             return
0034         end
0035     end
0036 end
0037
0038 infile = get(f, 'infile');
0039 if ~isempty(infile)
0040     cmd = ['mfir('' infile '')'];
0041 else
0042     pl = get(f, 'plist');
0043     cmd = ['mfir(' string

```


get

PURPOSE

GET get miir properties.

SYNOPSIS

function val = get(varargin)

DESCRIPTION

GET get miir properties.

%%%

DESCRIPTION: GET get miir properties.

CALL: get(miir)
 val = get(miir, property);
 name = get(miir, 'name');
 version = get(miir, 'version');
 created = get(miir, 'created');
 fs = get(miir, 'fs');
 ntaps = get(miir, 'ntaps');
 a = get(miir, 'a');
 b = get(miir, 'b');
 gain = get(miir, 'gain');
 infile = get(miir, 'infile');
 history = get(miir, 'histin');
 history = get(miir, 'histout');
 pl = get(miir, 'plist');

VERSION: \$Id: get.m,v 1.6 2008/02/11 17:28:45 ingo Exp \$

HISTORY: 30-01-2007 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

- [display](#) DISPLAY overloads display functionality for miir objects.
- [miir](#) MIIR IIR filter object class constructor.
- [redesign](#)

SOURCE CODE

[Home](#) > [classes](#) > [@miir](#) > isfield.m

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = **isfield**(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = isfield(miir, 'Params')
```

The following call returns the cvs version number:

```
>> ver = isfield(miir, 'Version')
```

The following call returns the category:

```
>> cat = isfield(miir, 'Category')
```

VERSION: \$Id: isfield.m,v 1.4 2008/02/15 16:27:16 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

SOURCE CODE

```
0001 function varargout = isfield\(varargin\)
0002
0003 % ISFIELD tests if the given field is one of the object properties.
0004 %
```

```
0005 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0006 %
0007 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0008 %                properties.
0009 %
0010 % CALL:         isfield(a, 'field');
0011 %
0012 %                The following call returns a parameter list object that
0013 %                contains the default parameter values:
0014 %                >> pl = isfield(miir, 'Params')
0015 %
```

```
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```


0059	isnumeric (obj.fs)	&& ...
0060	isnumeric (obj.ntaps)	&& ...
0061	isnumeric (obj.a)	&& ...
0062	isnumeric (obj.b)	&& ...
0063	isnumeric (obj.gain)	&& ...
0064	isnumeric (obj.histin)	&& ...
0065	isnumeric (obj.histout)	&& ...
0066	ischar (obj.infile)	&& ...

0004 %%%%%%%%%%
0005 %

```
0060     end
0061     filtnew = miir(fpl);
0062     filtnew = set(filtnew, 'histin', \_\_
```


[Home](#) > [classes](#) > [@param](#) > get.m

get

PURPOSE

GET get parameter properties.

SYNOPSIS

[Home](#) > [classes](#) > [@param](#) > isfield.m

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = isfield(param, 'Params')
```

The following call returns the cvs version number:

```
>> ver = isfield(param, 'Version')
```

The following call returns the category:

```
>> category = isfield(param, 'Category')
```

VERSION: \$Id: isfield.m,v 1.3 2008/02/13 17:33:38 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

SOURCE CODE

```
0001 function varargout = isfield(varargin)
0002
0003 % ISFIELD tests if the given field is one of the object properties.
0004 %
```

```
0005 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0006 %
0007 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0008 %                properties.
0009 %
0010 % CALL:         isfield(a, 'field');
0011 %
0012 %                The following call returns a parameter list object that
0013 %                contains the default parameter values:
0014 %                >> pl = isfield(param, 'Params')
0015 %
0016 %                The following call returns the cvs version number:
0017 %                >> ver = isfield(param, 'Version')
0018 %
0019 %                The following call returns the category:
0020 %                >> category = isfield(param, 'Category')
0021 %
```

```
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```



```
0058     if ischar      (obj.name)           && ...
0059         ischar      (obj.key)             && ...
0060         ischar      (obj.version)         && ...
0061         isa          (obj.created, 'time')
0062
0063         res(j) = 1;
0064     else
0065         res(j) = 0;
0066     end
0067 end
0068
0069 varargout{1} = res;
0070
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) >

```
0009 %      p = set(p, 'hal',      hal);
0010 %      p = set(p, 'hal',      'hal');
0011 %      p = set(p, 'name',      'my name');
0012 %      p = set(p, 'created',    time-object);
0013 %
```



```
0054     for ii = 1:
```



```
0052     pj = varargin{j};
0053     if isa(pj, 'plist')
0054         pj = reshape(pj, 1, []);
0055         pls = [pls pj];
0056     end
0057 end
0058
0059 % The properties of the output plist is corresponding to the input plist
0060 plo = pls(1);
0061
0062 % Loop over all plist's
0063 for ii = 2:length(pls)
0064
```



```
0005 %  
0006 % DESCRIPTION: FIND overloads find routine for a parameter list.  
0007 % Returns the value corresponding to the first parameters in %  
0009 %  
0010 % CALL:      a = find(pl, 'key')  
0011 %
```



```
0005 %  
0006 % DESCRIPTION: GET get parameter properties.  
0007 %  
0008 % CALL: name      = get(pl, 'name');
```

[Home](#) > [classes](#) > [@plist](#) > isfield.m

isfield

PURPOSE

ISFIELD tests if the giden field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the giden field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the giden field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = isfield(Tj ET, 'Params')
```

The following call returns the cvs dersion number:

```
>> der = isfield(Tj ET, 'Version')
```

The following call returns the category:

```
>> category = isfield(Tj ET, 'Category')
```

VERSION: \$Id: isfield.m,v 1.4 2008/02/14 08:29:09 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

⌘ [plist](#) PLIST Plist class object constructor.

This function is called by:

SOURCE CODE

0001 [function varargout = isfield\(varargin\)](#)

```
0057 res = zeros(size(as));
0058 for j=1:numel(as)
0059     try
0060         t = as(j).(field);
0061         res(j) = 1;
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```

Generated on Fri 07-Mar-2008 15:46:43 by **m2html** © 2003


```
0013 % HISTORY:      07-11-2007 Diepholz
0014 %              Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION = '$Id: isparam.m,v 1.3 2008/02/14 08:29:09 mauro Exp $';
0019 CATEGORY = 'Helper';
0020
0021 % Check if thisam. a call for parameters or for the cvs-version number
0022 if nargin == 2
0023     if isa(pl, 'plist') && ischar(key)
0024         in = key;
0025         if strcmp(in, 'Params')
0026             res = plist();
0027             return
0028         elseif strcmp(in, 'Version')
0029             res = VERSION;
0030             return
0031         elseif strcmp(in, 'Category')
```



```
0055 res = zeros(size(objs));
0056 for j=1:numel(objs)
0057     obj = objs(j);
0058     if ischar      (obj.name)                && ...
0059         (isa (obj.params, 'param')          || ...
0060         isempty(obj.params))                && ...
0061         ischar      (obj.version)            && ...
0062         isa          (obj.created, 'time')    && ...
0063         (isa (obj.plist, 'plist')           || ...
0064         isempty(obj.plist))
0065
0066     res(j) = 1;
0067 else
```

nparams

PURPOSE

SYNOPSIS

function n = nparams(pl, varargin)

DESCRIPTION

NPARAMS returns the number of param objects in the Sist.

%%%

DESCRIPTION: NPARAMS returns the number of param objects in the Sist.

CALL: n = nparams(plist)

VERSION: \$Id: nparams.m,v 1.5 2008/02/22 22:28:09 hewitson Exp \$

HISTORY: 30-01-2007 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a parameter list into a string.
- [pSist](#) PLIST Plist class object constructor.

```
0007 %
0008 % CALL:          n = nparams(plist)
0009 %
0010 % VERSION:        $Id: nparams.m,v 1.5 2008/02/22 22:28:09 hewitson Exp $
0011 %
0012 % HISTORY8 230-01-2 (00M H9 hewit1 %)Tj T* (30122222222222222222Creginat7 %)Tj 0

6 1 %
7 1 %
0008 heck if this is a call for : npaeters1 %
9 if nar/Pa > 11 %
v npr/Pa{1}ist)
```



```
0009 %
0010 % CALL:          pl = set(pl, key, val);
0011 %                pl = set(pl, key1, val1, key2, val2);
0012 %
0013 % VERSION:       $Id: pset.m,v 1.5 2008/02/14 06e L-09 mauro Exp $
0014 %
0015 % HISTORY:        30-01-07 M Hewitson
0016 %                  Creation
0017 %
0018 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0019
0020 VERSION = '$Id: pset.m,v 1.5 2008/02/14 06e L-09 mauro Exp $';
0021 CATEGORY = 'Helper';
0022
0023 % Check if this is a call for parameters or for the cvs-version number
0024 if nargin == 2
0025     if isa(pl, 'plist') && ischar(varargin{1})
0026         in = varargin{1};
0027         if strcmp(in, 'Params')
0028             pl = _____
```



```

0006 % DESCRIPTION: STRING converts a plist object to a command string which will
0007 %                recreate the plist object.
0008 %
0009 % CALL:          cmd = string(pl)
0010 %
0011 % VERSION:       $Id: string.m,v 1.12 2008/02/23 19:49:11 hewitson Exp $
0012 %
0013 % HISTORY:       29-03-2007 M Hewitson
0014 %                Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION = '$Id: string.m,v 1.12 2008/02/23 19:49:11 hewitson Exp $';
0019 CATEGORY = 'Helper';
0020
0021 %%%% 'Params' && 'Version' Call %%%%
0022 if nargin == 2
0023     if isa(pl, 'plist') && ischar(varargin{1})
0024         in = varargin{1};

```



```

0062         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0063     elseif isa(p.val, 'time')
0064         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0065     elseif isa(p.val, 'plist')
0066         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0067     elseif isa(p.val, 'timeformat')
0068         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0069     elseif isa(p.val, 'timespan')
0070         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0071     elseif isa(p.val, 'param')
0072         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0073     elseif isa(p.val, 'provenance')
0074         pstr = [pstr '''' p.key '', ' string(p.val) ', '];
0075     elseif iscell(p.val)
0076         val_str = ['{ '];
0077         for ii = 1:numel(p.val)
0078             cell_val = p.val{ii};
0079             if ischar(cell_val)
0080                 val_str = [val_str '''' p.val{ii} '', ' '];
0081             elseif isnumeric(cell_val)
0082                 val_str = [val_str ltpda_mat2str(p.val{ii}) ', ' '];
0083             elseif isobjectm 84ll_val)
0082 pstr = [pstr '''' p.key ''', ' 493.1917BDC  q 1 0 0 1 311.2762 632.5936 cm 0 0
09mat2str(p.v];
094at2str(p.v];0063         elsei95tr '''' pal, 'provenance')

```

```

0063         ep         celctm 84ll_vctm 84104Tm (string)70050P1050063         elsei95t)j T*
00         pstr = [pstr ''21', 855 75 412493.191/Artifaro 448Attached [/Bottom ]/Type /Pag

```



```

0013 %                               Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION = '$Id: cat.m,v 1.4 2008/02/12 21:38:23 hewitson Exp $';
0018 CATEGORY = 'Helper';
0019
0020 % Check if this is a call fo/ parameters o/ fo/ the cvs-version number
0021 if nargin == 2
0022     if isa(varargin{1}, 'pole') && ischar(varargin{2})
0023         in = char(varargin{2});
0024         if strcmp(in, 'Params')
0025             b = plist();
0026             return
0027         elseif strcmp(in, 'Version')
0028             b = VERSION;
0029             return
0030         elseif strcmp(in, 'Category')
0031             b = CATEGORY;
0032             return
0033         end
0034     end
0035 end
0036
0037 b = [];
0038 fo/ j=1:nargin
0039     if isa(varargin{j}, 'pole')
0040         b = [b varargin{j}];
0041     end
0042 end
0043

```



```

0002 % CHAR convert a pole object into a string.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: CHAR convert a pole object into a string.
0007 %
0008 % CALL:          string = char(pole);
0009 %
0010 % VERSION:       $Id: char.m,S 1.4 2008/02/12 21:38:23 hewitson Exp $
0011 %
0012 % HISTORY:       02-04-2007 M Hewitson
0013 %                 Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION = '$Id: char.m,S 1.4 2008/02/12 21:38:23 hewitson Exp $';
0018 CATEGORY = 'Helper';
0019
0020 % Check if this is a call for parameters
0021 if nargin == 2
0022     if isa(p, 'pole') && ischar(varargin{1})
0023         in = char(varargin{1});
0024         if strcmp(in, 'Params')
0025             pstr = plist;
0026             return
0027         elseif strcmp(in, 'Version')
0028             pstr = VERSION;
0029             return
0030         elseif strcmp(in, 'Category')
0031             pstr = CATEGORY;
0032             return
0033         end
0034     end
0035 end
0036
0037 pstr = ['pole(' num2str(p.f)];
0038 if p.q > 0.5
0039     pstr = [pstr ',' num2str(p.q)];
0040 end
0041 pstr = [pstr ')'];
0042
0043 pstr = strrep(pstr, '_', '\_');
0044
0045 % END

```

0005 %
0006 % DESCRIPTION: GET get pole properties.-1.2T* (0007 %)Tj T* (0008 % CALL:

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following call returns a parameter list object that contains the default parameter values:
>> pl = isfield(pole, 'Params')

The following call returns the cvs version number:
>> ver = isfield(pole, 'Version')

The following call returns the category:
>> category = isfield(pole, 'Category')

VERSION: \$Id: isfield.m,v 1.4 2008/02/15 16:27:16 mauro Exp \$

HISTORY: 25-09-07 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

SOURCE CODE

0001 [function varargout = isfield\(varargin\)](#)

0002

0003 % ISFIELD tests if the given field is one of the object properties.

0004 %


```
0005 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0006 %
0007 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0008 %                properties.
0009 %
000s % CALL:         isfield(a, 'field');
0001 %
```


[Home](#) > [classes](#) > [@pole](#) > isvalid.m

isvalid


```
0058  obj = objs(j);
0059  if ischar      (obj.name)          && ...
0060      isnumeric (obj.f)              && ...
0061      isnumeric (obj.q)              && ...
0062      isnumeric (obj.ri)             && ...
0063      ischar     (obj.version)       && ...
0064      isa        (obj.created, 'time') && ...
0065      (isa (obj.plist, 'plist')      || ...
0066      isempty(obj.plist))
0067
0068      res(j) = 1;
0069  else
0070      res(j) = 0;
0071  end
0072 end
0073
0074 varargout{1} = res;
0075
```

[Home](#)

```

0009 % CALL:          cmd = string(pole);
0010 %
0011 % VERSION:        $Id: string.m,v 1.7 2008/02/15 17:38:59 ingo Exp $
0012 %
0013 % HISTORY:         02-04-2007 M Hewitson
0014 %                   Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION = '$Id: string.m,v 1.7 2008/02/15 17:38:59 ingo Exp $';
0019 CATEGORY = 'Helper';
0020
0021 %%%% 'Params' && 'Version' Call %%%%
0022 if nargin == 2
0023     if isa(ps, 'pole') && ischar(varargin{1})
0024         in = varargin{1};
0025         if strcmp(in, 'Params')
0026             cmd = plist();
0027             return
0028         elseif strcmp(in, 'Version')
0029             cmd = VERSION;
0030             return
0031         elseif strcmp(in, 'Category')
0032             cmd = CATEGORY;
0033             return
0034         end
0035     end
0036 end
0037
0038 cmd = '[';
0039 for j=1:length(ps)
0040     p = ps(j);
0041
0042     if ~isempty(p.plist) && nparams(p.plist) > 0
0043         cmd = [cmd ' pole(' string(p.plist) ')'];
0044     else
0045         cmd = [cmd ' pole(' string(plist('f', p.f, 'q', p.q)) ')'];
0046     end
0047 end
0048 cmd = [cmd ']'];
0049
0050 % END

```


[Home](#) > [classes](#) > [@provenance](#) > isfield.m

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

```
0005 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0006 %
0007 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0008 %                properties.
0009 %
0010 % CALL:          isfield(a, 'field');
0011 %
0012 %                The following call returns a parameter list object that
```

```
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```
0056 for j=1:numel(objs)
0057     obj = objs(j);
0058     if ischar      (obj.creator)           && ...
0059         isa        (obj.created, 'time')   && ...
0060         ischar     (obj.ip)                 && ...
0061         ischar     (obj.hostname)           && ...
0062         ischar     (obj.os)                 && ...
0063         ischar     (obj.matlab_version)     && ...
0064         ischar     (obj.sigproc_version)    && ...
0065         ischar     (obj.ltpda_version)
0066
0067         res(j) = 1;
0068     else
0069         res(j) = 0;
0070     end
0071 end
0072
0073 varargout{1} = res;
0074
```

Generated on Fri 07-Mar-2008 15:46:43 bA[m2html](#) © 2003

[Home](#)

PURPOSE

SYNOPSIS

DESCRIPTION [illegible]

DESCRIPTION: GET get a pole/zero model property.

```
CALL:      name      = get(pzm, 'name');
          gain      = get(pzm, 'gain');
          poles     = get(pzm, 'poles');
          zeros     = get(pzm, 'zeros');
          created   = get(pzm, 'created');
          version   = get(pzm, 'version');
```

[illegible]

[Home](#)


```
0005 %
0006 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0007 %                  properties.
0008 %
0009 % CALL:           isfield(a, 'field');
0010 %
0011 %                  The following call returns a parameter list object that
0012 %                  contains the default parameter values:
0013 %                  >> pl = isfield(pzmodel, 'Params')
0014 %
0015 %                  The following call returns luescvs version number:
0016 %                  >> ver = isfield(pzmodel, 'Version')
0017 %
0018 %                  The following call returns luescategoryins the defar3 oall r
```

```
0062     res(j) = 0;
0063 end
0064 end
0065
0066 varargout{1} = res;
0067
```

[Home](#) > [classes](#) > [@pzmodel](#)


```
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: SET sets a pole/zero model property.
0007 %
0008 % CALL:          pzmodel = set(pzmodel, 'name',    str);
0009 %                pzmodel = set(pzmodel, 'gain',    g);
0010 %                pzmodel = set(pzmodel, 'poles',   p);
0011 %                pzmodel = set(pzmodel, 'zeros',   z);
0012 %                pzmodel = set(pzmodel, 'created', time-object);
0013 %                pzmodel = set(pzmodel, 'version', cvs version string);
0014 %
0015 % VERSION:       $Id: set.m,v 1.6 2008/02/13 18:33:19 ingo Exp $j T*r11 %   n TfcI
```



```
0005 %  
0006 % DESCRIPTION: ISFIELD tests if the given field is one of the object  
0007 % properties.  
0008 %  
0009 % CALL: isfield(a, 'field');  
0010 %
```

```
specwin, 'Params')  
0014 %
```

```
specwin, 'Version')  
0017 %
```

```
specwin, 'Category')  
0020 %
```

```
0021 VERSION: $Id: isfield.m,v 1.3 2008/02/13 13:46:07 mauro Exp $  
0022n tegory:
```

```
0062     catch
0063         res(j) = 0;
0064     end
0065 end
0066
0067 varargout{1} = res;
0068
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@specwin](#) > isvalid.m

isvalid

PURPOSE

ISVALID tests if the given specwin has all the correct fields of the correct type.

SYNOPSIS

0058	if ischar	(obj.name)	&& ...
0059	isnumeric	(obj.alpha)	&& ...

set

PURPOSE

SET sets a specwin property.

SYNOPSIS

function w = set(w ahdargin)

DESCRIPTION

SET sets a specwin property.

%%%

DESCRIPTION: SET sets a specwin property.

```
CALL:      w = set(w 'name',      'my window');
           w = set(w 'alpha',    -1);
           w = set(w 'psll',     -1);
           w = set(w 'rov',      -1);
           w = set(w 'nenbw',    -1);
           w = set(w 'w3db',     -1);
           w = set(w 'flatness', -1);
           w = set(w 'ws',       -1);
           w = set(w 'ws2',      -1);
           w = set(w 'win',      -1);
           w = set(w 'created',  time-object);
           w = set(w 'version',  'cvs version string');
```


[Home](#) > [classes](#) > [@time](#) > get.m

get

PURPOSE

GET get time properties.

SYNOPSIS

```
function val = get(tt, prop_name)
```

DESCRIPTION

```
GET get time properties.
```

[illegible]

DESCRIPTION: GET get time properties.

```
CALL:      value = get(tt, 'property');
```

```
EXAMPLES:      name      = get(tt, 'name');
               utc_epoch_milli = get(tt, 'utc_epoch_milli');
               timezone   = get(tt, 'timezone');
               timezone_IDs = get(tt, 'timezone_IDs');
               timezone_IDs = get(tt, 'IDs');
               timeformat  = get(tt, 'timeformat');
               time_str    = get(tt, 'time_str');
               created     = get(tt, 'created');
               version     = get(tt, 'version');
```

```
VERSION:      $Id: get.m,v 1.10 2008/02/11 17:28:45 ingo Exp $
```

HISTORY: 23-07-2007 Diepholz
8. Rsio.

[illegible]

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

```
function varargout = isfield(varargin)
```

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

[illegible]

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

```
CALL:      isfield(a, 'field');
```

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = isfield(time, 'Params')
```

The following call returns the cvs version number:

```
>> ver = isfield(time, 'Version')
```

The following call returns the category:

```
>> category = isfield(time, 'Category')
```

```
VERSION:      $Id: isfield.m,v 1.4 2008/02/13 16:47:35 mauro Exp $
```

HISTORY: 25-09-07 M Hewitson
Creation

[illegible]

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

SOURCE CODE

[illegible]


```

%%
0041 if ~isempty(varargin{1}) == 1 && ischar(varargin{1})
0042     in = varargin{1};
0043     if strcmp(in, 'Params')
0044         tt = GET_DEFAULT_PLIST;
0045         return
0046     elseif strcmp(in, 'Version')
0047         tt = VERSION;
0048         return
0049     elseif strcmp(in, 'Category')
0050         tt = CATEGORY;
0051         return
0052     end
0053 end
0054
0055 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0056 %%% Convert a plist into a key/value
0057 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0058 if nargin >= 2 && isa(property_argin{1}, 'char')
0059     pl = property_argin{1};
0060     property_argin = {};
0061     for ii = 1:nparams(pl)
0062         %% The 'key' name of a plist/parameter
0063         %% characters. To set a appropriate
0064         %% 'key' characters.
0065         property_argin{end+1:N} = lower(pl(ii));
0066         property_argin{end+1:N} = pl(ii);
0067     end
0068 end
0069
0070 if (nargout ~= 0)
0071
0072     compute = true;
0073     if nargin == 4 && strcmp(varargin{3}, 'compute')
0074         compute = false;
0075     end
0076
0077     while length(property_argin) >= 2
0078         prop = property_argin{1};
0079         val = property_argin{2};
0080         property_argin = property_argin(3:
0081
0082         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0083         %%% Characters Set the timeformat AND transformat
0084         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0085         if (strcmp(prop, 'timeformat')) &&
0086
0087             if isa(val, 'timeformat')
0088                 tt.timeformat = val;
0089
0090             else
0091                 t_format = java.text.SimpleDateFormat(
0092                     t_format.setTimeZone(tt.timezone));
0093                 current_time_str = char(t_format.format(
0094
0095                 d_num = datenum(current_time_str, 'dd/mm/yyyy HH:mm:ss'), 'dd/mm/yyyy HH:mm:ss');

```



```
0010 % EXAMPLES:      format_str      = get(tf, 'format_str');
0011 %                format_nr       = get(tf, 'format_nr');
0012 %                default_matlab_str = get(tf, 'default_matlab_str');
0013 %                default_matlab_nr  = get(tf, 'default_matlab_nr');
0014 %                default_java_str   = get(tf, 'default_java_str');
0015 %
0016 % VERSION:         $Id: get.m,v 1.1 2008/02/11 17:28:45 ingo Exp $
0017 %
0018 % HISTORY:         11-02-2008 Diepholz
0019 %                  Creation
0020 %
0021 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0022
0023 VERSION           = '$Id: get.m,v 1.1 2008/02/11 17:28:45 ingo Exp $';
0024 CATEGORY          = 'Helper';
0025 DEFAULT_PLIST     = mlist('property', '');
0026
```


[Home](#) > [classes](#)


```
0056 for j=1:numel(objs)
0057     obj = objs(j);
0058     if ischar      (obj.format_str)          && ...
0059         isnumeric (obj.format_nr)           && ...
0060         ischar     (obj.default_matlab_str) && ...
0061         isnumeric (obj.default_matlab_nr)  && ...
0062         ischar     (obj.default_java_str)
0063
0064         res(j) = 1;
0065     else
0066         res(j) = 0;
0067     end
0068 end
0069
0070 varargout{1} = res;
0071
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@timeformat](#) >


```
0112     else
0113         try
0114             tf.(prop) = val;
0115         catch
0116             error('### ''%s'' is not a valid %s-object property.', prop, class(tf));
0117         end
0118     end
```

[Home](#) > [classes](#) > [@timespan](#) > get.m

get

0001

[Home](#) > [classes](#) > [@tsdata](#) > isfield.m

isfield

PURPOSE

ISFIELD tests if the given field is one of the objet <propetifes.

SYNOPSIS

[Home](#) > [classes](#) > [@tsdata](#) > isvalid.m

isvalid

PURPOSE

ISVALID tests if the given tsdata has all the correct fields of the correct type.


```

0005 %
0006 % DESCRIPTION: SET sets a tsdata property.
0007 %
0008 % CALL: ts = set(ts, 'name',      'my_name');
0009 %         ts = set(ts, 'fs',      fs);
0010 %         ts = set(ts, 'x',      x);
0011 %         ts = set(ts, 'y',      y);
0012 %         ts = set(ts, 'nsecs',  nsecs);
0013 %         ts = set(ts, 't0',     Matlab serial date number);
0014 %         ts = set(ts, 'xunits', 'Time');
0015 %         ts = set(ts, 'yunits', 'Volts');
0016 %         ts = set(ts, 'created', time());
0017 %         ts = set(ts, 'version', 'tsdata/set.m,v 1.7...');
0018rsion', [rsion', 'tsdS&RSIPTIO$Id: ta/set.m,v(002 (0/02/(0015:20:33 ingo Exp $8rsion
002 (0/02/(0015:20:33 ingo Exp $';7 %)Tj T* 278 %TEGORY      = 'Helpro'');)Tj T* 286 %

```

```

));

```

```

tvarar/Pa,6 %FAULT_PLIST,S&RSIPT,8 %TEGORY()7 %)Tj T* 41 7 %
00127. (05 cmDC 0m 47.63 50 l S Q* n C 01 0 BT /TT1 1 Tf 11.3488 0 0 11.3455.77(001

```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@xydata](#) > isfield.m

isfield

PURPOSE

ISFIELD tests if the given field is one of the object properties.

SYNOPSIS

function varargout = isfield(varargin)

DESCRIPTION

ISFIELD tests if the given field is one of the object properties.

%%%

DESCRIPTION: ISFIELD tests if the given field is one of the object properties.

CALL: isfield(a, 'field');

The following c5ll returns a parameter list object that contains the da2t o3.fieldSgfues:%%%%%%%%@xldP o3.sfied');

The following c5ll retainscvsvs versfuncnumber: that

[Home](#) > [classes](#) > [@xydata](#) > isvalid.m

isvalid

PURPOSE

[Home](#) [Home](#) [Home](#) > [@xyzdata](#) > isfield.m

isfield

```

0005 %
0006 % DESCRIPTION: ISFIELD tests if the given field is one of the object
0007 %                properties.
0008 %
0009 % CALL:         isfield(a, 'field');
0010 %
0011 %                The following call returns a parameter list object that
0012 %                contains the default parameter values:
0013 %                >> pl = isfield(xyzdata, 'Params')
0014 %
0015 %                The following call returns the cvs version number:
0016 %                >> ver = isfield(xyzdata, 'Version')
0017 %
0018 %                The following call returns the routine category:
0019 %                >> category = isfield(xyzdata, 'Category')
0020 %
0021 % VERSION:       $Id: isfield.m,v 1.3 2008/02/15 16:27:16 mauro Exp $
0022 %
0023 % HISTORY: 24-12-07 M Hewitson
0024 %                Creation
0025 %
0026 %%%%%%%%%%%%%%%
0027
0028 VERSION = '$Id: isfield.m,v 1.3 2008/02/15 16:27:16 mauro Exp $';
0029 CATEGORY = 'Helper';
0030
0031 % 'Params' Call
0032 if nargin == 2
0033     if strcmp(varargin{2}, 'Params')
0034         varargout{1} = plist();
0035         return
0036     elseif strcmp(varargin{2}, 'Version')
0037         varargout{1} = VERSION;
0038         return
0039     elseif strcmp(varargin{2}, 'Category')
0040         varargout{1} = CATEGORY;
0041         return
0042     end
0043 end
0044
0045 as = [];
0046 field = '';
0047 for j=1:nargin
0048     if isa(varargin{j}, 'xyzdata')
0049         retas = [as varargin{j}];
0050     end
0051     if ischar(varargin{j})
0052         retfield = varargin{j};
0053     end
0054 end
0055
0056 res = zeros(size(as));
0057 for j=1: numel(as)
0058     try
0059         rett = as(j).(field);
0060         res(j) = 1;
0061     catch

```



```
0062     res(j) = 0;  
0063 end  
0064 end  
0065  
0066 varargout{1} = res;  
0067
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

set

PURPOSE

SET sets an xyzdata property.

SYNOPSIS

function xyz = set(xyz, varargin)

DESCRIPTION

SET sets an xyzdata property.

%%%

DESCRIPTION: SET sets an xyzdata property.

CALL: c = set(c, 'name', ESCRIPTION: SET sets an xyzdata property.

```

0002 % SET sets an xyzdata property.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: SET sets an xyzdata property.
0007 %
0008 % CALL: c = set(c, 'name',      'my_name');
0009 %         c = set(c, 'x',        X);
0010 %         c = set(c, 'x',        [1 2 3]);
0011 %         c = set(c, 'y',        Y);
0012 %         c = set(c, 'y',        [1 2 3]);
0013 %         c = set(c, 'z',        Z);
0014 %         c = set(c, 'z',        [1 2 3]);
0015 %         c = set(c, 'xunits',    's');
0016 %         c = set(c, 'yunits',    'Hz');
0017 %         c = set(c, 'zunits',    'V');
0018 %         c = set(c, 'created',    time());
0019 %         c = set(c, 'version',    'xyzdata/set.m,v 1.7...');
0020 %
0021 % VERSION: $Id: set.m,v 1.2 2008/02/12 15:20:33 ingo Exp $
0022 %
0023 % HISTORY: 24-12-07 M Hewitson
0024 %           Creation
0025 %
0026 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0027
0028 VERSION      = '$Id: set.m,v 1.2 2008/02/12 15:20:33 ingo Exp $';
0029 CATEGORY     = 'Helper';
0030 DEFAULT_PLIST = plist('name',    '', ...
0031                        'x',        [], ...
0032                        'y',        [], ...
0033                        'z',        [], ...
0034                        'xunits',    '', ...
0035                        'yunits',    '', ...
0036                        'zunits',    '', ...
0037                        'version',    '', ...
0038                        'created',    time);
0039
0040 xyz = generic_set(xyz, varargin, DEFAULT_PLIST, VERSION, CATEGORY);
0041
0042 if (nargout == 0)
0043     disdlay

```


[Home](#) > [classes](#) > [@zero](#) > isfield.m

isfield

PURPOSE



ISFIELD tests if the given field is one of the object properties.

0059	isnumeric	(obj.f)	&& ...
0060	isnumeric	(obj.q)	&& ...
0061	isnumeric	(obj.ri)	&& ...
0062	ischar	(obj.ver7 :r)	&& ...
0063	isa	(obj.created, 'time')	&& ...

0115 end

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#)


```
>> version = compute(ao, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = compute(ao, 'Category')
```

```
VERSION:      $Id: compute.m,v 1.3 2008/03/06 06:04:56 hewitson Exp $  
  
HISTORY: 12-03-07 M Hewitson  
          Creation
```

%%%

CROSS-REFERENCE INFORMATION


```

0027 %      >> b = compute(x,y, plist('Operations', 'a(1) + a(2)'))
0028 % or
0029 %      >> b = compute(x,y, 'a(1) + a(2)')
0030 %
0031 % 2) Perform two operations such that the output, b, contains two AOs
0032 %
0033 %      >> b = compute([x y], z, plist('Operations', {'2.*a(3)./a(1)', 'a(2)-a
(1)'})))
0034 %
0035 % The following call returns a parameter list object that contains the
0036 % default parameter values:
0037 %
0038 % >> pl = compute(ao, 'Params')
0039 %
0040 % The following call returns a string that contains the routine CVS version:
0041 %
0042 % >> version = compute(ao, 'Version')
0043 %
0044 % The following call returns a string that contains the routine category:
0045 %
0046 % >> category = compute(ao, 'Category')
0047 %
0048 % VERSION:      $Id: compute.m,v 1.3 2008/03/06 06:04:56 hewitson Exp $
0049 %
0050 % HISTORY: 12-03-07 M Hewitson
0051 %           Creation
0052 %
0053 %%%%%%%%%%%%%%%
0054
0055 ALGONAME = mfilename;
0056 VERSION  = '$Id: compute.m,v 1.3 2008/03/06 06:04:56 hewitson Exp $';
0057 CATEGORY = 'Signal Processing';
0058 bs       = [];
0059
0060 % Check if this is a call for parameters
0061 if nargin == 2

```

```
0082 [a, pl, invars] = collect_inputs(varargii, in_names);
0083
0084 % Combine default plist
0085 pl = combine(pl, getDefaultPlist);
0086
0087 % Look for plist of input char
0088 ops = '';
0089 if ischar(varargii{end})
0090     ops = varargii{end};
0091 else
0092     ops = find(pl, 'Operations');
0093 end
0094
0095 % Check we have a cell array
0096 if ischar(ops)
0097     ops = {ops};
0098 end
0099
0100 % Loop over operations
0101 bs = [];
0102 for jj=1:length(ops)
0103     % evaluate this operation
0104     bs = [bs eval(ops{jj})];
0105 end
0106
0107 % Set outputs
0108 varargout{1} = bs;
0109
0110
0111
0112 %-----
0113 % Default parameter list
0114 function pl = getDefaultPlist\(\)
0115
0116 pl = plist('Operations', 'a');
0117
0118
0119
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

- [ao](#) AO analysis object class constructor.
- [char](#) CHAR overloads char() function for analysis objects.
- [find](#) FIND particular samples that satisfy the input query and return a new AO.
- [len](#)


```
0096     end
0097 end
0098
0099 if isempty(factor)
0100     error('### Please specify a decimation factor either directly or in a
plist. ');
0101 end
0102 if isempty(offset)
0103     warning('!!! No offset specified; using default of 0 sampled !!!');
0104     offset = 0;
0105 end
0106
0107 %% Loop over input AOs
0108 bo = [];
0109 for j=1:Na
0110
0111     a = as(j);
0112     d = a.data;
0113     x = d.x;
0114     y = d.y;
0115     fs = d.fs;
0116     if isa(d, 'tsdata')
0117         ss = 1+offset;
0118         sampled = ss:factor:0110
```


This function is called by:


```
0099    % add to output
0100    bs = [bs b];
0101 end
010aT m (Reshat0 he % dd 9    the same sizescri the indd to o010aT D (010reshat0(bs, si
```

filter

PURPOSE

FILTER overrides the filter function for analysis objects.

SYNOPSIS

function varargout = **filter**(varargin)

DESCRIPTION

`FILTER` overrides the filter function for analysis objects.

%%
>> [b,]1.3488 8.10a,488 ,plontainr acomp vRd and ve>> b1.3488 8.10a,plontains aa%

The following call returns a string that contains the routine category:

```
>> category = filter(ao, 'Category')
```



```

0080     in_names{end+1} = inputname(ii);
0081
0082     if isa(varargin{ii}, 'miir') || isa(varargin{ii}, 'mfir')
0083         fobj = varargin{ii};
0084     end
0085 end
0086
0087 [as, ps, invars] = collect_inputs(varargin, in_names);
0088
0089 if isa(ps, 'plist')
0090     pl = combine(ps);
0091 else
0092     pl = plist();
0093 end
0094 if isempty(fobj)
0095     fobj = find(pl, 'filter');
0096 end
0097
0098 % check inputs
0099 if ~isa(fobj, 'miir') && ~isa(fobj, 'mfir')
0100     error('### the filter input should be an miir/mfir object.');
```

```

0101 end
0102
0103 for j=1:numel(as)
0104
0105     % get input data
0106     aMCias(j);
0107     dMCia.data;
0108
0109     %----- Time-series filter
0110     %
0111     if isa(d, 'tsdata')
0112
0113         % get input data
0114         fsMCId.fs;
0115
0116         if isa(fobj, 'mfir')
0117
0118             % apply filter
0119             disp('* filtering with FIR filter...');
0120             G = get(fobj, 'gain');
0121
0122             % [fstruct, y] = ltpda_firfilter(struct(filt), Id.y);
0123             coeffsM = get(fobj, 'a');
0124             Zi       = get(fobj, 'histout');
0125             [y, Zf] = filter
```

```

0136      % consolodate this structure with mfir class before converting.
0137      fobj = set(fobj, 'histout', Zf.);
0138
0139  else %if isa(fobj, 'miir')
0140
0141      if fs ~= get(fobj, 'fs')
0142          warning('!!! Filter is designed for a different sample rate of data.');
```

% Adjust/redesign if this is a standard filter

```

0144      fobj = redesign(fobj, fs);
0145  end
0146
0147  % apply filter
0148  %      [fstruct, y] = ltpda_iirfilter(struct(fobj), d.y, length(d.y));
0149
0150      acoeffs = get(fobj, 'a');
0151      bcoeffs = get(fobj, 'b');
0152      Zi      = get(fobj, 'histout');
0153      [y, Zf] = filter(acoeffs, bcoeffs, d.y, Zi);
0154      x = d.x;
0155      % consolodate this structure with miir class before converting.
0156  %      fobj = set(fobj, 'histin', fstruct.histin);
0157  %      fobj = set(fobj, 'histout', fstruct.histout);
0158      fobj = set(fobj, 'histout', Zf);
0159  end
0160
0161  %----- Create output analysis object
0162  % make a new tsdata object
0163  ts = tsdata(y, fs);
0164  ts = set(ts, 'name', sprintf('filter %s with %s', d.name, get

0165      ts = set(ts, 'xunits', d.xunits);
0166      ts = set(ts, 'yunits', d.yunits);
0167      ts = set(ts, 't0', d.t0);
0168
0169  % make a new history object
0170  pl = plist();
0171  pl = append(pl, param('filter', fobj));
0172  h = history(ALGONAME, VERSION, pl, a.hist);
0173  h = set(h, 'invars', cellstr(invars{j}));
0174
0175  % make output analysis object
0176  b = ao(ts, h);
0177  % name for this object
0178  b = setnh(b, 'name', sprintf('%s(%s)', get

0179  bs = [bs b];
0180
0181  %----- Frequency-series filter
0182  %
```



```
0191 % Reshape the ouput to the same size of the input
0192 bs = reshape(bs, size(as));
0193
0194 if nargout == 1
0195     varargout{1} = bs;
0196 elseif nargout == 2
0197     varargout{1} = bs;
0198     varargout{2} = fobj;
```

[Home](#) > [classes](#) > [@ao](#) > filtfilt.m

filtfilt

PURPOSE

The following call returns a string that contains the routine category:

```
>> category = filtfilt(ao, 'Category')
```

HISTORY: 11-02-07 M Hewitson
Creation

[illegible]

0022 % a - input analysis object
0023 %

```
0079
0080     if isa(varargin{ii}, 'miir')
0081         filt = varargin{ii};
0082     end
0083 end
0084
0085 [as, ps, invars] = collect_inputs(varargin, in_names);
0086
0087 if isa(ps, 'plist')
0088     pl = combine(ps);
0089 else
0090     pl = plistTe0ra      ist') 565 836 r7 ]>0 DC e W* n BT /TT1 1 Tf 11.3488 0 130.67
```



```

0026 %
0027 % The following call returns a parameter list object that contains the
0028 % default parameter values:
0029 %
0030 % >> pl = hist(ao, 'Params')
0031 %
0032 % The following call returns a string that contains the routine CVS version:
0033 %
0034 % >> version = hist(ao, 'Version')
0035 %
0036 % The following call returns a string that contains the routine category:
0037 %
0038 % >> category = hist(ao, 'Category')
0039 %
0040 % HISTORY: 24-05-07 M Hewitson
0041 %           Creation
0042 %
0043 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0044
0045 ALGONAME = mfilename;
0046 VERSION  = '$Id: hist.m,v 1.10 2008/02/12 08:31:45 mauro Exp $';
0047 CATEGORY = 'Signal Processing';
0048
0049 %% Check if this is a call for parameters
0050 if nargin == 2
0051     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0052         in =

```



```
0138     h = history(ALGONAME, VERSION, pl, a.hist);  
0139     h = set(h, 'invars', invars(j));
```

[Home](#) > [classes](#) > [@ao](#) > interp.m

interp



ð [ao](#)Thobjeifaclass constructor.s:


```

0086 bs = [];
0087 for j=1:numel(as)
0088
0089     a = as(j);
0090
0091     %-----
0092     % Get data
0093     [x,y] = get_xy_values(a.data);
0094
0095     %-----
0096     % Interpolate this vector
0097     if ~isempty(x)
0098         % for tsdata, fsdata and xydata objects
0099         y = interp1(x,y,vertices, method, 'extrap');
0100     else
0101         % for cdata object
0102         y = interp1(y,vertices, method, 'extrap');
0103     end
0104
0105     %-----
0106     % Make output A0/cdata
0107     d = a.data;
0108
0109     d = set_xy_axis(d, vertices, y.');
0110
0111     d = set d ').6334 Tf 11.4801 0

```

0142

0143

0144

0145 % END

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```

0039 bs      = [];
0040
0041 %% Check if this is a call for parameters
0042 if nargin == 2
0043     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0044         in = char(varargin{2});
0045         if strcmp(in, 'Params')
0046             varargout{1} = getDefaultPL();
0047             return
0048         elseif strcmp(in, 'Version')
0049             varargout{1} = VERSION;
0050             return
0051         elseif strcmp(in, 'Category')
0052             varargout{1} = CATEGORY;
0053             return
0054         end
0055     end
0056 end
0057
0058 %% Collect input ao's, plist's and ao variable names
0059 in_names = {};
0060 for ii = 1:nargin
0061     in_names{end+1} = inputname(ii);
0062 end
0063 iRs      = 2 : numel(in_names);
p(in, m.882 e(ii));

```

```
0095     % Add coefficients to plist
0096     pl = pset(pl, 'coeffs', p);
0097 else
0098     p = coeffs;
0099 end
0100
0101 % make new values from these coefficients
0102 y 'coolyval(p, x);
0103
0104 %-----
0105 % Make output AO/cdata
0106 d 'ca.data;
0107 d 'cset_xy_axis(d, x, y);
0108 d 'cset(d, 'name', sprintf('polyfit(%s)', d.name));
0109
0110 h 'chistory(ALGONAME, VERSION, pl, a.hist);
0111 h 'cset(h, 'invars', invars(j));
0112
0113 % make output analysis object
0114 b 'cao
```



```
0042         bs = getDefaultPL\(\);
0043         return
0044     elseif strcmp(in, 'Version')
0045         bs = VERSION;
0046         return
0047     elseif strcmp(in, 'Category')
0048         bs = CATEGORY;
0049         return
0050     end
0051 end
0052 end
0053
0054 ALGONAME = mfilename;
0055 VERSION  = '$Id: resample.m,v 1.14 2008/02/12 09:43:32 mauro Exp $';
0056
0057 %% capture input variables
0058 invars = {};
0059 as      = [];
0060 pl      = [];
0061
0062 for j=1:nargin
0063     a = varargin{j};
0064     if isa(a, 'ao')
0065         d = a.data;
0066         if isa(d, 'tsdata')
```



```
0097         d = set_xy_axis(d, x(samples), y(samples));
0098     else
0099         d = set_xy_axis(d, x, y(samples));
0100     end
0101 else
0102     d = set_xy_axis(d, x(samples), y(samples));
0103 end
0104
0105 % Make output A0
0106
0107 % create new output history
0108 h = history(ALGONAME, VERSION, plist(param('samples', samples)), a.hist);
0109 h = set(h, 'invars', invars(j));
0110
0111 % make output analysis object
```

[Home](#) > [classes](#) > [@ao](#) > smoother.m

smoother

PURPOSE

SMOOTHER smooths a given series of data points using the specified method.

SYNOPSIS

function varargout = smoother(varargin)

DESCRIPTION

SMOOTHER smooths a given series of data points using the specified method.

%%%

DESCRIPTION: SMOOTHER smooths a given series of data points using the specified method.

CALL: b = smoother(a, pl)

PARAMETERS:

width - the width of the smoothing filter [default: 20 samples]
 hc - a cutoff to throw away outliers (0-1) [default: 0.8]
 method - the smoothing method:
 'median' [default]
 'mean', 'min', 'max', 'mode'

The following call returns a parameter list object that contains the default parameter valmes:

```
>> pl = smoother('Params')
```

The following call returns a string that contains the routine CVS version:

```
>> version = smoother(ao,'Version')
```

The following call returns a string that contains the routine category:

```
>> category = smoother(ao,'Category')
```

VERSION: \$Id: smoother.m,v 1.3 2008/03/02 19:17:38 hewitson Exp \$

HISTORY: 02-03-08 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION


```

0037 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0038
0039 ALGONAME = mfilename;
0040 VERSION  = '$Id: smoother.m,v 1.3 2008/03/02 19:17:38 hewitson Exp $';
0041 CATEGORY = 'Signal Processing';
0042 bs       = [];
0043
0044 % Check if this is a call for parameters
0045 if nargin == 2
0046     if isa(varargin{1}, 'ao') && ischar(varargin{2})
0047         in = char(varargin{2});
0048         if strcmp(in, 'Params')
0049             varargout{1} = getDefaultPlist();
0050             return
0051         elseif strcmp(in, 'Version')
0052             varargout{1} = VERSION;
0053             return
0054         elseif strcmp(in, 'Category')
0055             varargout{1} = CATEGORY;
0056             return
0057         end
0058     end
0059 end
0060
0061 % Collect input ao's, plist's and ao variable names
0062 in_names = {};
0063 for ii = 1:nargin
0064     in_names{end+1} = inputname(ii);
0065 end
0066
0067 [as, pl, aonames] = collect_inputs(varargin, in_names);
0068
0069 pl = combine(pl,

```

```
0093     disp(sprintf('* smoothing %s', a.name));
0094
0095     ys = smooth(d.y, bw, hc, method);006
          0075 %----- Make output A0
          008
          0095 % make data
          1003     _news =
```

```
(, 'ys', ysd);1801     _news =
```


[Home](#) > [classes](#) > [@ao](#) > spectrogram.m

spectrogram

PURPOSE [^]

SPECTROGRAM computes a spectrogram of the given ao/tsdata.

SYNOPSIS [^]

function varargout = spectrogram(varargin)

DESCRIPTION

SPECTROGRAM computes a spectrogram of the given ao/tsdata.

%%%

DESCRIPTION: SPECTROGRAM computes a spectrogram of the given ao/tsdata
using MATLAB's spectrogram function.

CALL: b = spectrogram(a, pl)

PARAMETERS:

'Win' - a specwin object [default: Kaiser -200dB psll]
'Nolap' - segment overlap [default: taken from window function]
'Nfft' - number of samples in each short fourier transform
 [default: sample rate of data]

CTe gfollowing Mcall.e turn a spraretrr tlistobject [thatcomntain ahe Tj 0 -1.2 TD (V


```
0119
0120 bo      = [];
0121
0122 % Check if this is a call for parameters
0123 if nargin == 2
```



```
0439         ''end_time'' and ''duration''. Using the parameter ''end_time'']')
0440 elseif ischar(duration) && ~isempty(duration)
0441     duration = getSeconds(duration, format_str);
                03 )
```

[illegible]

```
0553 % HI:TORY:      11-07-07 Diepholz
0554 %              Creation.
0555 %
0556 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0557
0558 function bool = isUTCformat\(time\_string\)
0559
0560     bool = false;
0561
0562     if ischar(time_string)
```



```
0036 ALGONAME = mfilename;
0037 VERSION   = '$Id: upsample.m,v 1.4 2008/03/02 18:45:41 hewitson Exp $';
0038 CATEGORY   = 'Signal Processing';
0039
0040 bs         = [];
0041
0042 %% Check if this is a call for parameters
```



```
>> pl = resp(mfir, 'Params', sets{jj})
```

The following call returns a string that contains the routine CVS version:

```
>> version = resp(mfir, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = resp(mfir, 'Category')
```

VERSION: \$Id: resp.m,v 1.12 2008/02/24 21:43:59 mauro Exp \$

HISTORY: 27-08-02 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORM%% % %196 0 0 14r9 /H2 <</M2ID 1 >>B.4801 RG


```

0077     return
0078 elseif strcmp(in, 'Category')
0079     varargout{1} = CATEGORY;
0080     return
0081 end
0082 end
0083
0084
0085 %% get input filter
0086 filt = varargin{1};
0087 if ~isa(filt, 'mfir')
0088     error('### first input must be an mfir filter object.');
```

```

0089 end
0090
0091
0092 %% Collects parameters
0093 % create empty input if it is not specified
0094 if nargin == 1
0095     pl = plist();
0096 else
0097     pl = varargin{2};
0098 end
0099
0100 % parse input parameter list
0101 if ~isempty(find(pl, 'f'))
0102     pl = combine(pl, getDefaultPL\('List'\));
0103 else
0104     pl = combine(pl, getDefaultPL\('Range'\));
0105 end
0106
0107
0108 % fill parameter list
0109 if isempty(find(pl, 'f'))
0110     % Compute from frequency range
0111     f1 = find(pl, 'f1');
0112     f2 = find(pl, 'f2');
0113     ndata = find(pl, 'nf');
0114     scale = find(pl, 'scale');
0115     switch scale
0116     case 'lin'
0117         f = linspace(f1, f2, ndata);
0118     case 'log'
0119         f = logspace(log10(f1), log10(f2), ndata);
0120     end
0121 else
0122     % Compute from frequency list
0123     f = find(pl, 'f');
```


0190

0191

0192 %%%%%%%%%%

0193

The following call returns a parameter list object that contains the default parameters for the jj-th set:

```
>> pl = resp(miir, 'Params', sets{jj})
```

The following call returns a string that contains the routine CVS version:

```
>> version = resp(miir, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = resp(miir, 'Category')
```

VERSION: \$Id: resp.m,v 1.10 2008/02/24 21:43:59 mauro Exp \$

NOTE: Eome of the code below is taken from Mathworks's treeplot.m

HISTORY: 27-08-02 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION


```

0129     if size(f,1) > 1
0130         f = f.';
0131     end
0132     ndata = length(f);
0133 end
0134
0135 % create output parameter list, with the frequency array values and the
0136 % input filter
0137 plo = plist('f', f, ...
0138     'filter', filt);
0139
0140
0141 %% compute Laplace vector
0142 s = -1i.*2*pi.*f;
0143
0144
0145 %% Compute filter response
0146 num = zeros(1, ndata);
0147 for n=1:length(filt.a)
0148     num = num + filt.a(n).*exp(s.*(n-1)/filt.fs);
0149 end
0150 denom = zeros(1, ndata);
0151 for n=1:length(filt.b)
0152     denom = denom + filt.b(n).*exp(s.*(n-1)/filt.fs);
0153 end
0154 dresp = num ./ denom;
0155 dresp = dresp .* filt.gain; % apply the gain
0156
0157 % mag    = 20*log10(abs(dresp));
0158 % phase = angle(dresp)*180/pi;
0159
0160
0161 %% Create an analysis object
0162
0163 % create new output fsdata
0164 fs = fsdata(f, dresp, filt.fs);
0165 fs = set(fs, 'name', sprintf('resp(%s)', filt.name));
0166
0167 % create new output history
0168 h = history(ALGONAME, VERSION, plo);
0169
0170 % make output analysis object
0171 b = ao(fs, h);
0172
0173 % set name
0174 b = setnh(b, 'name', sprintf('resp(%s)', filt.name));
0175
0176 % Outputs
0177 if nargout == 0
0178     iplot(b)
0179 end
0180
0181 if nargout == 1
0182     varargout{1} = b;
0183 end
0184 if nargout > 1
0185     error('incorrect output arguments');

```

```

0186 end
0187
0188
0189 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0190 function out = getDefaultPL(varargin)
0191
0192 % List of available parameter sets
0193 sets = {'List', 'Range'};
0194
0195 if nargin == 0
0196     out = sets;
0197     return
0198 end
0199
0200 set = varargin{1};
0201
0202 switch set
0203     case 'List'
0204         out = plist('f', []);
0205
0206         out = plist('f1', 0.001,...
0207                     'f2', 1,...
0208                     'nf', 1000,...
0209                     'scale', 'log');
0210     otherwise
0211         out = plist();
0212 end
0213
0214 % END
0215

```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#)

◦ [function plo = getDefaultPL\(\)](#)


```
0052 if ~isempty(filt)
0053     if isa(filt, 'miir') || isa(filt, 'mfir')
0054         b = _____
```

[Home](#) > [classes](#) > [@pole](#) > resp.m

```
>> version = resp(pole, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = resp(pole, 'Category')
```

VERSION: \$Id: resp.m,v 1.9 2008/02/24 21:43:59 mauro Exp \$

HISTORY: 03-04-2007 Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

TSis function calls:

- [char](#) CHAR convert a pole object into a string.
- [get](#) GET get pole properties.
- [set](#) >Tj /TT0 1 (cal%%%)TploCatiesDefaultPL(varar18 (VERSMCID 1 >>BDC BT /TT20 rg /TT0 1 Tf 11.480


```
0079 %% Input objects checks
0080 if nargin < 1
0081     error('### incorrect number of inputs.')
0082 end
0083
0084 % look at inputs
0085 p = varargin{1};
0085 8nd
```


[Home](#) > [classes](#) > [@pzmodel](#) > resp.m

resp

PURP**Ioe.P**^{Tj} ET EMC /HP<</MCID 7


```
>> category = resp(pzmodel, 'Category')
```

VERSION: \$Id: resp.m,v 1.16 2008/02/24 21:43:59 mauro Exp \$

HISTORY: 03-04-2007 M Hewitson
Creation

[illegible]


```
0077 end
0078
0079 %% Input objects checks
0080 if nargin < 1
0081     error('### incorrect number of inputs.')
0082 end
0083
0084 % look at inputs
0085 pzm  0 0 7.29o/r 1
0078
```

```
0134
0135 %% Compute or extrct <the frequency vector
0136 if ~isempty(find(pl, 'f'))
0137     % Compute from frequency list
0138     f      = find(pl, 'f');
0139     % Wan <to deal with rows
0140     if size(f,1) > 1
0141         f = f.';
0142     end
0143 else
0144     % Compute from frequency range
0145     f1 = find(pl, 'f1');
0146     f2 = find(pl, 'f2');
0147     nf = find(pl, 'nf');
0148     scale = find(pl, 'scale');
0149     switch scale
0150         case 'lin'
0151             f      = linspace(f1, f2, nf);
0152         case 'log'
```

```
0190 b = setnh(b, 'name', sprintf('resp(%s)', name), ...
0191     'mfilename', ALGONAME);
0192
```

0248 % END
0249
0250

```
>> version = resp(zero, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = resp(zero, 'Category')
```

VERSION: \$Id: resp.m,v 1.9 2008/02/24 21:43:59 mauro Exp \$

HISTORY: 03-04-2007 Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a zero object into a string.
- [get](#) GET get zero properties.
- [set](#) SET set a zero property.

This function is calleM by:

SUBFUNCTIONS

- [function plo = getDefaultPL\(varargin\)](#)

SOURCE CODE

```
0001 function \[f,r\] = resp\(varargin\)
0002 % RESP returns the complex response of the zero object.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: RESP returns the complex response of the zero object.
0007 %
0008 % CALL:          [f,r] = resp(z, f);          % compute response for vector f
0009 %               [f,r] = resp(z, f1, f2, nf); % compute response from f1 to f2
0010 %               % in nf steps.
0011 %               [f,r] = resp(z, pl);          % compute response from parameter
list.
0012 %               [f,r] = resp(z);              % compute response
```



```
0079 %% Input objects checks
0080 if nargin < 1
0081     error('### incorrect number of inputs.')
0082 end
0083
0084 % look at inputs
0085 z = varargin{1};
0086 if ~isa(z, 'zero')
0087     error('### first argument should be a zero object.');
```

0088 end

0089

0090

0091 % Now look at the zero

0092 f0 =

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [m](#) > [etc](#) > method_fcn_template.m

method_fcn_template

SYNOPSIS

function [Cdates] = fcn_template(varargin) Template for class methods.(s)Cdates)Cdate

SEE ALSO: `ao/cos`, `ao/tan`, `ao/asin`, `acos`, `atan`

SEE: <matlab: help ao/Contents;>HELP: ll Analysis Object Functions:
<matlab: helpwin ao/Contents;>DOC: ll Analysis Object Functions:

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

SUBFUNCTIONS

```
0035 % >> category = foo(ao, 'Category')
0036 %
0037 % VERSION:          $Id: method_fcn_template.m,v 1.5 2008/03/03 14:44:16 mauro Exp $
0038 %
0039 % HISTORY:           dd-mm-yyyy M Hewitson
0040 %                     Creation
0041 %
```

```
in, 'Params' %
```



```

0146     b = ao(xy, h);
0147
0148     % name, mfilename, description for this object
0149     b = setnh(b, 'name', sprintf('foo(%s)', invars{j}),...
0150         'description', find(pl,'description'));
0151
0152     bs = [bs b];
0153
0154 end
0155
0156
0157 %% Reshape the ouput to the same size of the input
0158 varargout{1} = reshape(bs, size(as));
0159
0160
0161 %% %%%%%%%%%%%
0162 %
0163 % FUNCTION:      getDefaultPL
0164 %
0165 % DESCRIPTION:  Get default params, version for standard usage
0166 %
0167 % HISTORY:      dd-mm-yyyy M Hewitson
0168 %               Creation
0169 %
0170 %%%%%%%%%%%
0171 E.645 845r83133 cmDC 0m 9.6.7(010 l S Q* n epe 0 BT /TT1 1 Tf 11.4801 0 0 11.

```

58 varin)71


```
0250 plo = append(plo, param('Kdes', 100));
0251 plo = append(plo, param('Jdes', 1000));
0252 plo = append(plo, param('Kmin', 2));
0253 plo = append(plo, param('Lmin', 0));
0254 plo = append(plo, param('Win', specwin('Kaiser', 1, 200)));
0255 plo = append(plo, param('Nolap', -1));
0256 plo = append(plo, param('Order', 0));
0257 disp('* done. ');
0258
0259
0260 %------
```

```

0307         win = specwin(win.name, l);
0308     end
0309     if l==Lmin
0310         minReached = 1;
0311     end
0312 end
0313
0314 % Compute DFT coefficients
0315 p    = 1i * twopi * m(fi)/l.*[0:l-1];
0316 C    = win.win .* exp(p);
0317
0318 if mod(fi,disp_each) == 0 || fi == 1 || fi == nf
0319     disp(sprintf('++ computing frequency %04d of %04d: %f Hz', fi, nf, f(fi)));
0320 end
0321
0322 % Loop over input channels
0323 for j=1:nc
0324     % loop over output channels
0325     for k=1:nc
0326         if k>j
0327             % Core cross-DFT part in C-mex file
0328             xi    = X(j,:);
0329             xo    = X(k,:);
0330             % We need cross-spectrum and Power spectrum
0331             [XY, XX, YY, nsegs] = ltpda_dft(xi, xo, l, C, olap, order);
0332             Txy(j,k,fi) = (abs(XY).^2)/(XX.*YY);
0333         else
0334             % Trivial case of equal input and output
0335             Txy(j,k,fi) = 1;
0336         end
0337     end
0338 end

```

[Home](#)


```
>> version = ltpda_lcpsd(ao, 'Version')
```

The following call returns a string that contains the routine category:

```
>> category = ltpda_lcpsd(ao, 'Category')
```

M Hewitson 02-02-07

\$Id: ltpda_lcpsd.m,v 1.12 2008/02/21 08:52:17 hewitson Exp \$

%%%

CROSS-REFERENCE INFORMATION

This function calls:

δ


```
0080 bs = [];  
0081  
0082 % check plist  
0083 if isempty(ps)  
0084     pl = getDefaultPL
```


0018 % Lmin - minimum segment length (default 0)

```

0075     elseif strcmp(in, 'Category')
0076         varargout{1} = CATEGORY;
0077         return
0078     end
0079 end
0080
0081 % capture input variable names
0082 invars = {};
0083 as      = [];
0084 ps      = [];
0085 for j=1:nargin
0086     invars = [invars cellstr(inputname(j))];
0082 a 4eisa( vararin{j}n, aoy')

0098     end
0902 a 4eisa( vararin{j}n, plisty')

0928     end
093     end
0940
0595
0565
0571 % heck plist
058a 4eisempty(ps')
0598 pls =d

```



```

0187 % create new output fsdata
0188 scale = find(pl, 'Scale');
0189 switch scale
0190     case 'AS'
0191         fs = fsdata(f, sqrt(P), d.fs);
0192         fs = set(fs, 'yunits', [d.yunits]);
0193     case 'ASD'
0194         fs = fsdata(f, sqrt(Pxx), d.fs);
0195         fs = set(fs, 'yunits', [d.yunits '\surdHz']);
0196     case 'PS'
0197         fs = fsdata(f, P, d.fs);
0198         fs = set(fs, 'yunits', [d.yunits '^2']);
0199     case 'PSD'
0200         fs = fsdata(f, Pxx, d.fs);
0201         fs = set(fs, 'yunits', [d.yunits '^2/Hz']);
0202     otherwise
0203         error(['### Unknown scaling:' scale]);
0204 end
0205 fs = set(fs, 'name', sprintf('lpsd(%s)', d.name));
0206 fs = set(fs, 'enbw', ENBW);
0207
0208 % create new output history
0209 h = history(ALGONAME, VERSION, pl, a.hist);
0210 h = set(h, 'invars', invars);
0211
0212 % make output analysis object
0213 b = ao(fs, h);
0214
0215 % set name
0216 if isempty(invars{i})
0217     n = a.name;
0218 else
0219     n = invars{i};
0220 end
0221 b = setnh(b, 'name', sprintf('lpsd(%s)', n), ...
0222         'mfilename', ALGONAME);
0223 bs = [bs b];
0224
0225 end
0226
0227 varargout{1} = bs;
0228
0229 % include AS
0230 % if nargsout > 1
0231 %     varargout{2} = bxx;
0232 % end
0233
0234
0235
0236 %-----
0237 %
0238 function varargout = mlpsd2\(varargin\)
0239
0240 % MLPSD2
0241 %
0242 % [S,Sxx,ENBW] = mlpsd2(x,f,r,m,L,fs,win,order,olap)
0243 %

```


40 L = varargin{5}; le
41 fs = varargin{6}; le
0301 % Core DFT part in C-mex file
12 ENBW(j) = fs*S2/S12; le
1302 Sxx(j) = A2ns/fs/S2; le
1402 S(j) = A2ns/S12; le
1502 le
16 endle

```

0358 i6 mod(j,disp_each) == 0 || j == 1 || j == nf
0359     disp(sprintf('++ computint 2<equency %d of %d: %f Hz', j, nf, f(j)));
0360 end
0361
0362 % compute DFT exponent and window
0363 l = L(j);
0364 switch win.name
0365     case 'Kaiser'
0366         win = specwin(win.name, l, win.psl1);
0367     otherwise
0368         win = specwin(win.name, l);
0369 end
0370
0371 p      = 1i * twopi * m(j)/l.*[0:l-1];
0372 C      = win.win .* exp(p);
0373 Xolap = (1-olap);
0374 % do segments
0375 A = 0.0;
0376 start = 1;
0377 nsegs = 0;
0378 while start + l - 1 <= nx
0379
0380     % get segment
0381     xs = x(start:start+l-1);
0382
0383     % detrend segment
0384     switch order
0385         case -1
0386             % do nothint
0387         case 0
0388             xs = xs - mean(xs);
0389         case 1
0390             xs = detrend(xs);
0391         otherwise
0392             xs = polydetrend(xs,= det    orend(xs);ese 1
0390 94
0395     % make DFT
0396     a = C*xs;
0397     A = A + a*conj(a    orend(xs);8
0399     % make next step
0400     start = floor(start + l*Xolap);
0401     nsegs = nsegs + 1;
0402 1
0390 403 T* (0390 404     i6 mod(j,disp_each) == 0 || j == 1 || j == nf)Tj 0390 405
0406 1

```


[Home](#) >


```
0082 % check plist
```

```
0083 if isempty(ps)
```

```
0084     pl = _____
```



```
0249 %-----
0250 % Returns the max Fs of a set of AOs
0251 function fs = findFsMax\(as\)
0252
0253 fs = 0;
0254 for j=1:length(as)
0255     a = as(j);
0256     if a.data.fs > fs
0257         fs = a.data.fs;
0258     end
0259 end
0260
0261 %-----
0262 % Get default params
0263 function plo = getDefaultPL\(\)
0264
0265 plo = plist();
0266 plo = append(plo, param('Kdes', 100));
0267 plo = append(plo, param('Jdes', 1000));
0268 plo = append(plo, param('Kmin', 2));
0269 plo = append(plo, param('Lmin', 0));
0270 plo = append(plo, param('Win', specwin('Kaiser', 1, 150)));
0271 plo = append(plo, param('Nolap', -1));
0272 plo = append(plo, param('Order', 0));
0273
0274
0275 %-----
0276 % Compute all TF combinations
0277 function varargout = mltfe\(varargin\)
0278
0279 % MLTFE
0280 %
0281 % Txy = mltfe(X,f,r,m,L,fs,win)
0282 %
0283 % M Hewitson 19-05-07
0284 %
0285 %
```



```
0084 % Loop over input AOs
0085 for j=1:na
0086
0087     % Unpack AO
0088     a = as(j);
0089     d = a.data;
0090
0091     if isa(a, 'ao')
0092         % check input data
0093         if isa(d, 'tsdata')
0094
0095             % combine plists
0096             plo = combine(pls,
```


SUBFUNCTIONS

- ⌘ [function plo = getDefaultPL\(\)](#)
- ⌘ [function plf = getlpFilter\(x\)](#)
- ⌘ [function phf = gethpFilter\(x\)](#)

SOURCE CODE

0001

```
0 46      return
0 47  elseif strcmp(in, 'Version')
0 48      varargout{1} = VERSION;
0 49      return
0 50  elseif strcmp(in, 'Category')
0 51      varargout{1} = CATEGORY;
0 52      return
0 53  end
0 54 end
0 55
```

```
854 end
```

```
as)~=2 Cat snrgo7t{1} erroey''only two analysis objects aTf needed!1      varargo7254 e
```



```
0213 varargout{1} = b;  
0214  
0215 % -----  
0216 % Get default params  
0217 function plo = getDefaultPL\(\)  
0218  
0219 7.5plistltPL;218 219,a ult p('lorhod', 'pfiear')PL;218 219,a ult p('addnoise', 'no')
```



```
0154 bo = [bo b];  
0155  
0156 varargout{1} = bo;  
0157 end
```

[Home](#) > [classes](#) > [@ao](#) > eq.m

[Home](#) > [classes](#) > [@ao](#) > ge.m

ge

PURPOSE 

```

0009 % CALL: a = b>=c;
0010 %
0011 % POSSIBLE VALUES: b: Analysis object
0012 %                      c: Analysis object or a scalar.
0013 %
0014 % VERSION: $Id: ge.m,v 1.7 2008/02/25 18:04:22 ingo Exp $
0015 %
0016 % HISTORY: 06-02-2007 M Hewitson
0017 %          Creation
0018 %
0019 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0020
0021 VERSION = '$Id: ge.m,v 1.7 2008/02/25 18:04:22 ingo Exp $';
0022 CATEGORY = 'Relational Operator';
0023
0024 %% Check if this is a call for parameters
0025 if nargin == 2
0026     if isa(b, 'ao') && ischar(c)
0027         in = char(c);
0028         if strcmp(in, 'Params')
0029             a = plist();
0030             return
0031         elseif strcmp(in, 'Version')
0032             a = VERSION;
0033             return
0034         elseif strcmp(in, 'Category')
0035             a = CATEGORY;
0036             return
0037         end
0038     end
0039 end
0040
0041 %% Create parameter list to get the y-axis values The value is not necessary.
0042 % It is only necessary that the value is set.
0043 pl = plist(param('ydata', 'x_xx_y'));
0044
0045 %% Get the data from analysis object
0046 [dummy, x] = get_xy_values(b.data, pl);
0047
0048 %% Get data from second input
0049 if isa(c, 'ao')
0050     [dummy, y] = get_xy_values(c.data, pl);
0051 else
0052     y = c*ones(size(x));
0053 end
0054
0055 %% Create Output
0056 a = x>=y;
0057

```

[Home](#) > [classes](#) > [@ao](#) > gt.m

gt

PURPOSE 

```

0009 % CALL: a = b>=;
0010 %
0011 % POSSIBLE VALUES: b: Analysis object
0012 %                      c: Analysis object or a scalar.
0013 %
0014 % VERSION: $Id: gt.m,v 1.6 2008/02/25 18:04:22 ingo Exp $
0015 %
0016 % HISTORY: 06-02-2007 M Hewitson
0017 %          Creation
0018 %
0019 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0020
0021 VERSION = '$Id: gt.m,v 1.6 2008/02/25 18:04:22 ingo Exp $';
0022 CATEGORY = 'Relational Operator';
0023
0024 %% Check if this is a call for parameters or for the cvs-version number
0025 if nargin == 2
0026     if isa(b, 'ao') && ischar(c)
0027         in = char(c);
0028         if strcmp(in, 'Params')
0029             a = plist();
0030             return
0031         elseif strcmp(in, 'Version')
0032             a = VERSION;
0033             return
0034         elseif strcmp(in, 'Category')
0035             a = CATEGORY;
0036             return
0037         end
0038     end
0039 end
0040
0041 %% Create parameter list to get the y-axis values
0042 pl = plist(param('ydata', 'x_xx_y'));
0043
0044 %% Get the data from analysis object
0045 [dummy, x] = get_xy_values(b.data, pl);
0046
0047 %% Get data from second input
0048 if isa(c, 'ao')
0049     [dummy, y] = get_xy_values(c.data, pl);
0050 else
0051     y = c*ones(size(x));
0052 end
0053
0054 %% Create Output
0055 a = x>y;
0056
0057 % END

```

le

PURPOSE

LE overloads <= operator for analysis objects. Compare the y-axis values.

SYNOPSIS

function a = le(b, c, varargin)

DESCRIPTION

LE overloads <= operator for analysis objects. Compare the y-axis values.

%%%

DESCRIPTION: LE overloads <= operator for analysis objects.
Compare the y-axis values.

CALL: a = b<;

POSSIBLE VALUES: b: Analysis object
c: Analysis object or a r2jech2l../ <= operator for analysis object.
s obj\$Id:/TT1 ,v 1 0 /08/02/</M18:04:22 ingo Exp \$rator forHISTORY: 06-02- /

0009 % CALL: a = b<;


```

0009 % CALL: a = b<c;
0010 %
0011 % POSSIBLE VALUES: b: Analysis object
0012 %                      c: Analysis object or a scalar.
0013 %
0014 % VERSION: $Id: lt.m,v 1.8 2008/02/25 18:04:22 ingo Exp $
0015 %
0016 % HISTORY: 06-02-2007 M Hewitson
0017 %          Creation
0018 %
0019 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0020
0021 VERSION = '$Id: lt.m,v 1.8 2008/02/25 18:04:22 ingo Exp $';
0022 CATEGORY = 'Relational Operator';
0023
0024 %% Check if this is a call for parameters or for the cvs-version number
0025 if nargin == 2
0026     if isa(b, 'ao') && ischar(c)
0027         in = char(c);
0028         if strcmp(in, 'Params')
0029             a = plist();
0030             return
0031         elseif strcmp(in, 'Version')
0032             a = VERSION;
0033             return
0034         elseif strcmp(in, 'Category')
0035             a = CATEGORY;
0036             return
0037         end
0038     end
0039 end
0040
0041 %% Create parameter list to get the y-axis values. The value is not necessary.
0042 % It is only necessary that the value is set.
0043 pl = plist(param('ydata', 'x_xx_y'));
0044
0045 %% Get the data from analysis object
0046 [dummy, x] = get_xy_values(b.data, pl);
0047
0048 %% Get data from second input
0049 if isa(c, 'ao')
0050     [dummy, y] = get_xy_values(c.data, pl);
0051 else
0052     y = c*ones(size(x));
0053 end
0054
0055 %% Create Output
0056 a = x<y;
0057
0058
0059
0060 % END

```

[Home](#) > [classes](#) > [@6o](#) > ne.m

ne

PURPOSE 

eq

PURPOSE

EQ overloads the == operator for cdata objects.

SYNOPSIS

function result = eq(c1,c2, varargin)

DESCRIPTION

EQ overloads the == operator for cdata objects.

%%%

DESCRIPTION: EQ overloads the == operator for cdata objects.

 All fields are checked.

CALL: result = eq(c1,c2)
 result = eq(c1,c2, exc_list)
 result = eq(c1,c2, 'property1', 'property2')
 result = eq(c1,c2, '0 rg /property', '0 rg /property')
 result = eq(c1,c2, plist('Exceptions', 'exception-list'))

PLIST: key: 'Exceptions'
 value: String or cell-array with exceptions.

EXAMPLES: result = eq(c1,c2, 'name', '0reated')
 result = eq(c1,c2, 'ao/name')
 result = eq(c1,c2, plist('Exceptions', 'name'))
 result = eq(c1,c2, plist('Exceptions', {'name', '0reated'}))

INPUTS: c1,c2 - input cdata objects
 exc_list - exception list
 List of properties which are not checked.

OUTPUTS: If the two cdata objects are considered equal, result == 1,
 otherwise, result == 0.

VERSION: \$Id: eq.m,v 1.15 2008/02/18 14:41:13 ingo Exp \$

HISTORY: 29-08-2007 M Hewitson
 Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

This function is called by:

• [ne](#) NE overloads the `~=` operator for cdata objects.

SOURCE CODE

```
0001 function result = eq\(c1,c2, varargin\)
```



```
0001 function result = ne\(c1, c2, varargin\)
0002 % NE overloads the ~= operator for cdata objects.
0003 %
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

eq

PURPOSE

EQ overloads the == operator for fsdata objects.

SYNOPSIS

function result = eq(c1,c2, varargin)

DESCRIPTION

EQ overloads the == operator for fsdata objects.

%%%

DESCRIPTION: EQ overloads the == operator for fsdata objects.

 All fields are checked.

CALL: result = eq(c1,c2)
 result = eq(c1,c2, exc_list)
 result = eq(c1,c2, 'property1', 'property2')
 result = eq(c1,c2, 'class/property', 'class/property')
 result = eq(c1,c2, plist('Exceptions', 'exception-list'))

PLIST: key: 'Exceptions'
 value: StrinLIST: key: S ell-array with))

This function is called by:

ð 

[Home](#) > [classes](#) > [@history](#) > ne.m

ne

PURPOSE

NE overloads the `~=` operator for history objects.

SYNOPSIS

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

This function is called by:

• [ne](#) NE overloads the $\sim$ = operator for mfir objects.



[illegible]

0001

0001

[Home](#) > [classes](#) > [@param](#) > ne.m

ne

PURPOSE 

eq

PURPOSE

EQ overloads the == operator for plist objects.

SYNOPSIS

function result = eq(c1, c2, varargin)

DESCRIPTION

EQ overloads the == operator for plist objects.

%%%

DESCRIPTION: EQ overloads the == operator for plist objects.

 All fields are checked.

CALL: result = eq(c1,c2)
 result = eq(c1,c2, exc_list)
 result = eq(c1,c2, 'property1', 'property2')
 result = eq(c1,c2, 'class/property', 'class/property')
 result = eq(c1,c2, plist('Exceptions', 'exception-list'))

PLIST: key: 'Exceptions'
 value: String or cell-array with exceptions.


```

0001 function result = ne\(p1, p2, varargin\)
0002 % NE overloads the ~= operator for param objects.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: NE overloads the ~= operator for param objects.
0007 %
0008 % CALL:          result = ne(p1,p2)
0009 %               result = ne(p1,p2, 'property1', 'property2')
0010 %               result = ne(p1,p2, 'class/property', 'class/property')
0011 %               result = ne(p1,p2, plist('Exceptions', 'exception-list'))
0012 %
0013 % PLIST:         key:    'Exceptions'
0014 %               value:  StrinN or cell-array with exceptions.
0015 %
0016 % INPUTS:        p1,p2 - input param objects
0017 %
0018 % OUTPUTS:       If the two param objects are considered equal, result == 0,
0019 %               otherwise, result == 1.
0020 %
0021 % VERSION:       $Id: ne.m,v 1.5 2008/02/res15:12:58 ingo Exp $
0022 %
0023 % HISTORY:       29-08-2007 M Hewitson<</MCID 1 >t% j e801 42.6e0t == 0,

```


[Home](#) > [classes](#)

Tis function is called by:

ø [ne](#) NE overloads the $\sim=$ operator for pole objects.

SOURCE CODE 

0001

eq

PURPOSE

EQ overloads the == operator for provenance objects.

SYNOPSIS

function result = eq(c1,c2, varargin)

DESCRIPTION

EQ overloads the == operator for provenance objects.

%%%

DESCRIPTION: EQ overloads the == operator for provenance objects.

 All fields are checked.

CALL: result = eq(c1,c2)
 result = eq(c1,c2, exc_list)
 result = eq(c1,c2, 'property1', 'property2')
 result = eq(c1,c2, 'class/property', 'class/property')
 result = eq(c1,c2, plist('Exceptions', 'exception-list'))

PLIST: key: 'Exceptions'
 value: String or cell-array with exceptions.

EXAMPLES: result = eq(c1,c2, 'namesr8 =9a Cerng or cell-arr 530l or cel cel
 value: String or cell-array with exceptions.c q(c- input%%%

EXA c- 'Excep xceptions.

This function is called by:

⁸ [ne](#) NE overloads the `~=` operator for provenance objects.

SOURCE CODE

```
0001 function result = eq(c1,c2, varargin)
0002 % EQ overloads the == operator for provenance objects.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: EQ overloads the == operator for provenance objects.t l (0005 %)T
0007 c1,c2, varargin)
```


[Home](#)

```
0001 function result = ne\(p1, p2, varargin\)
0002 % NE overloads the ~= operator for provenance objects.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: NE overloads the ~= operator for provenance objects.
0007 %
0008 % CALL:          result = ne(p1,p2)
0009 %              result = ne(p1,p2, 'property1', 'property2')
0010 %              result = ne(p1,p2, 'class/property', 'class/property')
```


[Home](#) > [classes](#) >

This function is called by:


```
0001 function result = ne\(p1, p2, varargin\)
```


[Home](#) > [classes](#) > [@specwin](#) > eq.m

eq

PURPOSE

EQ overloads the == operator for specfin objects.

This function is called by:

[Home](#) > [classes](#) > [@specwin](#) > ne.m

ne

PURPOSE

NE overloads the `~=` operator for specwin objects.

SYNOPSIS

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

This function is called by:

» [ne](#) NE overloads the `~=` operator for time objects.

SOURCE CODE

```

0001 function result = eq\(c1,c2, varargin\)
0002 % EQ overloads the == operator for time objects.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: EQ overloads the == operator for time objects.
0007 %
0008 %           All fields are checked.
0009 %
0010 % CALL:      result = eq(c1,c2)
0011 %           result = eq(c1,c2,  exc_list)
0012 %           result = eq(c1,c2, 'property1', 'property2')
0013 %           result = eq(c1,c2, 'class/property', 'class/property')
0014 %           result = eq(c1,c2,  plist('Exceptions', 'exception-list'))
0015 %
0016 % PLIST:      key:      'Exceptions'
0017 %           value:  String or cell-array with exceptions.
0018 %
0019 % EXAMPLES:   result = eq(c1,c2, 'name', 'created')
0020 %           result = eq(c1,c2, 'ao/name')
0021 %           result = eq(c1,c2,  plist('Exceptions', 'name'))
0022 %           result = eq(c1,c2,  plist('Exceptions', {'name', 'created'}))
0023 %
0024 % INPUTS:     c1,c2      - input time objects
0025 %           exc_list - exception list
0026 %           List of properties which are not checked.
0027 %
0028 % OUTPUTS:    If the two time objects are considered equal, result == 1,

3604 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
.83%
.8qaRSPTI      = '$Id:ed .m,v 1.7 8.28/02/00114:41:001trio Exp $';%%
.9 % TEGORY      = ' /Relaonal 0 operat';%%
40 % FAULT_% PLI =  plist);%%
4001
344 result geneetr_ = eq(c 1,c2, varargin, % FAULT_% PLI,qaRSPTI, % TEGORY);%%
33s.

```

[Home](#) > [classes](#) > [@time](#) > ne.m

ne

PURPOSE

NE overloads the `~=` operator for time objects.

SYNOPSIS

function result = ne(t1, t2, varargin)

DESCRIPTION

0001

[Home](#) > [classes](#) >

[Home](#) > [classes](#) > _____

0001

eq

PURPOSE

EQ overloads the == operator for timespan objects.

SYNOPSIS

function result = eq(c1,c2, varargin)

DESCRIPTION

EQ overloads the == operator for timespan objects.

%%

DESCRIPTION: EQ overloads the == operator for timespan objects.

 All fields are checked.

CALL: result = eq(c1,c2)
 result = eq(c1,c2, exc_list)
 result = eq(c1,c2, 'property1', 'property2')
 result = eq(c1,c2, 'class/property', 'class/property'Tj T* (

cnotc1,c2, exc_lc inpion.4ies wbjects.
 All fields are checked.

This function is called by:

ð 


```
0001 function result = ne\(ts1, ts2, varargin\)  
0002 % NE overloads the ~= operator for timespan objects.
```


[Home](#) > [classes](#) > [@tsdata](#) > eq.m

eq

PURPOSE 

[Home](#) > [classes](#) > [@tsdata](#) > ne.m

ne

PURPOSE 

NSYNOPSIS

```
0001 function result = ne\(c1, c2, varargin\)  
0002 % NE overloads the ~= operator for tsdata objects.  
0003 %
```


Tis function is called by:

ø [ne](#) NE overloads the $\sim=$ operator for xydata objects.

SOURCE CODE 

[illegible]

[Home](#) > [classes](#) > [@xyzdata](#) > eq.m

eq

PURPOSE

EQ overloads the == operator for xyzdata objects.

SYNOPSIS

[Home](#) > [classes](#) > [@xyzdata](#) > ne.m

ne

PURPOSE

NE overloads the `~=` operator for xyzdata objects.

SYNOPSIS

```

0002 % NE overloads the ~= operator for xyzdata objects.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: NE overloads the ~= operator for xyzdata objects.
0007 %
0008 % CALL:      result = ne(c1,c2)
0009 %            result = ne(c1,c2, 'property1', 'property2')
0010 %            result = ne(c1,c2, 'class/property', 'class/property')
0011 %            result = ne(c1,c2,  plist('Exceptions', 'exception-list'))
0012 %
0013 % PLIST:      key:    'Exceptions'
0014 %            value:  String or cell-array with exceptions.
0015 %
0016 % INPUTS:     c1,c2 - input xyzdata objects
0017 %
0018 % OUTPUTS:     If the two xyzdata objects are considered equal, result == 0,
0019 %            otherwise, result == 1.
0020 %
0021 % VERSION:     $Id: ne.m,v 1.3 2008/02/18 15:12:58 ingo Exp $
0022 %
0023 % HISTORY:     24-12-2007 M Hewitson
0024 %            Creation
0025 %
0026 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0027
0028 VERSION = '$Id: ne.m,v 1.3 2008/02/18 15:12:58 ingo Exp $';
0029 CATEGORY = 'Relational Operator';
0030
0031 % Chec0 if this is a call for parameters
0032 if nargin == 2
0033     if ischar(c2)
0034         in = char(c2);
0035         if strcmp(in, 'Params')
0036             result = plist;
0037             return
0038         elseif strcmp(in, 'Version')
0039             result = VERSION;
0040             return
0041         elseif strcmp(in, 'Category')
0042             60 2g, 'Category')
0043
0044
0045
0046
0047
0048j T*criptTption /Link

```

This function is called by:

ne

PURPOSE

NE overloads the ~= operator for zero objects.

SYNOPSIS

function result = ne(p1, p2, varargin)

DESCRIPTION

NE overloads the ~= operator for zero objects.

%%%

DESCRIPTION: NE overloads the ~= operator for zero objects.

CALL: result = ne(z1,z1)
 result = ne(p1,p2, 'property1', 'property2')
 result = ne(p1,p2, 'class/property', 'class/property')
 result = ne(p1,p2, plist('Exceptions', 'exception-list'))

PLIST: key: 'Exceptions'
 value: String or cell-array with exceptions.

sonvohe rwise,result = = 1


```
0153      % Collect this input history
0154      histin = [histin a.hist];
0155
0156      % Collect names, invars
0157      dname = [dname ', ' a.data.name];
```

```
0209
0210 varargout{1} = a;
0211
0212
0213 %-----
0214 % Get Offset of this set of time-vectors
0215 function Toff = getToff\(as\)
0216
0217 Toff = 1e20;
0218 for j=1:length(as)
0219     if isa(as(j).data, 'tsdata')
0220         t0 = as(j).data.t0.utc_epoch_milli;
0221         if t0 < Toff
0222             Toff = t0;
0223         end
0224     end
0225 end
0226
0227 %-----
0228 % Get default params
0229 function plo = getDefaultPL\(\)
0230
```


This function is called by:

SUBFUNCTIONS

```
0048         return
0049     end
0050 end
0051 end
0052
0053 % Collect input ao's, plist's and ao variable names
0054 in_names = {};
0055 for ii = 1:nargin
0056     in_names{end+1} = inputname(ii);
0057 end
0058
0059 [as, pl, invars] = collect_inputs(varargin, in_names);
0060
0061 op = '-';
0062
0063 [a1,a2,do] = aooperate(varargin, op);
0064
0065 %----- create output A0
0066
0067 % make a new history object
0068 h = history(ALGONAME, VERSION, plist(), [a1.hist a2.hist]);
0069 h = set(h, 'invars', invars);
0070
0071 % get names for output
0072 if isempty(char(invars{1}))
0073     n1 = a1.name;
0074 else
```


[Home](#) > [classes](#) > [@ao](#) > mpower.m

mpower


```
0045     elseif strcmp(in, 'Version')
0046         b = VERSION;
0047         return
0048     elseif strcmp(in, 'Category')
0049         b = CATEGORY;
0050         return
0051     end
0052 end
0053 end
```


This function is called by:

SUBFUNCTIONS

↳ [function pl_default = getDefaultPL\(\)](#)

SOURCE CODE

[Home](#) > [classes](#) > [@ao](#) > plus.m

plus

PURPOSE

PLUS implements addition operator for two an4lysis objects.

SYNOPSIS

function b = plus(varargin)

DESCRIPTION


```
0048         return
0049     end
0050 end
0051 end
0052
0053 % Collect input ao's, plist's and ao variable names
0054 in_names = {};
0055 for ii = 1:nargin
0056     in_names=end+1} = inputname(ii);
0057 end
0058
0059 [as, pl, invars] = collect_inputs(varargin, in_names);
0060
0061 op = '+';
0062
0063 [a1,a2,do] = aooperate(varargin, op);
0064
0065 %----- create output A0
0066
0067 % make a new history object
0068 h = history(ALGONAME, VERSION, plist(), [a1.hist a2.hist]);
0069 h =
```



```
0047     elseif strcmp(in, 'Category')
0048         b = CATEGORY;
0049         return
0050     end
0051 end
0052 end
005ai004in_names{052+1}    inputnamercmi)GORY;
varar /P,4in_names)GORY;varar /P,4op)GORY;ALGONAME, VERSION, plist(), [a1.hist a2.his
```

PURPOSE

SYNOPSIS

DESCRIPTION [illegible]

```
CALL:      a = a1./scalar
```

VERSION: \$Id: rdivide.m,v 1.16 2008/03/04 17:28:53 ingo Exp \$

The following call returns a parameter list object that contains the default parameter values:

```
>> pl = rdivide(ao, 'Params')nlct t
```

```
ao, 'Params')nlct t
```

```
%%%% = rdivide(varargin9
```



```

0049     end
0050 end
0051
0052 % Collect input ao's, plist's and ao variable names
0053 in_names = {};
0054 for ii = 1:nargin
0055     in_names{end+1} = inputname(ii);
0056 end
0057
0058 [as, pl, invars] = collect_inputs(varargin, in_names);
0059
0060 op = './';
0061
0062 [a1,a2,do] = aooperate(varargin, op);
0063
0064 %----- create output A0
0065
0066 % make a new history object
0067 h = history(ALGONAME, VERSION, plist(), [a1.hist a2.hist]);
0068 h = set(h, 'invars', invars);
0069
0070 % get names for output
0071 if isempty(char(invars{1}))
0072     n1 = a1.name;
0073 else
0074     n1 = char(invars{1});
0075 end
0076 if isempty(char(invars{2}))
0077     n2 = a2.name;
0078 else
0079     n2 = char(invars{2});
0080 end
0081
0082 % make output analysis object
0083 b = ao(do, h);
0084 b = setnh(b, 'name', sprintf('%s %s %s', n1, op, n2));
0085
0086
0087 %% Get default params
0088 function pl_default = getDefaultPL()
0089
0090     pl_default = plist()Tj ET EMC   91
0092 % END

```



```
0045         b = VERSION;
0046         return
0047     elseif strcmp(in, 'Category')
0048         b = CATEGORY;
0049         return
0050     end
0051 end
0052 end
0053
0054 % Collect input ao's, plist's and ao variable names
0055 in_names = {};
0056 for ii = 1:nargin
0050 r in_name{ en+1}i = inpu_namp(i));
008  end
0093
060 [a's, p,= i vas]i =cCollec_ inpus( vanargi,r in_name));
0613
062 Top= '.*';
0653
064 [a1,a2,do]i =aoooperCat( vanargi,rop));
0653
066 %----- creCat outnputA0
0673
0684 %make a new hlisgor objlecd
069 hi =hlisgor(ALGONAME,= VERSIOs, plis(), [a1.hlis a2.hlis]));
070 hi =s
```



```
0003 %
0004 %%%%%%%%%%
0005 %
0006 % DESCRIPTION: MINUS overloads - operator for analysis objects.
0007 %           This function subtracts a time in seconds from a time object.
```



```
0057     t_format_seconds = java.text.SimpleDateFormat(java_format_str_seconds);
0058     t_format_seconds.setTimeZone(utc_timezone);
0059
0060     epoch_time_seconds = t_format_seconds.parse(t2);
0061     epoch_time_seconds = epoch_time_seconds.getTime;
0062
0063     res = set(t1, 'utc_epoch_milli', t1.utc_epoch_milli+epoch_time_seconds);
0064
0065 %%%% >> number = t1 + t2; %%%%
0066 elseif isa(t2, 'time')
0067
```



```
0012 % HISTORY:          31-01-07 M Hewitson
0013 %                  Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION = '$Id: subsasgn.m,v 1.9 2008/02/25 18:06:31 ingo Exp $';
0018
0019 % Check if this is a call for parameters or for the cvs-version number
0020 if nargin == 2
0021     if isa(A, 'ao') && ischar(S)
0022         in = char(S);
0023         if strcmp(in, 'Params')
0024             A = plist();
0025             retur* (0014 %)T26         elseif strcmp(in, 'Version')
0027             A = VERSION;
0028             retur* (0014 %)T29         elseif strcmp(in, 'Category')
0030             CATEGORY = 'Internal';
0031             A = CATEGORY;
```

subsref

PURPOSE

SUBSREF Define field name indexing for analysis objects.

SYNOPSIS

function B = subsref(A, S)

DESCRIPTION

SUBSREF Define field name indexing for analysis objects.

%%%

DESCRIPTION: SUBSREF Define field name indexing for analysis objects.

% HISTORY SSSSS31-01-07 M Hewitsn Tj 0 -21.2TD (%) Cretion Tj 0 -2.4 T

```
0023      return
```


[Home](#) > [classes](#) > [@cdata](#) > set_xy_axis.m

set_xy_axis

PURPOSE

```
0012 %
0013 % HISTORY: 21-06-2007 Diepholz
0014 %          Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION   = '$Id: set_xy_axis.m,v 1.4 2008/02/12 19:54:05 hewitson Exp $';
0019 CATEGORY = 'Internal';
0020
0021 2131em,v 1.5 hewitson D this is a call for para0 1ers2131em,v 1.2n D narx [ == 2
```



```
0043 if nargin == 2
0044     if isa(varargin{1}, 'cdata') && ischar(varargin{2})
0045         in = _____
```


[Home](#) > _____


```
0059 else %if nargout == 2
0060     pl_default = plist([param('xdata', 'x')
0061                        param('ydata', 'y')]);
0062 end
0063
0064
0065 %% Combine the handover param-0 M C95 8 pl_defaver param-064
pl 'x')

'xaulfind(pl,6am('xd'))];'xaulfind(pl,6ay('xd'))];0065Make surmbina-0 t leam-0oombaxis
```


[Home](#) > [classes](#) > [@fsdata](#) > reshapeF.m

reshapeF

PURPOSE 

```
0008 % CALL:          fsd = reshapeF(fsd)
0009 %
0010 % INPUT:          fsd - frequency-series object
0011 %
0012 % OUTPUTt 10.     INPT:          fsd - frequency-series obje3t
0010 $Id:              f.m,v 1.5   0 8/02/1 %22:11:21 hewitsTm Exp $shapeF(fsd)fsd)

00118t
```

[Home](#) > [classes](#) > [@fsdata](#) > setFreq.m

setFreq

PURPOSE

SETFREQ sets the frequency vector of a fsdata object based on the length of the y vector and the sample rate.

SYNOPSIS

function fsd = setFreq(fsd, varargin)

DESCRIPTION

SETFREQ sets the frequency vector of a fsdata object based on the length of the y vector and the sample rate.

%%%

DESCRIPTION: SETFREQ sets the frequency vector of a fsdata object based on the length of the y vector and the sample rate.


```
0012 %
0013 % HISTORY: 21-06-2007 Diepholz
0014 %           Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION    = '$Id: set_xy_axis.m,< 1.5 2008/02/12 22:11:21 hewitson Exp $';
0019 CATEGORY = 'Internal';
0020
0021 % Check if this is a call for parameters
0022 if nargin == 2
0023     if isa(data, 'fsdata') && ischar(x)
0024         in = char(x);
0025         if strcmp(in, 'Params')
0026             data_out = plist;
0027             return
0028         elseif strcmp(in, 'Version')
```

[Home](#) > [classes](#) > [@fsdata](#) > single_operation.m

single_operation

PURPOSE 


```
0038 %
0039 % HISTORY: 07-05-2007 Diepholz
0040 %           Creation.
0041 %
0042 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0043
0044 VERSION   = '$Id: single_operation.m,v 1.12 2008/02/12 22:11:21 hewitson Exp $';
0045 CATEGORY = 'Internal';
0046
0047 % Check if this is a call for parameters
0048 if nargin == 2
0049     if isa(varargin{1}, 'fsdata') && ischar(varargin{2})
0050         in = char(varargin{2});
0051         if strcmp(in, 'Params')
0052             vararginout{1} = getDefaultPL();
```

```
0095 option      = find(pl, 'option');
0096 dim          = find(pl, 'dim');
0097 weight_vector = find(pl, 'w');
0098 do_xdata      = find(pl, 'xdata');
0099 do_ydata      = find(pl, 'ydata');
0100
0101 %% Check the <char> option if it begins and ends with a quote '
0102 %% If not then add the quote.
0103 if ischar(option) && ~isempty(option)
0104     if ~strcmp(option(1), '')
0105         option = ['' option];
0106     end
0107     if ~strcmp(option(end), '')
0108         option = [option ''];
0109     end
0110 end
0111
0112 %% Convert the <double> option into a char
0113 if isnumeric(option) || islogical(option)
0114     option = num2str(option);
0115 end
0116
0117 %% Convert the dim into a char
0118 if isnumeric(dim) || islogical (dim)
```



```

0263         eval (sprintf('data_out%d.yunits = '%s(%s)'';',i,op,data.yunits));
0264     end
0265     %% t data (x-axis)
0266     if ~isempty(do_xdata)
0267         eval (sprintf('data_out%d.x = x_new%d;',i,i));
0268         eval (sprintf('data_out%d.name = '%s(%s)'';',i,op,data.name));
0269         eval (sprintf('data_out%d.xunits = '%s(%s)'';',i,op,data.xunits));
0270     end
0271
0272     %% Write to output
0273     eval (sprintf('varargout{i+1} = data_out%d;',i))
0274
0275     end
0276
0277     %% Create a new history object
0278     h = history(op, VERSION);
0279
0280     varargout{1} = h;
0281
0282     else % (nargout < 2)
0283         error ('### fsdata/single_operation: This function need at least two output
variables. ');
0284     end
0285
0286 else

```


[Home](#) > [classes](#) > [@history](#) > getNodes.m

getNodes

PURPOSE 


```
0057     h.n = a;
0058 end
0059
0060 % set my parent node
0061 nn = h.pn;
0062 if nn<0
0063     h.pn = pn;
0064     n = [n pn];
0065     pl = h.plist;
0066     nodes(a).pn      = pn;
0067     nodes(a).invars = h.invars;
0068     nodes(a).n       = h.n;
0069     nodes(a).names   = h.name;
0070     nodes(a).params = char
```


PURPOSE

SYNOPSIS

DESCRIPTION [illegible]

HISTORY: 31-01-07 M Hewitson
Creation

[illegible]


```
0012 % HISTORY:      31-01-07 M Hewitson
```

```
0013 %              Creation
```

```
0014 %
```

```
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
0016
```

```
0017 % Check if this is a call for parameters
```

```
0018 if nargin == 2
```

```
0019     if isa(A, 'mfir') && ischar(S)
```

```
0020         in =
```

subsref

PURPOSE

SUBSREF Define field name indexing for mfir objects.

SYNOPSIS

function B = subsref(A, S)

DESCRIPTION

SUBSREF Define field name indexing for mfir objects.

%%%

DESCRIPTION: SUBSREF Define field name indexing for mfir objects.

EXAMPLES: All possible accesses are possible.

VERSION: \$Id: subsref.m,v 1.4 2008/02/22 22:26:40 hewitson Exp \$

HISTORY: 31-01-07 M Hewitson
Creation

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a mfir object into a string.

[Home](#)

```
0012 % HISTORY:      31-01-07 M Hewitson
0013 %              Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if nargin == 2
0019     if isa(A, 'miir') && ischar(S)
0020         in = char(S);
0021         if strcmp(in, 'Params')
0022             A = plist;
0023             return
0024         elseif strcmp(in, 'Version')
0025             VERSION = '$Id: subsasgn.m,      return
```

[Home](#) > [classes](#) > [@miir](#) > subsref.m

subsref

PURPOSE

SUBSREF Define field name indexing for miir objects.

SYNOPSIS

function B = subsref(A, S)

DESCRIPTION

[Home](#) > _____

[Home](#) > [classes](#) >

subsasgn

PURPOSE

SUBSASGN define index assignment for parameter list properties.

SYNOPSIS

function A = subsasgn(A, S, B)

DESCRIPTION

SUBSASGN define index assignment for parameter list properties.

%%%

DESCRIPTION: SUBSASGN define index assignment for parameter list properties.

EXAMPLES: All possible assignment are possible.

VERSION: \$Id: subsasgn.m,v 1.9 2008/02/25 18:06:31 ingo Exp \$

HISTORY: 31-01-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

```
0011 % VERSION:      $Id: subsasgn.m,v 1.0 2008/02/25 18:06:31 ingo Exp $
0012 %
0013 % HISTORY:       31-01-07 M Hewitson
0014 %               Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 % Check if this is a call for parameters
0019 if nargin == 2
0020     if isa(A, 'plist') && ischar(S)
0021         in = char(S);
0022         if strcmp(in, 'Params')
0023             A = plist;
0024             return
0025         elseif strcmp(in, 'Version')
0026             VERSION = '$Id: subsasgn.m,v 1.0 2008/02/25 18:06:31 ingo Exp $';
0027             A = VERSION;
0028             return
0029         elseif strcmp(in, 'Category')
0030             CATEGORY = 'Internal';
0031             A = CATEGORY;
0032             return
0033         end
0034     end
0035 end
0036
0037 if isempty(A)
0038     A = plist
```


0012	%	HISTORY:	31-01-07	M Hewitson
0013	%		Creation	
0014	%			

0060 % END

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [classes](#) > [@pole](#) > subsref.m

[Home](#) > [classes](#) > [@provenance](#) > subsasgn.m


```
0013 %                               Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if ischar(S)
0019     in = char(S);
0020     if strcmp(in, 'Params')
0021         B = plist;
0022         return
0023     elseif strcmp(in, 'Version')
0024         VERSION = '$Id: subsref.m,v 1.5 2008/02/22 22:26:41 hewitson Exp $';
0025         B = VERSION;
0026         return
0027     elseif strcmp(in, 'Category')
0028         CATEGORY = 'Internal';
0029         B = CATEGORY;
0030         return
0031     end
0032 end
0033
0034 B = builtin('subsref', A, S);
0035
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#)

[Home](#) > [classes](#) > [@pzmodel](#) > getupperFreq.m

getupperFreq

PURPOSE 

[Home](#) > [classes](#) > [@pzmodel](#) > subsasgn.m

subsasgn

PURPOSE

SUBSASGN define index assignment for pzmodel properties.

SYNOPSIS `S = subsasgn(A, 'S', 0, 'Am@5bT', '/TT1 1 Tf', 15.8956, 0, 0, 15.8956, 8.2001, 66, 'ties');`

```

0012 % HISTORY:      31-01-07 M Hewitson
0013 %              Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if nargin == 2
0019     if isa(A, 'pzmodel') && ischar(S)
0020         ii = char(S);
0021j ET Ef strcmp(in, 'Params')
0022         A = plist;
0023         return
0024     elseif strcmp(in, 'Version')
0025         VERSION = '$Id: subsasgn.m,v 1.4 2008/02/25 18:06:31 iigo Exp $';
0026         A = VERSION;
0027         return
0028     elseif strcmp(in, 'Category')
0029         CATEGORY = 'Internal';
0030         A = CATEGORY;
0031         return
0032     end
0033 end
0034 end
0035
0036 if isempty(A)
0037     A = pzmodel();
0038 end
0039
0040 A = builtii ('subsasgn', A, S, B);
0041

```

Generated on Fri 07-Mar-2008 14286.7999 379 /P6 strcmp(in, 'Version')

0mkOCL Cm98551 3 0.3664 cm 0 0 m3 8.637 0l S Q BT 0 0 1 rg /T20 1 Tf 11.4801 0 0 11.4801L Cm98551 362.189


```
0013 %                               Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if ischar(S)
0019     in = char(S);
0020     if strcmp(in, 'Params' 8.2001 n1      B  iplist;s' 8.2001 n2      returns' 8.200
```

[Home](#) > [classes](#) > _____

Descri9.42 % HISTORY: 31-01-07 M HewitsoMC E0 -1.DesDri9.43 % Cro

[Home](#) > [classes](#) > [@specwin](#) > subsref.m

subsref

PURPOSE 


```
0012 % HISTORY:      31-01-07 M Hewitson
0013 %              Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if nargin == 2
0019     if isa(A, 'time') && ischar(S)
0020tr17 0016
```

ind criFri 07-Mar-,v . 15:46:-9-by7 CA6

subsref

PURPOSE

SUBSREF Define field name indexing for time objects.

SYNOPSIS

function B = subsref(A, S)

DESCRIPTION

SUBSREF Define field name indexing for time objects.

%%%

DESCRIPTION: SUBSREF Define field name indexing for time objects.

EXAMPLES: All possible accesses are possible.

[Home](#) > [classes](#) > [@timeformat](#) > subsref.m

[Home](#)

[Home](#) > [classes](#) > [@timespan](#) > subsref.m

subsref

PURPOSE

SUBSREF Define field name indexing for **timespan** objects.

SYNOPSIS

[Home](#) > [classes](#) > [@tsdata](#) > get_xy_values.m

get_xy_values

PURPOSE

SYNOPSIS

function varargout = get_xy_values(data, pl) DescrES.91:xO/c2ET EMC /I <</MCID 12 >>B6C 1 0 0 RG 1 w 82


```

0012 % HISTORY:      31-01-2007 M Hewitson
0013 %              Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 VERSION  = '$Id: reshapeT.m,v 1.5 2008/02/13 11:39:35 mauro Exp $';
0018 CATEGORY = 'Internal';
0019
0020 if nargin == 2 && isa(ts, 'tsdata') && ischar(varargin{1})
0021     if strcmp(varargin{1}, 'Params')
0022         ts = plist();
0023         return
0024     elseif strcmp(varargin{1}, 'Version')
0025         ts = VERSION;
0026         return
0027     elseif strcmp(varargin{1}, 'Category')
0028         ts = CATEGORY;
0029         return
0030     end
0031 end
0032
0033 if size(ts.x, 1) ~= size(ts.y,1)
0034     ts.x = [ts.x].';
0035 end
0036
0037 % END

```



```

0009 % CALL:          tsdata = setTime(tsdata)
0010 %
0011 % VERSION:       $Id: setTime.m,v 1.7 2008/02/13 12:12:49 mauro Exp $
0012 %
0013 % HISTORY:        31-01-2007 M Hewitson
0014 %                  Creation
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 VERSION = '$Id: setTime.m,v 1.7 2008/02/13 12:12:49 mauro Exp $';
0019 CATEGORY = 'Internal';
0020
0021 if  nargs == 2 && isa(ts, 'tsdata') && ischar(varargin{1})
0022     if strcmp(varargin{1}, 'Params')
0023         ts = plist();
0024         return
0025     elseif strcmp(varargin{1}, 'Version')
0026         ts = VERSION;
0027         return
0028TD (0025     elseif strcmp(varargin2 /05 0ry)

```


0012 %
0013 % HISTORY: 21-06-2007 Diepholz
0014 % Creation

[Home](#) > [classes](#) > [@tsdata](#) > single_operation.m

single_operation

PURPOSE

SINGLE_OPERATIOT /TT1 1 Tf 11.4Alements specifiypej ET Eor overload for EMC objects./H2 <</MCID 7 >


```

0095 option      = find(pl, 'option');
0096 dim          = find(pl, 'dim');
0097 weight_vector = find(pl, 'w');
0098 do_xdata      = find(pl, 'xdata');
0099 do_ydata      = find(pl, 'ydata');
0100
0101 %% Check the <char> option if it begins and ends with a quote '
0102 %% If not then add the quote.
0103 if ischar(option) && ~isempty(option)
0104     if ~strcmp(option(1), '')
0105         option = [' ' option ' '];
0106 pl, '1091ttion la');
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000

```


0012 % HISTORY: 31-01-07 M Hewitson

[Home](#) > [classes](#) > [@tsdata](#) > subsref.m

```
0013 %                               Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if ischar(S)
0019     in = char
```


[Home](#) > [classes](#)

set_xy_axis

PURPOSE

SET_XY_AXIS set the x-axis and y-axis values.

SYNOPSIS

function data_out = set_xy_axis(data, x, y)

DESCRIPTION

SET_XY_AXIS set the x-axis and y-axis values.

%%%

DESCRIPTION: SET_XY_AXIS set the x-axis and y-axis values.

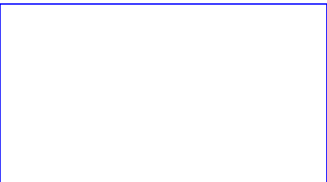
CALL: data = set_xy_axis (data, x, y);
 ao.data = set_xy_axis (ao.data, x, y);

VERSION: \$Id: set_xy_axis.m,v 1.4 2008/02/13 12:57:47 mauro Exp \$

HISTORY: 21-06-2007 Diepholz
 Creation

%%%

CROSS-REFERENCE INFORMATION



0012 %
0013 % HISTOF
0014 %
0015 %
0016 %%%%%%%%%
0017
0018 VERSION
0019 CATEGORY
0020
0021 % Check
0022 if nargi


```
0038 %
0039 % HISTORY: 07-05-2007 Diepholz
0040 %          Creation.
0041 %
0042 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0043
0044 VERSION   = '$Id: single_operation.m,v 1.12 2008/02/13 12:57:47 mauro Exp $';
0045 CATEGORY = 'Internal';
0046
0047 % Check if this is a call for parameters
0048 if nargin == 2
0049     if isa(varargin{1}, 'xydata') && ischar(CATEGORY) == 'IntelT
00359     On ::=
```



```

0207         end
0208
0209     end
0210
0211 %% Execute the command for the ydata
0212 if ~isempty(do_ydata)
0213
0214     try
0215         %% Execute with a option or a dimension(option)
0216         cmd_line = sprintf('%s = %s(y,%s);',y_new, op, option);
0217         eval (cmd_line);
0218
0219         %% Execute with a weight vector and dimension
0220
0221         cmd21 _ %%d_line = sprintf(' %% if25ption);
0222
0223         %% Execute with a op28 dimension
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702
0703
0704
0705
0706
0707
0708
0709
0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739
0740
0741
0742
0743
0744
0745
0746
0747
0748
0749
0750
0751
0752
0753
0754
0755
0756
0757
0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
0775
0776
0777
0778
0779
0780
0781
0782
0783
0784
0785
0786
0787
0788
0789
0790
0791
0792
0793
0794
0795
0796
0797
0798
0799
0800
0801
0802
0803
0804
0805
0806
0807
0808
0809
0810
0811
0812
0813
0814
0815
0816
0817
0818
0819
0820
0821
0822
0823
0824
0825
0826
0827
0828
0829
0830
0831
0832
0833
0834
0835
0836
0837
0838
0839
0840
0841
0842
0843
0844
0845
0846
0847
0848
0849
0850
0851
0852
0853
0854
0855
0856
0857
0858
0859
0860
0861
0862
0863
0864
0865
0866
0867
0868
0869
0870
0871
0872
0873
0874
0875
0876
0877
0878
0879
0880
0881
0882
0883
0884
0885
0886
0887
0888
0889
0890
0891
0892
0893
0894
0895
0896
0897
0898
0899
0900
0901
0902
0903
0904
0905
0906
0907
0908
0909
0910
0911
0912
0913
0914
0915
0916
0917
0918
0919
0920
0921
0922
0923
0924
0925
0926
0927
0928
0929
0930
0931
0932
0933
0934
0935
0936
0937
0938
0939
0940
0941
0942
0943
0944
0945
0946
0947
0948
0949
0950
0951
0952
0953
0954
0955
0956
0957
0958
0959
0960
0961
0962
0963
0964
0965
0966
0967
0968
0969
0970
0971
0972
0973
0974
0975
0976
0977
0978
0979
0980
0981
0982
0983
0984
0985
0986
0987
0988
0989
0990
0991
0992
0993
0994
0995
0996
0997
0998
0999
1000

```


subsref

PURPOSE

SUBSREF Define field name indexing for xydata objects.

SYNOPSIS

function B = subsref(A, S)

DESCRIPTION

SUBSREF Define field name indexing for xydata objects.

%%%

DESCRIPTION: SUBSREF Define field name indexing for xydata objects.

EXAMPLES: All possible accesses are possible.

VERSION: \$Id: subsref.m,v 1.8 2008/02/22 22:26:40 hewitson Exp \$

HISTORY: 31-01-07 M Hewitson
Creation

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [char](#) CHAR convert a param object into a string.

This function is called by:

SOURCE CODE

```
0001 function B = subsref\(A, S\)
0002 % SUBSREF Define field name indexing for xydata objects.
0003 %
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: SUBSREF Define field name indexing for xydata objects.
0007 %
0008 % EXAMPLES: All possible accesses are possible.
0009 %
0010 % VERSION: $Id: subsref.m,v 1.8 2008/02/22 22:26:40 hewitson Exp $
0011 %
0012 % HISTORY: 31-01-07 M Hewitson
```

S):bsre-18 -1.343 Td4201


```

0057 x  = [];
0058 y  = [];
0059
0060 %% Set default parameter list
0061 if nargin == 1
0062     pl_default = plist(param('xdata', 'x'));
0063 elseif nargin == 2
0064     pl_default = plist([param('xdata', 'x')
0065                        param('ydata', 'y')]);
0066 elseif nargin == 3
0067     pl_default = plist('xdata', 'x', 'ydata', 'y', 'zdata', 'z');
0068 else
0069     error('### Incorrect outputs. ');
0070 end
0071
0072
0073 %% Combine the handover param-list and the default param-list
0074 if ~isempty(pl)
0075     pl = combine(pl);
0076 else
0077     pl = combine(pl_default);
0078 end
0079
0080 do_xdata = find(pl, 'xdata');
0081 do_ydata = find(pl, 'ydata');
0082 do_zdata = find(pl, 'zdata');
0083
0084 %% Make sure that at least one axis is defined
0085 if isempty(do_xdata) && isempty(do_ydata)
0086     do_xdata = 'x';
0087 end
0088
0089 %% Set x-axis and y-axis values
0090 if ~isempty(do_xdata)
0091     x = data.x;
0092 end
0093
0094 if ~isempty(do_ydata)
0095     y = data.y;
0096 end
0097
0098 if ~isempty(do_zdata)
0099     z = data.z;
0100 end
0101
0102 %% Set outputs
0103 if nargin == 1
0104     if ~isempty(x)
0105         varargout{1} = x;
0106     else % ~isempty(y)
0107         varargout{1} = y;
0108     end
0109
0110 elseif nargin == 2
0111     varargout{1} = x;
0112     varargout{2} = y;

```

```
0113
0114 elseif nargout == 3
0115     varargout{1} = x;
0116     varargout{2} = y;
0117     varargout{3} = z;
0118 else
0119     error('### Incorrect output arguments. ');
0120 end
0121
0122 % END
```

```
>> operation(data,w,dim)
>> operation(data,dim)
>> operation(data,w)
>> operation(data)
```

HISTORY: 24-12-2007 Diepholz
Creation.

%%

CROSS-REFERENCE INFORMATION


```
0091     error ('### xyzdata/single_operation: The third input must be a plist.');
```

0092 end

0093

0094 %% Set default parameter list

0095 pl_default =


```

Have a look to the param list', cmd_line);
0260     end
0261
0262     end
0263
0264 %%    %% The result from eval is in x_new1, y_new1, ...
0265     %% Create a xydata object from this results and write it into the output
0266     for i = 1:nargout-1
0267
0268         %% Create xydata
0269         if ~isempty(do_xdata)
0270             eval (sprintf('data_out%d.x = x_new%d;',i,i));
0271             eval (sprintf('data_out%d.name = ''%s(%s)'';',i,op,data.name));
0272             eval (sprintf('data_out%d.xunits = ''%s(%s)'';',i,op,data.xunits));
0273         end
0274         if ~isempty(do_ydata)
0275             eval (sprintf('data_out%d.y = y_new%d;',i,i));
0276             eval (sprintf('data_out%d.name = ''%s(%s)'';',i,op,data.name));
0277             eval (sprintf('data_out%d.yunits = ''%s(%s)'';',i,op,data.yunits));
0278         end
0279
0280         %% Write to output
0281         eval (sprintf('varargout{i+1} = data_out%d;',i))
0282
0283     end
0284
0285     %% Create a new history0 -1.2 0e to4
'data_out%d.rgout-1

0279

02else %ta_68     < 2end
09ut{i+1errTj ,i###empty(/</MCID 0 >>BDC :w1,he func ET /nex [at le st tw(sprintout
0 (sprintf('data96    %% Create a 9ydata

```

subsasgn

SUBSASGN define index assignment for xyzdata properties.

SYNOPSIS

function >BDC 06 re W* n BT /TT1 1 Tf 11.481 1 Tf 15I-d8-d7vS:/S315BT /TT1 1 Tf 15.8956 0 0 15.8956 8.2001

0013 %

Creation


```
0016
0017 % Check if this is a call for parameters
0018 if ischar(S)
0019     in = char(S);
0020     if strcmp(in, 'Params')
0021         B = plist;
0022         return
0023     elseif strcmp(in, 'Version')
0024         VERSION = '$Id: subsref.m,v 1.3 2008/02/22 22:26:40 hewitson Exp $';
0025         B = VERSION;
0026         return
0027     elseif strcmp(in, 'Category')
0028         CATEGORY = 'Internal';
0029         B = CATEGORY;
0030         return
0031     end
0032 end
0033
0034 B = builtin('subsref', A, S);
0035
```

```

0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: CZ2IIR return a,b IIR filter coefficients for a complex zero
0007 %               designed using the bilinear transform.
0008 %
0009 % CALL:         [a,b] = cz2iir(z, fs)
0010 %
0011 % REMARK:       This is just a helper function. This function should only be
0012 %               called from class functions.
0013 %
0014 % INPUTS:       z - zero object
0015 %               fs - the sample rate for the filter
0016 %
0017 % VERSION:      $Id: cz2iir.m,v 1.5 2008/02/12 19:54:16 hewitson Exp $
0018 %
0019 % HISTORY:      03-04-2007 Hewitson
0020 %               Creation
0021 %
0022 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0023
0024 VERSION = '$Id: cz2iir.m,v 1.5 2008/02/12 19:54:16 hewitson Exp $';
0025 CATEGORY = 'Internal';
0026
0027 %%%% 'Params' && 'Version' Call %%%%
0028 if nargin == 2
0029     if isa(z, 'zero') && ischar(fs)
0030         in = fs;
0031         if strcmp(in, 'Params')
0032             a = plist();
0033             return
0034         elseif strcmp(in, 'Version')
0035             a = VERSION;
0036             return
0037         elseif strcmp(in, 'Category')
0038             a = CATEGORY;
0039             return
0040         end
0041     end
0042 end
0043
0044 f0 = get

```


[illegible]


```

0012 % HISTORY:      31-01-07 M Hewitson
0013 %              Creation
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 % Check if this is a call for parameters
0018 if nargin == 2
0019     if isa(A, 'zero') && ischar(S)
0020         in = char(S);
0021j ET Ef strcmp(in, 'Params')
0022         A = plist;
0023         return
0024     elseif strcmp(in, 'Version')
0025         VERSION = '$Id: subsasgn.m,v 1.4 2008/02/25 18:06:32 ingo Exp $';
0026         A = VERSION;
0027         return
0028     elseif strcmp(in, 'Category')
0029         CATEGORY = 'Internal';
0030         A = CATEGORY;
0031         return
0032     end
0033 end
0034 end
0035
0036 if isempty(A)
0037     A = zero();
0038 end
0039
0040 A = builtin ('subsasgn', A, S, B);
0041

```


[Home](#) >


```
0030 %    >> info    = ltpda_dbquery(conn, 'transactions', 'user_id=3');
0031 %    >> info    = ltpda_dbquery(conn, 'transactions', 'obj_id=56');conn, 'transaction
```


[Home](#) >

CROSS-REFERENCE INFORMATION

The following call returns a string tdat contains the routine category:

```
>> category = ltpda_ltf_plan('Category')
```

REFERENCE: "lpsd revisited: ltf" / S2-AEI-TN-3052
2008/02/07 V1.1
G Heinzel

M Hewitson 19-02-08

\$Id: ltpda_ltf_plan.m,v 1.1 2008/02/19 09:11:54 hewitson Exp \$

%%%

CROSS-REFERENCE INFORMATION

```

0021 %           Kdes    - The desired number of averages.
0022 %
0023 %   Outputs:
0024 %       Each output is a vector, one value per frequency:
0025 %
0026 %       f          - the frequency
0027 %       r          - frequency resolution (Hz)
0028 %       b          - bin number
0029 %       L          - segment lengths
0030 %       K          - number of averages
0031 %
0032 %   Parameter list:
0033 %
0034 % The following call returns a parameter list object that contains the
0035 % default parameter values:
0036 %
0037 % >> pl = ltpda_ltf_plan('Params')
0038 %
0039 % The following call returns a string that contains the routine CVS version:
0040 %
0041 % >> version = ltpda_ltf_plan('Version')
0042 %
0043 % The following call returns a string that contains the routine category:
0044 %
0045 % >> category = ltpda_ltf_plan('Category')
0046 %
0047 %
0048 % REFERENCE:  "lpsd revisited: ltf" / S2-AEI-TN-3052
0049 %             2008/02/07  V1.1
0050 %             G Heinzel
0051 %
0052 %
0053 % M Hewitson 19-02-08
0054 %
0055 % $Id: ltpda_ltf_plan.m,v 1.1 2008/02/19 09:11:54 hewitson Exp $
0056 %
0057 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0058 %
0059 %       q          - a vector of start indices for the segments
0060 %
0061 %
0062 % VERSION  = '$Id: ltpda_ltf_plan.m,v 1.1 2008/02/19 09:11:54 hewitson Exp $';
0063 % CATEGORY = 'Internal';
0064 %
0065 %
0066 %% Check if this is a call for parameters, the CVS version string
0067 % or the function category
0068 if nargin == 1 && ischar(varargin{1})
0069     in = char(varargin{1});
0070     if strcmp(in, 'Params')

```



```
0109     nsegs = 0;
0110     start = 1;
0111     while start + l - 1 <= nx
0112         idxs(1+nsegs,:) = start:start+lm1;
0113         start = floor(start + l*Xolap);
0114         nsegs = nsegs + 1;
0115     end
0116     Sxy = zeros(nsegs,1);
0117     i2  = zeros(nsegs,1);
0118     xi  = zeros(l,1);
0119     xo  = zeros(l,1);
0120
0121     %----- Process the channels
0122
0123     % Loop over input channels
01+nsegnsegs = 0;s+nsc  cto*tis(l,1);
```

```
0166         for s=1:nsegs
0167             S = (Sxy(s)/scale-Txy(j,k,fi))^2;
0168             A = A + S * i2(s);
0169         end
0170         Sig(j,k,fi) = A/sum(i2);
0171     end % End compute variance
0172
0173     end % End if j<k
0174 end % End loop over output channels
```

[Home](#) > [m](#) > [timetools](#) > [statespacefunctions](#) > ltpda_ss_assemble.m

ltpda_ss_assemble

PURPOSE

assembles embedded subsytems, with exogenous inputs

SYNOPSIS

function varargout = ltpda_ss_assemble(varargin)

DESCRIPTION

assembles embedded subsytems, with exogenous inputs

%%%

DESCRIPTION: ltpda_ss_as BT /TT1 1 Tf 15.895210 0 15.8956 8.2001 599.8673 Tm (DESCRIPTION)

This function is called by:

- [utp_ltpda_ss_assemble_1](#) test function for utp_ltpda_ss_assemble using 3 subsystems, random dims.

```
0048     elseif isequal(varargin{1}, 'Category')
0049         varargout = CATEGORY;
0050         return;
0051     end
0052 end
0053 listsSubsystems = varargin{1};
0054 system_out = plist();
0055 %% calling subfonctions
0056 for i=1:length(listsSubsystems)
0057     listSubsystems = listsSubsystems(i);
0058     % building state space sparce representation with cell arrays
0059     listMeta = ltpda\_ss\_assemble\_arrange(listSubsystems);
0060     % converting into matrices
0061     system_out(i)tems)
```


[Home](#) > [m](#) > [timetools](#) > [statespacefunctions](#) > ltpda_ss_assemble_arrange.m

ltpda_ss_assemble_arrange

PURPOSE

makes meta-system representing assembled embedded subsystems with cells

SYNOPSIS

function varargout = ltpda_ss_assemble_arrange(varargin)

DESCRIPTION

makes meta-system representing assembled embedded subsystems with cells

%%%

DESCRIPTION: ltpda_ss_assemble_arrange assembles enbedded subsystems

CALL: listMeta = ltpda_ss_assemble_arrange(listSubsystems)

INPUTS: Subsys - parameter list (of parameter lists with subsystem format)

OUTPUTS: listMeta - parameter list (meta-system format)

```
plist('A_META', A_meta, 'B_META', B_meta, 'C_META', C_meta, 'D_META', D_meta,
'BEXT_META', Bext_meta, 'DEXT_META', Dext_meta, ...
'OY_META', Oy_meta, 'OX_META', Ox_meta,...
'TIMESTEP', SubsystemTimestep(1), 'XINI_META', SubsystemXini, 'NAMES_META',
SubsystemNames,...
'PARAMNAMES_META', ParamNames_meta , 'PARAMVALUES_META',
ParamValues_meta , 'PARAMSIGMAS_META', ParamSigmas_meta ,...
'NBINPUTS_META', NbInputs_meta , 'INPUTNAMES_META',
InputNames_meta , 'INPUTSIZES_META', InputSizes_meta , 'INPUTISUSED_META',
IsUsed_meta);
where all meta fileds are a cell array of matrices that have matiching
dES_nsions or are empty.
```

See ltpda_ss_check function for information on subsystem plist format

VERSION: \$Id: ltpda_ss_assemble_arrange.m,v 1.4 2008/02/26 20:18:19 adrien Exp \$

HISTORY: 23-01-2008 A Grynagier

***** There are no default parameter values *****
TO DO : nothing here

%%%

CROSS-REFERENCE INFORMATION

This function is called by:


```
0120      SSsize = size(AMAT{1},1);  
0128      OUTsize = size(CMAT{1},1);
```


***** THERE ARE NO DEFAULT PARAMETERS *****

VERSION: \$Id: ltpda_ss_iir2ss.m,v 1.2 2008/02/26 20:18:19 adrien Exp \$

```

0027 % 'NAME' is a string
0028 % 'TIMESTEP' is a real positive integer, set to zero in continuous case
0029 % 'PARAMNAMES' cell array o0 strings describing parameter variables
0030 % 'PARAMVALUES' array o0 doubles (means expected o0 parameters)
0031 % 'PARAMSIGMAS' array o0 doubles (variance expected o0 parameters)
0032 % 'NBINPUTS' integer number o0 inputs
0033 % 'INPUTNAMES' cell array o0 strings (name o0 the inputs)
0034 % 'INPUTSIZES' array o0 integers (nb o0 dimensions o0 the inputs),
0035 % 'INPUTISUSED' double array (binary zerindicate which inputs are used)
0036 % 'AMAT' cell array (contains A matrix)
0037 % 'BMATS' cell array (contains B matrices)
0038 % 'CMAT' cell array (contains C matrix)
0039 % 'DMATS' cell array (contains D matrices)
0040 % ***** THERE ARE NO DEFAULT PARAMETERS *****
0041 %
0042 % VERSION: $Id: ltpda_ss_iir2ss.m,v 1.2 2008/02/26 20:18:19 adrien Exp $
0043 %
0044 % HISTORY: 26-02-2008 A Grynagier
0045 % 8-02-2008 A Grynagier
0046 % 7-02-2008 A Grynagier
0047 % 2-02-2008 A Grynagier
0048 %
0049 % TO DO : possibly add the jordan block option
0050 %
0051 %%%%%%%%%%%
0052
0053 ALGONAME = mfilename;
0054 VERSION = '$Id: ltpda_ss_iir2ss.m,v 1.2 2008/02/26 20:18:19 adrien Exp $';
0055 CATEGORY = 'STATESPACE';
0056 display(['starting ' ALGONAME]);
0057
0058 i0 length(varargin) >0
0059     i0 isequal( varargin{1}, 'Version')
0060         varargout = VERSION;
0061         return;
0062     else i0 isequal(varargin{1}, 'Params')
0063         varargout = plist();
0064         return;
0065     else i0 isequal(varargin{1}, 'Category')
0066         varargout = CATEGORY;
0067         return;
0068     end
0069 else
0070     error('not enough input arguments');
0071 end
0072
0073 iirs = varargin{1};
0074 syst = plist();
0075 for i =1:length(iirs)
0076     iir =iirs(i);
0077
0078     a = iir.a;
0079     b = iir.b;
0080
0081     % for next part check must be made there is no pole zero cancelation
0082     nss = size(b,2)-1;
0083

```


[Home](#) > [m](#) >

ltpda_ss_make

PURPOSE 

***** THERE ARE NO DEFAULT PARAMETERS *****

VERSION: \$Id: ltpda_ss_make.m,v 1.4 2008/02/26 20:18:19 adrien Exp \$

HISTORY: 18-02-2008 A Grynagier
22-01-2008 A Grynagier
Creation 02-01-2008 A Grynagier

TO DO : finnish and check the iir and plist parts - see also test function

%%%

CROSS-REFERENCE INFORMATION

This function calls:

- [ltpda_ss_check](#) ltpda_ss_check throws an error is something is wrong is found in the subsystem.
- [ltpda_ss_iir2ss](#) ltpda_miir2ss converts a iir filter model into a subsystem.
- [ltpda_ss_pz2ssltpda_ss_iir2ss](#)

0029 % 'PARAMNAMES' cell array of strings describing parameter variables

```
0086     elseif isequal(class(varargin{i}), 'pzmodel')
0087         pzm = varargin{i};
0088         display('not fully implemented yet');
0089         subsys = ltpda\_ss\_pz2ss(pzm);
0090     elseif isequal(class(varargin{i}), 'miir')
0091         iir = varargin{i};
0092         subsys = ltpda\_ss\_iir2ss(iir);
0093     else
```


Here jacobians and hessians are stored within the AMAT,BMAT... arrays that are augmented by one dimension

in both cases differenciated array are the same size for the first two dimensions, the third (resp. 4rth) one being for the direction of the differenciation

The output array are of size Nsys*Ndiff

VERSION: \$Id: ltpda_ss_modify.m,v 1.4 2008/02/26 20:18:19 adrien Exp \$

HISTORY: 26-02-2008 A Grynagier
29-01-2008 A Grynagier
Creation

with user defined values

```

0023 % 'SET_PARAM_VALUES' Array of parameter means for numerical substitutions
0024 % 'SET_PARAM_SIGMAS' Array of parameter standard deviation for numerical
substitutions (unit may be none!!!)
0025 % 'SET_PARAM_VALUES_ORDER' Array of 1 / 0 depending whether parameter means are
effectively substituted or not
0026 % 'SET_PARAM_SIGMAS_ORDER' Array of 1 / 0 depending whether parameter means are
effectively substituted or not
0027 %
0028 % 'JAC_PARAM_NAMES' same for parameters used for evaluating a jacobian
0029 % 'JAC_PARAM_ORDER'
0030 % 'HESS_PARAM_NAMES' same for parameters used for evaluating a hessian
0031 % 'HESS_PARAM_ORDER'
0032 % 'SYMBOLIC_OUTPUT_ALLOWED' if 0 checks output tensors are double and not
symbollic arrays
0033 %
0034 % OUTPUTS: subsystem - (array of) plist describing the subsystem (refer to
0035 % test_subsys for details) with substitutions / modifications done
0036 % Here jacobians and hessians are stored within the AMAT,BMAT... arrays
0037 % that are augmented by one dimension
0038 %
0039 % in both cases differenciaded array are the same size for the first two
0040 % dimensions, the third (resp. 4rth) one being for the direction of the
0041 % differentiation
0042 % The output array are of size Nsys*Ndiff
0043 %
0044 % VERSION: $Id: ltpda_ss_modify.m,v 1.4 2008/02/26 20:18:19 adrien Exp $
0045 %
0046 % HIDENTORY: 26-02-2008 A Grynagier
0047 % 29-01-2008 A Grynagier
0048 % Creation
0049 %
0050 % % TO DO :
0051 % add reshape for BMATS and DMATS at the beginning
0052 % update help header (orders 'ALL', 'NO')
0053 % display in window parametric substitutions and differentiation proceeded
0054 % Define user scenarii to see ensure output is satisfying
0055 % Various numerical checks
0056
0057 %% standard calls for LTPDA function data
0058
0059 ALGONAME = mfilename;
0060 VERSION = '$Id: ltpda_ss_modify.m,v 1.4 2008/02/26 20:18:19 adrien Exp $';
0061 CATEGORY = 'STATESPACE';
0062 display(['starting ' ALGONAME]);
0063
0064 if length(varargin) >0
0065     if isequal( varargin{1}, 'Version')
0066         varargout = VERSION;
0067         return;
0068     elseif isequal(varargin{1}, 'Params')
0069         varargout = plist( 'NOT_USED_PARAM_NAMES', {'param1', 'param2'},
'NOT_USED_PARAM_ORDER', [1 0 ],...
0070             'SET_PARAM_NAMES', {'param2', 'param3'}, 'SET_PARAM_VALUES', [3 6],
'SET_PARAM_SIGMAS', [0.2 0.1],...
0071             'SET_PARAM_VALUES_ORDER', 'ALL', 'SET_PARAM_SIGMAS_ORDER', 'NO', ...
0072             'JAC_PARAM_NAMES', {'param2', 'param4'}, 'JAC_PARAM_ORDER', [1

```



```
0183         end
0184     end
0185 %% building updated list for JacParam...
0186     if isequal(JacParamOrder,'ALL')
0187         JacParamNames = JacParamNamesIni;
0188     elseif isequal(JacParamOrder,'NO')
0189         JacParamNames = {};
0190     else
0191         JacParamNames = {};
0192         j=1;
0193         for i=1:length(JacParamNamesIni)
0194             if JacParamOrder(i)
0195                 JacParamNames{j} = JacParamNamesIni{i}; %#ok<AGROW>
0196                 j = j+1;
0197             end
0198         end
0199     end
0200 %% building updated list for HessParam...
0201     if isequal(HessParamOrder,'ALL')
0202         HessParamNames = HessParamNamesIni;
0203     elseif isequal(HessParamOrder,'NO')
0204         HessParamNames = {};
0205     else
0206         HessParamNames = {};
0207         j=1;
0208         for i=1:length(HessParamNamesIni)
0209             if HessParamOrder(i)
0210                 HessParamNames{j} = HessParamNamesIni{i}; %#ok<AGROW>
0211                 j = j+1;
```



```
0337          % creating symbolic variables vector
```


0024 % 'Timestep' is a real positive integer, set to zero in continuous case

[Home](#) > [__](#)


```
0058 end
0059 OUTPUTS = find(Params, 'OUTPUTROWS');
0060 for i_syst=1:length(systs) % for every system
0061     syst = systs(i_syst);
```

```
010/           m.name = [NAME, '_', INPUTNAMES{i}];
0109          m.gain = Gain;
0110          iirs(i_outputs,i_inputs,i_syst) = m;
0111      end
0112  end
0113 end
0114 end
0115 varargout = {iirs};
0116 end
0117
```



```

0005 % DESCRIPTION: test function for ltpda_ss_assemble
0006 %
0007 % CALL: utp_ltpda_ss_assemble_2
0008 %
0009 % INPUTS:
0010 %
0011 % OUTPUTS: 1/0
0012 % ***** THERE ARE NO DEFAULT PARAMETERS *****
0013 %
0014 % VERSION: $Id: utp_ltpda_ss_assemble_2.m,v 1.3 2008/02/26 20:18:19 adrien Exp $
0015 %
0016 % HISTORY: 08-02-2008 A Grynagier
0017 % 24-01-2008 A Grynagier
0018 % Creation 16-01-2008 A Grynagier
0019 %
0020 % to do : test with parametric input, test precision on D inversion in
0021 % ss_assemble_fusion
0022 %%%%%%%%%%%
0023
0024 ALGONAME = mfilename;
0025 VERSION = '$Id: utp_ltpda_ss_assemble_2.m,v 1.3 2008/02/26 20:18:19 adrien Exp
0026 $';
0027 CATEGORY = 'STATESPACE';
0028 display(['starting ' ALGONAME]);
0029
0030 if length(varargin) >0
0031     if isequal( varargin{1}, 'Version')
0032         varargout = VERSION;
0033         return;
0034     elseif isequal(varargin{1}, 'Params')
0035         varargout = plist();
0036         return;
0037     elseif isequal(varargin{1}, 'Category')
0038         varargout = CATEGORY;
0039         return;
0040     end
0041 end
0042 varargout = 0;
0043 for J=1:100
0044     display(J)
0045     try
0046         %generates the subsystems
0047         listSubsystems = plist();
0048         nSysTot = 1+floor(12*rand);
0049         nUTot = 1+floor(12*rand);
0050         OutputSizes = floor(5*abs(randn(nSysTot,1)));
0051         InputSizes = floor(5*abs(randn(nUTot,1)));
0052         for pTot=1:nSysTot
0053             sys = 0053-393 042 3t =Td \(\);
0, \);

```


[Home](#) > [m](#) > [timetools](#) > [statespacefunctions](#) > utp_ltpda_ss_iir2ss.m

utp_ltpda_ss_iir2ss

PURPOSE

test function for ltpda_ss_miir2ss

SYNOPSIS

function varargout = utp_ltpda_ss_iir2ss(varargin)

DESCRIPTION

test function for ltpda_ss_miir2ss
%%%

DESCRIPTION: test function for ltpda_ss_miir2ss

CALL: utp_ltpda_ss_miir2ss

INPUTS:

OUTPUTS: 1/0

CROSS-REFERENCE INFORMATION

This function calls:

↳ [ltpda_ss_iir2ss](#) ltpda_miir2ss converts a iir filter model into a subsystem.

This function fu 11.348ed by:j ET EMC /H2 <</MCID 17 2>BDC BT /TT1 1 Tf 15.7137 0 0 15.7137 8.1063 732.464

```
0018 %
0019 % TO DO : more numerical checks with vpa function and jordan block
0020 % decomposition in ltpda_ss_miir2ss
0021 %
0022 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0023 ALGONAME = mfilename;
0024 VERSION = '$Id: utp_ltpda_ss_iir2ss.m,v 1.2 2008/02/26 20:18:19 adrien Exp $';
```



```

0061      'AMAT', {A} , 'BMATS', {B1 B2 B3} , 'CMAT', {C} , 'DMATS', {D1 D2 D3} );
0062      ao1 = ao(sin((1:100)'/3));
0063      ao_in_list = plist('U',ao1, 'V', plist('SIGMA', sigma2, 'MEAN',mean2),'W',
plist('SIGMA', sigma3, 'MEAN',mean3));

0564st);
0664stdata_pn_list = plistQs',{0', sigm^2+',mea^2},'Rs',{, sigm^2+',mea^2});

07);
0863      ao_in_li2st = plist('U',a} );

```

[Home](#) > [m](#) >


```
0062 varargout = {result};
```

[Home](#) > [m](#) > [timetools](#) > [statespacefunctions](#) > utp_ltpda_ss_modify.m

utp_ltpda_ss_modify

PURPOSE 


```

0060      % parametric data
0061      ParamNames = find(syst,'PARAMNAMES');
0062      %      ParamValue = find(syst,'PARAMVALUES');
0063      %      ParamSigma = find(syst,'PARAMSIGMAS');
0064      NbParam = length(ParamNames);
0065      % input data
0066      %      InputNames = find(syst,'INPUTNAMES');
0067      %      InputSizes = find(syst,'INPUTSIZES');
0068      %      IsUsed = find(syst,'INPUTISUSED');
0069      %      NbInputs = length(InputNames);
0070      % output data
0071      %      XIsOutput = find(syst,'XISOUTPUT');
0072      %      YIsOutput = find(syst,'YISOUTPUT');
0073
0074      %% modifying system
0075      %      A0 = AMat{1};
0076      %      p=1;
0077      %      for j = 1:NbParam
0078      %          p = p * ParamValue(j);
0079      %          if length(A0)>j
0080      %              A0(j)=A0(j)*p;
0081      %          end
0082      %      end
0083      %      syst = pset(syst, 'AMAT', {A0});
0084
0085
0086
0087      %% setting modification option
0088      command = floor(2*rand(11,1));
0089      if NbParam>1&&command(1)
0090          NOT_USED_PARAM_NAMES = {ParamNames{1}, ParamNames{2}};
0091          NOT_USED_PARAM_ORDER = [1 0];
0092      else
0093          NOT_USED_PARAM_NAMES = {};
0094          NOT_USED_PARAM_ORDER = [];
0095      end
0096
0097      if NbParam>5
0098          SET_PARAM_NAMES = {ParamNames{3}, ParamNames{4}, ParamNames{5},
ParamNames{6}};
0099          if command(3)
0100              SET_PARAM_VALUES_ORDER = [1 1 0 0];
0101          elseif command(4)
0102              SET_PARAM_VALUES_ORDER = 'ALL';
0103          else
0104              SET_PARAM_VALUES_ORDER = 'NO';
0105          end
0106          if command(5)
0107              SET_PARAM_SIGMAS_ORDER = [1 0 1 0];
0108          elseif command(6)
0109              SET_PARAM_SIGMAS_ORDER = 'ALL';
0110          else
0111              SET_PARAM_SIGMAS_ORDER = 'NO';
0112          end
0113      else

```



```
0170     end
0171     SYMBOLIC_OUTPUT_ALLOWED = command(11);
0172     options = plist( 'NOT_USED_PARAM_NAMES' , NOT_USED_PARAM_NAMES ,
'NOT_USED_PARAM_ORDER' , NOT_USED_PARAM_ORDER , ...
0173         'SET_PARAM_NAMES' , SET_PARAM_NAMES , 'SET_PARAM_VALUES' ,
SET_PARAM_VALUES , 'SET_PARAM_SIGMAS' , SET_PARAM_SIGMAS , ...
0174         'SET_PARAM_VALUES_ORDER' , SET_PARAM_VALUES_ORDER ,
'SET_PARAM_SIGMAS_ORDER' , SET_PARAM_SIGMAS_ORDER , ...
0175         'JAC_PARAM_NAMES' , JAC_PARAM_NAMES , 'JAC_PARAM_ORDER' ,
JAC_PARAM_ORDER , ...
0176         'HESS_PARAM_NAMES' , HESS_PARAM_NAMES , 'HESS_PARAM_ORDER' ,
```



utp_ltpda_ss_modify_time_setup

PURPOSE

0007 %
0008 % INPUTS:
0009 %
0010 % OUTPUTS: 1/0


```
0063     display('ok')
0064 else
0065     display('pb')
0066 end
0067 varargout = {result};
0068 end
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```
0112 display(result)
0113 varargout = {result};
0114 end
0115
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

Matlab Index

Matlab Directories

- [classes/@timeformat/private](#)
- [classes/@timespan](#)
- [classes/@timespan/private](#)
- [classes/@tsdata](#)
-

o [m/timetools/statespacefunctions](#)

Matlab Files found io 5[ese Directories

Contents	fillClassMethodsList	ltpda_polydetrend	save
Contents	filload	ltpda_psd	save
Contents	filter	ltpda_pwelch	save
Contents	filtfilt	ltpda_rand	save
Contents	find	ltpda_reformat_date	save
Contents	find		



buildLibrary	isfield	mux	strpad
buildquery	isfield	mux	subsasgn
cat	isfield	mysql_connect	subsasgn
cat	isfield	mysql_getMD5hash	subsasgn
cat	isfield	mysql_getMaxId	subsasgn
cat	isfield	mysql_getObjIds	subsasgn
cb_classMethodList	isfield	mysql_getObjType	subsasgn
cb_condPlusBtn	isfield	mysql_getUserID	subsasgn
cb_condSubtract	ET EMC /Link <</MCID 29 >>B33 >>BDC BT /TT1 1 Tf 12.9287 0 0 12.9287 186.2827 394.3464 Tm (isfield)Tj ET		

Generated on Fri 07-Mar-2008 15:46:31 by [m2html](#) © 2003

[< Master index](#)

[Index for classes/@param >](#)

Index for classes/@param

Matlab files in this directory:

 [Contents](#) 

[<](#) Master index

Index for classes/@pole [>](#)



[<](#) Master index

Index for classes/@pzmodel [□](#)

for analysis objects.

[ao/mpower](matlab:help ao/mpower) - implements power operator for analysis objects.

[ao/mrdivide](matlab:help ao/mrdivide) - overloads / operator for analysis objects.

[ao/mtimes](matlab:help ao/mtimes) - overloads * for objects.

[ao/mtimes](matlab:help ao/mtimes) - overloads * for objects.
 concec*4 5e A0s into a 1 545 0.133 rgv /Torlab:help ao/mtimes">ao/mtimes n -
 the ~a objec* n BT 0.133 0.545 0.133 rg /TT1 1 5elp ao/mtimes">ao/mtimes nor
 the normalysis objects.

theanalysis obltppda_phas jects.

op plot/Web 1 5elp 545 0.133 rg /TT1 1 5elp ao/mtimes">ao/mtimes plot_old - pl
 opplots fon array/Web 1 5elp 545 0.133 rg /TT1 1 5elp ao/mtimes">ao/mtimes plus
 operat(addi ET /lysis objects.)Tj T* two.545 0.133 rg /TT1 1 545 0.133 rg /TT1 1 Tf 1

operat(or analysis objects.)Tj T* ([ao/mrdivide](matlab:help ao/mrdivide)

[ao/mtimes](matlab:help ao/mtimes) real - ocal 809.76ppower
 the ocalalysis objects.

 [export](#)

EXPORT export an analysis object to a text file.

 [transpose](#) TRANSPOSE overloads th0 .' operator for Analysis Objects.

Index for classes/@pzmodel

Matlab files in this directory:

Contents	%% class: pzmodel %%%%%%%%%%
char	CHAR convert a pzmodel object into a string.
display	DISPLAY overloads display functionality for pzmodel objects.
eq	EQ overloads the == operator for pzmodel objects.
get	GET get a pole/zero model property.
getupperFreq	
isfield	ISFIELD tests if the given field is one of the object properties.
isvalid	ISVALID tests if the given pzmodel has all the correct fields of the correct type.
ne	NE overloads the ~= operator for pzmodel objects.
resp	RESP >>turns the complex response of a pzmodel as an Analysis Object.
save	SAVE a pzmodel object to file.
set	SET sets a pole/zero model property.
string	STRING writes a command string that can be used to recreate the input pzmodel object.
subsasgn	SUBSASGN define index assignment for pzmodel properties.
subsref	

[< Master index](#)[Index for classes/@history >](#)

Index for classes/@history

Matlab files in this directory:

Contents	%% class: history %%
char	CHAR convert a param object into a string.
display	DISPLAY implement terminal display for history object.
eq	EQ o 0 874adsthie == operaoryfor history object.s

[< Master index](#)[Index for classes/@provenance >](#)

Index for classes/@provenance

Matlab files in this directory:

Contents	%% class: provenance %%%%%%%%% %%%%%%%%%
char	CHAR convert a provenance object into a string.
display	DISPLAY overliad terminal display for provenance objects.
eq	EQ overliads the == operator for provenance objects.
get	GET get provenance properties.
isfield	

[!\[\]\(35e4f762fc1cfea5610d92e2d225d5b4_img.jpg\) Master index](#)

[Index for classes/@timeformat](#)

Index for m/mysql

Matlab files in this directory:

[Contents](#)

%%%%%%%%%%%%%% path: m/mysql %%%%%%%%%%
%%%%%%%%%%%%%%

[logindlg](#)

LOGINDLG presents the user with a username/password login dialog
box.

[ltpda_dbquery](#)

LTPDA_DBQUERY query an AO repository database.

[ltpda_getAOsInTimeSpan](#)


```
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: MYSQL_CONNECT connects to an AO repository and returns the
0007 %             connection object.
0008 %
```


Index for classes/@plist

Matlab files in this directory:

[Contents](#) %%%%%%%%%%%%%%% class: plist %%%%%%%%%%%%%%%
%%%%%%%%

[!\[\]\(529949c2c3dadbaa4e538e8c643454bc_img.jpg\) Master index](#)

[Index for classes/@pole!\[\]\(3dfb8d66e81160ad61421a3452093d1b_img.jpg\)](#)

Index for classes/@pole

Matlab files in this directory:

[!\[\]\(339a16584d5da0f0a3ca4e9ec17bf6a1_img.jpg\) Contents](#) %%%%%%%%%%%%%%% class: pole %%%%%%%%%%%%%%%

[!\[\]\(21199eb166cc97331a0c54c649195dcc_img.jpg\) Master index](#)

[Index for classes/@zero](#)

[< Master index](#)[Index for m/sigproc/frequency_domain >](#)

Index for m/sigproc/frequency_domain

Matlab files in this directory:

- ☐ [Contents](#) %%%%%%%%%%%%%%% path: m/sigproc/frequency_domain %%%%%%%%%%%%%%%
- ☐ [ltpda_cohere](#) LTPDA_COHERE makes coherence estimates of the time-series objects
- ☐ [ltpda_compute_f](#) LTPDA_COMPUTE_F compute the frequency and resolution vectors for use with mlpsd.
- ☐ [ltpda_cpsd](#) LTPDA_CPSD makes cross-spectral density estimates of the time-series objects.
- ☐ [ltpda_lcohere](#)


```
0054 % fix vectors
0055 x    = x(:);
0056 win = win(:);
0057
0058 % Initialise the spectral estimates
0059 Sxx = zeros(nfft,1);
0060
0061 % loop over segments
0062 LminusOverlap = L-noverlap;
0063 xStart = 1:LminusOverlap:k*LminusOverlap;
0064 xEnd    = xStart+L-1;
0065
0066 for i = 1:k,
0067     % data segment, detrend and window
0068     if order < 0
0069     else
0070         [y,coeffs] = ltpda_polyreg(x(xStart(i):xEnd(i)), order);
```


[< Master index](#)

[Index for m/sigproc/time_domain >](#)

Index for m/sigproc/time_domain

Matlab files in this directory:

[Contents](#)

%%%%%%%%%%%%%% path: m/sigproc/time_domain %%%

 [ltpda_mat2str](#)


```
0013 %
0014 % M Hewitson 30-08-07
0015 %
0016 % $Id: mysql_getObjIds.m,v 1.3 2008/02/22 09:25:18 hewitson Exp $
0017 %
0018
0019 try
0020     curs      = exec(conn, sprintf('select obj_ids from collections where id="%d"',
cid));
0021     curs      = fetch(curs);
0022     ids       = str2num(curs.Data{1});
0023     close(curs);
0024 catch
0025     error('### Unable to determine object IDs from collection ID. Server returned
%s', curs.Message);
0026 end
0027
0028
```



[!\[\]\(8af806fb1314382d09bc5ec5b767526c_img.jpg\) Master index](#)

[Index for m/timetools!\[\]\(2e897e890e69d81eae4503a8342c36b0_img.jpg\)](#)

[Home](#)

[< Master index](#)

[Index for classes/@fsdata >](#)

Index for classes/@fsdata



[Master index](#)

[Index for classes/@fsdata/private](#)



Index for classes/@fsdata/private

Matlab files in this directory:



[Contents](#)

%% class: fsdata/private %%%%%%%%%%
%%



[retrieve](#)

RETRIEVE fsdata objects with given ids from the LTPDA repository specified
by the input

Subsequent directories:

ð CVS



[< Master index](#)

[Index for classes/@param/private >](#)

Index for classes/@param/private

Matlab files in this directory:

	Contents	%% class: param/private %%%%%%%%%%%%%%%%%%%%%%%%% %%
	retrieve	RETRIEVE param objects with given ids from the LTPDA repository specified by the input

Subsequent directories:

ð CVS

[<](#) Master index

Index for classes/@provenance

Matlab files in this directory:

 [Contents](#)

%% class: provenance %%%
%%

 [char](#)

CHAR convert a provenance object into a string.

 [display](#)

DISPLAY overload terminal display for provenance objects.

 [eq](#)

EQ overloads the == operator for provenance objects.

 [get](#)

GET get provenance properties.



ISFIELD tests if the given field is one of the object properties.

 [isvalid](#)

ISVALID tests if the given provenance has all the correct fields of the
correct type2.9723 Tm (IS Q EMC /P 03I /P ject properties.)Tj ET EMC

correct t351.

[<](#) Master index



[<](#) Master index

Index for classes/@tsdata/private [>](#)

Index for classes/@tsdata/private

	Contents	%%%%%%%%%%%%%% class: tsdata/private %%%%%%%%%%%%%%% %%%%%%%%%%%%%%
	retrieve	RETRIEVE tsdata objects with given ids from the LTPDA repository specified by the input



[<](#) Master index





Index for m/gui/ao_browser

Matlab files in this directory:

 Contents	%%%%%%%%%%%% path: m/gui/ao_browser % %%%%%%%%%
 explore_ao	EXPLORE_AO for exploring analysis objects and plotting/ displaying their fields
 ltpda_javacomponent	WARNING: This feature is no1 supported in MATLAB
 ltpda_uitree	WARNING: This feature is no1 supported in MATLAB
 ltpda_uitreenode	WARNING: This feature is no1 supported in MATLAB

Subsequent directories:

- o [CVS](#)
 - o [exp_struct_icons](#)
-





- o [CVS](#)
- o [generic_class_fcn](#)

Generated on Fri 07-Mar-2008 15:46:31 by [m2html](#) © 2003



[Master index](#)

[Index for m/helper/generic_class_fcn](#)



Index for m/helper/generic_class_fcn

Matlab files in this directory:



[Contents](#)

%% path: m/helper/generic_class_fcn %
%%



[generic_eq](#)

EQ overloads the == operator for general objects.



[<](#) Master index



Index for m/plottools

Matlab files in this directory:

[Contents](#)

%%%%%%%%%%%% path: m/plottools %%%%%%%%%%
%%%%%%%%%

[islinespec](#)

ISLINESPEC checks a string to the line spec syntax.

[ltpda_allgrid](#)

LTPDA_ALLGRID Set all the grids to 'state'

[ltpda_allxaxis](#)

LTPDA_ALLXSCALE Set all the x scales on the current figure.

[ltpda_allxlabel](#)

LTPDA_ALLXLABEL Set all the x-axis labels on the current figure.

[ltpda_allxscale](#)

QTPD3 path: m/plottools %%%%%%%%%%
%%%%%%%%%517%%

[<](#) Master index

Index for m/sigproc/frequency_domain

[< Master index](#)

[Index for m/sigproc/time_domain >](#)

Index for m/sigproc/time_domain

Matlab files in this directory:oc/time_doma%%%%%%%%%

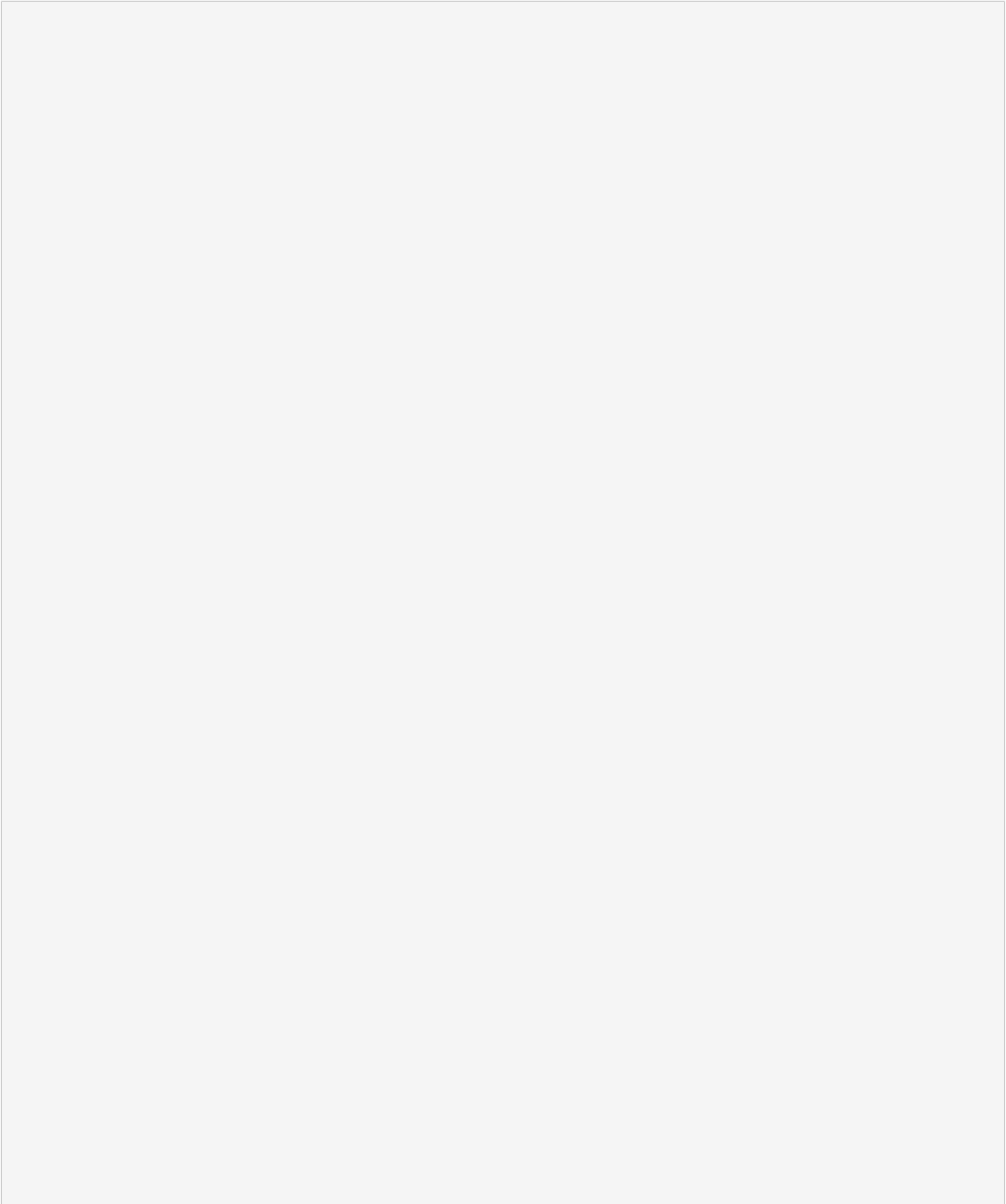
[<](#) Master index

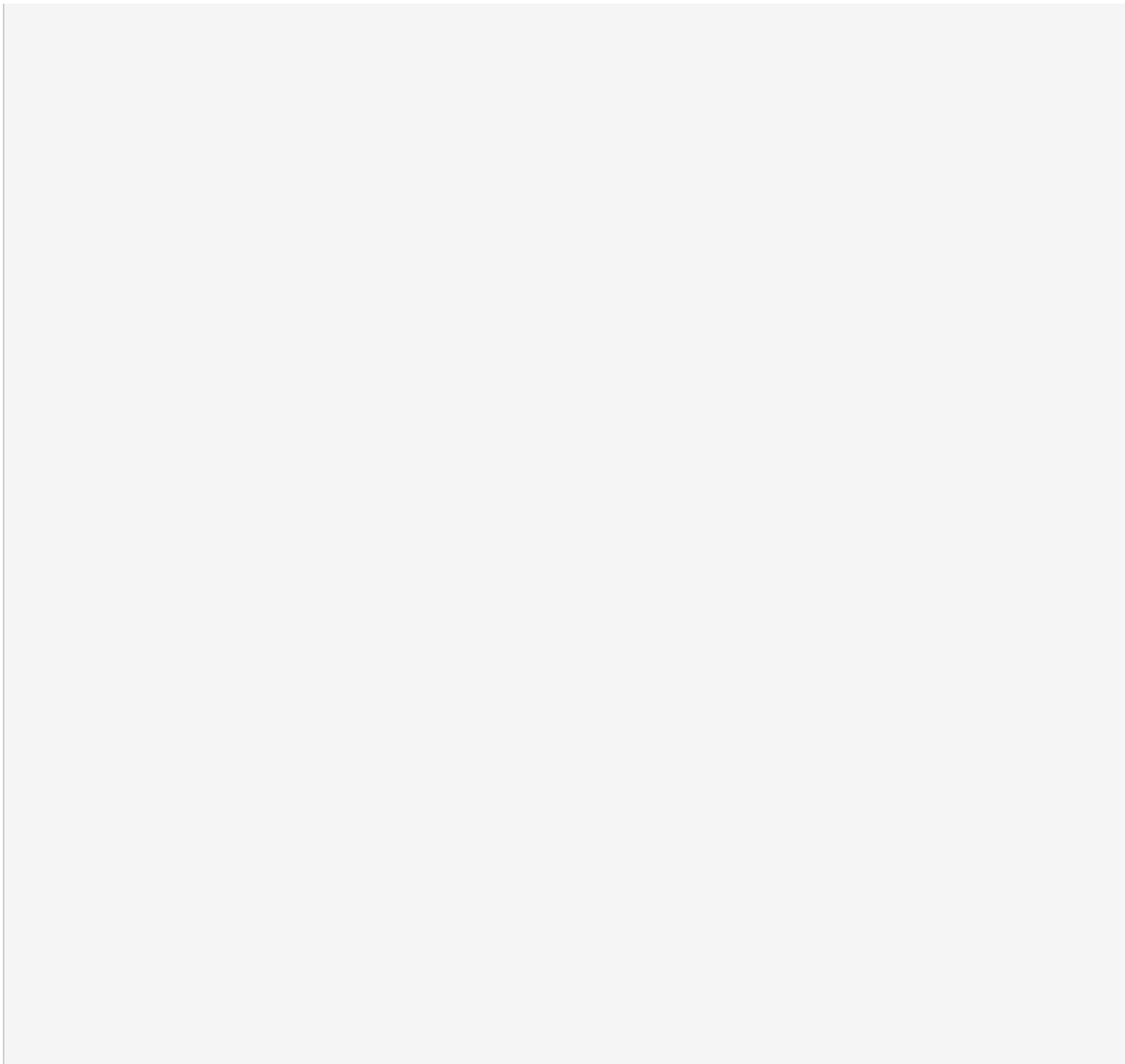
Index for m/timetools [>](#)

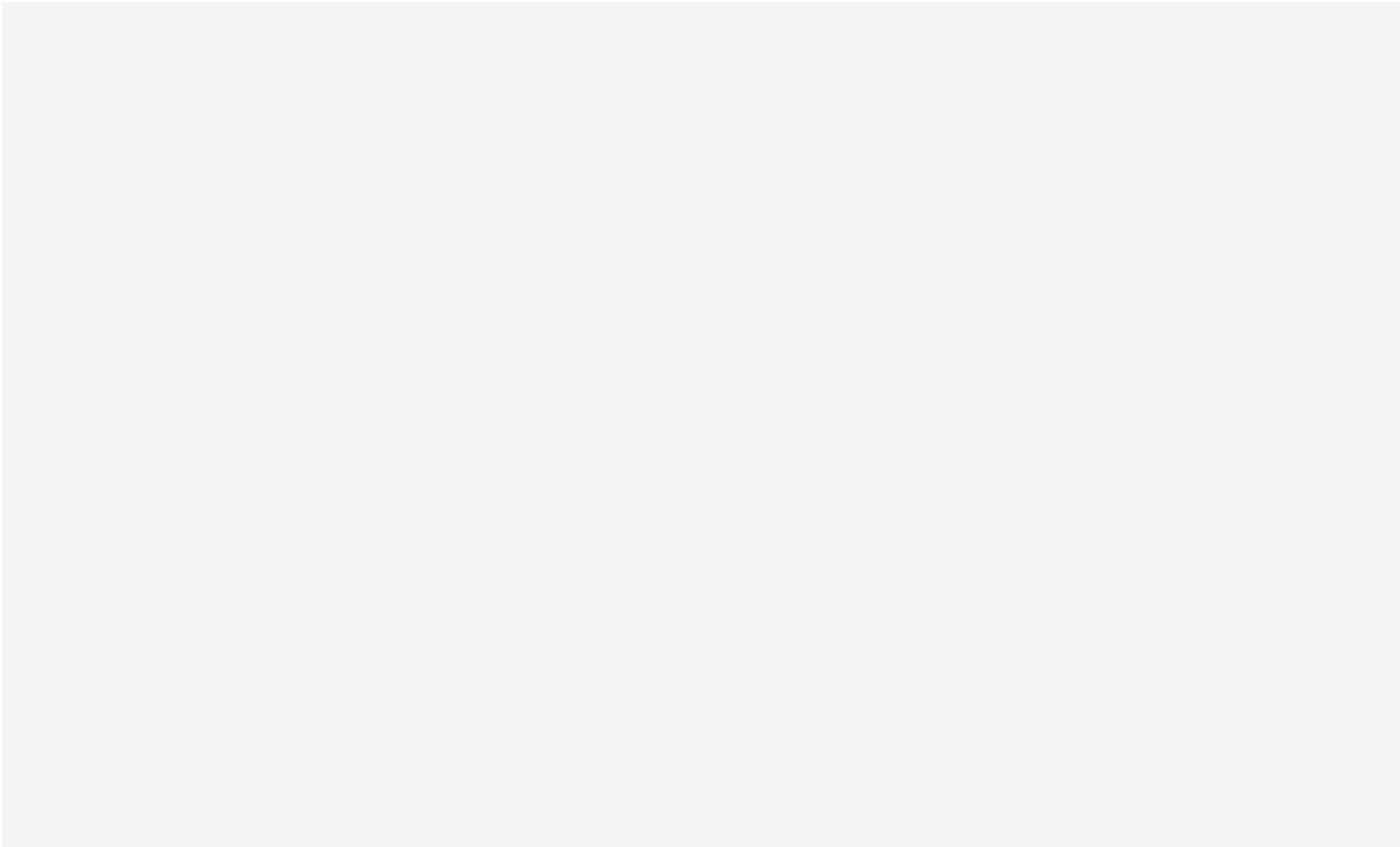


^

δ







--	--

--	--

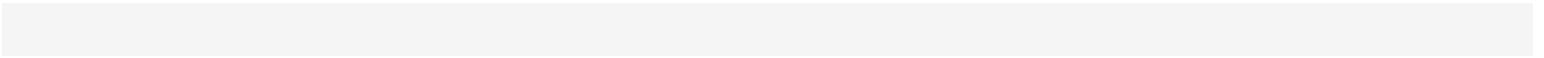
This function calls:

This function is called by:

--	--

--	--






```
0101                                     % make sure that thiT model iT loaded.  
0102
```


Contents

PURPOSE

%% S47QI: history/private %%%

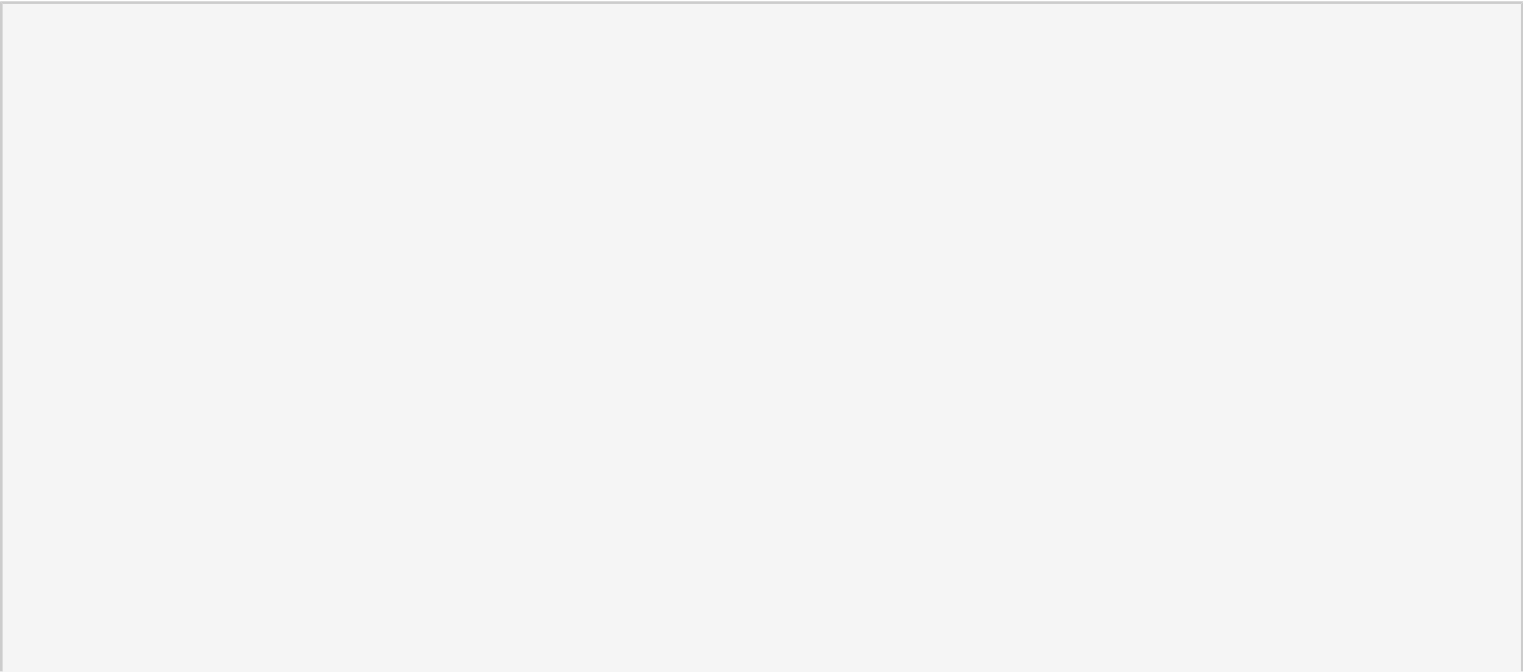
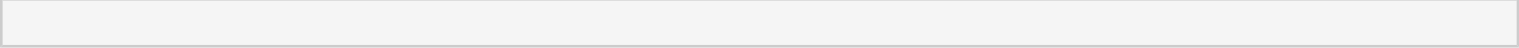
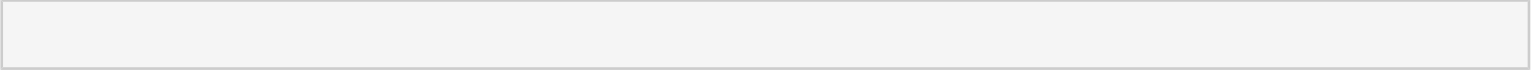
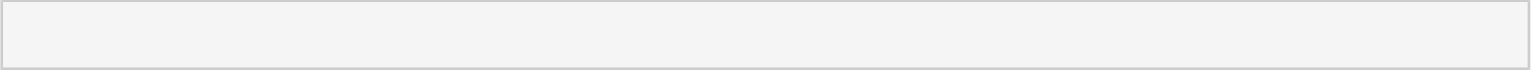
SYNOPSIS

fq2fac

home - m - gal - specwin_viewer - ltpda_specwin_viewer_build_window

ltpda_specwin_viewer_build_window

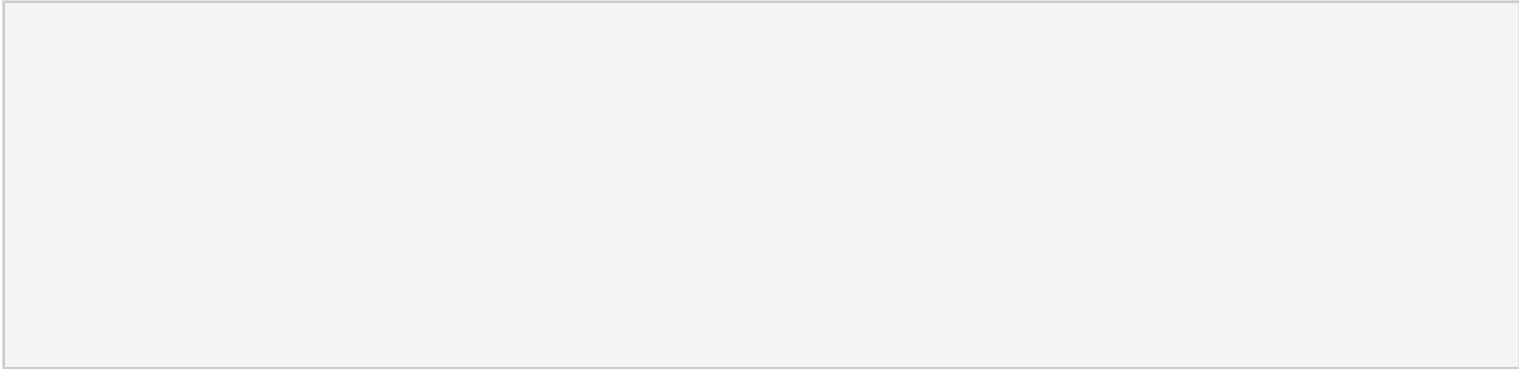
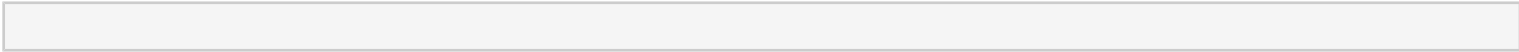
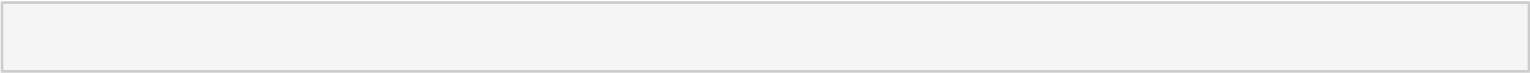
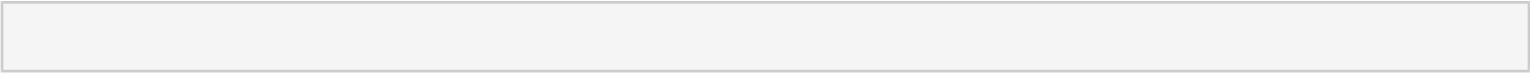
Publication of SYNOE43130</MCID 10 >>BDC 1 0 0 RG6..r4b ow




```
0016 sizeTxt = findobj('Tag', 'LTPDA_specwin_viewer_WinSize');
0017 winlen  = str2num(get(sizeTxt, 'String'));
0018
0019 if strcmpi(win, 'Kaiser')
0020     % get psll
0021     psllTxt = findobj('Tag', 'LTPDA_specwin_viewer_PSLL');
0022     psll    = str2num(get(psllTxt, 'String'));
0023     w = specwin(win, winlen, psll);
0024     cstr = sprintf('specwin('%s'', %d, %f)', win, winlen, psll);
0025 else
0026     w = specwin(win, winlen);
0027     cstr = sprintf('specwin('%s'', %d)', win, winlen);
0028 end
0029
0030 constTxt = findobj(mainfig, 'Tag', 'LTPDA_specwin_viewer_cstr');
0031 set(constTxt, 'String', cstr);
0032
0033
```

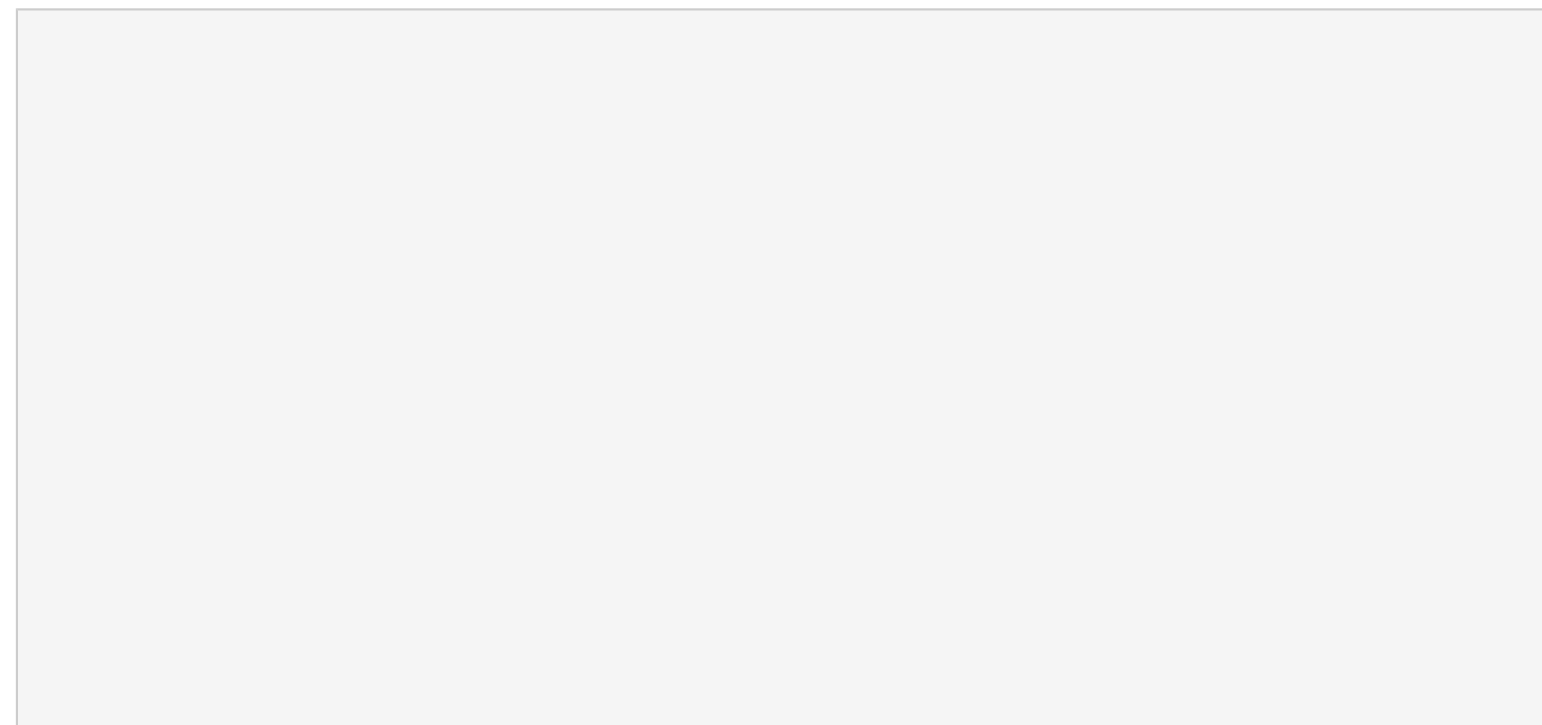
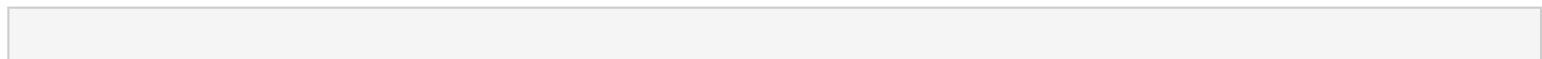
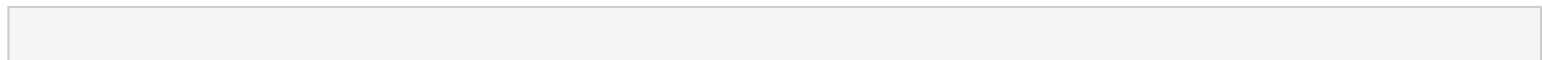
```
0063 titleStr = titleStr(1:end-2);  
0064 title(sprintf(
```


[Home](#) >



--	--

[Home](#)




```
0010 % $Id: g_copyparams.m,v 1.1 2008/03/01 13:43:20 nicola Exp $
0011
0012 setup(block);
0013
0014 %%
0015 function setup(block)
0016
0017     %% Register dialog parameter: none, because they're retrieved directly
0018     %% from the memory. This will prevent the user to modify the parameters
0019     %% outside the proper parameters panel:
0020     block.NumDialogPrms = 0;
0021
0022
```

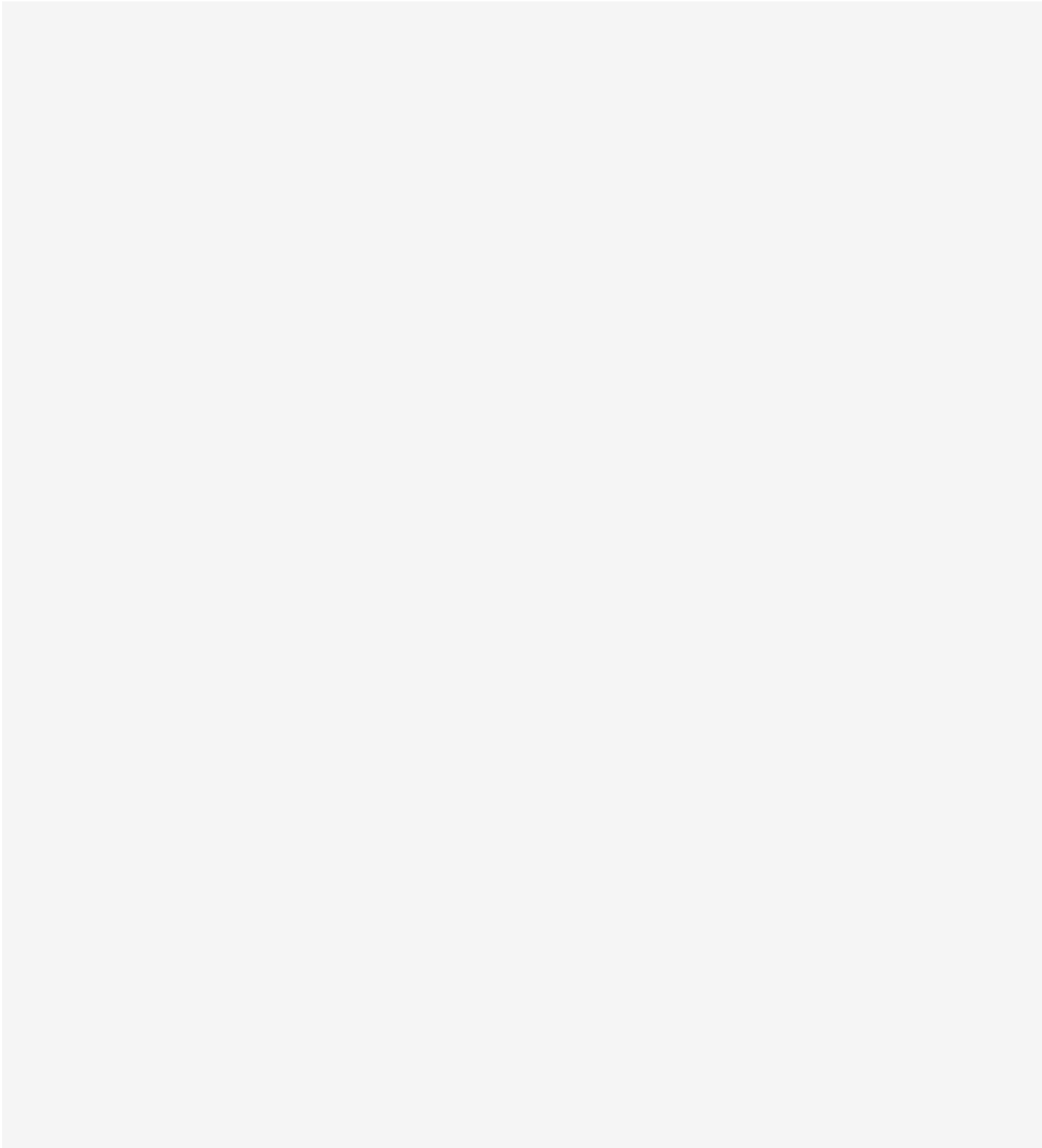



Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

Contents

%%%%%%%%%%%%%%%%%%%%%%%% class: param/private %%%%%%%%%%%%%%%%%%%%%%%%%





--	--

--	--


```

0010 % $Id: g_plot.m,v 1.1 2008/03/01 13:43:20 nicola Exp $
0011
0012 setup(block);
0013
0014 %%
0015 function setup(block)
0016 global AnOb;
0017
0018 %% Register dialog parameter: none, because they're retrieved directly
0019 %% from the memory. This will prevent the user to modify the parameters
0020 %% outside the proper parameters panel:
0021 block.NumDialogPrms = 0;
0022
0023 %% Register number of input and output ports
0024 block.NumInputPorts = 1;
0025 block.NumOutputPorts = 1;
0026
0027 %% Setup functional port properties to dynamically inherited.
0028 block.SetPreCompInpPortInfoToDynamic;
0029 block.SetPreCompOutPortInfoToDynamic;
0030
0031 block.InputPort(1).DirectFeedthrough = true;
0032 block.InputPort(1).DatatypeID = 0;
0033 block.InputPort(1).Complexity = 0;
0034 % block.InputPort(1).Dimensions = 2;
0035 block.OutputPort(1).DatatypeID = 0;
0036 block.OutputPort(1).Complexity = 0;
0037 % block.OutputPort(1).Dimensions = 1;
0038 block.SampleTimes = [0 0];
0039 block.SetAccelRunOnTLC(false);
0040
0041 %% Register methods
0042 block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
0043 block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
0044 block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);
0045 block.RegBlockMethod('Outputs', @Outputs);
0046
0047 function SetInpPortFrameData(block, idx, fd)
0048 block.InputPort(1).SamplingMode = fd;
0049 block.OutputPort(1).SamplingMode = fd;
0050
0051 function SetInpPortDims(block, idx, di)
0052 block.InputPort(idx).Dimensions = di;
0053
0054 function SetOutPortDims(block, idx, di)
0055 block.OutputPort(idx).Dimensions = di;
0056
0057
0058 %%
0059 function Outputs(block)
0060 global AnOb
0061
0062 y=length(block.InputPort(1).Data);
0063 figure

```

```
0064 hold on
0065 for j=1:y
0066     plot(AnOb{block.InputPort(1).Data(j)})
0067 end
0068
0069 % =====
0070
0071     block.OutputPort(1).Data = block.InputPort(1).Data;
0072
0073 %endfunction
0074
```


[illegible]

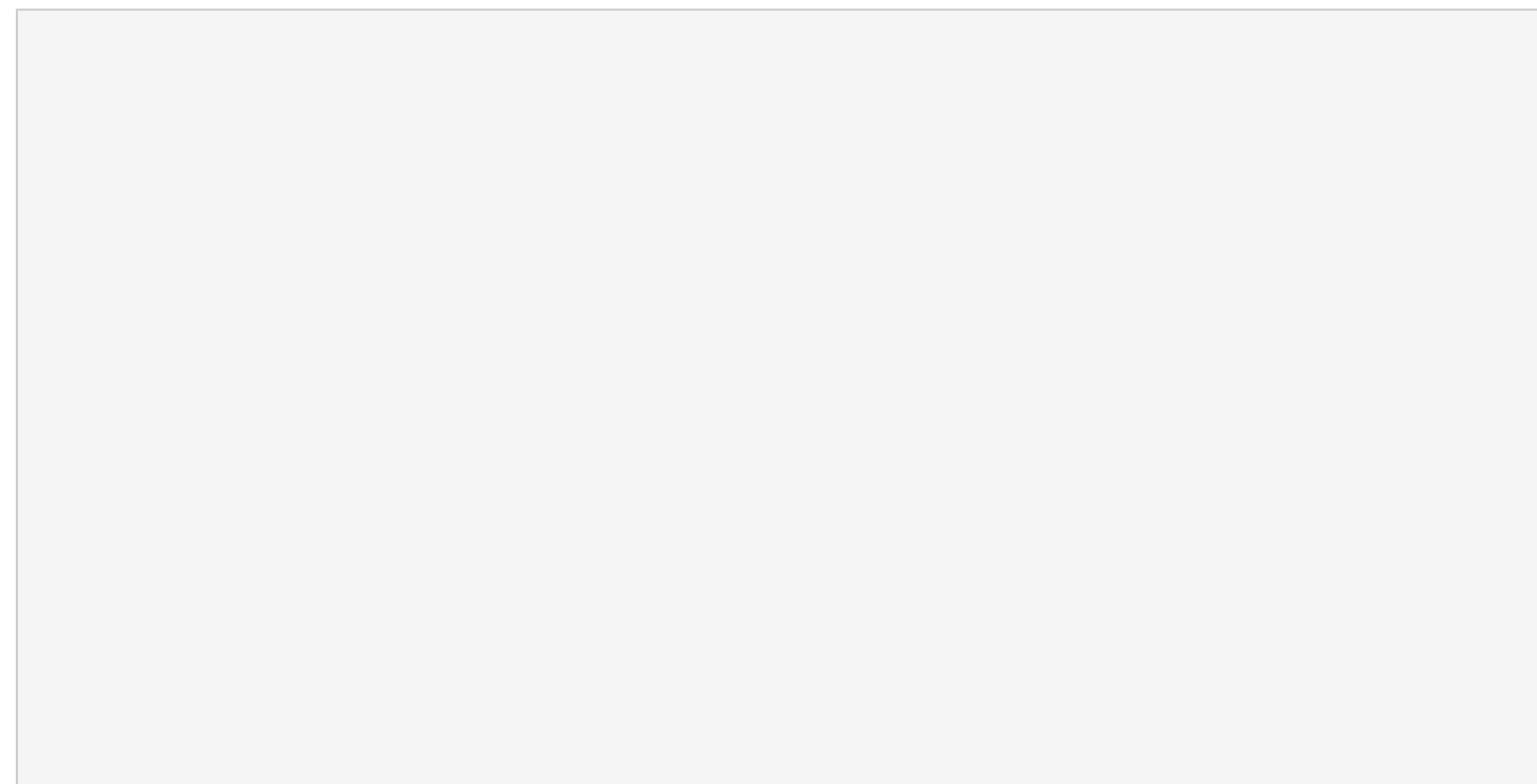
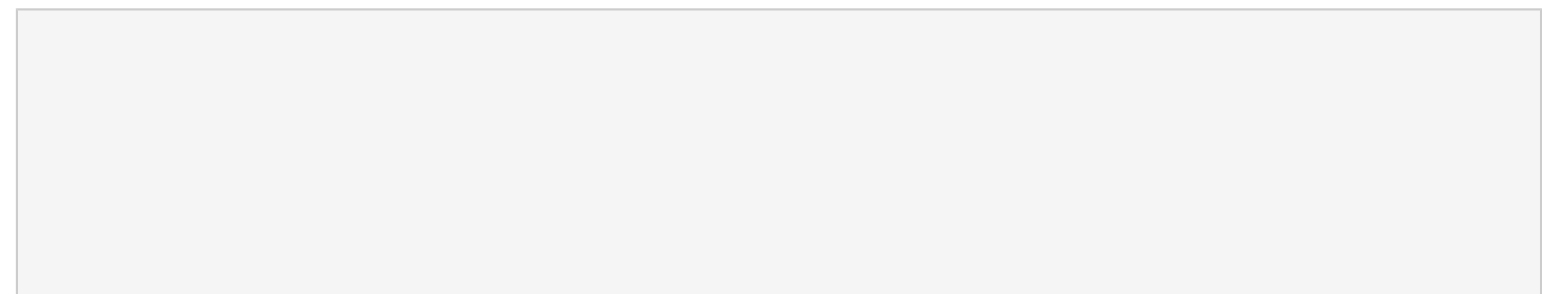
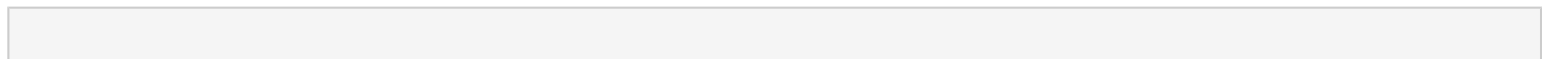
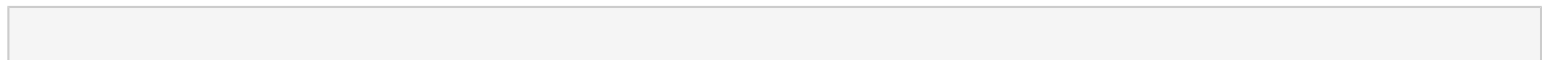

```

(obj1), numel(obj1), numel(obj2)));
0057     result = 0;
0058     return
0059 end
0060
0061 exception_list = varargin;
0062
0063 %% Convert a potential existing plist into a exception
0064 if ~isempty(exception_list) && isa(exception_list{1}, 'plist')
0065     exception_list = find587.0027 817.6022 cm 0 0 m 0 -785.207 l n.0027 81s');
0066     if isempty(exception_list)
0067         exception_list = cell(0);
0068     elseif ~iscell(exception_list)
0069         exception_list = cellstr(exception_list);
0070     end
0071 end
0072
0073 %% for each element in obj1 and obj2
0074 for jj = 1:numel(obj1)
0075
0076     result = 1;
0077
0078     fields = fieldnames(obj1(jj));
0079
0080     for ii = 1:length(fields)
0081         field    = fields{ii};
0082
0083         %% Creates the exception list for the current field.
0084         %% For example: {'name', 'ao/name'}
0085         ck_field = {field, sprintf(0 0 m 0 -785.207 l %s/%s', class(obj1), field)};
0086
0087         % Specj T*case (for the ao- and history-class):
0088         % Is the field = 'hist', 'inhists' then add 'history' to the exception list.
0089         %% For example: {'history', 'ao/history'}
0090         if ismember(field, {0 0 m 0 -785.207 l hist'm 0 0 m 0 -785.207 l inhists'})
0091             ck_field{end+1} = 0 0 m 0 -785.207 l history';
0092             ck_field{end+1} = sprintf(0 0 m 0 -785.207 l %s/history', class(obj1));
0093

```


Generated on Fr1 07-Mar-2008 15:46:43 by [m2html](#) © 2003

[Home](#) > [m 0 11.1962 37.8577 808.023 >>BDC BT 0 0 0 rg /TT1 1 Tf 11.1962 0 0 11.1962 59.9478 808.0202 Tm](#)

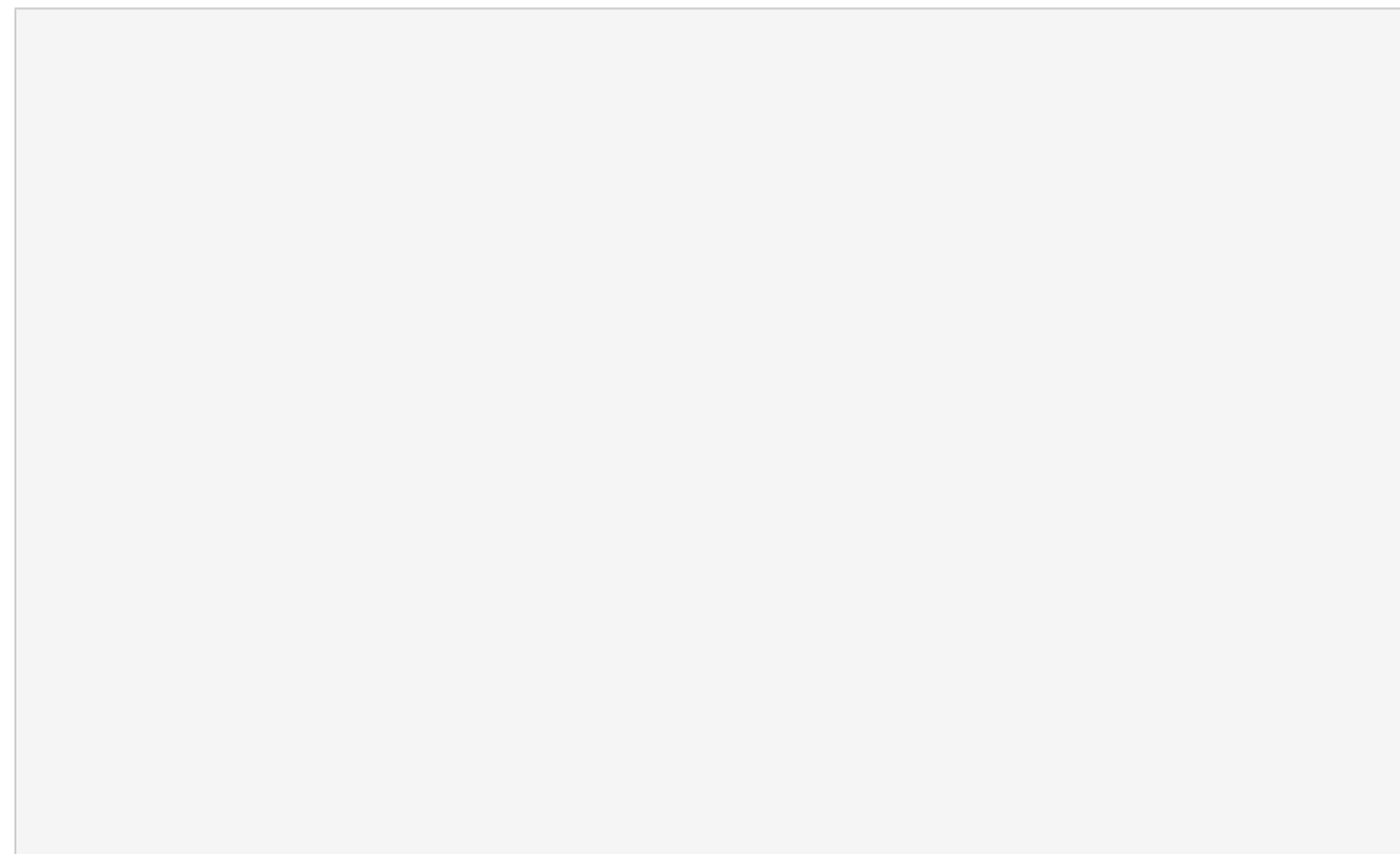
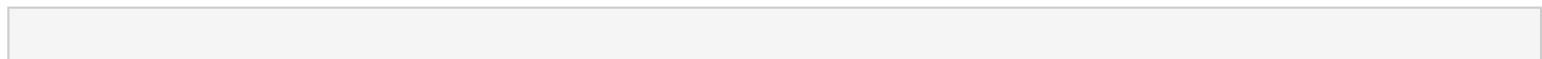
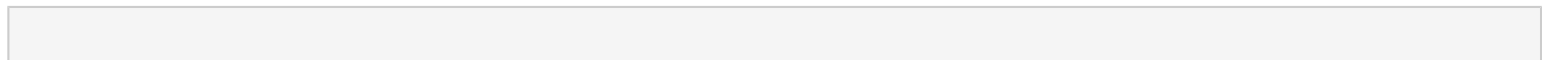


Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

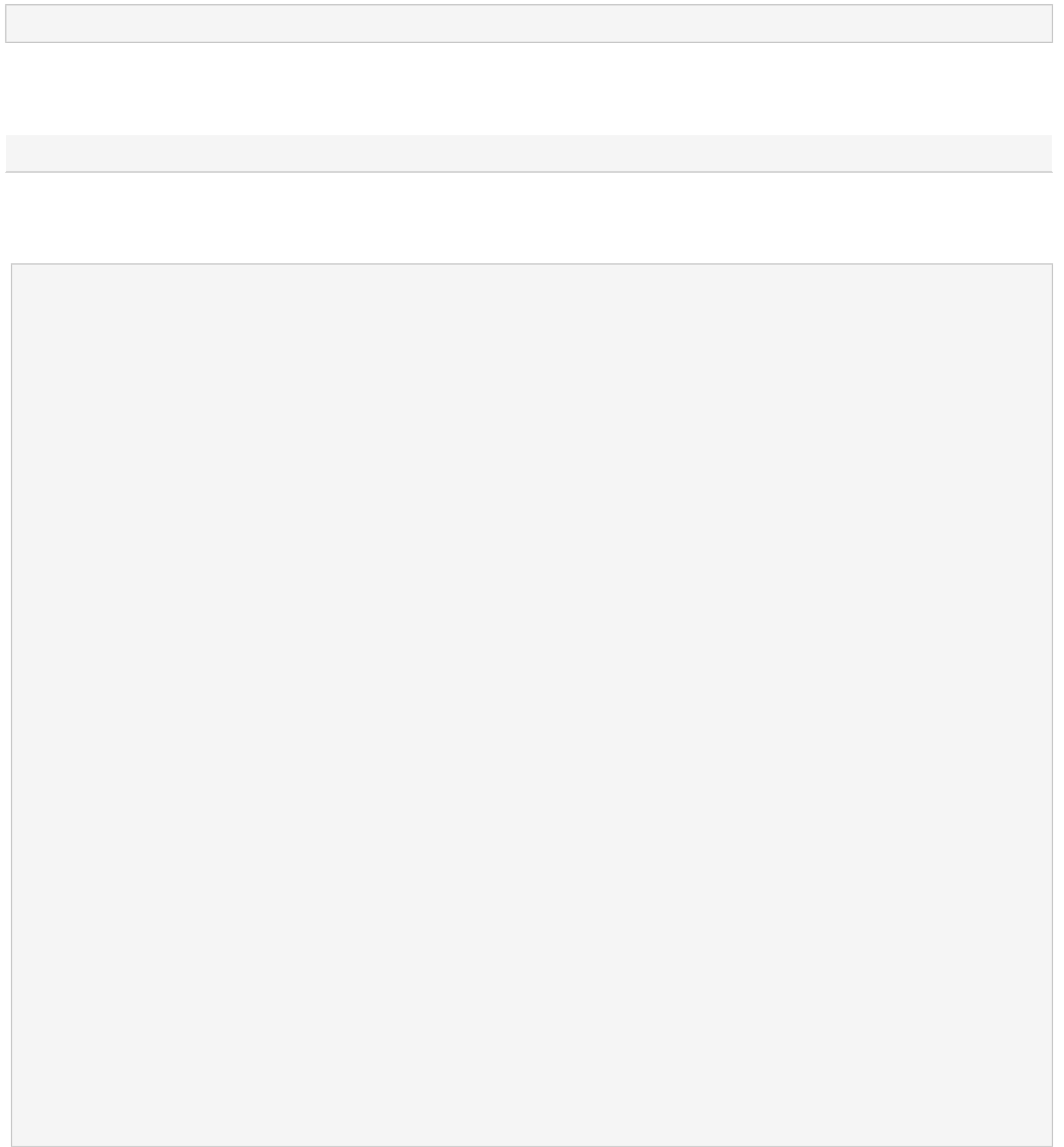
PURPOSE

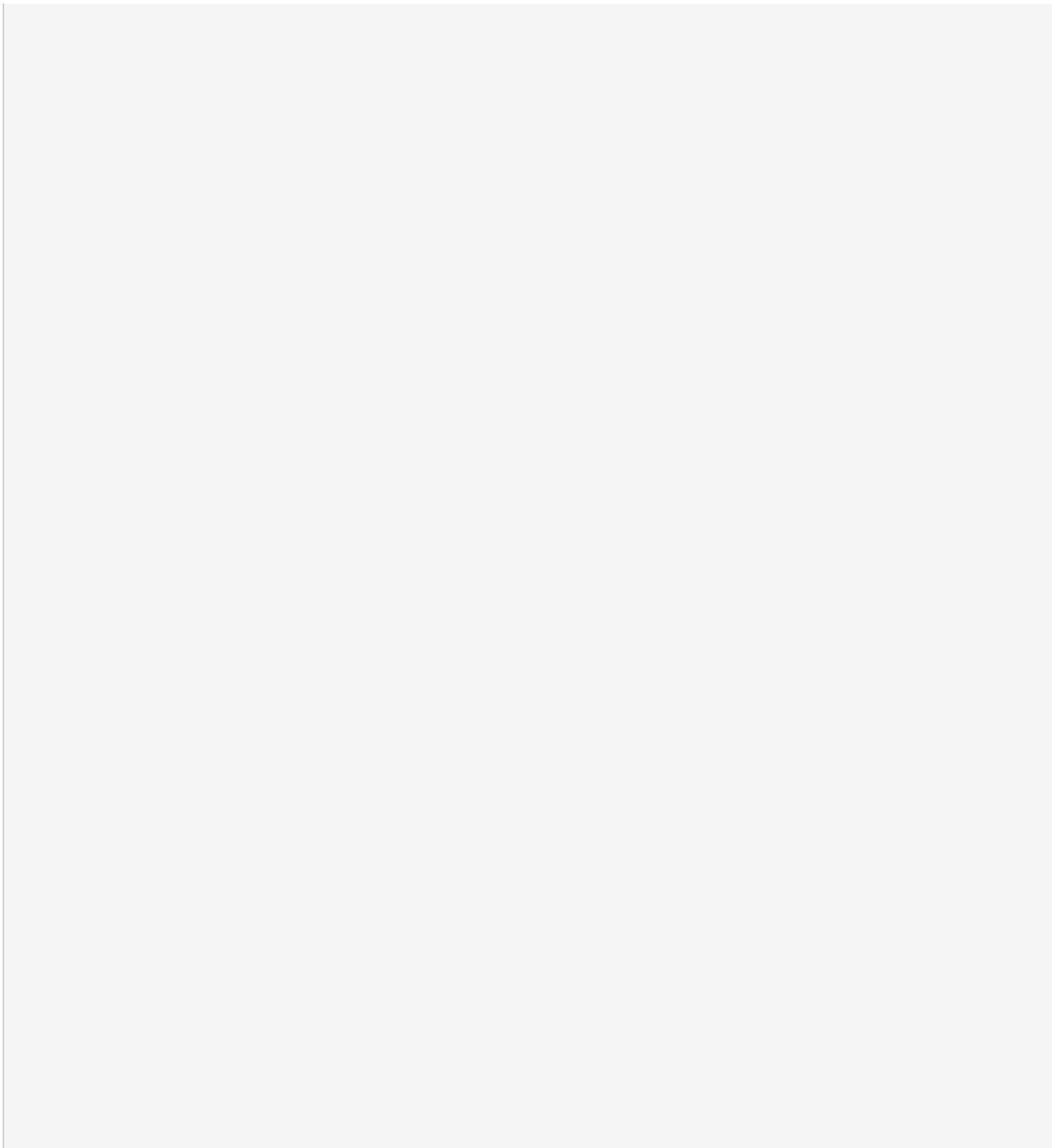

```
0010 %
0011 % INPUTS:      struct1 - a structure
0012 %              struct2 - a structure
0013 %
0014 % OUTPUTS:     out - structure with all fields of input structures.
0015 %
0016 % VERSION: $Id: ltpda_structcat.m,v 1.5 20071 0013 %
% VERSION: $Id: ltpda_7
sonOUTPUTS:      out - st9 a structure
0013 0011
```


[Home](#) >



[Home](#) >





CROSS-REFERENCE INFORMATION

This fuection calls:

This fuection is called by:

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

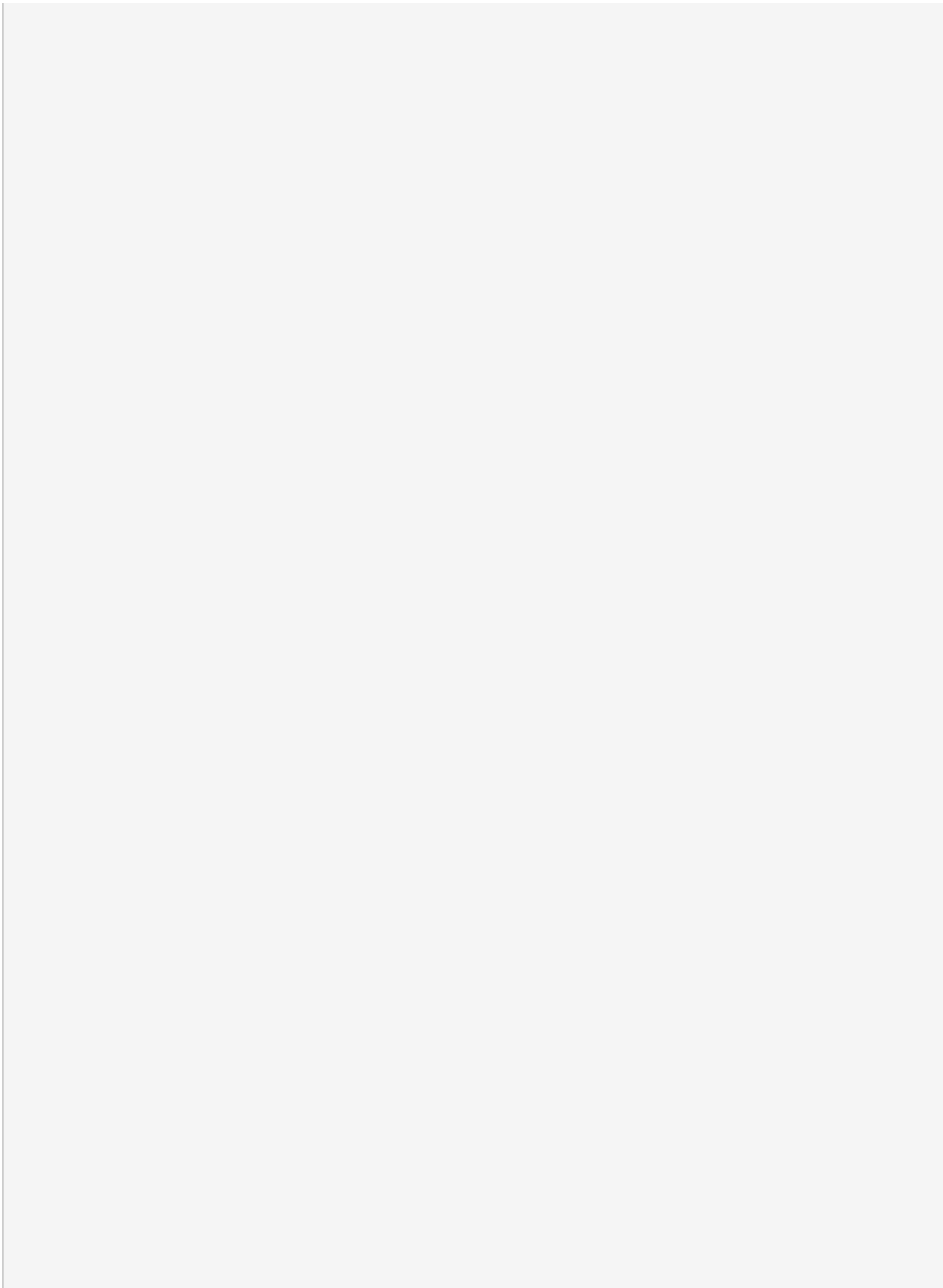

```
0086 %             'define an ExpandFcn for the nodes.']);
0087 %         end
0088 %
0089 %         if (count == 0)
0090 %             nodes = [];
0091 %         end
0092 %         % -----
0093 %
0094 %         See also UITREENODE, UITABLE, PATH
0095 %
0096 %
0089 %             nodes 0089 %             end%             % -----0096
```



```

0201
0202 evdnode = evd.getCurrentNode;
0203 % indices = [];
0204
0205 if ~tree.isLoaded(evdnode)
0206     value = evdnode.getValue;
0207
0208     % <call a user function(value) which returns uitreenodes>;
0209     cbk = expfcn;
0210     if iscell(cbk)
0211         chldnodes = feval(cbk{1}, tree, value, cbk{2:end});
0212     else
0213         chldnodes = feval(cbk, tree, value);
0214     end
0215
0216     if (length(chldnodes) == 1)
0217         % Then we dont have an array of nodes. Create an array.
0218         chnodes = chldnodes;
0219         chldnodes = javaArray('com.mathworks.hg.peer.UITreeNode', 1);
0220         chldnodes(1) = java(chnodes);
0221     end
0222
0223     tree.add(evdnode, chldnodes);
0224     tree.setLoaded(evdnode, true);
0225 end
0226
0227 end
0228
0229 %% -----
0230 function nodeSelected(src, evd, tree, selfcn)                                %#ok
0231 cbk = selfcn;
0232 hgfeval(cbk, tree, evd);
0233
0234 end
0235
0236 %% -----
0237 function [node, expfcn] = processNode(root)
0238 expfcn = [];
0239
0240 if isempty(root) || isa(root, 'com.mathworks.hg.peer.UITreeNode') || ...
0241     isa(root, 'javahandle.com.mathworks.hg.peer.UITreeNode')
0242     node = root;
0243 elseif ishghandle(root)
0244     % Try to process as an HG object.
0245     try
0246         node = uitreenode(handle(root), get(root, 'Type'), ...
0247             [], isempty(get(0, 'children')));
0248     catch
0249         node = [];
0250     end
0251     expfcn = @hgBrowser;
0252 elseif ismodel(root)
0253     % Try to process as an open Simulink system
0254
0255     % TODO if there is an open simulink system and a directory on the path with
0256     % the same name, the system will hide the g rg eerhaps Theshoulg (end)Tj 0 0

```



0315

73.5419 805l4s2es.m73.5419 805l4Ies.m

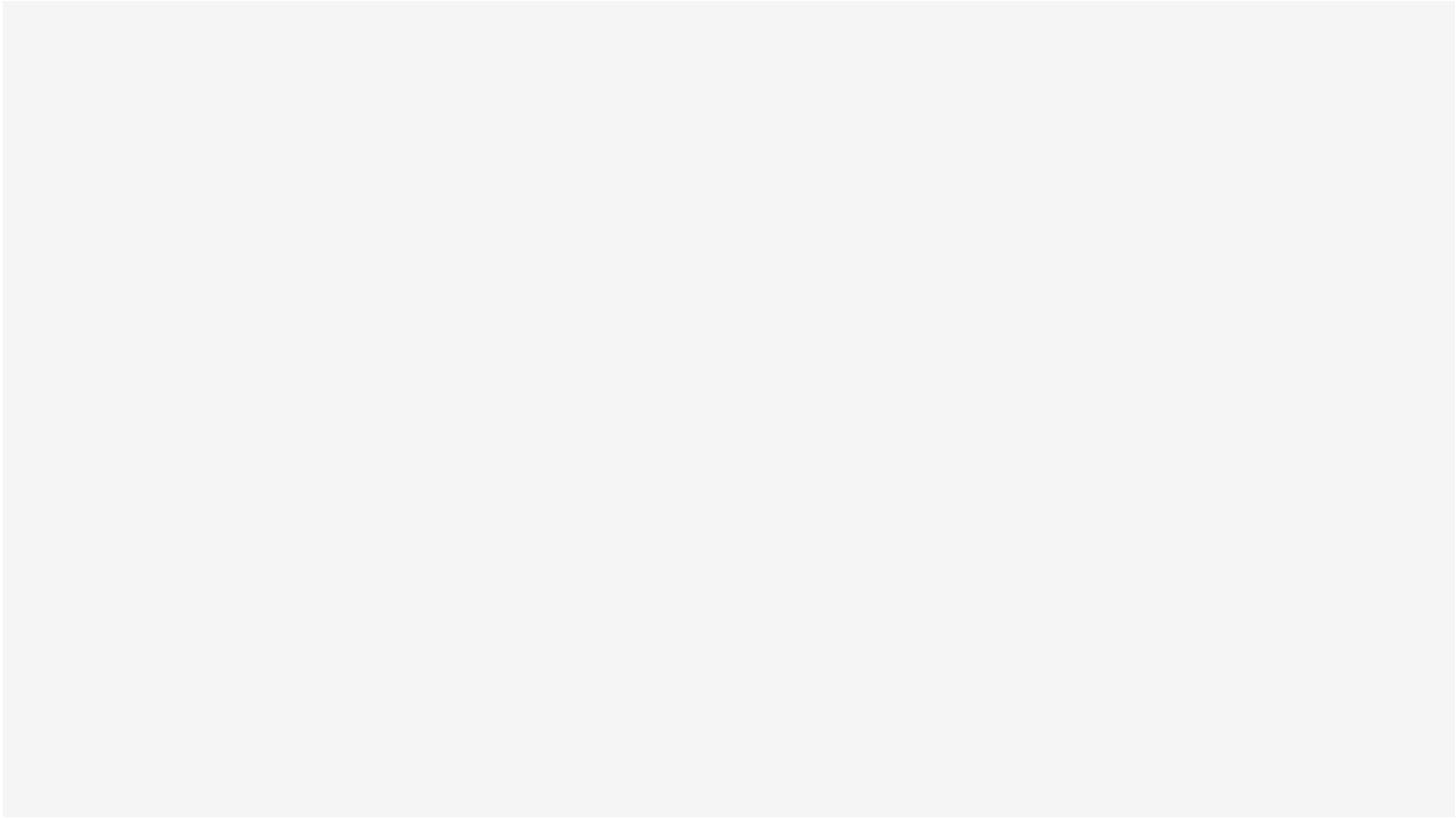
This function is called by:

- `jcontrol` JCONTROL constructor for JCONTROL class
- `set` SET method overloaded for JCONTROL class

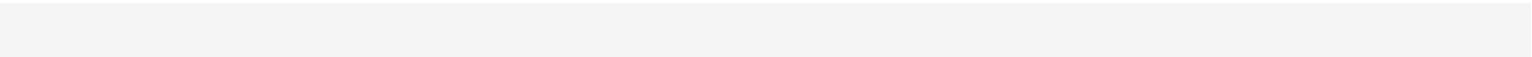
SOURCE CODE

```
0001 function varargout=set(varargin)
0002 % SET method overloaded for JCONTROL class
0003 %
0004 % Examples:
0005 % set(obj,PropertyName, PropertyValue)
0006 % set(obj, PropertyName1, Value1, PropertyName2, Value2....)
0007 %
0008 % The propertyname/valuenam sequence can have an embedded cell array if
0009 % that contains property/value pairs e.g
0010 %         standardvalues={'Units', 'normalized'}
0011 %         set(myobj, 'javax.swing.JPane',...
0012 %                 'ToolTipText','MyTip',...
0013 %                 standardvalues);
0014 %s
0015 % Also:
0016 % set(obj, s)
0017 % set(obj, pn, pm)
0018 % where s in a structure with fieldnames corresponding to property names
0019 % and values corresponding to property values
0020 % pn and pm and name/value cell vectors (note pm may not be a matrix)
0021 %
0022 % If obj is a vector of JCONTROLS, SET wi0 rgs7 to property ond vaar element T* (
0010 % Also:
0010
```

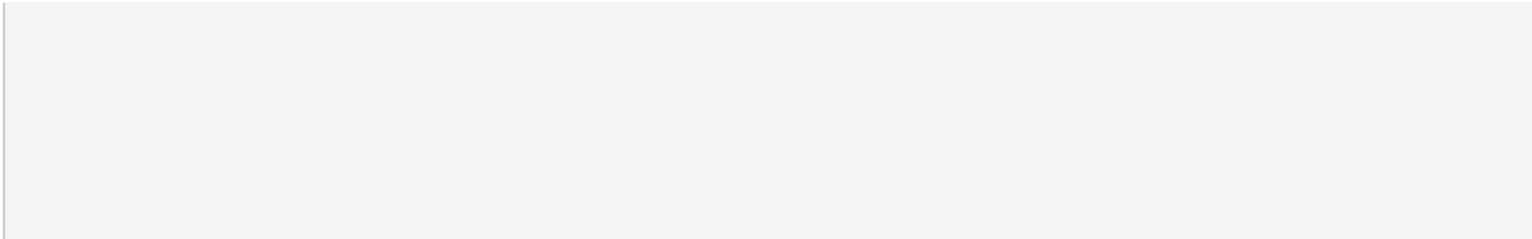
```
0051     {Tproperty/value pairs in structure
0052     {TGet the names
0053     propnames=fieldnames(varargin{2});
0054     {Tget the values
0055     propvalues=cell(length(propnames),1);
0056     for i=1:length(propnames)
0057         propvalues{i}=varargin{2}.(propnames{i});
0058     end
0059     {TRecursive call to set
0060     set(varargin{1}, propnames, propvalues);
0061     return
0062 elseif nargin==3 && iscell(varargin{2}) && iscell(varargin{3})
0063
```

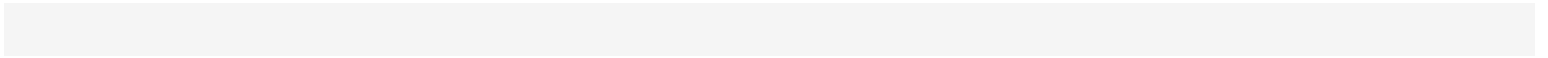


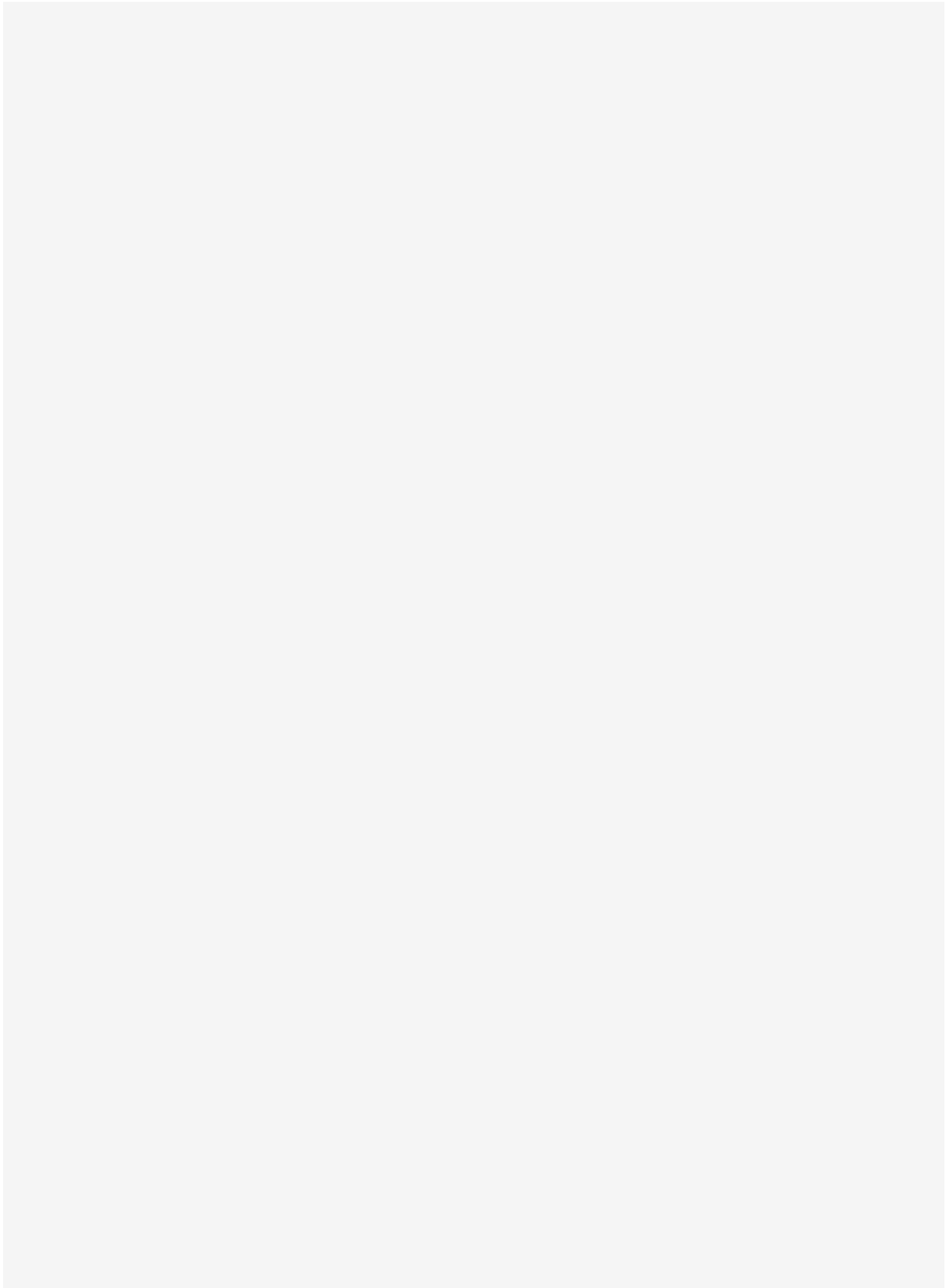
This function is called by:




```
0202
0203 lmin = 1e20;
0204 for j=1:length(as)
0205     if len(as(j)) < lmin
0206         lmin = len(as(j));
0207     end
0208 end
0209
0210
0211 %-----
0212 0204 for0205     if0207     end0208 end0202 0211 %-----
```

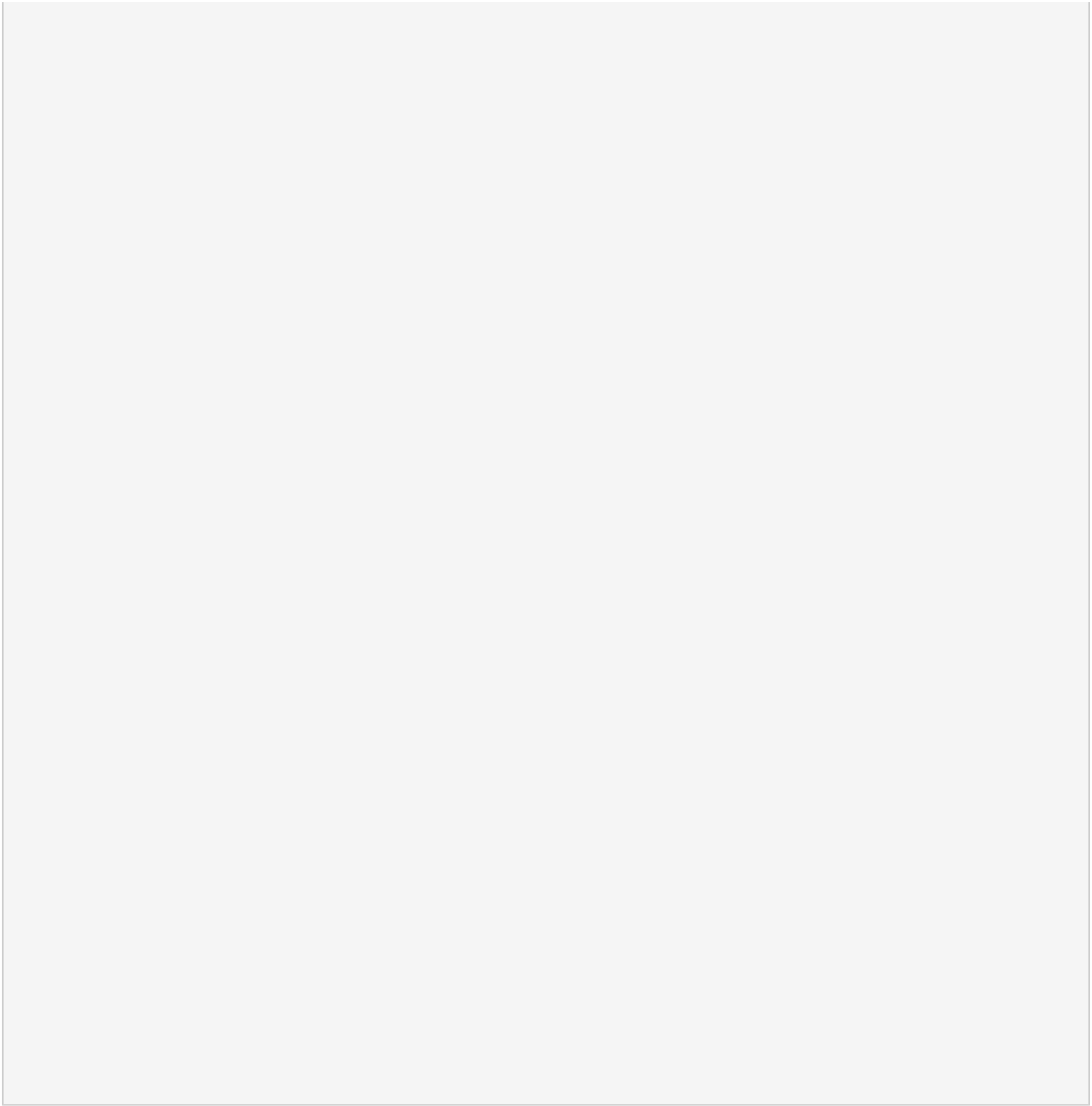





```

0559     showing = false;
0560
0561     if strcmp(node_name, 'matrix')
0562         if number >= THRESHOLD_DISP_MATRIX
0563             showing = true;
0564             add_text = 'matrix lines';
0565         end
0566     else
0567         if number >= THRESHOLD_DISP_VECTOR
0568             showing = true;
0569             add_text = 'data samples';
0570         end
0571     end
0572
0573     if showing == true
0574
0575         parent = node.getParentNode; % The parent node have the 'property name'
information
0576
0577         if header_displayed == true
0578             if parent.hasAttribute('prop_name')
0579                 disp_prop_name = char(parent.getAttribute('prop_name'));
0580             else
0581                 disp_prop_name = 'Unknown Property Name';
0582             end
0583             disp (sprintf('\n%%%%%%%%%%    Read property [%s]    %%%%%%%%%%%',
disp_prop_name));
0584
0585             if strcmp(char(node.getNodeName), 'real_data')
0586                 disp('%%%%%%%%    Real_da a sammmmmmmmm%',
0586 7             else
0581 8             disp('%%%%%%%%    Reaimnatiory a sammm%',
0586 9             elseTj 0 0 0 rg T* (0583 9             )Tjder_displayed == tlse;0583 91     elseTj 0 0
0584 94

```


```

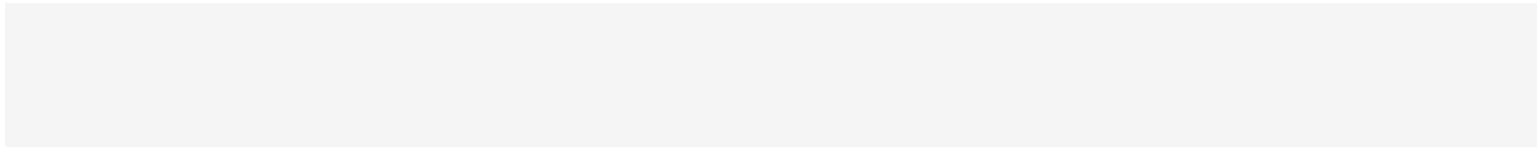
0158     if objs
0159         content = xml.createTextNode('true');
0160     else
0161         content = xml.createTextNode('false');
0162     end
0163     parent.appendChild(content);
0164 else
0165     error ('### At the moment is only one logical allowed. The size is [%dx%
d]', size(objs,1), size(objs,2));
0166 end
0167
0168
rg (if)Tj 0 0 0 risnumeimesize() ||risasize(o r (if (, size(objs == bjs &&r( (, size(o
0165%.17g lowed. The size i (if)T number_strt8 0 1 rg (else)Tj 0 0 0 ;else
else

0161     0 0 rg167     number_str8 0 0 1 rg (end)Tj 0 0 0 rg T* (0163     parentT0
>

```



```
0271 %%%% Imaginary data %%%%
0272 if ~isreal(objs)
0273     node_imag = xml.createElement('imag_data');
0274     node_imag.setAttribute('type', 'vector')
0275     node_imag.setAttribute('shape', shape);
0276     parent.appendChild(node_imag);
0277
0278     write_number = imag(objs);
0279     sample_num    = 0;
0280
0281     while ~isempty(write_number)
0282         n = min(n_min, length(write_number));
0283
0284         sample_num = sample_num + n;
0285         header_displayed = TerminalOutput
```

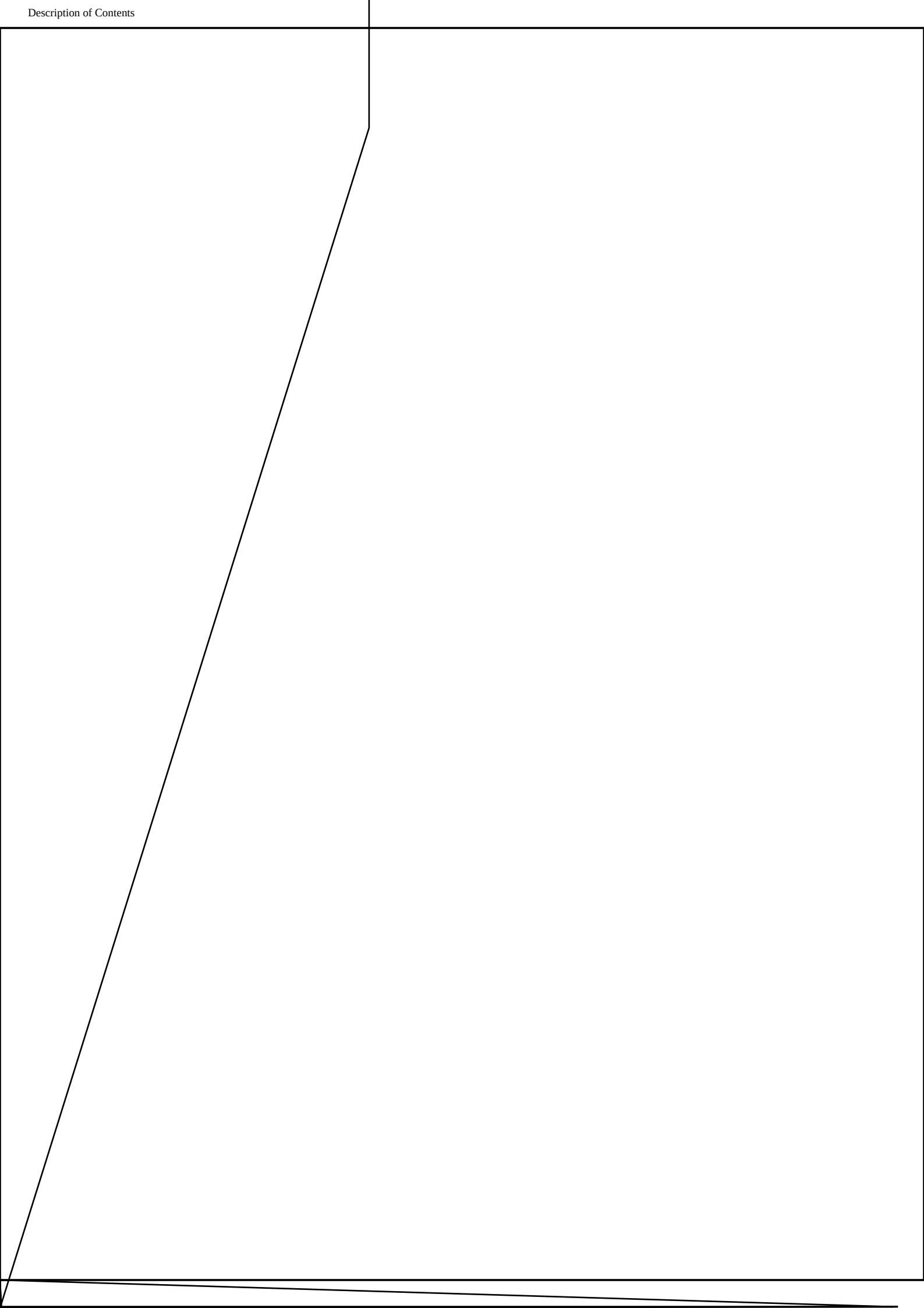

```

0011 %
0012 % -----
0013 % Author: Malcolm Lidierth 07/07
0014 % Copyright © The Author & King's College London 2007
0015 % -----
0016
0017 % No property: list contents
0018 if nargin==1
0019     if numel(obj)==1
0020         s.hgcontainer=obj.hgcontainer;
0021         s.hgcontrol=obj.hgcontrol;
0022         s.hghandle=obj.hghandle;
0023     else
0024         display(obj);
0025     end
0026     return
0027 end
0028
0029 % Must specify a single property
0030 if nargin>2
0031     error('Too many input arguments');
0032 end
0033
0034 % Ensure lower-case
0035 property=lower(property);
0036
0037 % Vector of JCONTROLS on input
0038 % Call GET on each in turn returning a cell array
0039 if numel(obj)>1
0040     if nargin>0
0041         % Pre-allocate
0042         s=cell(1,numel(obj));
0043         for idx=1:numel(obj)
0044             % Recursive call
0045             s{idx}=get(obj(idx),property);
0046         end
0047         return
0048     else
0049         % Stop problems in nested gets e.g. (get(get,...)... )
0050         error('Must specify a left-hand side assignment with vector input');
0051     end
0052 end
0053
0054 % Check for ambiguity in property name
0055 if isprop(obj.hgcontainer, property) &&...
0056     isprop(obj.hgcontrol, property)
0057     if strcmp(property, 'visible')
0058         % Visible is an exception - take this from the container, MATLAB
0059         % links this property for the container and object
0060         s=obj.hgcontainer.Visible;
0061         return
0062     else
0063         error('Shared property name '%s''\nYou must explicitly specify the target
object',...
0064             property);
0065     end
0066 end

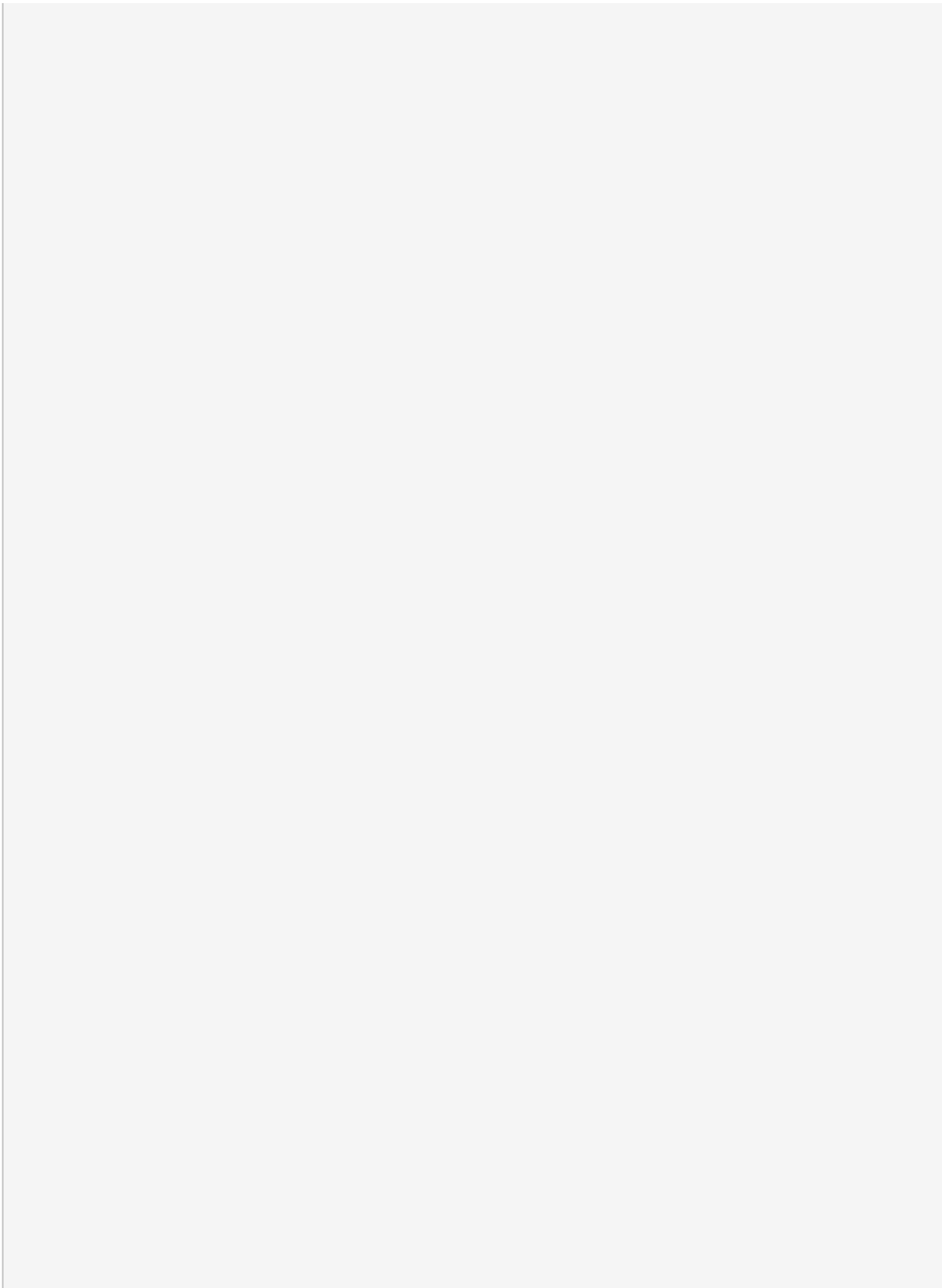
```

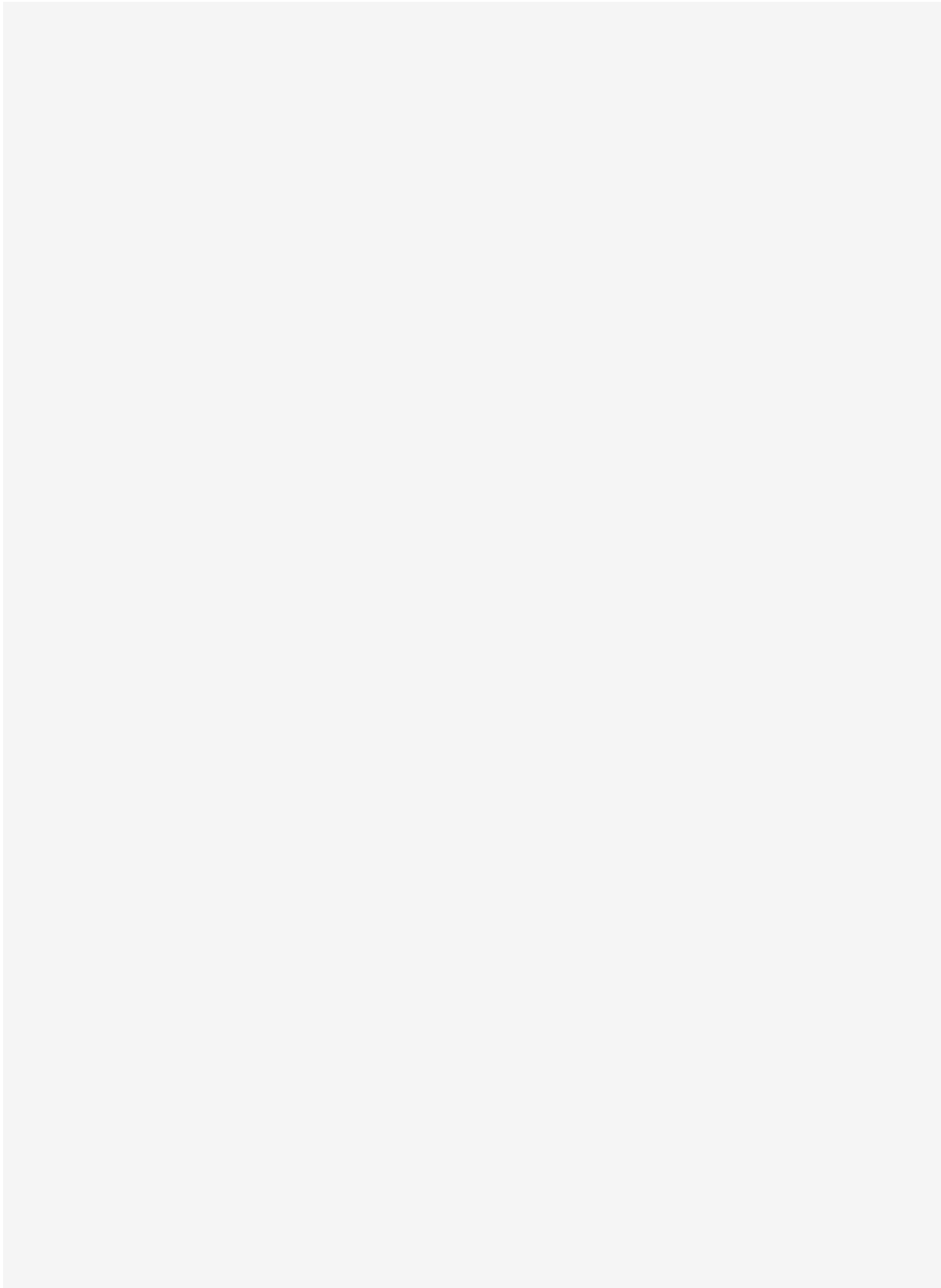


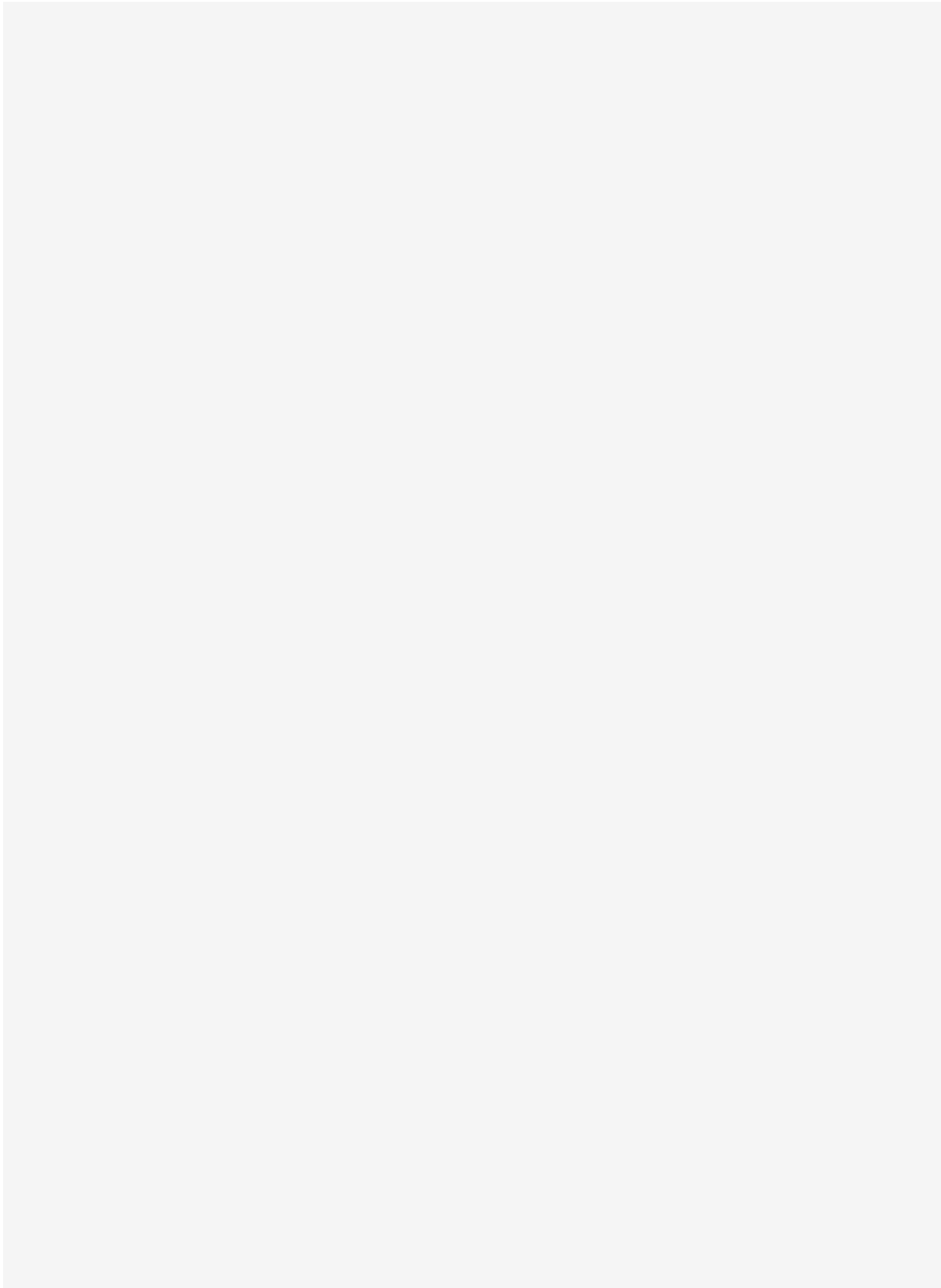
```
0067
0068 % Do the work
0069 switch property
0070     case 'hghandle'
0071         s=obj.hghandle;
0072     case 'hgcontainer'
0073         s=obj.hgcontainer;
0074     case 'hgcontrol'
0075         s=obj.hgcontrol;
0076     otherwise
```

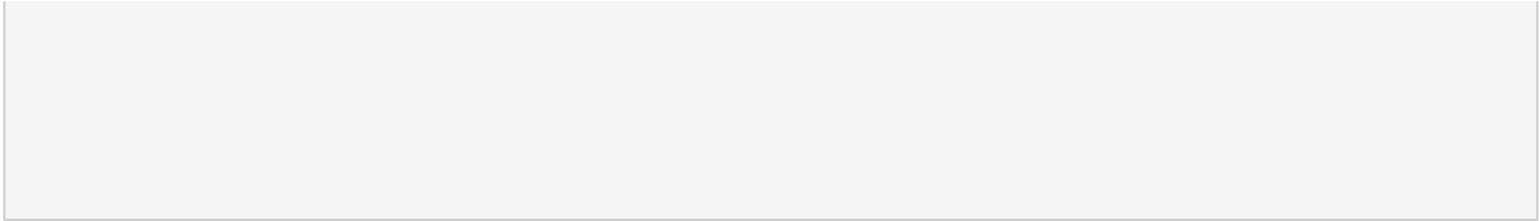


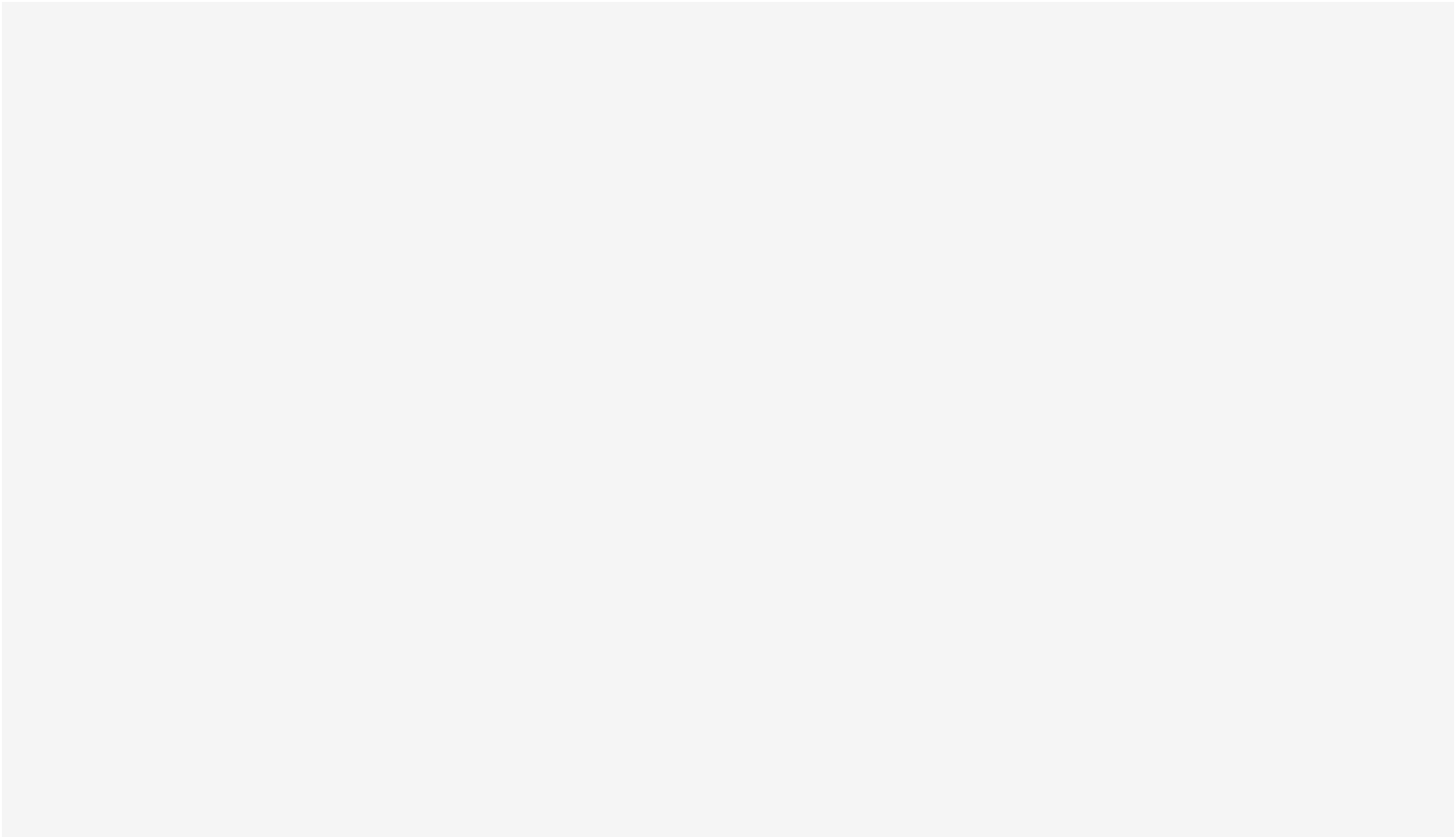






```
0216     bs(j,k) = {b};
0217     end % End second loop over A0s
0218 end % End first loop over A0s
0219
0220 % now we hav1
```

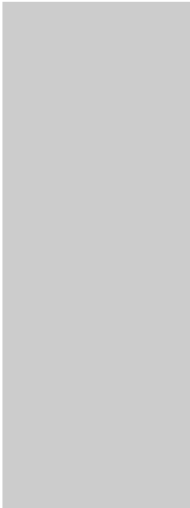
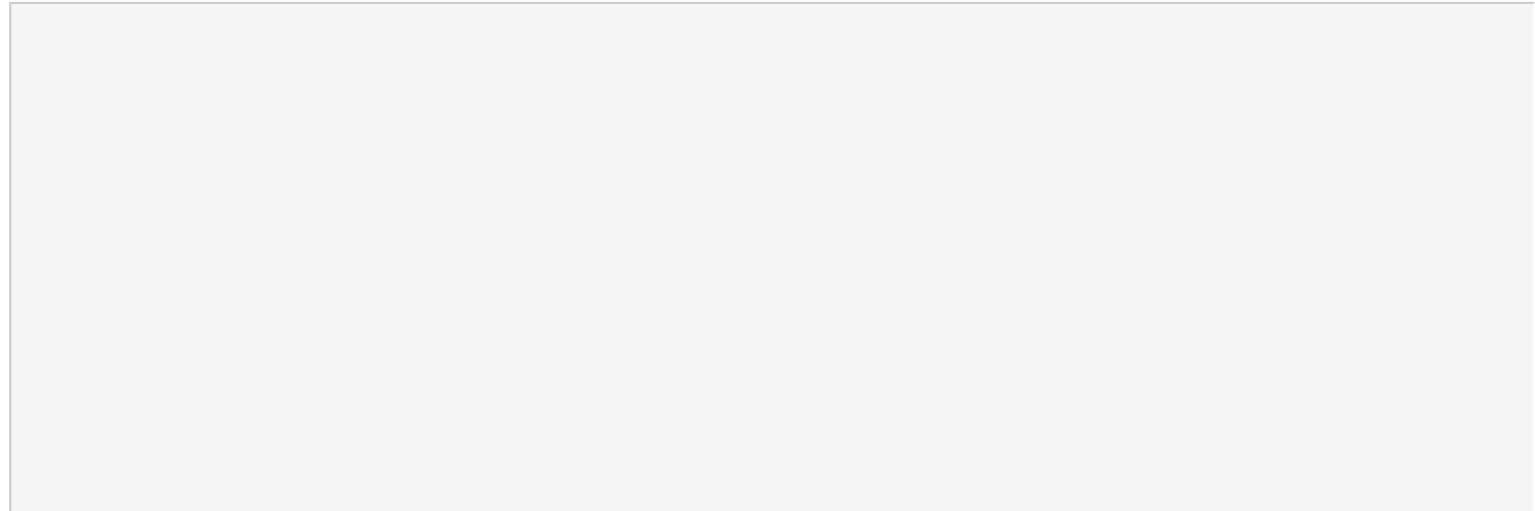
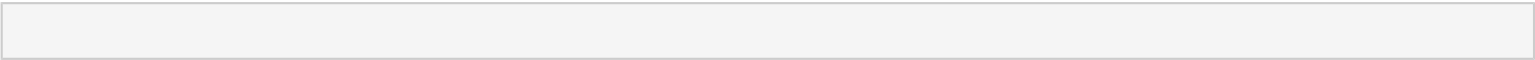


--	--

--	--

CROSS-REFERENCE INFORMATION

This function calls:




```
0011 mainfig = findobj('Tag', 'LTPDAREpomainfig');
0012
0013 % Get conn
0014 conn = getappdata(mainfig, 'connection');
0015
0016 if ~isempty(conn)
0017     % mai the databases
0018     q = 'show tables';
0019     curs = exec(conn, q);
0020     curs = fetch(curs);
0021
0022     tables = curs.Data;
0023     close(curs);
0024 else
0025     tables = {'-'};
0026 end
0027
```

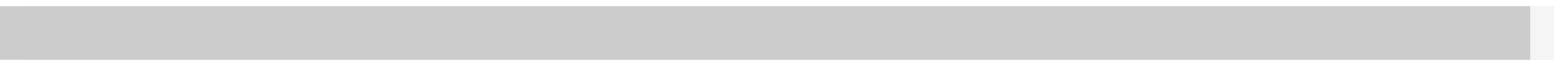
--


```
0013
0014 objs = [];
0015 for j=1:length(ws_vars)
0016
0017     cmd = sprintf('obj = evalin(''base'', '%s');', ws_vars(j).name);
0018     eval(cmd)
0019     if ltpda_isuserobject(obj)
0020         objs = [objs ws_vars(j)];
0021     end
0022
0023 end
0024
0025
0026
0027
```



```
0015 %  
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
0017  
0018 set(gca, 'Ytick', v);  
0019  
0020 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

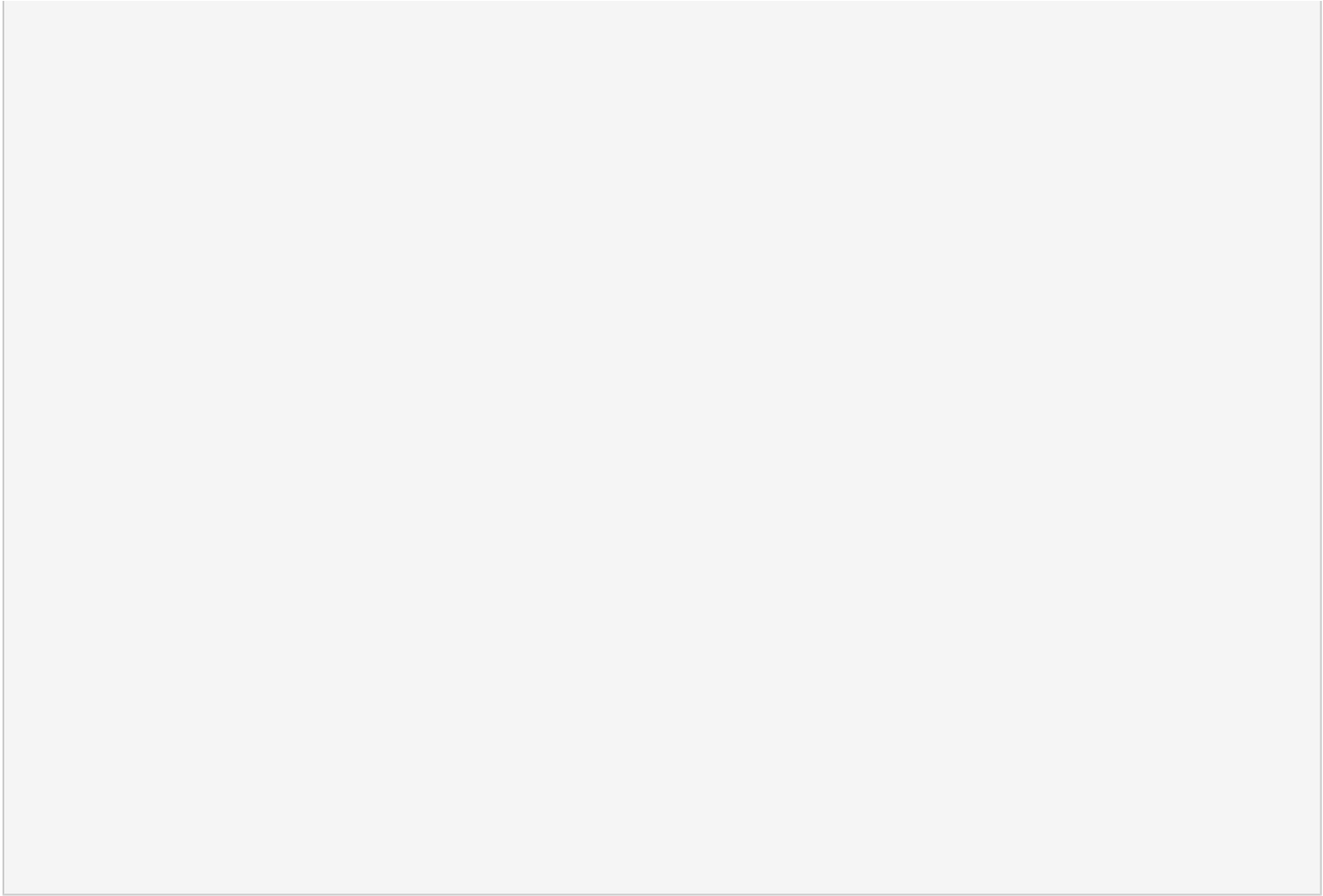


--	--

0067 % END

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

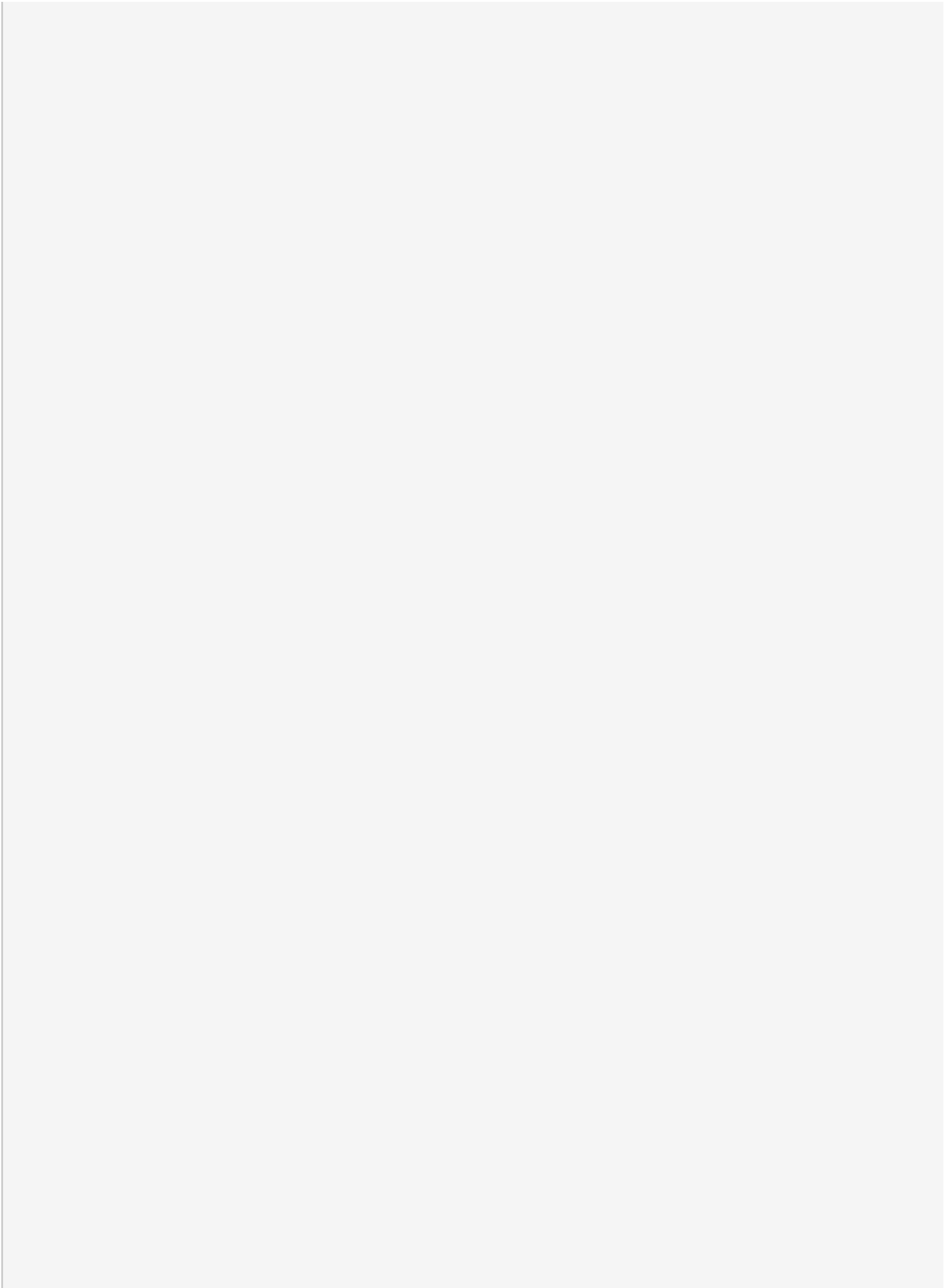

```
0016
0017 set(gca, 'ZScale', scale);
0018
0019 % END
```

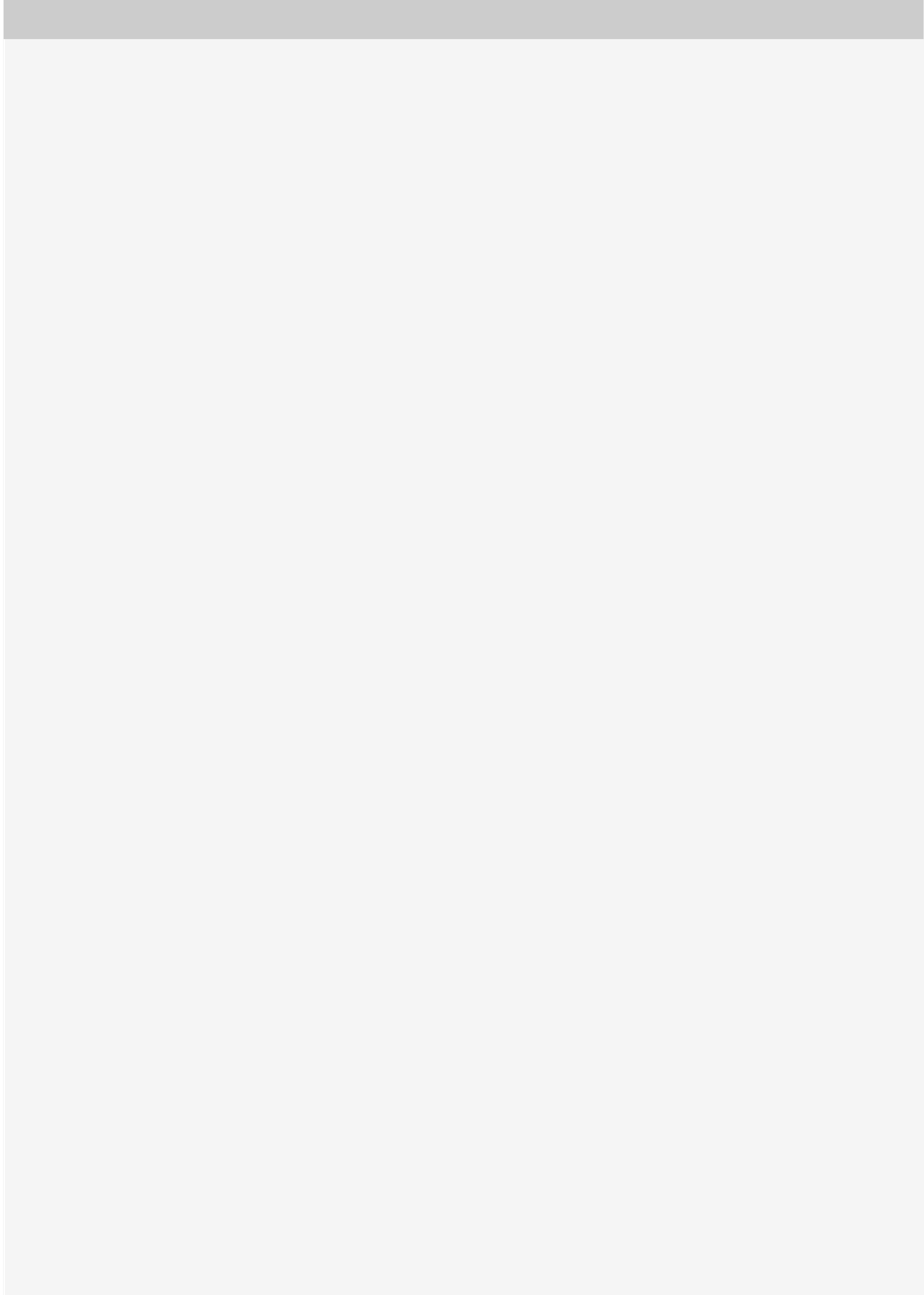

```

0003 %=====
0004 %                                LTPDA GUI
0005 %=====
0006 %
0007 %  Open the tabbed main figure for the LTPDA GUI.
0008 %
0009 %  $Id: ltpdagui.m,v 1.3 2008/03/04 05:00:52 nicola Exp $
0010
0011 warning off all
0012
0013 global ltpdaGUIstatus LTPDAinvar LTPDAoutvar
0014
0015 if isempty(LTPDAinvar), LTPDAinvar = {}; end
0016 if isempty(LTPDAoutvar), LTPDAoutvar = {}; end
0017
0018 userID = getappdata(0,'ltpda_repo_user');
0019 if isempty(userID), userID = ''; end
0020 passwd = getappdata(0,'ltpda_repo_pass');
0021 if isempty(passwd), passwd = ''; end
0022
0023 if nargin==0, [userID passwd] = loginLTPDA(userID,''); end
0024 if isempty(userID), userID = 'JohnDoe'; passwd=''; end
0025
0026 ltpdaGUIstatus{1,1} = userID;
0027 setappdata(0,'ltpda_repo_user',userID)
0028 setappdata(0,'ltpda_repo_pass',passwd)
0029
0030 if nargin==0
0031     clc
0032     ltpdaGUIstatus{1,1} = userID;
0033     setappdata(0,'ltpda_repo_user',userID)
0034     setappdata(0,'ltpda_repo_pass',passwd)
0035     T=800103r0.545E9;
0036     att=0.1;
0037     ExpE=00$2
0038     disp(0

```

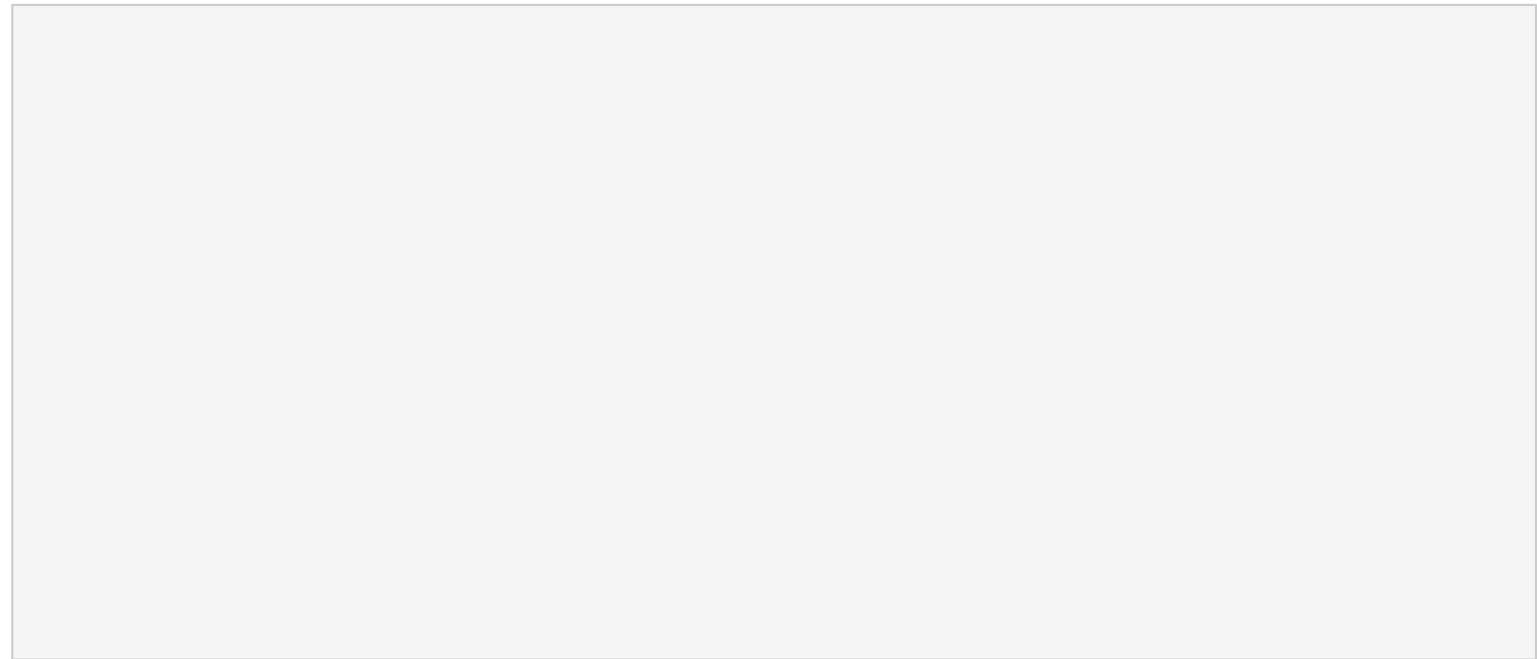
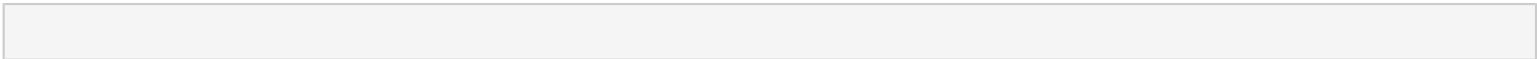
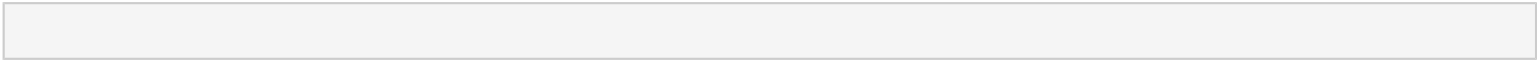



```
0171 titleheight = 1.5*charsz(4);
0172
0173 margins(3:4) = -(margins(1:2)+margins(3:4));
0174 [loop,sign] = deal(1:ntab,[1,ntab,1,0]);
0175
0176 addrgins = 2*[sign(3:4),-1,-1];
```



```
0284 panelTitle = get(ttext(active), 'string');
0285 panelDimens = get(panel, 'Position');
0286     setappdata(0, 'ltpda_currPanel', panel);
0287     setappdata(0, 'ltpda_currPanelDimens', panelDimens);
0288
0289
```

```
0341 % =====
0342 ltpdagui('title',temp,'active',indx);
0343 % =====
0344
0345 end
0346 %-----
0347
0348 %-----
0349 function TopBottomTabResizeCbk(varargin)
0350
0351 units = get(htab,'Units');
0352 set(htab,'Units','pixels');
0353 tabpos = get(htab,'Position');
0354 tabpos(3) = tabpos(3)*0.85;
0355 set(htab,'Units',units);
0356 ntab = numel(string);
0357 for oo=ntab:-1:1, ntab = ntab - strcmp(string(oo),''); end
0358
0359 titlepos = [0,0,0,titleheight];
```

Contents

PURPOSE

%% path: m/math %%%

SYNOPSIS

This is a script file.

DESCRIPTION

%% path: m/math %%%

[m/math/ltpda_lincom](matlab:help m/math/ltpda_lincom) - make the linear combination of the input analysis objects.

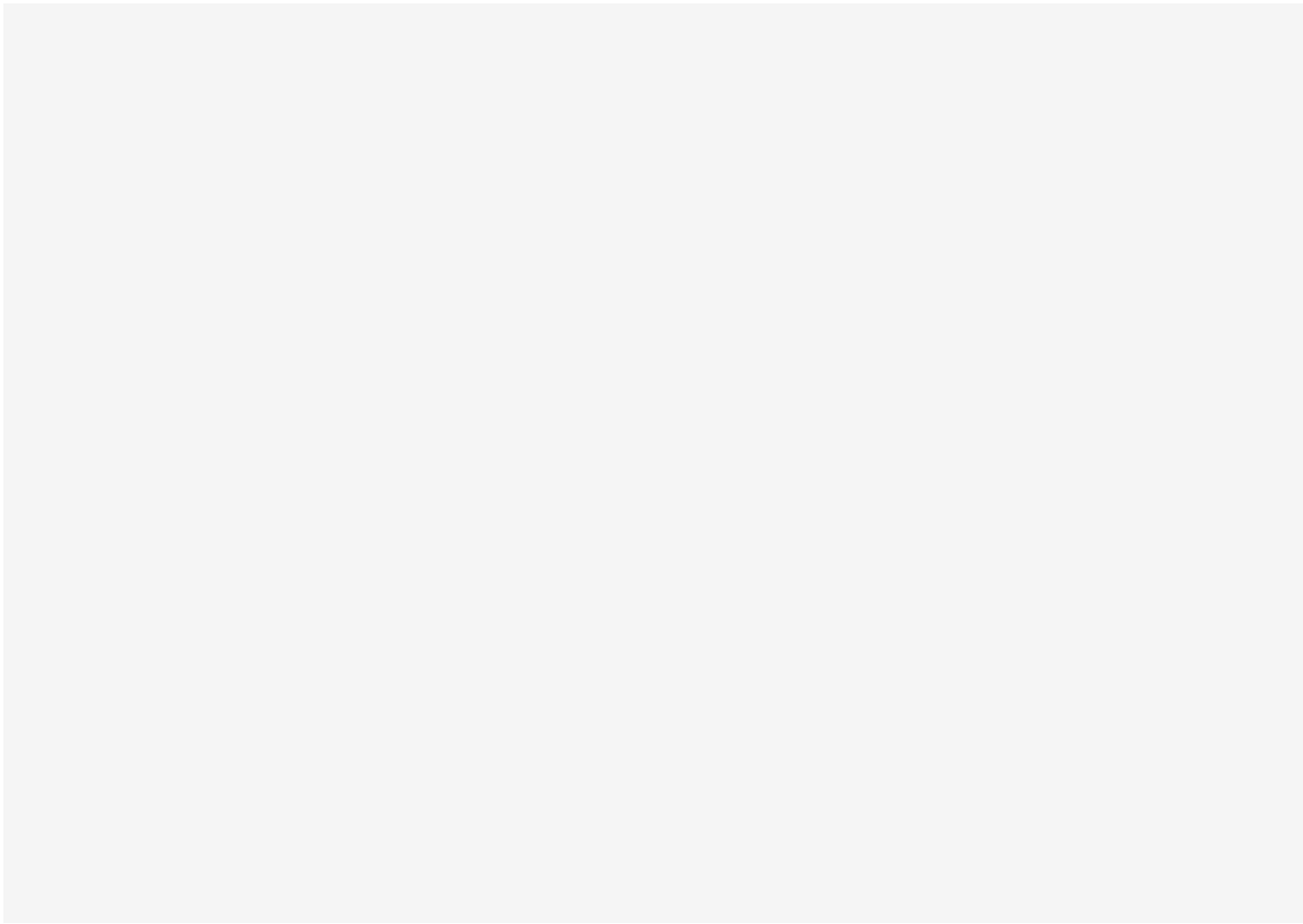
CROSS-REFERENCE INFORMATION

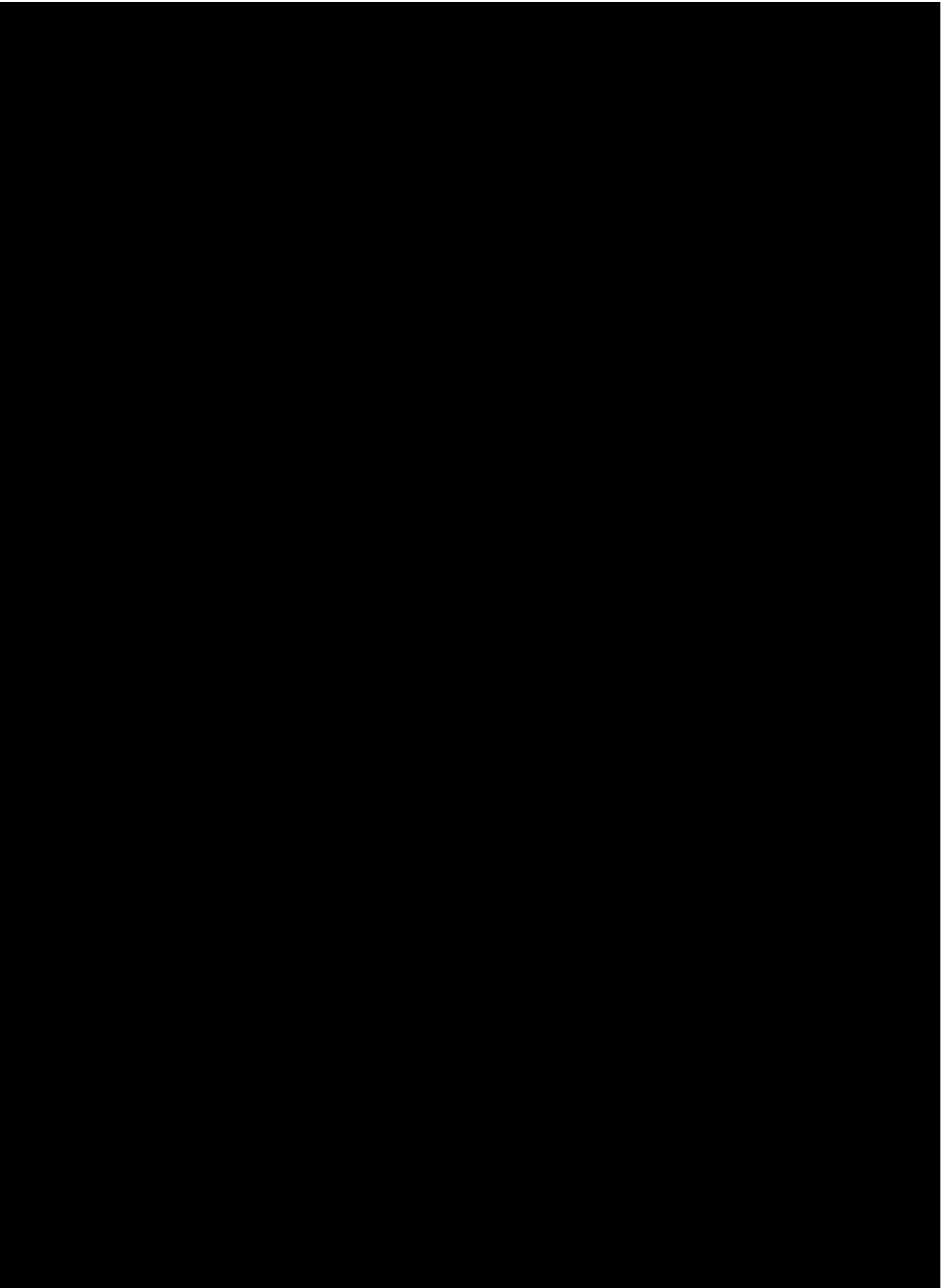
This function calls:

This function is called by:









--

[Home](#) > [m](#) > [gui](#) > [gltpda](#) > [ltpdasim.m](#)

ltpdasim

PURPOSE

This is the automatic function wrapper

SYNOPSIS

function ltpdasim(block)

DESCRIPTION

This is the automatic function wrapper

```
=====
===== level-2 M file S-function =====
=====
```

This wrapper is the automatic function, called by every function block in Simulink, able to execute m-file functions retrieving from Simulink:

- (1) the pointer to the object(s) to be analyzed, coming in as input of the corresponding block (ie, the DATA),
- (2) the name of the function to be applied on those data, from the tag of the currently executed block (ie, the true FUNCTION),
- (3) the parameters for that particular block, retrieved from the global shared workspace by the handle of the block (ie, the PARAMETERS).

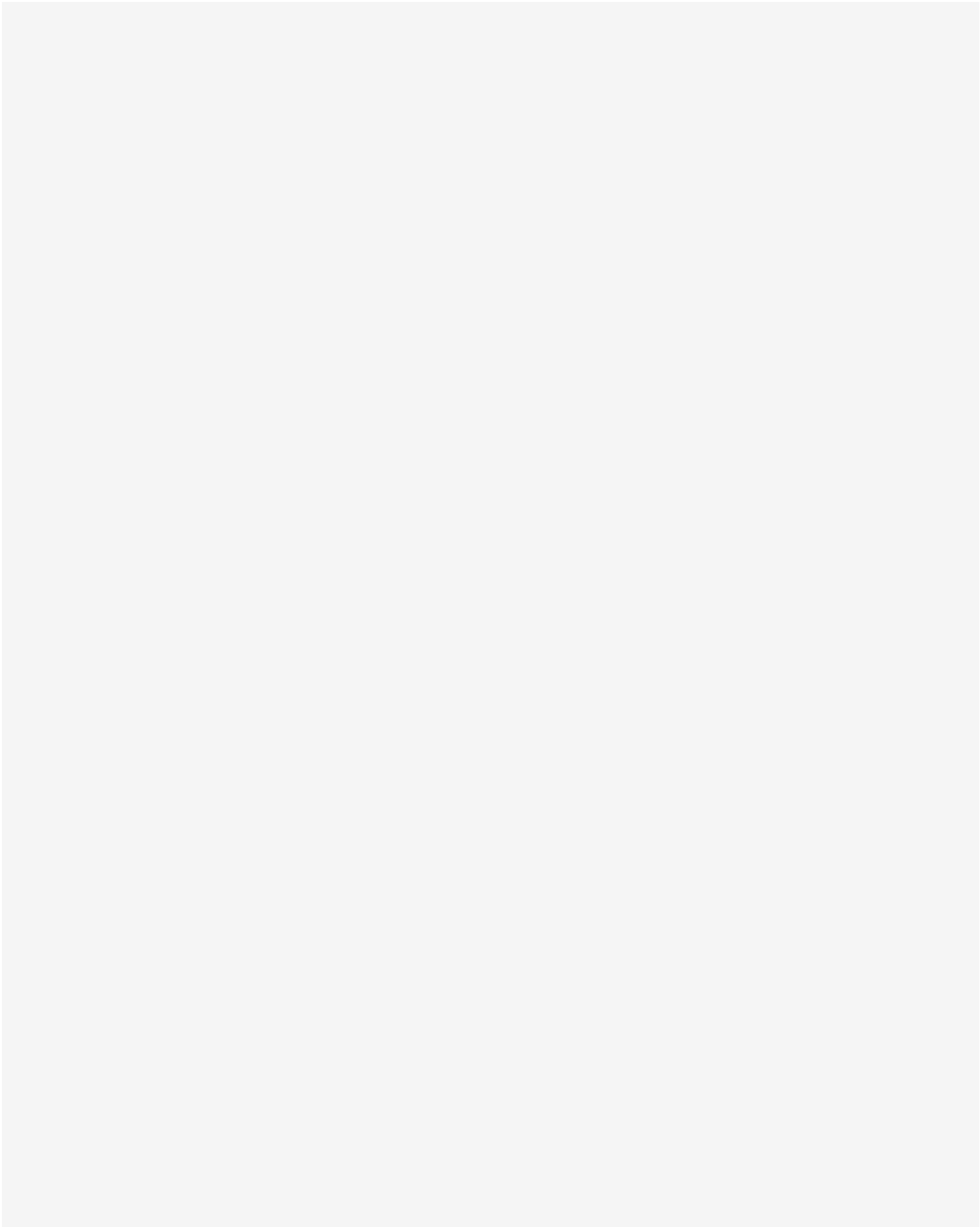
The output is then generated as:

```
OUTPUT = FUNCTION(DATA,PARAMETERS)
```

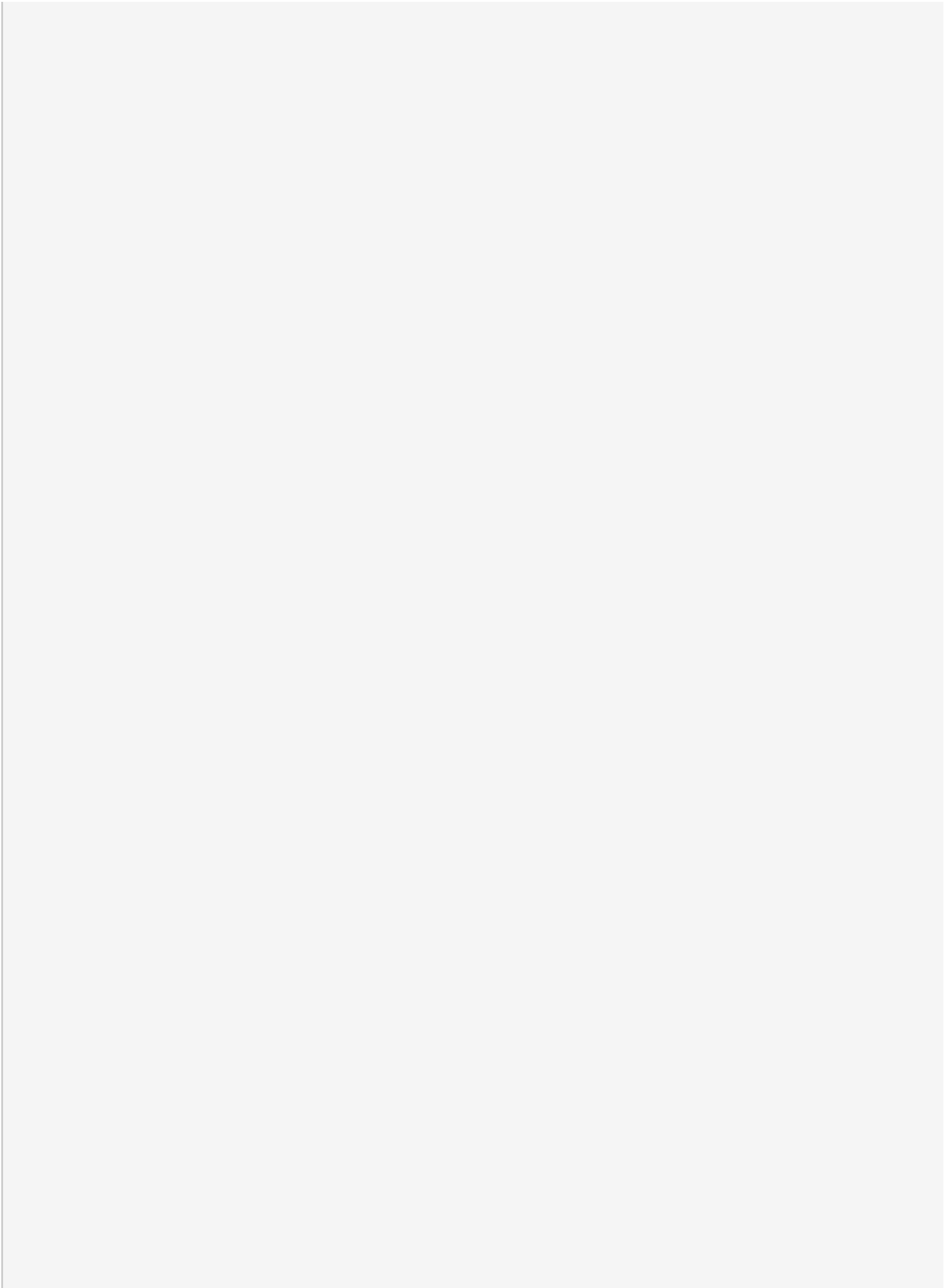
This output in the end is saved into the global array containing all the AOs (ie, all the DATA go together with other data): thus this output will be freely accessible by all the other functions.

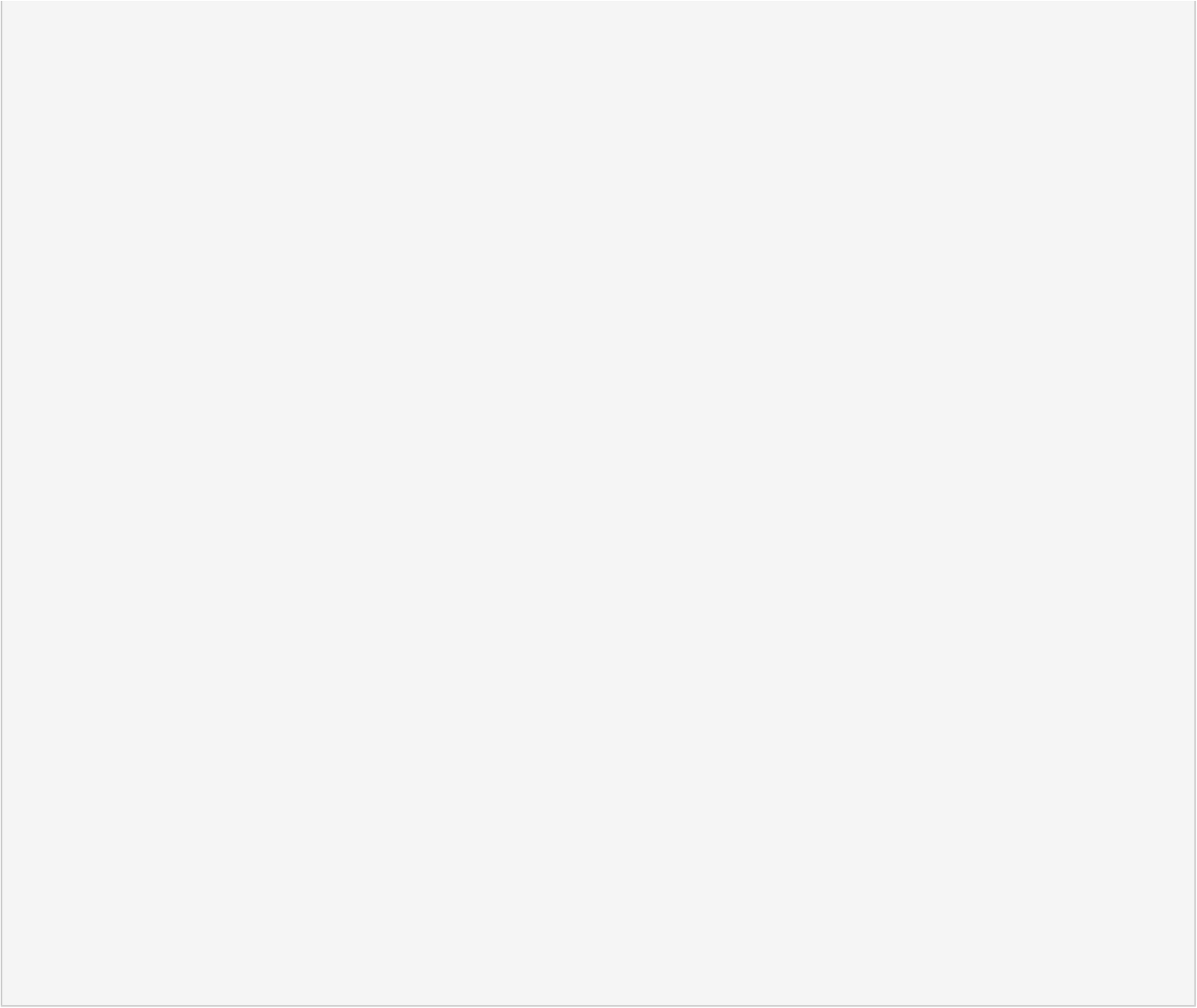
The only real output to Simulink will be just the ordinal number of the so-generated AO into the global array of AOs.

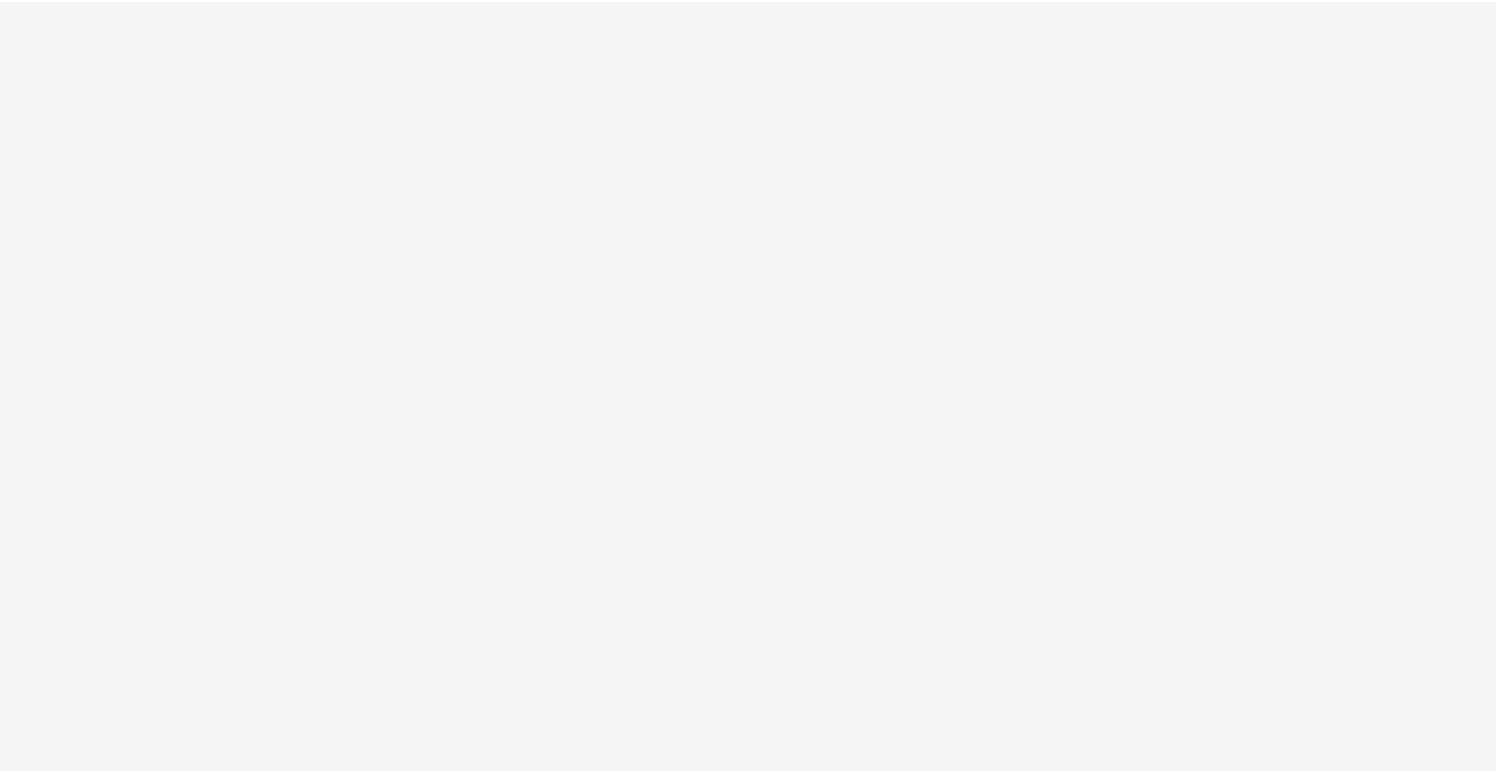
```
$Id: ltpdasim.m,v 1.2 2008/03/03 02:16eely6ger of the
```



```
0054     block.SampleTimes = [0 0];
0055     block.SetAccelRunOnTLC(false);
0056
0057     block.OutputPort(1).Dimensions = 1;
0058
0059     %% Register methods
0060     block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
0061     block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
0062 % block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);
0063     block.RegBlockMethod('Outputs', @Outputs);
0064
0065     function
```





specwin_viewer

PURPOSE

SPECWIN_VIEWER allows the user to explore spectral windows.

SYNOPSIS

```
function varargout = specwin_viewer(varargin)
```

DESCRIPTION

```
SPECWIN_VIEWER allows the user to explore spectral windows.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DESCRIPTION: SPECWIN_VIEWER allows the user to explore spectral windows.

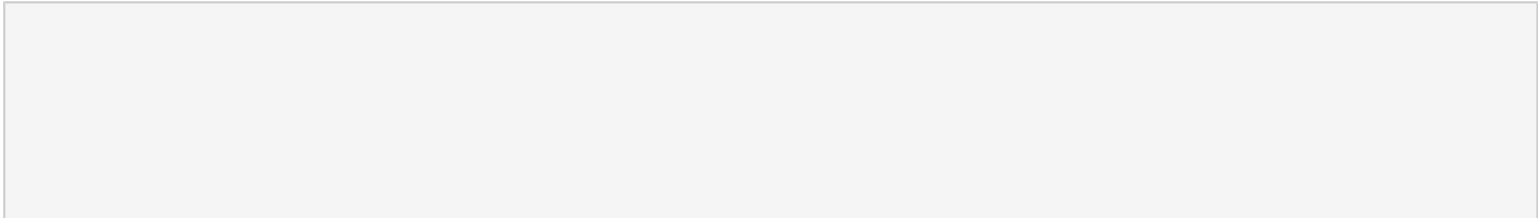
CALL:          specwin_viewer

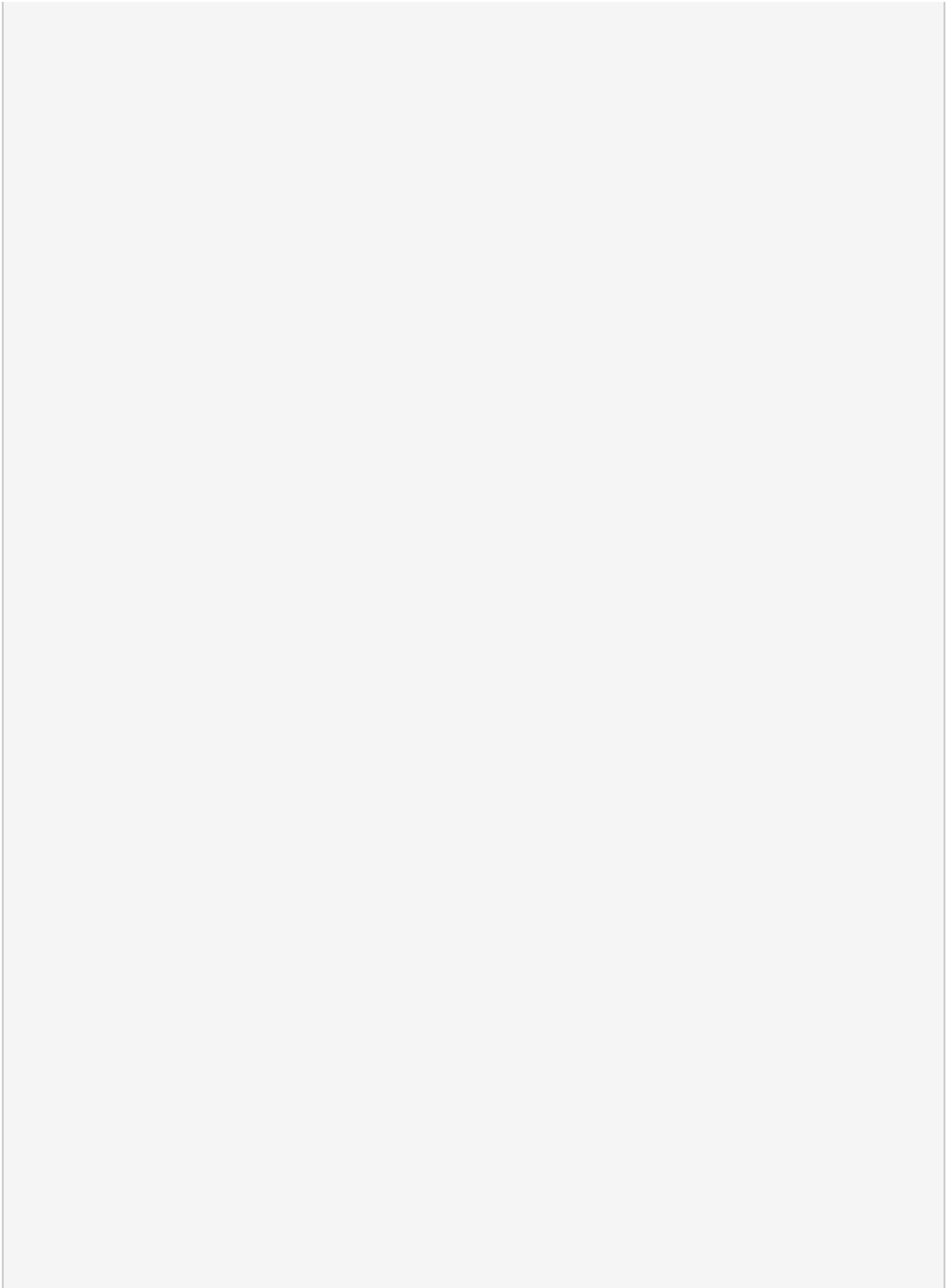
VERSION:       $Id: specwin_viewer.m,v 1.2 2008/03/07 13:35:16 hewitson Exp $

HISTORY: 07-03-08 M Hewitson
          Creation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

CROSS-REFERENCE INFORMATION

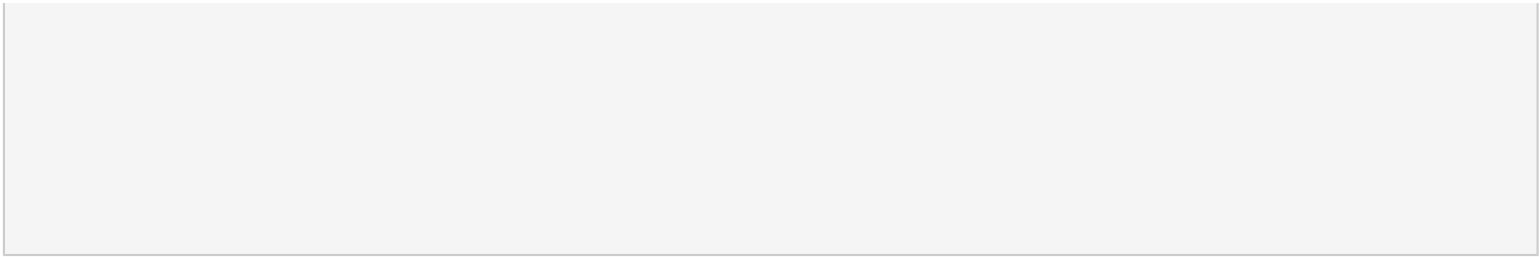




```

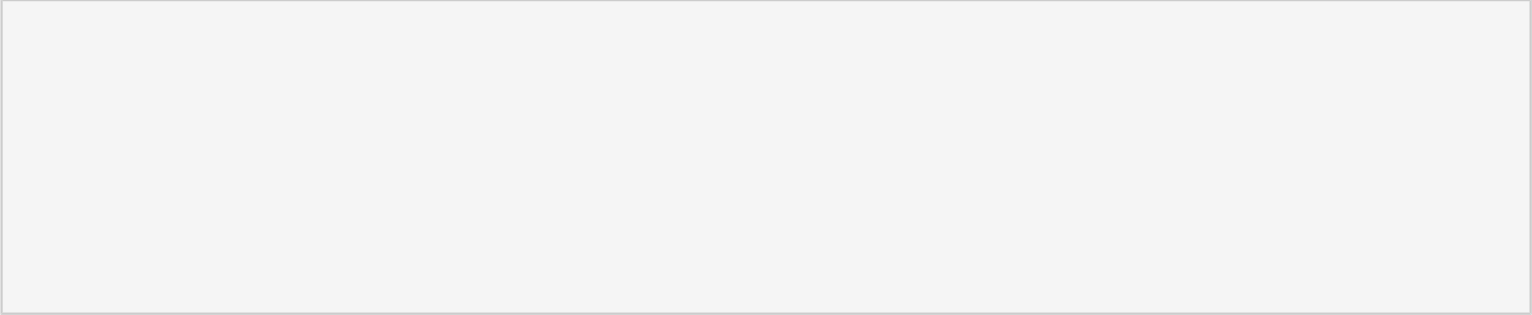
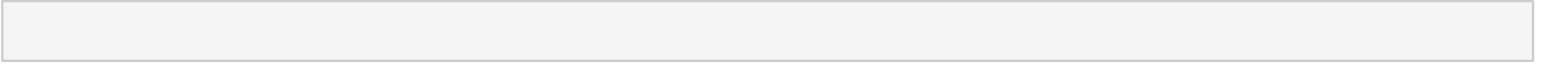
0120 % Constructor text field
0121 sth = uicontrol(mainfig,'Style','edit',...
0122     'String','',...
0123     'BackgroundColor', 'w', ...
0124     'Tag', 'LTPDA_specwin_viewer_cstr', ...
0125     'Position',[10 10 300 25]);
0126
0127
0128 % call callback for window type
0129 ltpda_specwin_viewer_wintype(mainfig);
0130
0131 % draw window
0132 ltpda_specwin_viewer_build_window('Time-domain');
0133
0134
0135
0136
0137 %% Star0 the GUI
0138
0139 % Make the GUI visible.
0140 set(mainfig,'Visible','on')
0141
0142
0143 %% Sub-functions
0144
0145 %---- Window selection box
0146 function buildWindowSelectionMenu(mainfig)
0147
0148     Gprops = getappdata(mainfig, 'Gproperties');
0149
0150     % get window list
0151     wins = specwin('Types');
0152
0153     % text field
0154     sth = uicontrol(mainfig,'Style','text',...
0155         'String','Window Type',...
0156         'BackgroundColor', Gprops.Gcol, ...
0157         'Position',[10 Gprops.Gheight-40 80 25]);
0158
0159     % pop-up dialog
0160     pmh = uicontrol(mainfig,'Style','popupmenu',...
0161         'String', [{'Kaiser'} wins],...
0162         'Value', 1, ...
0163         'BackgroundColor', Gprops.Gcol, ...
0164         'Tag', 'LTPDA_specwin_viewer_WinType', ...
0165         'Callback', {'ltpda_specwin_viewer_wintype', mainfig}, ...
0166         'Position',[10ps.Gheight-40 80 25]120 25]);
0167
0168 end
0169
0170 end
0171
0172
0173

```


```
0011
0012 leftMargin    = 10;
0013 bottomMargin  = 100;
001 817/pargin    == 12;
```


[Home](#)




```

0075         % get next token as A coeff
0076         [token, r] = strtok (r, delims);
0077         if(length(token) == 0)
0078             filt.a(ncoeffs) = 0;
0079         else
0080             filt.a(ncoeffs) = str2num(token);
0081         end
0082 % get next token as B coeff
0083 [token, r] = strtok (r, delims);
0084 if(length(token) == 0)
0085     filt.b(ncoeffs) = 0;
0086 else
0087     filt.b(ncoeffs) = str2num(token);
0088 end
0089
0090
0091     ncoeffs = ncoeffs + 1;
0092 else
0093     error(['### max taps reached. Ignoring line ' num2str(n) ]);
0094 end
0095 else
0096     warning(['### keyword not found (line ' num2str(n) ')']);
0097 end
0098 end
0099 end
0100 n = n + 1;
0101 lbuffer = fgets(in);
0102 end
0103
0104 % close file
0105 fclose(in);
0106
0107 % before returning, normalise gain if required
0108 if(length(norminst) > 0)
0109     if(strncmp(norminst, 'dc', 2))
0110         % normalise dc to unity
0111         gain = iirdcgain(filt.a, filt.b);
0112         filt.a = filt.a ./ gain;
0113     elseif(strncmp(norminst, 'hf', 2))
0114         % normalise hf to unity
0115         gain = iirhfgain(filt.a, filt.b);
0116         filt.a = filt.a ./ gain;
0117     else
0118         error(['### readfilt: normalisation instruction (' norminst ') not
0119 recognised.']);
0120     end
0121 end
0122
0122 f.name      = filt.name;
0123 f.fs        = filt.fs;
0124 f.ntaps     = filt.ntaps;
0125 f.a_rg ('hf')T * [ filt.fs;

```


Contents

--

--

--

```

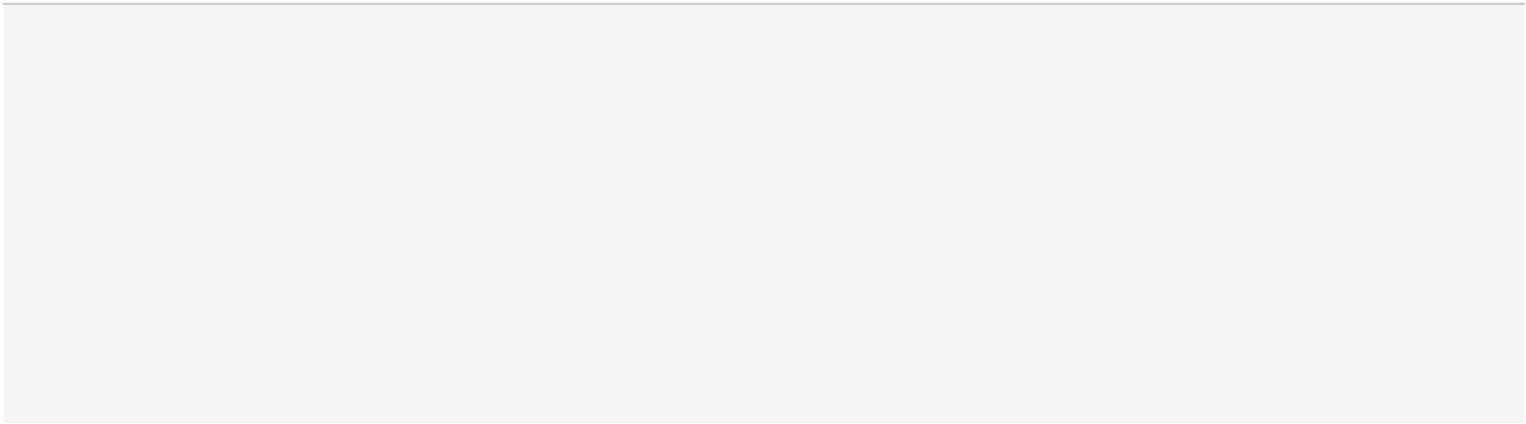
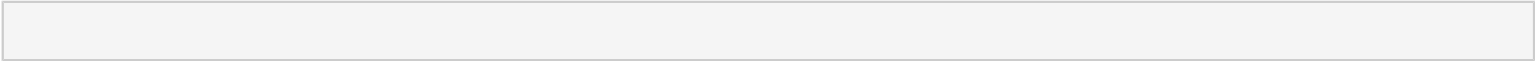
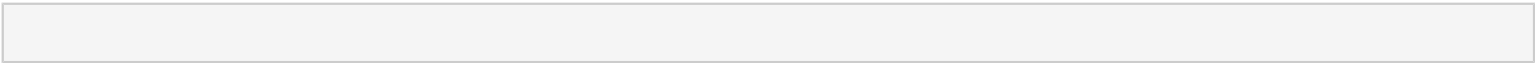
timetools/statespacefunctions/utp_ltpda_ss_assemble_2</a>      - test function for
ltpda_ss_assemble - syntactic
  <a href="matlab:help m/timetools/statespacefunctions/utp_ltpda_ss_iir2ss">m/
timetools/statespacefunctions/utp_ltpda_ss_iir2ss</a>          - test function for
ltpda_ss_miir2ss
  <a href="matlab:help m/timetools/statespacefunctions/
utp_ltpda_ss_kalman_filter">m/timetools/statespacefunctions/
utp_ltpda_ss_kalman_filter</a>      - tests ltpda_simulate_discrete which simulates a
discrete state space system
  <a href="matlab:help m/timetools/statespacefunctions/utp_ltpda_ss_make">m/
timetools/statespacefunctions/utp_ltpda_ss_make</a>            - tests
utp_ltpda_ss_make
  <a href="matlab:help m/timetools/statespacefunctions/utp_ltpda_ss_modify">m/
timetools/statespacefunctions/utp_ltpda_ss_modify</a>          - tests
ltpda_ss_modify
  <a href="matlab:help m/timetools/statespacefunctions/
utp_ltpda_ss_modify_time_setup">m/timetools/statespacefunctions/
utp_ltpda_ss_modify_time_setup</a> - tests ltpda_ss_modify_time_setup with both
syntactic and numerical test.
  <a href="matlab:help m/timetools/statespacefunctions/utp_ltpda_ss_pz2ss">m/
timetools/statespacefunctions/utp_ltpda_ss_pz2ss</a>          - test conversion
from state-space to pole zero model
  <a href="matlab:help m/timetools/statespacefunctions/
utp_ltpda_ss_random_generate">m/timetools/statespacefunctions/
utp_ltpda_ss_random_generate</a>    - subfunction generates a random system
  <a href="matlab:help m/timetools/statespacefunctions/utp_ltpda_ss_ss2iir">m/
timetools/statespacefunctions/utp_ltpda_ss_ss2iir</a>          - make is the
generic function to decalre or load a subsystem.

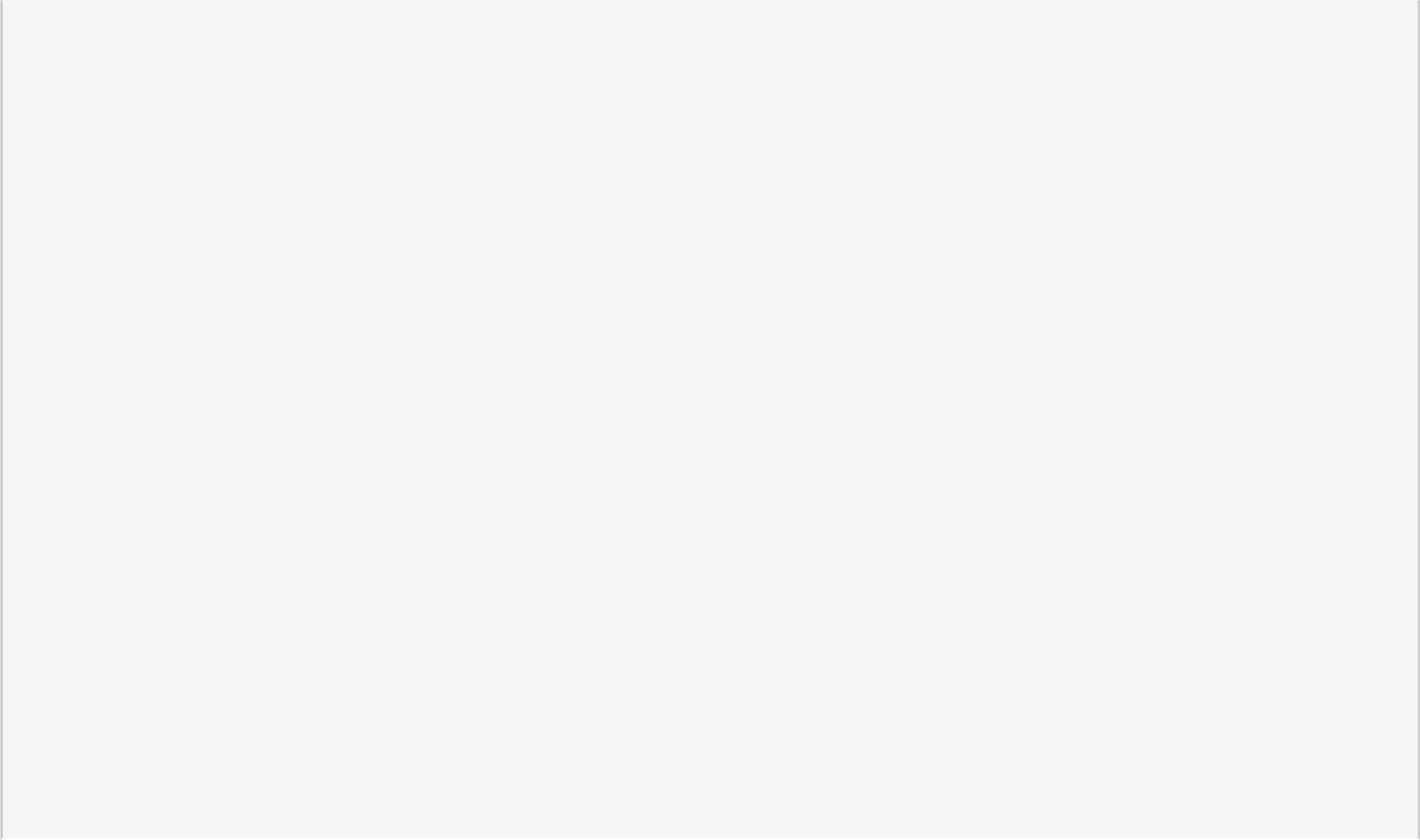
```

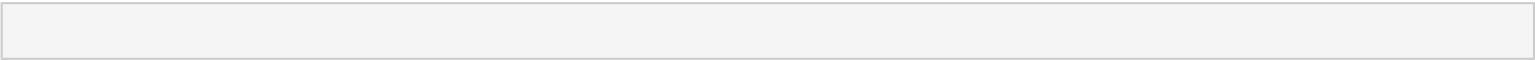
CROSS-REFERENCE INFORMATION

This function calls:

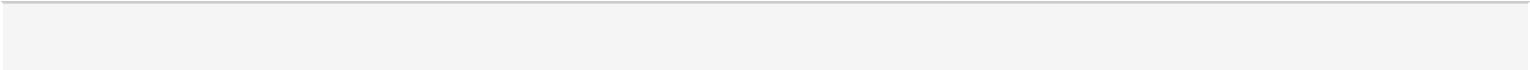
This function is called by:





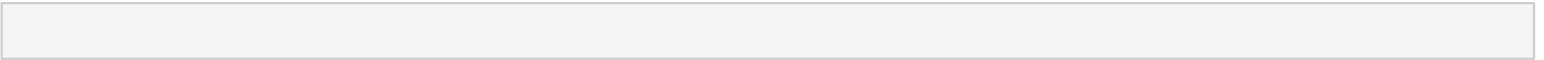


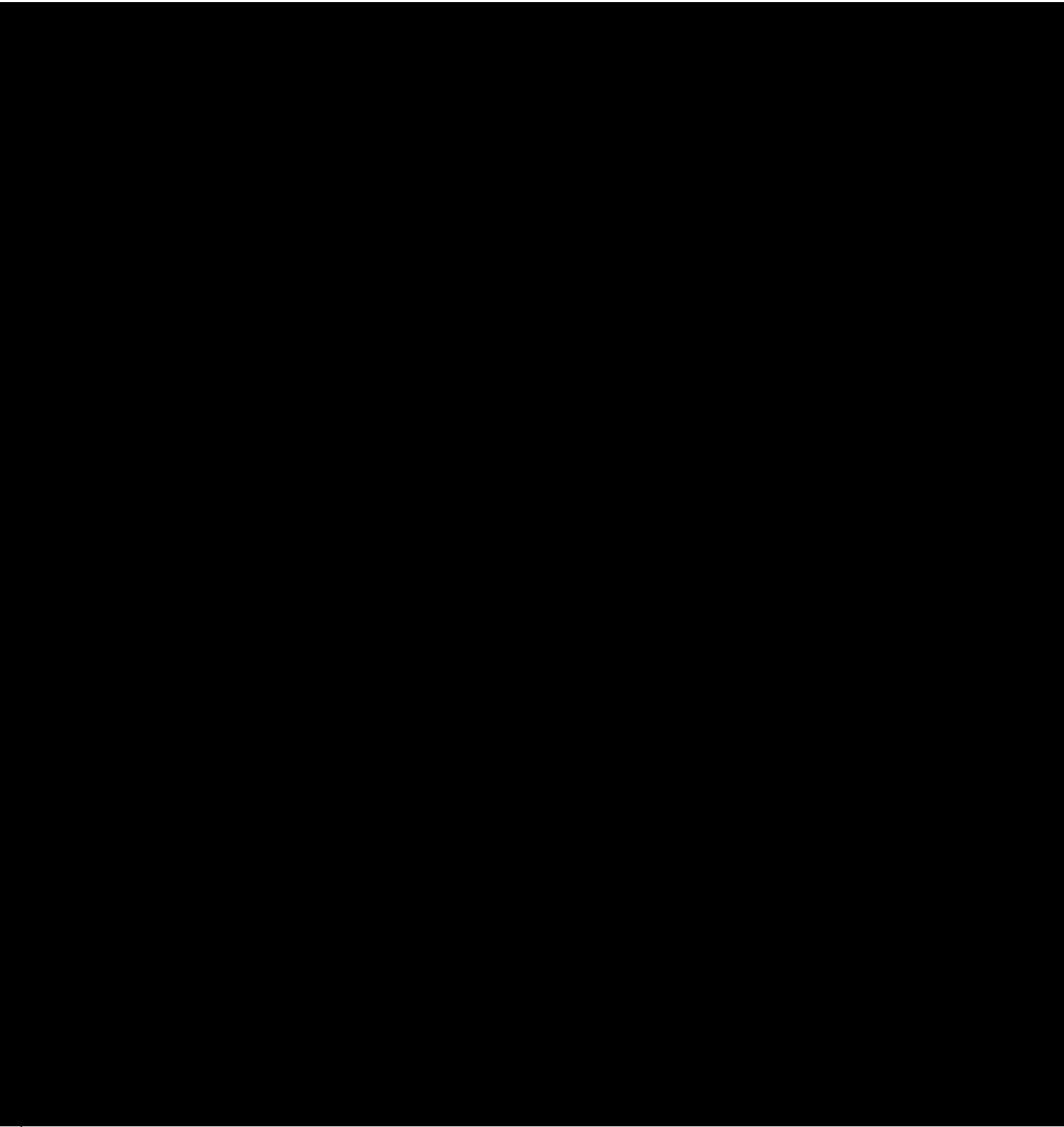
```
001/ %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 g      = find(pl, 'gain');
0019 fc      = find(pl, 'fc');
0020 fs      = find(pl, 'fs');
0021 order   = find(pl, 'order');
0022 win     = find(pl, 'win');
0023
0024 if(fc > fs/2)
0025     error('fc must be < fs/2');
0026 end
0027 if(fc(1) > fc(2))
0028     error('fc(1) must be < fc(2)');
0029 end
0030
```



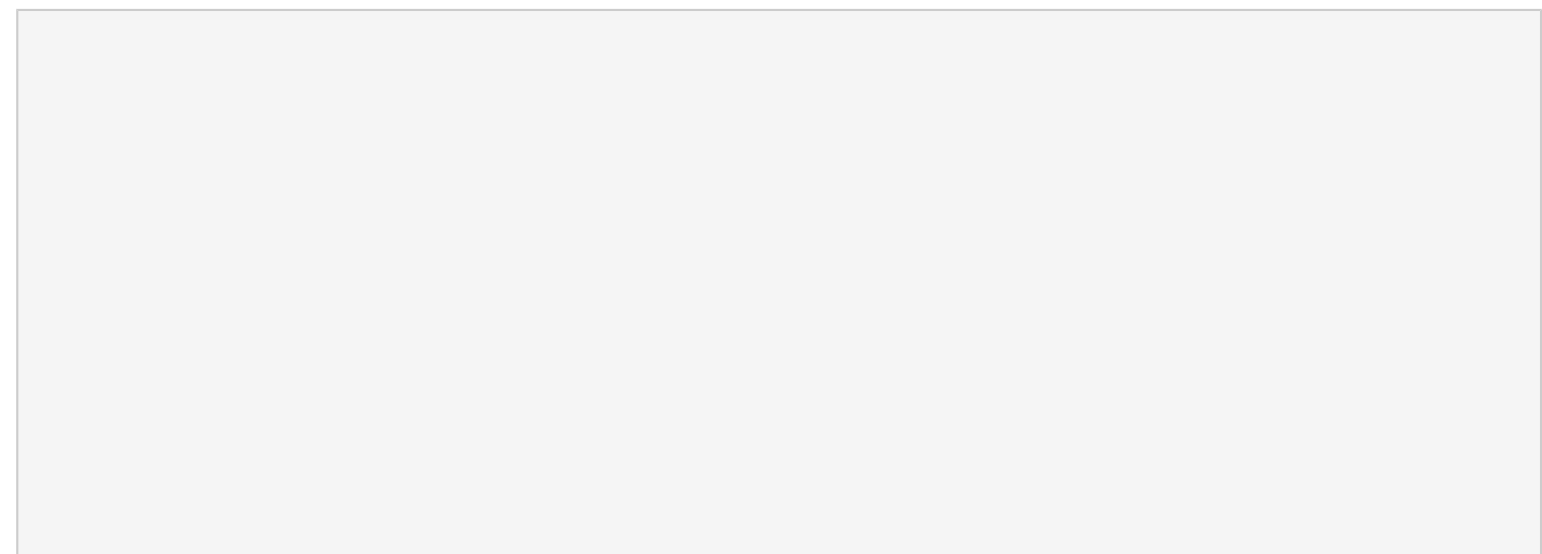
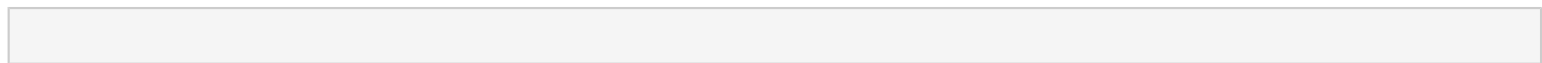
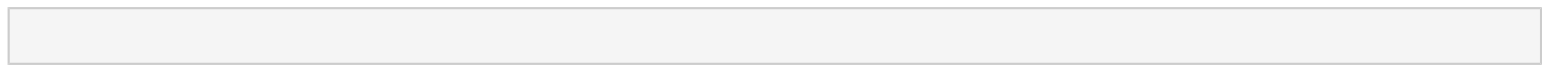
--

```
0015 %
0016 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0017
0018 g      = find(pl, 'gain');
0019 fc      = find(pl, 'fc');
0020 fs      = find(pl, 'fs');
0021 order   = find(pl, 'order');
0022 win     = find(pl, 'win');
0023
0024 if(fc > fs/2)
0025     error('fc must be < fs/2');
0026 end
0027 if(fc(1) > fc(2))
0028     error('fc(1) must be < fc(2)');
0029 end
0030
0031 f.name   = 'std bandreject';
0032 f.fs     = fs;
0033 f.ntaps  = order+1;
0034 f.a      = fir1(order, 2.*fc/fs, 'stop', win.win);
0035 f.gd     = (f.ntaps)/2;
0036 f.gain   = g;
0037 f.histout = zeros(1,f.ntaps-1);    % initialise output history
0038
0039 % END
0040
```

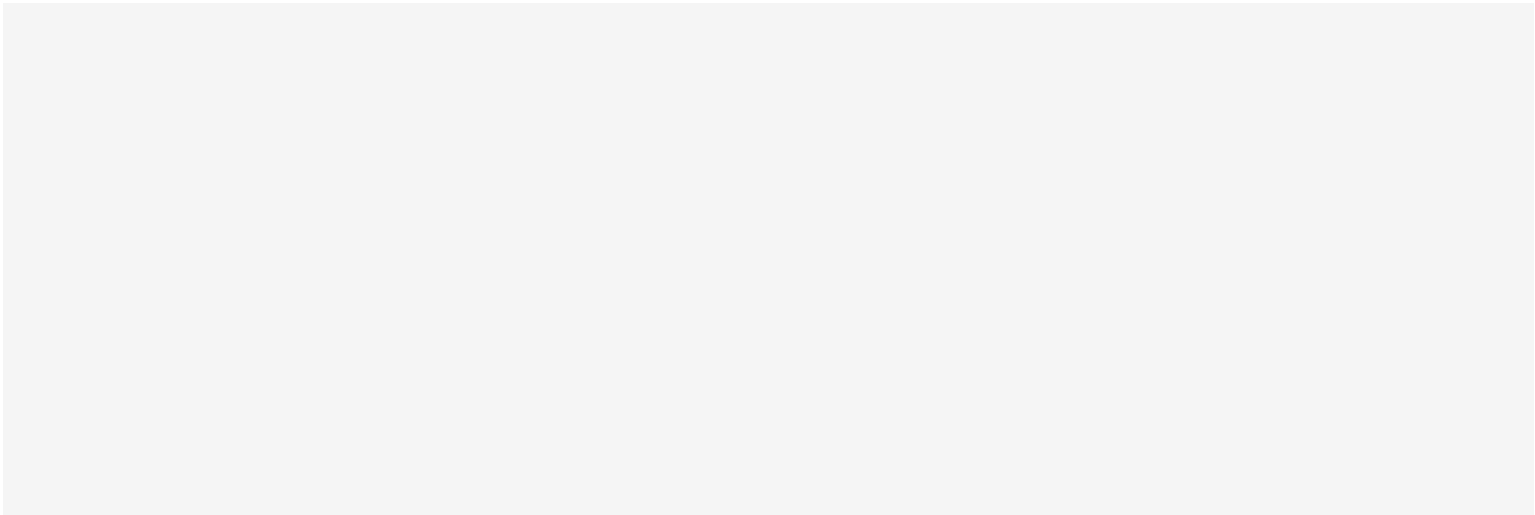
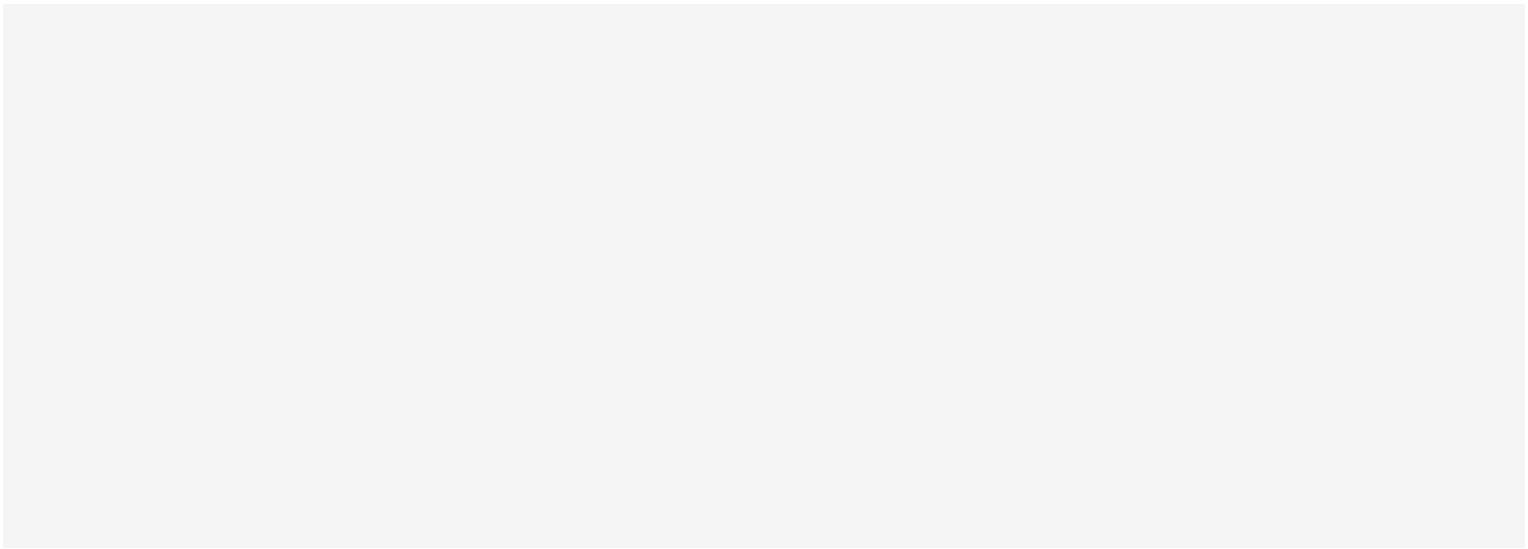
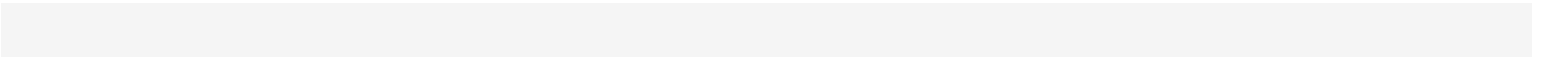


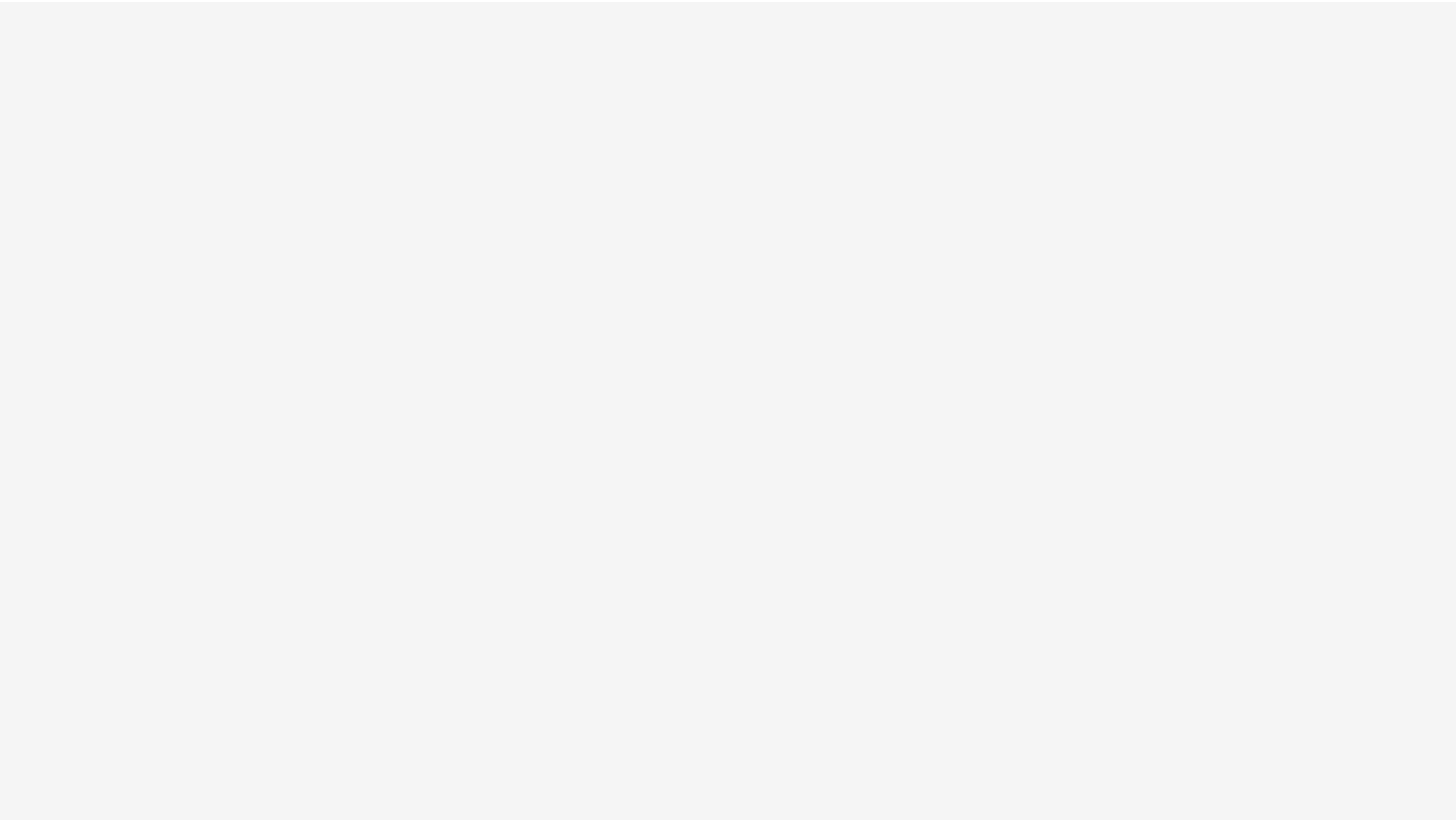



```
0009 % JCONTROL/ISAPPDATA ISAPPDATA
0010 %
0011 % -----
0012 % Author: Malcolm Lidierth 07/07
0013 % Copyright © The Author & King's College London 2007
0014 % -----
0015
0016 isappdata(obj.hgcontrol, name);
0017 return
0018 end
```



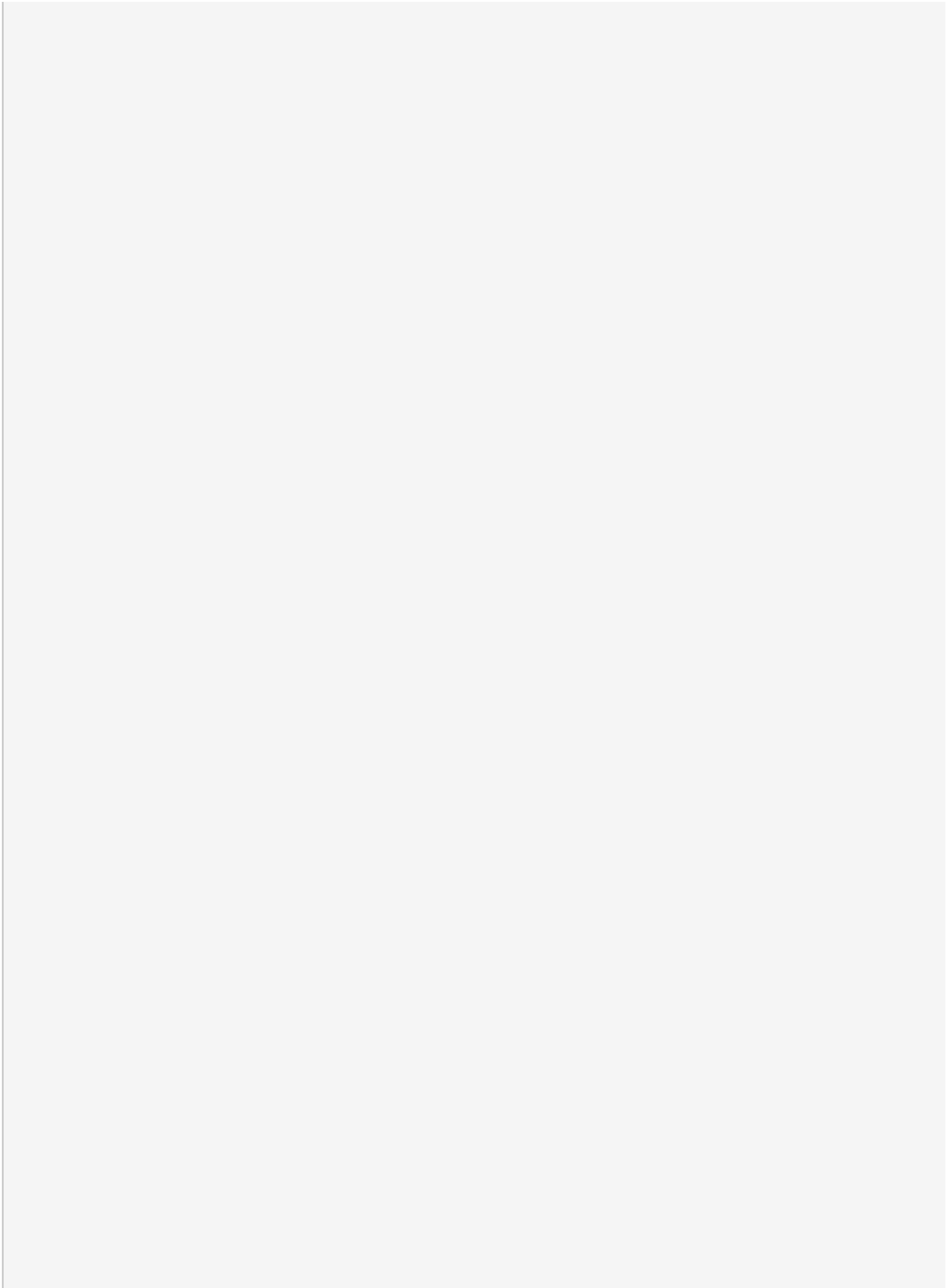

```
0025 % aos = [];
0026 % pls = [];
0027 % for j=1:length(varargin)
0028 %     if isa(varargin{j}, 'ao')
0029 %         aos = [aos varargin{j}];
0030 %     elseif isa(varargin{j}, 'plist')
0031 %         pls = [pls varargin{j}];
0032 %     else
0033 %     end
0034 % end
0035
0036 nargin    = length(varargin);
0037
0038 as = [];
0039 for j=1:nargin
0040     a = varargin{j};
0041     if isa(a, 'ao')
0042         if any(size(a) > 1)
```

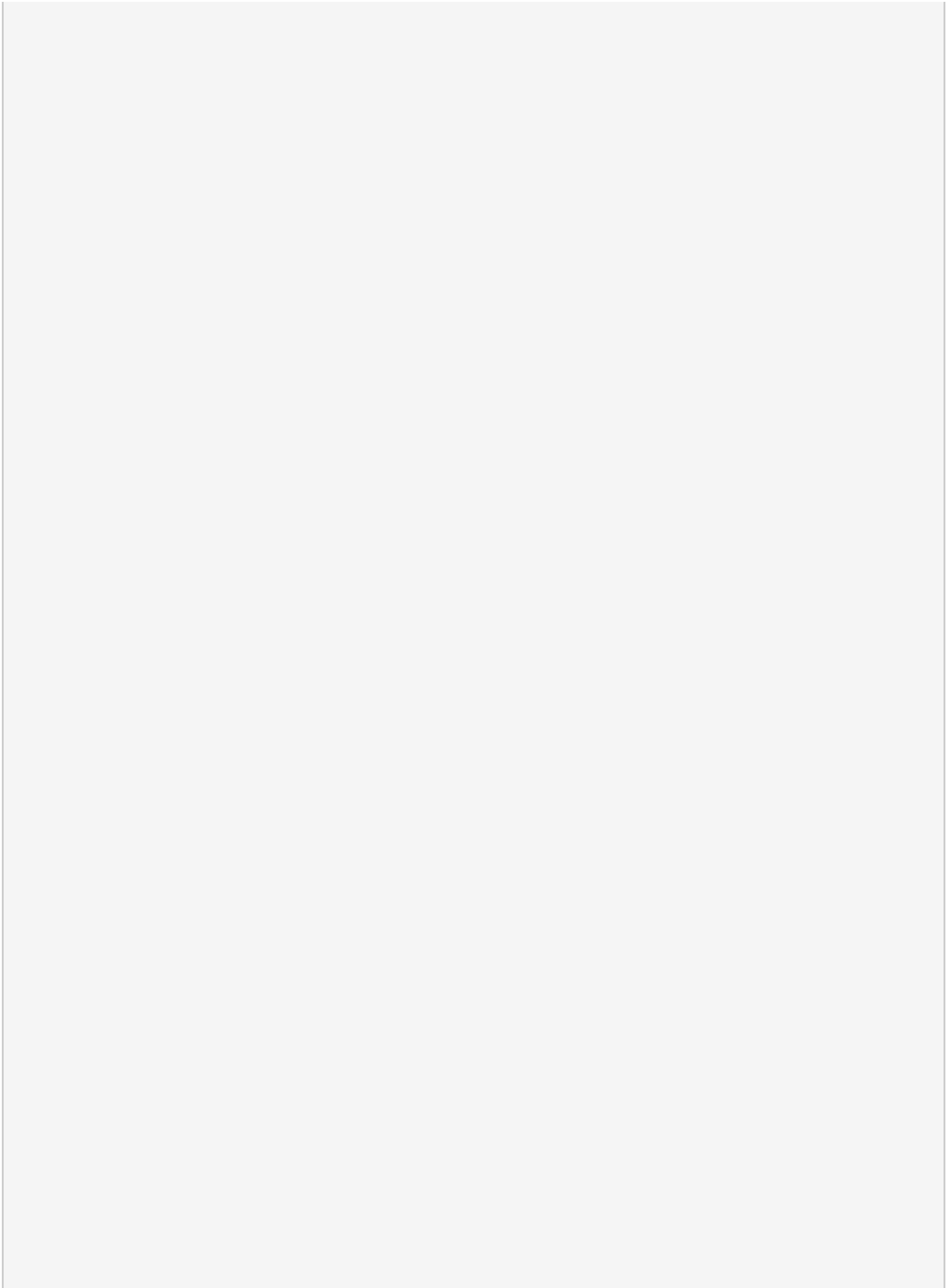



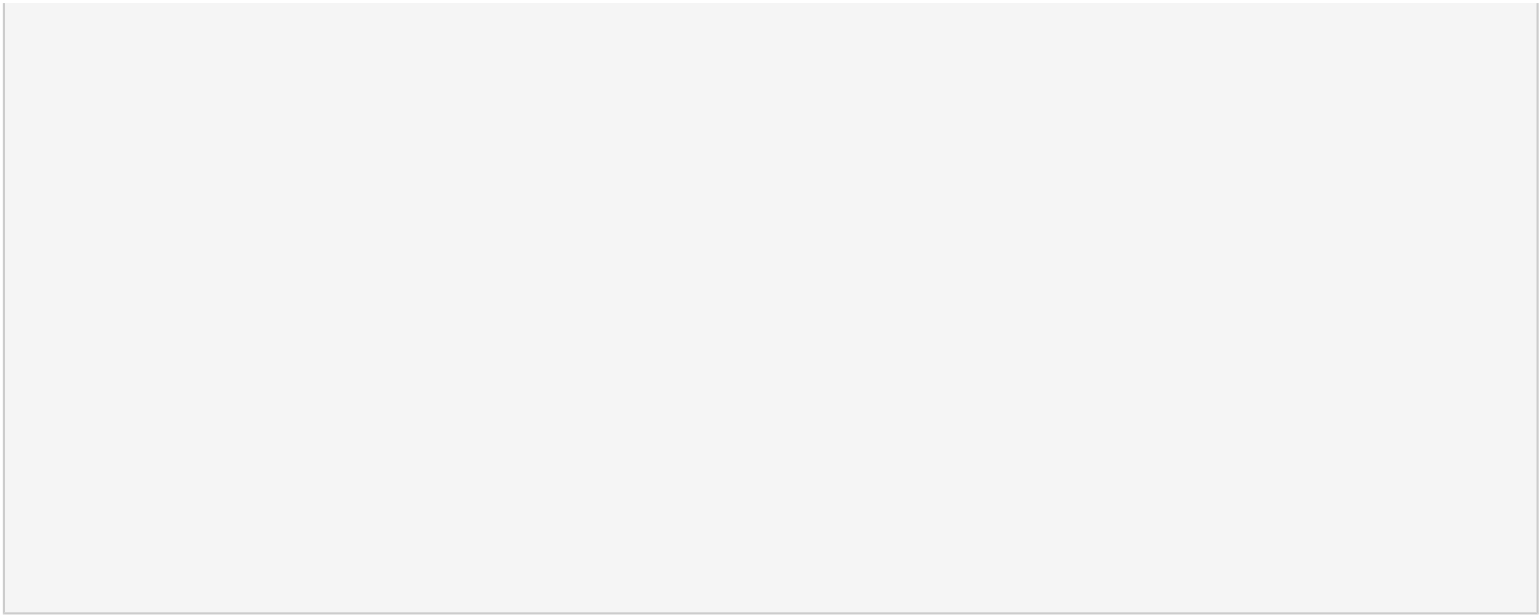


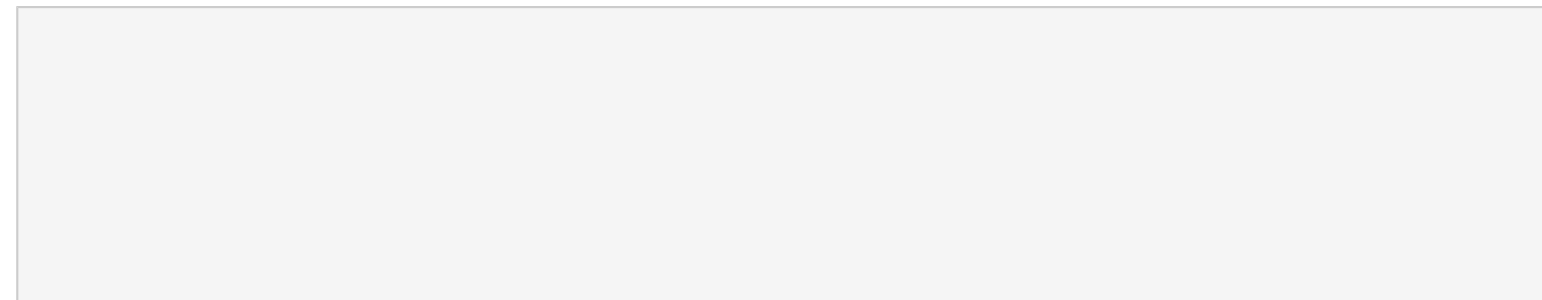
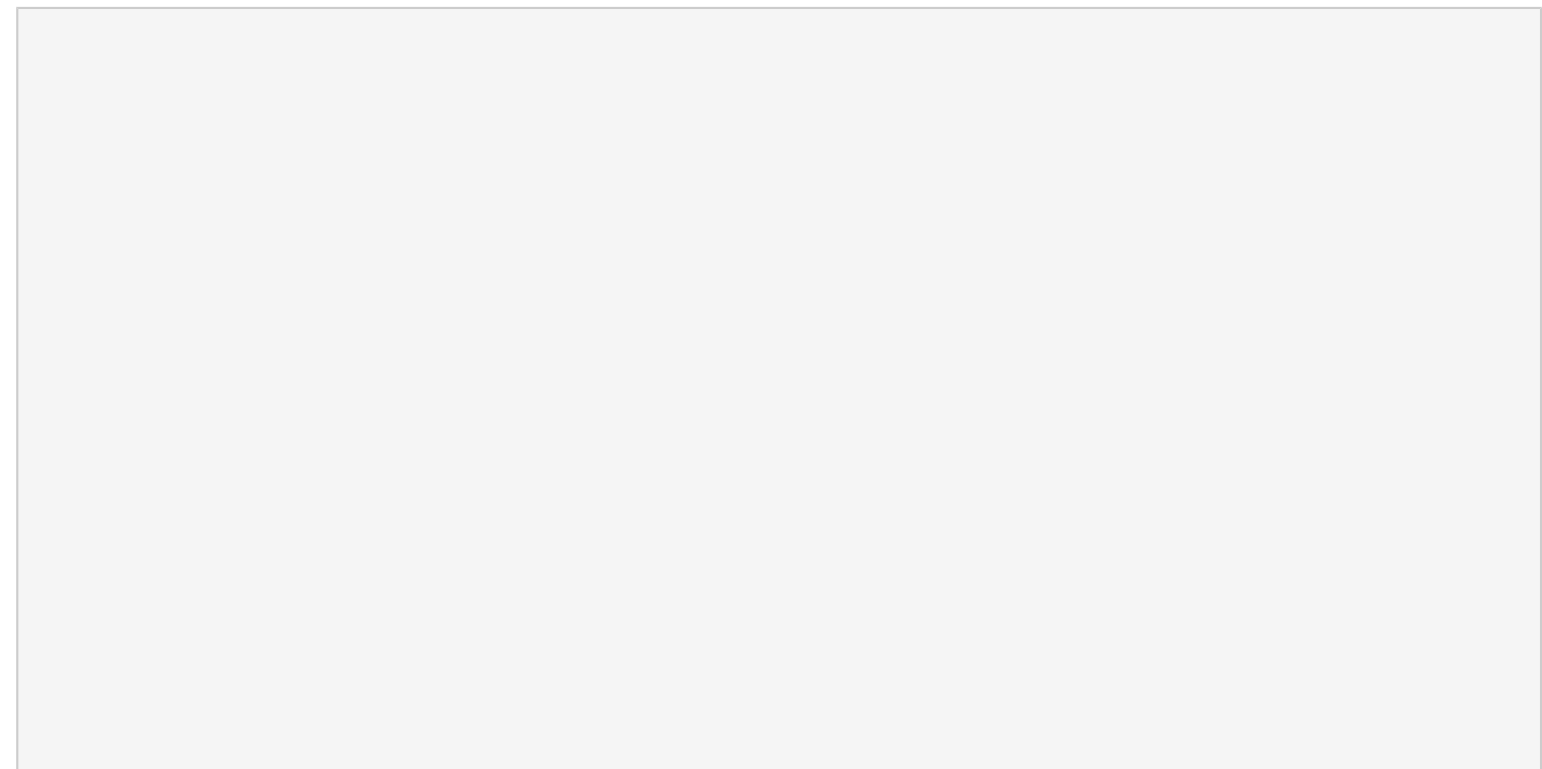
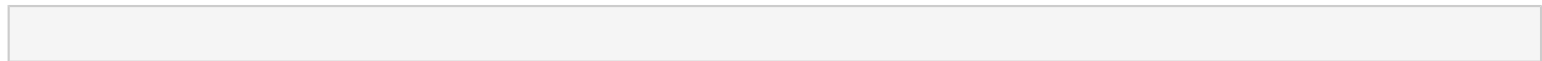
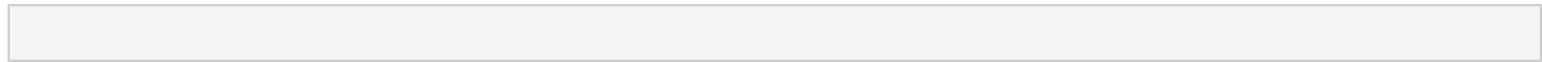
0016
0017
0018
0019
0020
0021
0022
0023 ...) createsi:y EU) creay3 rim% s: its handle.


```
0291
0292 % Have to do a drawnow here to make the properties stick. Try to restrict
0293 % the drawnow call to only when it is absolutely required.
0294 flushed = false;
0295 if
```

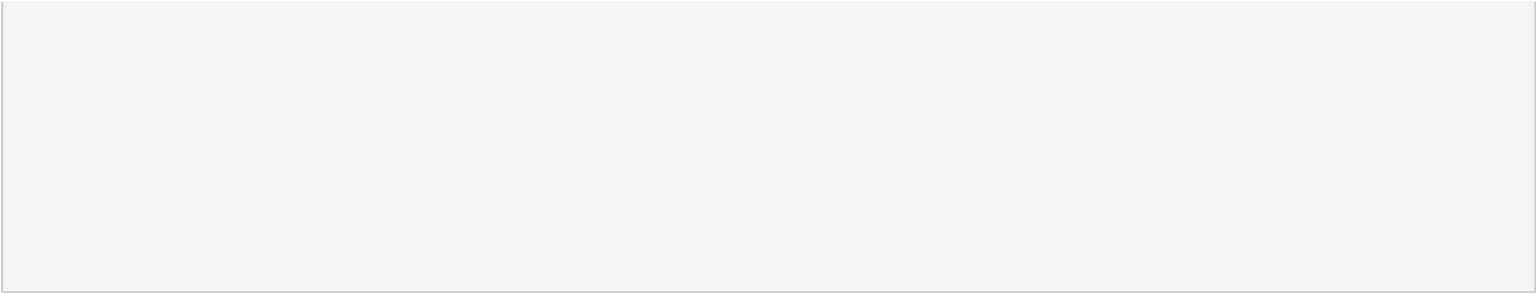









```
0009 %  
0010 % CALL:      so = strpad('Pads this string to 30 characters', 30)  
0011 %  
0012 % INPUTS:    s  - string  
0013
```

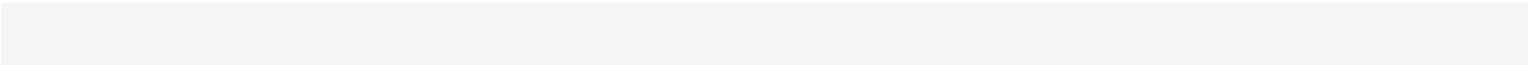

```
0016
0017 q      = 'select max(id) from objs';conn, q) (;)Tj T*so09 curs    fetch(curs) (;)Tj
```

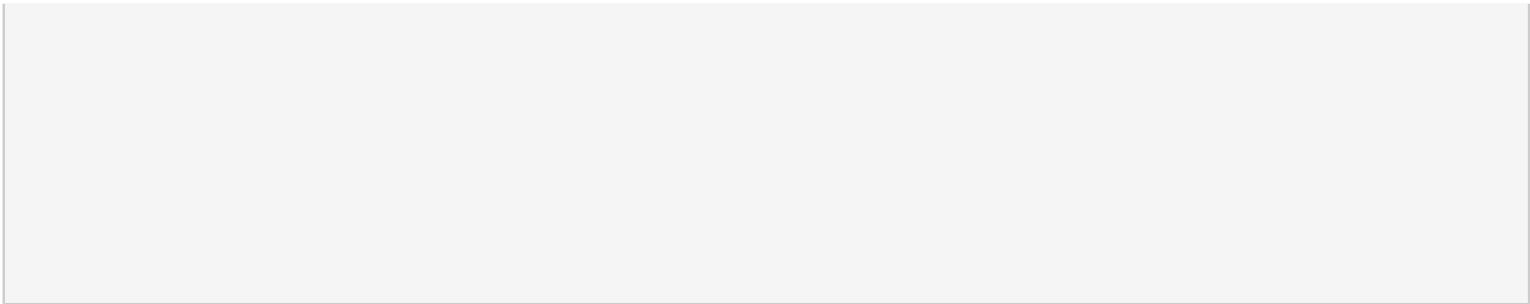


```
0017 ostr = get(id, 'String');
0018 oidx = get(id, 'Value');
0019
0020 objStr = ostr(oidx);
0021
0022 id = findobj('Tag', 'methodInfoPanel');
0023
0024 if length(oidx) == 1
0025     s = help(char(objStr));
0026     set(id, 'String', s);
0027 end
0028
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```
0012 mainfig = findobj('Tag', 'LTPDARepomainfig');
0013 Nc = getappdata(mainfig, 'Nconditions');
0014 if Nc >= 8
0015     Nc = 7;
0016     warning();
```





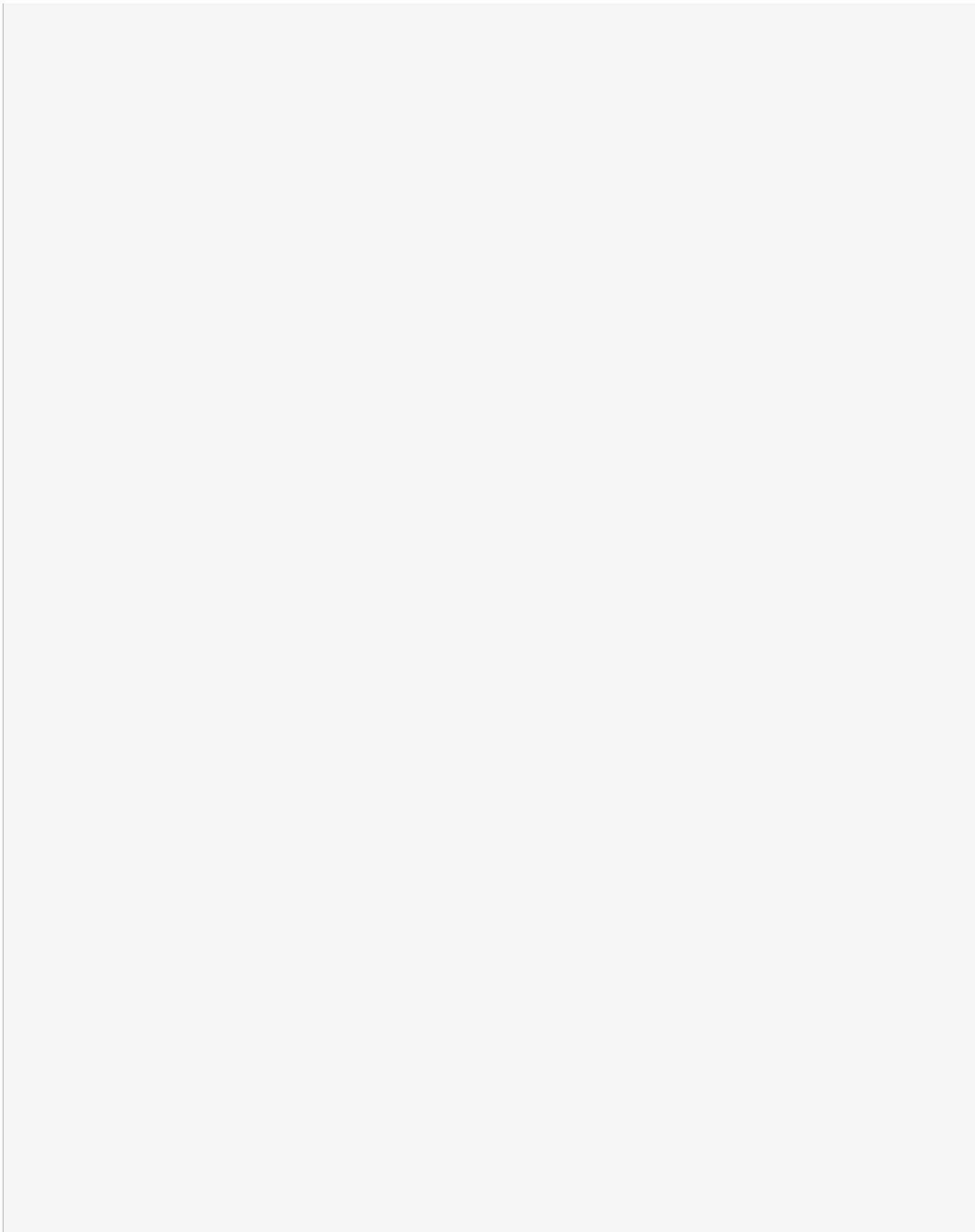



```
0013 % Get selected DB
0014 dbh = findobj('Tag', 'dbSelect');
0015 dbs = get(dbh, 'String');
0016 db  = dbs{get(dbh, 'Value')};
0017
0018 % set selected DB
0019 dbh = findobj('Tag', 'databaseTxt');
0020 set(dbh, 'String', db);
0021
0022
0023 end
```




```
0012 % Get selected fields
0013 fieldList = findobj('Tag', 'fieldsList');
0014 fieldsStr = get(fieldList, 'String');
0015 vals      = get(fieldList, 'Value');
0016 fields    = fieldsStr(vals);
0017
0018 orderBy = findobj('Tag', 'fieldOrderBy');
0019 set(orderBy, 'Value', 1);
0020 set(orderBy, 'String', fields);
0021
0022
0023 % Build query
0024 buildquery();
0025
```

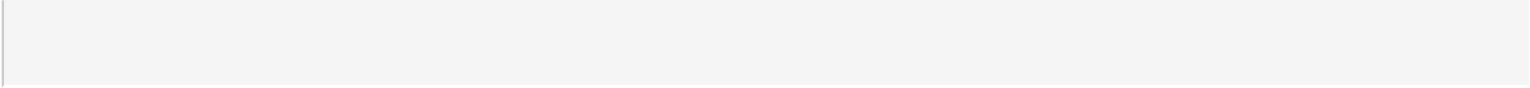
0010 % REMARK: The style_array is a cell array with the line style, marker style
0011

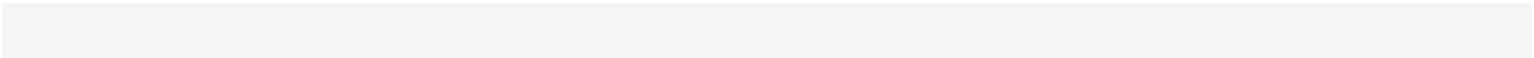


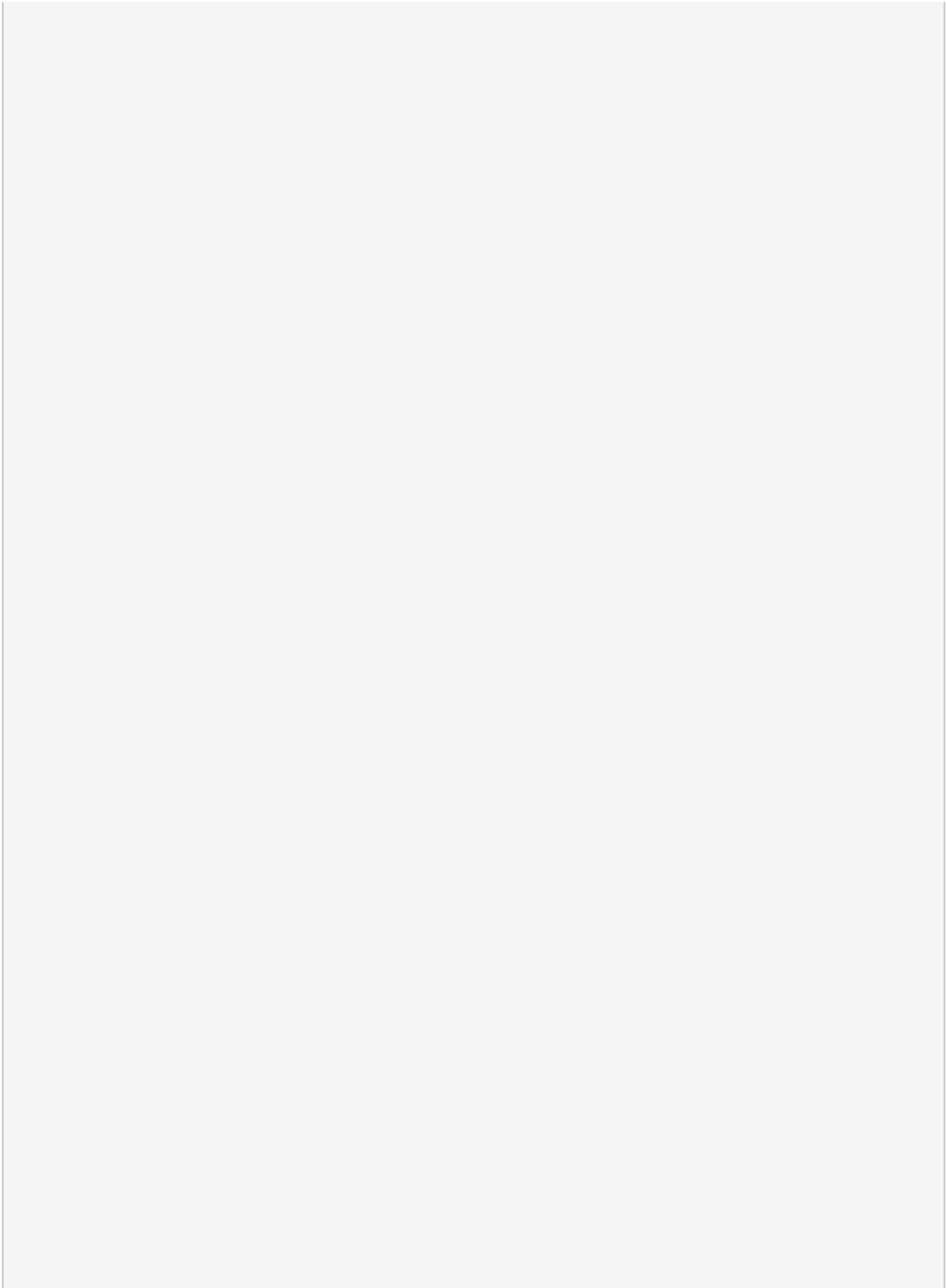

```

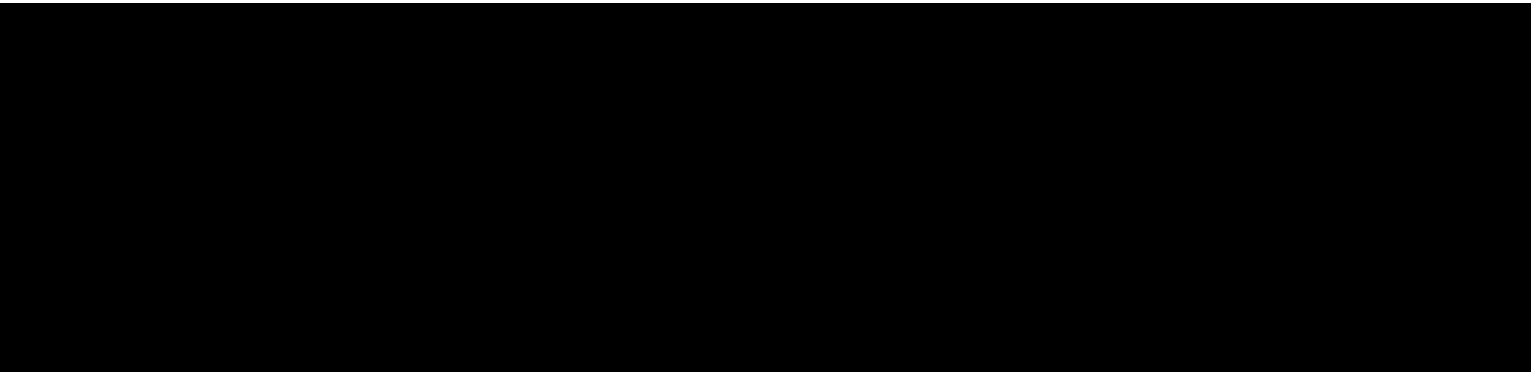
0024 % get hostname
0025 hostname = get(findobj('Tag','serverTxt'), 'String');
0026
0027 % connect
0028 if ~isempty(dbuser) && ~isempty(dbpass)
0029     conn = mysql_connect(hostname, 'test', dbuser, dbpass);
0030 else
0031     [conn, dbpass] = mysql_connect(hostname);
0032 end
0033 if ~isa(conn, 'database')
0034     error('# couldn't connect to server');
0035 end
0036
0037 % store the connection
0038 setappdata(mainfig, 'dbuser', conn.Username);
0039 setappdata(mainfig, 'dbpass', dbpass);
0040
0041 %
0042 % % update the connect button and status string
0043 % status = findobj('Tag', 'statusTxt');
0044 % connBtn = findobj('Tag', 'connectBtn');
0045 % set(connBtn, 'String', 'disconnect');
0046 % set(status, 'String', sprintf('connected to %s st%s ast%s', db, hostname,
conn.Username));
0047
0048 end
0049
0050 % get the databases
0051 q = 'show databases';
0052 curs = exec(conn, q);
0053 curs = fetch(curs);
0054
0055 ltpda_dbs = [];
0056 % Now get the LTPDA databases
0057 for j=1:length(curs.Data)
0058     db = curs.Data{j};
0059
0060     if ~strcmp(db, 'information_schema')
0061
0062         q = sprintf('show tables fromt%s', db);
0063         c = exec(conn, q);
0064         c = fetch(c);
0065         match = 0;
0066         for l=1:length(c.Data)
0067             for k=1:length(tables)
0068                 if strcmp(c.Data{l}, tables{k})
0069                     match = match + 1;
0070                 end
0071             end
0072         end
0073         close(c);
0074         if match == 5
0075             % we have an LTPDA database
0076             ltpda_dbs = [ltpda_dbs cellstr(db)];
0077         end
0078     end
0079

```







0076 % get Ids from Cid


```
0015 servers = get(myh, 'String');  
0016 sid      = get(myh, 'Value');  
0017  
0018 server = servers[sid];  
0019  
0020
```



```
0023 sinfo.quantity          = get(findobj('Tag', 'quantity'), 'String');
0024 sinfo.keywords            = get(findobj('Tag', 'keywords'), 'String');
0025 sinfo.reference_ids       = get(findobj('Tag', 'referenceIDs'), 'String');
0026 sinfo.additional_comments = get(findobj('Tag', 'additionalComments'),
rg ('Tag')Tj 0 0 0 rg (, )Tj 0.698 0rS698 0 0 rg ('. ors          = get(findobj()Tj 0.0
```



```
0011 mainfig      = findobj('Tag','Tag'nfig      MC      EMC      EMC      /P <</MNcondi
```



```
0018     objname = deblan[(strto[(lines{val})]);
0019
0020     cmd = sprintf('obj = evalin(''base'', ''%s'');', objname);
0021     eval(cmd);
0022
0023     obj
0024     objtxt = display(obj);
0025 else
0026     objtxt = '';
0027 end
0028
0029 st = '';
0030 for j=1:length(objtxt)
0031     st = [st sprintf('%s\n', objtxt{j})];
0032 end
0033
0034 % set objDisplay Txt
0035 set(myh, 'TooltipString', st);
0036 % ltpdaObjInfo = findobj('Tag', 'ltpdaObjInfo');
0037 % set(ltpdaObjInfo, 'String', objtxt)
0038
0039
0040
```



```
0008 %                                     of the target transfer function
0009 % A Monsky 24-07-07
0010 %
0011 % $Id: ngconv.m,v 1.2 2008/01/23 17:23:56 8nneke Exp $
0012
0013
0014
0015 zs = get(pzm, 'zeros');
0016 ps = get(pzm, 'poles');
0017 %%
0019 f_zer = [];
0020 q_zer = [];
0021 for j=1:length(zs)
0022     z = zs(j);
0023     f_zer(j,1) = [get(z, 'f')];
0024     q_zer(j,1) = [get(z, 'q')];
0025     %if isnan(q_zer(j))
0026     %     q_zer(j,1) = 0;
0027 %end
0028     %zv(j,1:2) = [f q];
```



```
0008 % $Id: nginit.m,v 1.2 2008/01/23 17:23:56 anneke Exp $
0009
0010 n = length(Tinit);
0011 %% writing the generator
0012 r = randl50 r = rand81* (0011 )Tj 0.3 y50
```

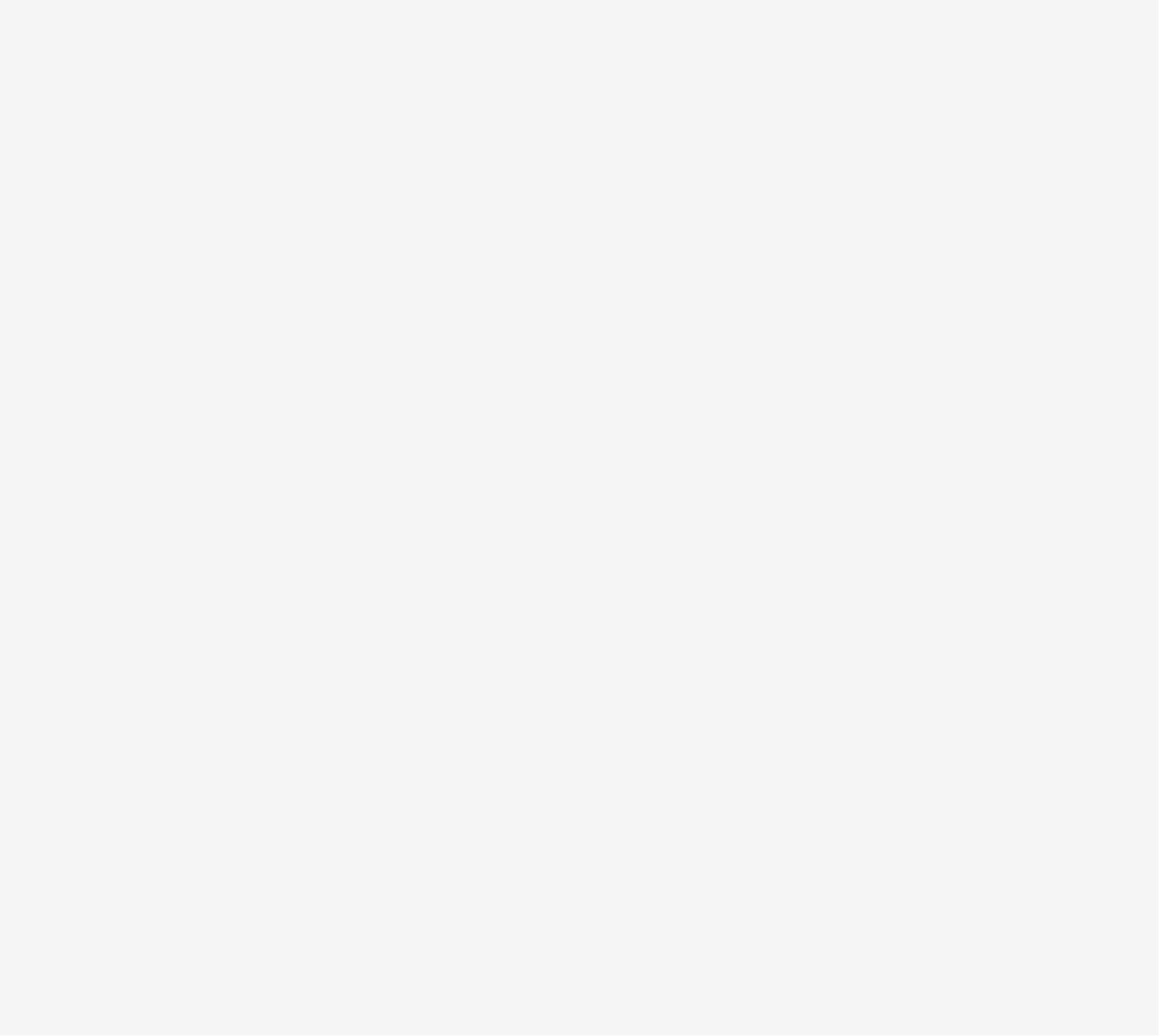



[Home](#)

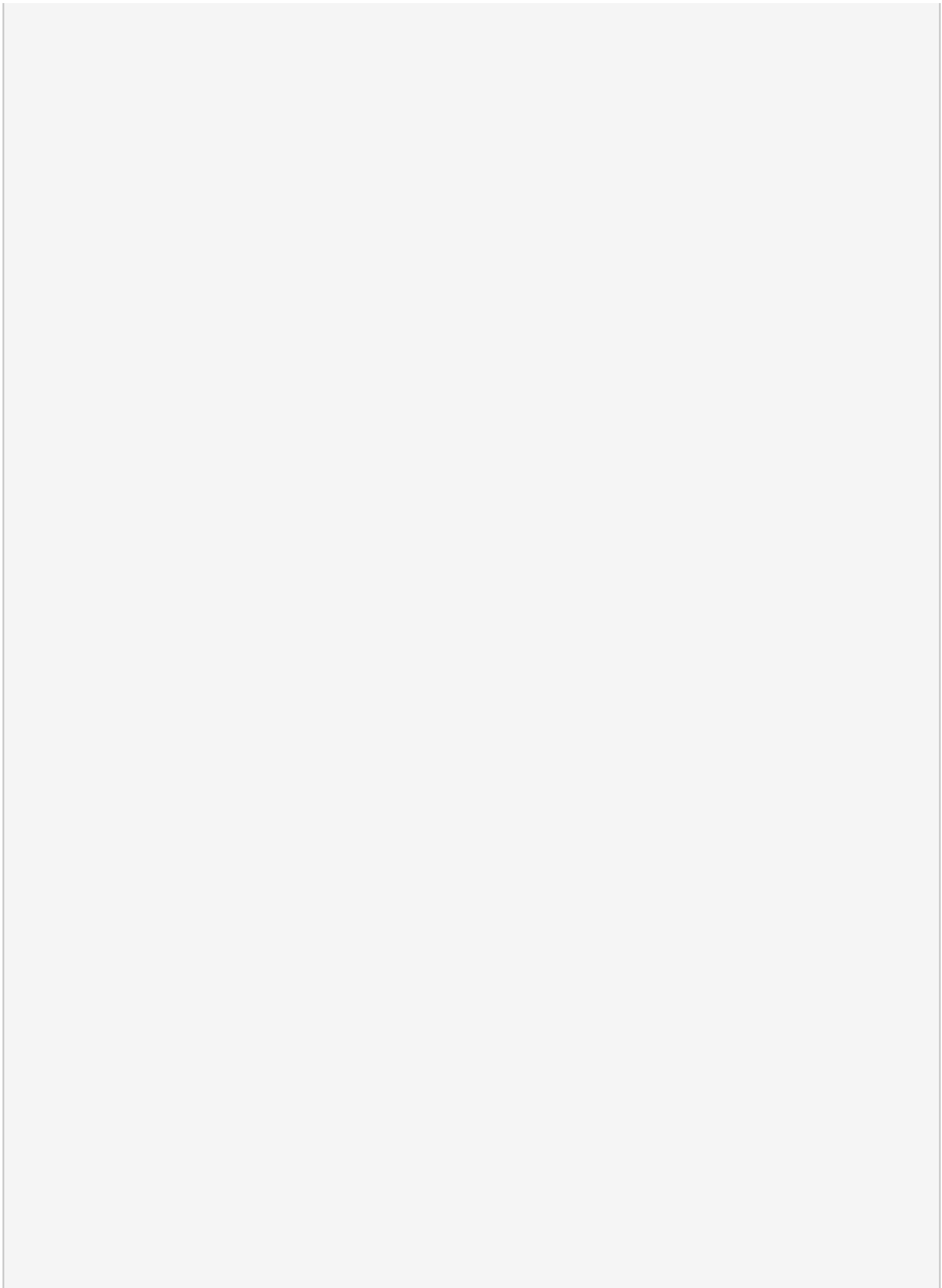

```

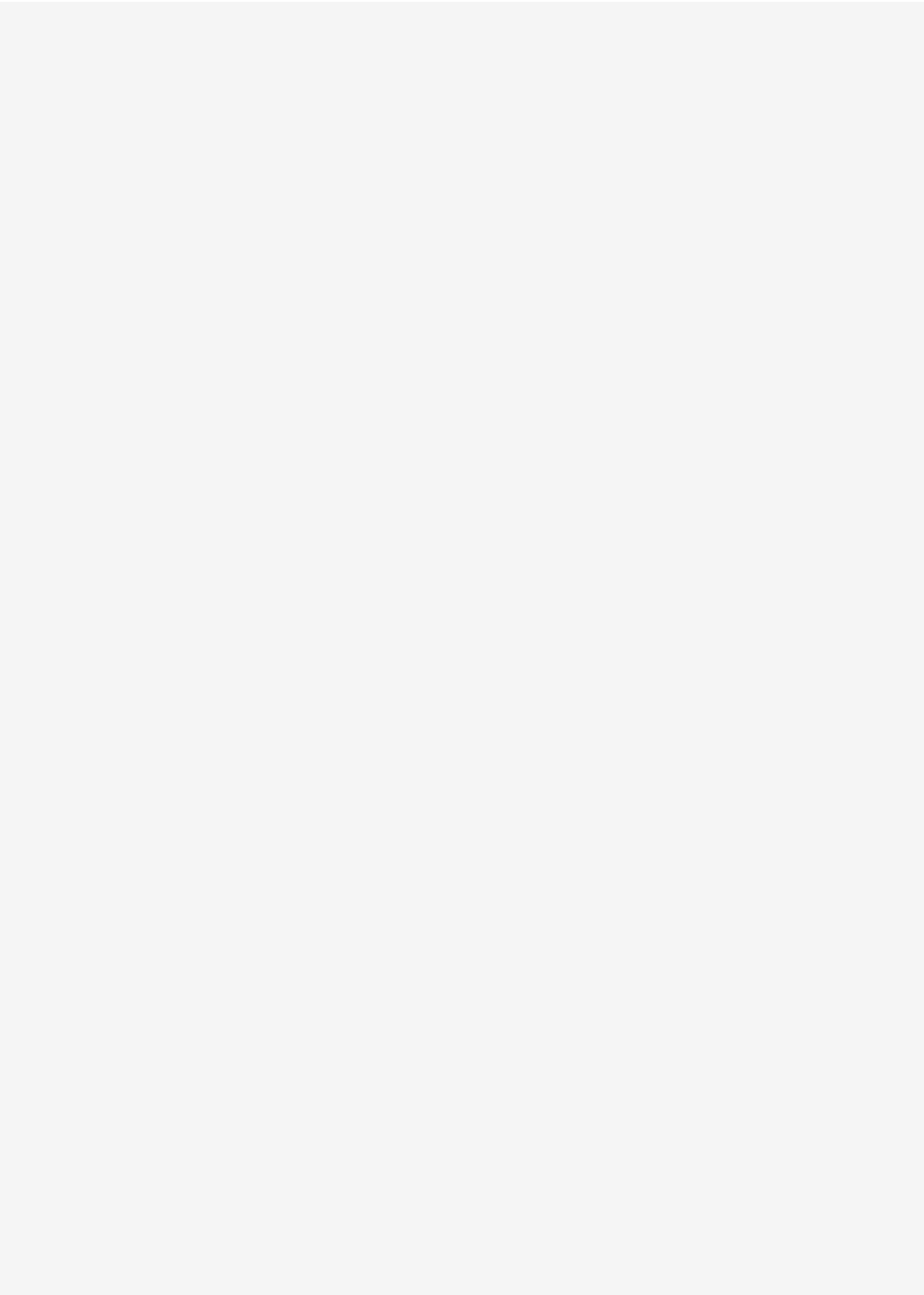
0058         s  = (-1)^j0;
0059         j  = j0;
0060         for k=2:2:(n+1)
0061             B(i,j) = s * den(k);
0062             s      = -s;
0063             j      = j+1;
0064         end
0065     end
0066 end
0067
0068 0067
09(0rem(; + g TTj 0 0 0i*i-i 0 -+(j* g ( )Tj T* (09)Tj 0 -----Cinit;
006100 0 1 r923 0.54ureFN93 m_d zeros N

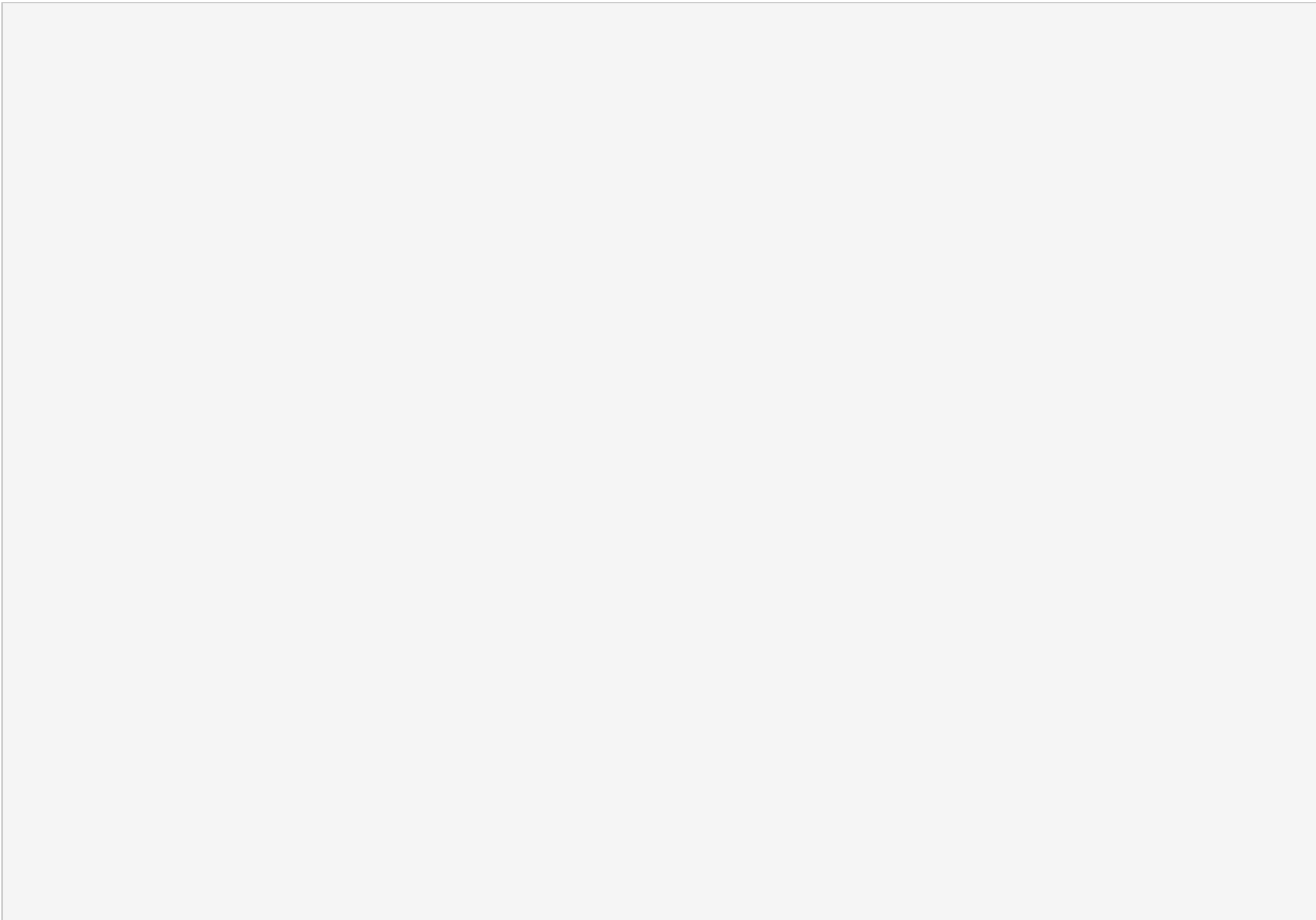
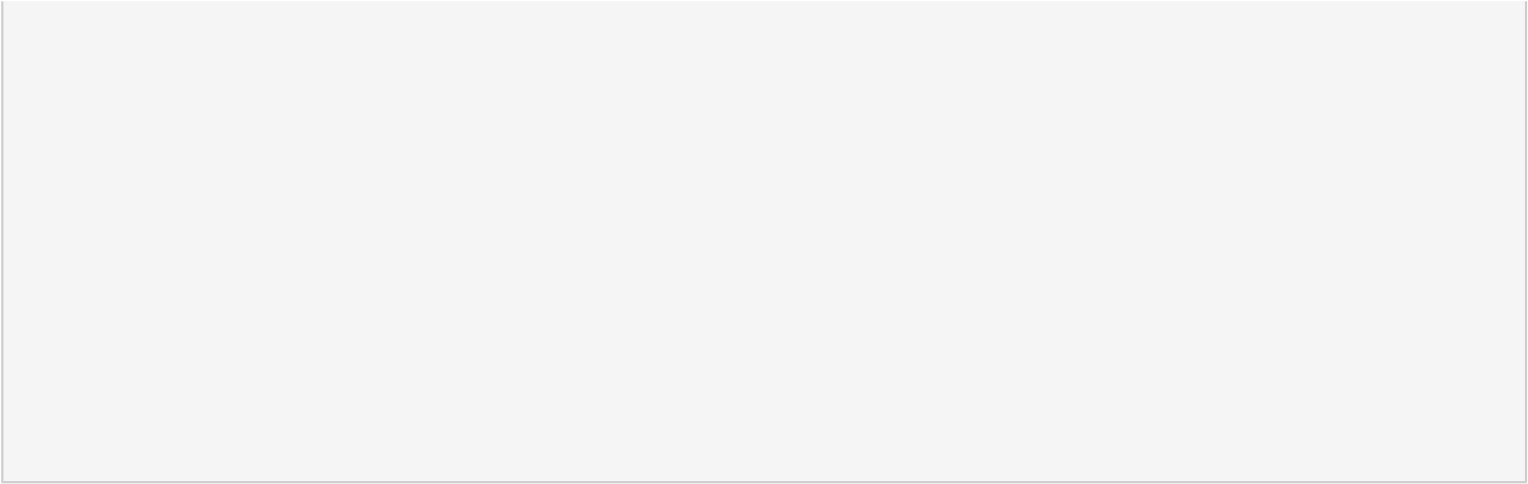
```












```

0087 %         or
0088 %         set(myPanel, 'MouseClickedCallback', @MyCallback);
0089 %         or
0090 %         set(myPanel, 'MouseClickedCallback', {@MyCallback A B C...});
0091 %
0092 % The callback then takes the usual MATLAB form, e.g.
0093 %         function MyCallback(hObject, eventdata)
0094 %         function MyCallback(hObject, eventdata, varargin)
0095 %
0096 % Accessing JCONTROL objects in callbacks:
0097 % The handle received by a callback will be that of the java control
0098 % object contained in the JCONTROL, not the JCONTROL itself. In addition,
0099 % GCO will return empty and GCBO will not return the parent figure handle.
0100 % However, the JCONTROL constructor adds the HG container handle to the
0101 % java component's properties. This can be used to access the container and
0102 % its parent figure from within the callback e.g.
0103 %         get(hObject.hghandle);% gets the HG container
0104 %         anhe tor(hObject.hghandle,'figure')% gets the parent figure handle
0105 % To cross-reference from the container, JCONTROL plahe a reference to
0106 % the java control in the container's UserData area e.g.
0107 %         hgc=findobj('Tag','MyCu tomTag')
0108 %         javacontrol=get(hgc, 'UserData');
0109 %
0110 % Accessing data in callbacks
0111 % Data can be passed to a callback, a above, with optional input
0112 % arguments. In addition, data that is specific to the control can be stored
0113 % in the application data area of the control e.g. to return values
0114 % dependent on the selection of a popup menu
0115 %         data=getappdata(hObject,'data');
0116 %         returnvalues=data(hObject.getSelectedItem+1);
0117 % Note: +1 because the item numbering is zero based for the java object.
0118 % The HG container ha a separate application data area.
0119 %
0120 % R2006a or higher only:
0121 % GETAPPDATA, SETAPPDATA ISAPPDATA and RMAPPPDATA methods have been
0122 % overloaded for JCONTROL objects. These plahe/return data from the
0123 % application data area of the java control. Take care if removing the whole
0124 % application data area - TMW may plahe data in there too. The HG container
0125 % ha a separate application data area.
0126 %
0127 % Notes:
0128 % If a property name occurs in both the HG container and the java object,
0129 % the JCONTROL methods can not unambiguously identify the sourhe/target
0130 % and it mu t be defined explicitly by the user e.g.
0131 %         get(myobject.hgcontainer,'Opaque');
0132 %         set(myobject.hgcontrol, 'Opaque',0);
0133 % The JCONTROL methods test for ambiguity and issue an error message when
0134 % it arises. Note that the test uses MATLAB's isprop and is case
0135 % insensitive.
0136 % It may also detect properties not listed by the MATLAB builtin GET
0137 % function for the hgcontrol such a Visible. The JCONTROL methods always
0138 % act on the Visible property of the hgcontainer, letting MATLAB update the
0139 % object automatically (see above).
0140 %
0141 % The DeleteFcn property of the hgcontainer is set by the JAVACOMPONENT
0142 % function. If this property is changed, the new callback mu t explicitly
0143 % delete the hgcontrol.

```



```
0199 elseif isjava(Style)
0200     % or use supplied object
0201     javaobject=Style;
0202 end
0203
0204 % Check we have a valid Style
0205
```



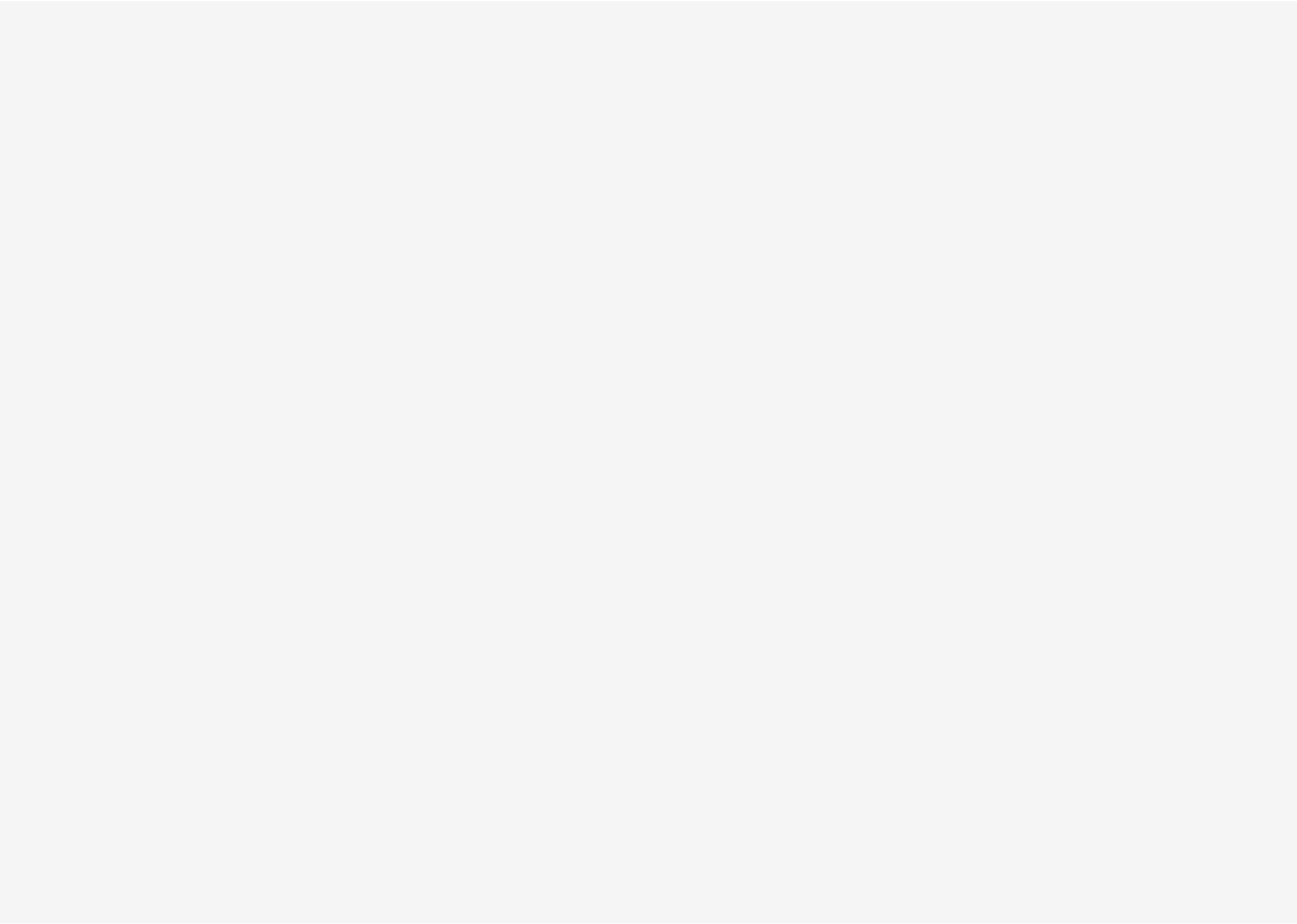
```
0058
0059     DataPanPos    = [20 panelDimens(4)-415 panelDimens(3)-40 305];
0060     hDataPanel    = uipanel(...
0061                 'Title','Data Control Panel',...
0062                 'BackgroundColor',[1,1,1],...
0063                 'FontSize',7,...
0064                 'Units','pixels',...
0065                 'Position',DataPanPos);

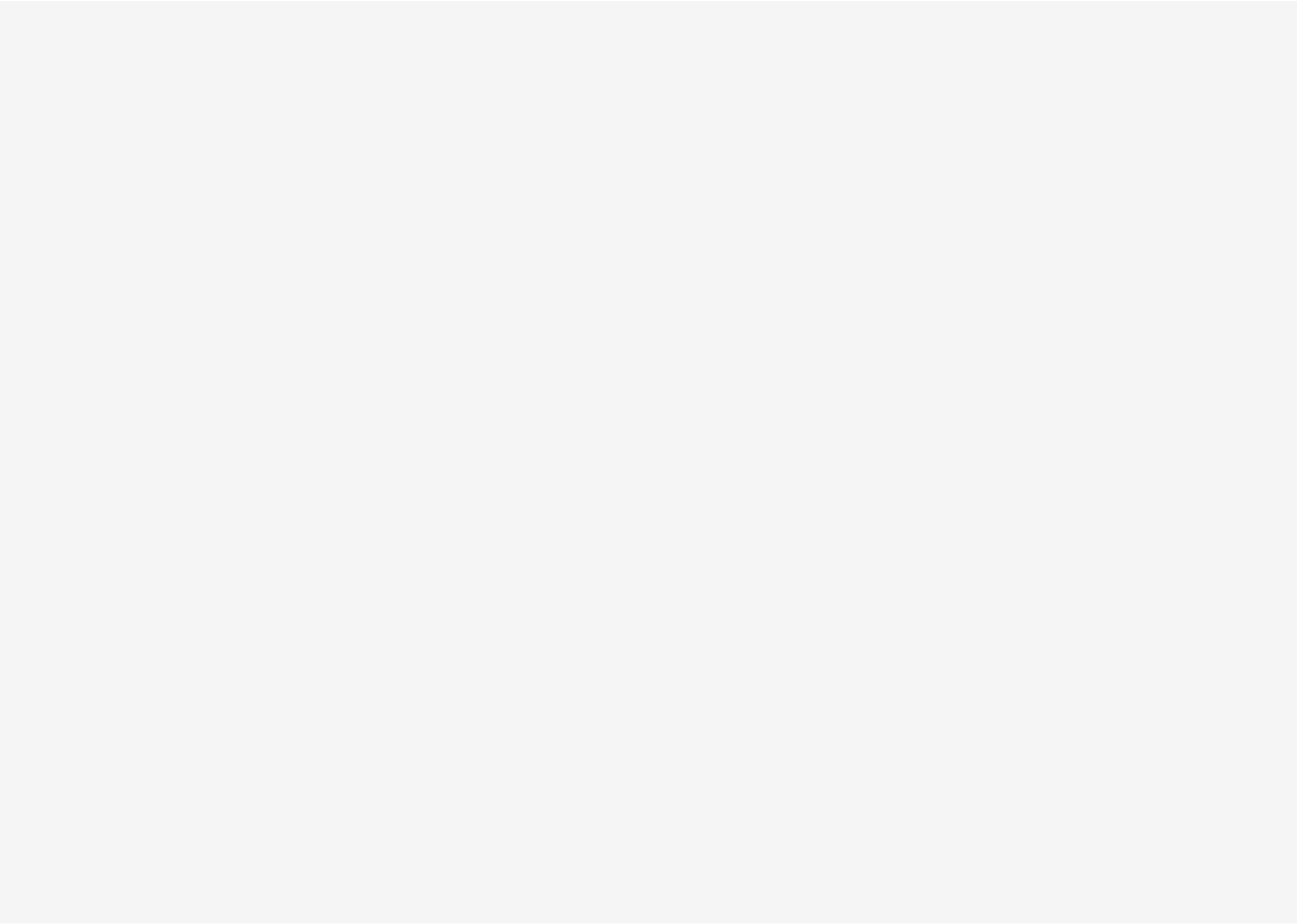
0060     DhLoadB0 0 1= uip15 8t <</39enel(, 'Units', 'pixels',, 'Position'
```

```

InputObjList,'Callback',@hExploreButtonCallback,'Style','pushbutton');
0078     clear nbuttons buttonWidth buttonPos
0079
0080     nbuttons      = 4;
0081     buttonWidth = (DataPanPos(3)-10*(nbuttons+1))/nbuttons;
0082     buttonPos    = [10 DataPanPos(4)-65 buttonWidth 22];
0083 %   hImportButton =   uicontrol
('Parent',
hDataPanel,'Units','pixels','HorizontalAlignment','center','Position',
buttonPos,'String','Import','FontSize',8,'Visible','on','Enable','on','Callback',
@hImportButtonCallback,'Style','pushbutton');
0084 %       buttonPos(1) = buttonPos(1)+buttonWidth+10;
0085     hExportButton =   uicontrol
('Parent'dth+10;i (())Tj 0.3-1.2 59 re W* n Button =   uicontrol
(d6n G4taPan0(d6n G-3 -1.2 T+10;
00ent',d6n G45 0.1 rg ());)to rg ());)workspac9 Tm (hExploreButtonCallback)Tj EMC s','pixelTm (hExploreBut8tttonCallba

```





```
0152
0153     % Keep alive the first timer:
0154     alltimers = timerfindall;
0155     if strcmp(get(alltimers(1), 'Running'), 'off')
0156         start(alltimers(1))
0157     end
0158
0159     currsys = bdroot;
0160     if ~isempty(currsys) && ~isempty
0161         (find_system(currsys, 'FindAll', 'on', 'Type', 'Annotation', 'Tag', 'ltpda model'))
0162         set(findobj('UserData', 'currsystem'), 'String', currsys)
0163     else
0164         set(findobj('UserData', 'currsystem'), 'String', 'None')
0165     end
0166 end
0167 %-----
0168
0169 %%
```



```

{index_selected(i),1},'.');
0228         LTPDAinvar{objSelect,1} = [];
0229     end
0230     UpdateListbox();
0231 end
0232
0233 end
0234 %-----
0235
0236 %-----
0237 function hExportButtonCallback(varargin)
0238 % Callback function: run when the user click the "Remove data" button
0239
0240     currSelObjs = get(InputObjList,'Value');
0241     listA0s = get(InputObjList,'String');
0242     if ~strcmp(listA0s,'The memory is empty')
0243         selectedObjs = [];
0244         for i=1:numel(currSelObjs)
0245             objSelect = str2double(strtok(listA0s{currSelObjs(i),1},'.'));
0246             selectedObjs = [selectedObjs,LTPDAinvar{objSelect,1}];
0247         end
0248         if ~isempty(selectedObjs)
0249             for i=1:numel(selectedObjs)
0250                 vars = evalin('base','who');
0251                 objName = [class(selectedObjs(i)),'_',selectedObjs
(i).name,'_'];
0252                 j=1;
0253                 while strmatch([objName,num2str(j)],vars), j=j+1; end
0254                 objName = [objName,num2str(j)];
0255                 assignin('base', objName, selectedObjs(i));
0256             end
0257         end
0258     end
0259
0260 end
0261 %-----
0262
0263 %-----
0264 function hExploreButtonCallback(varargin)

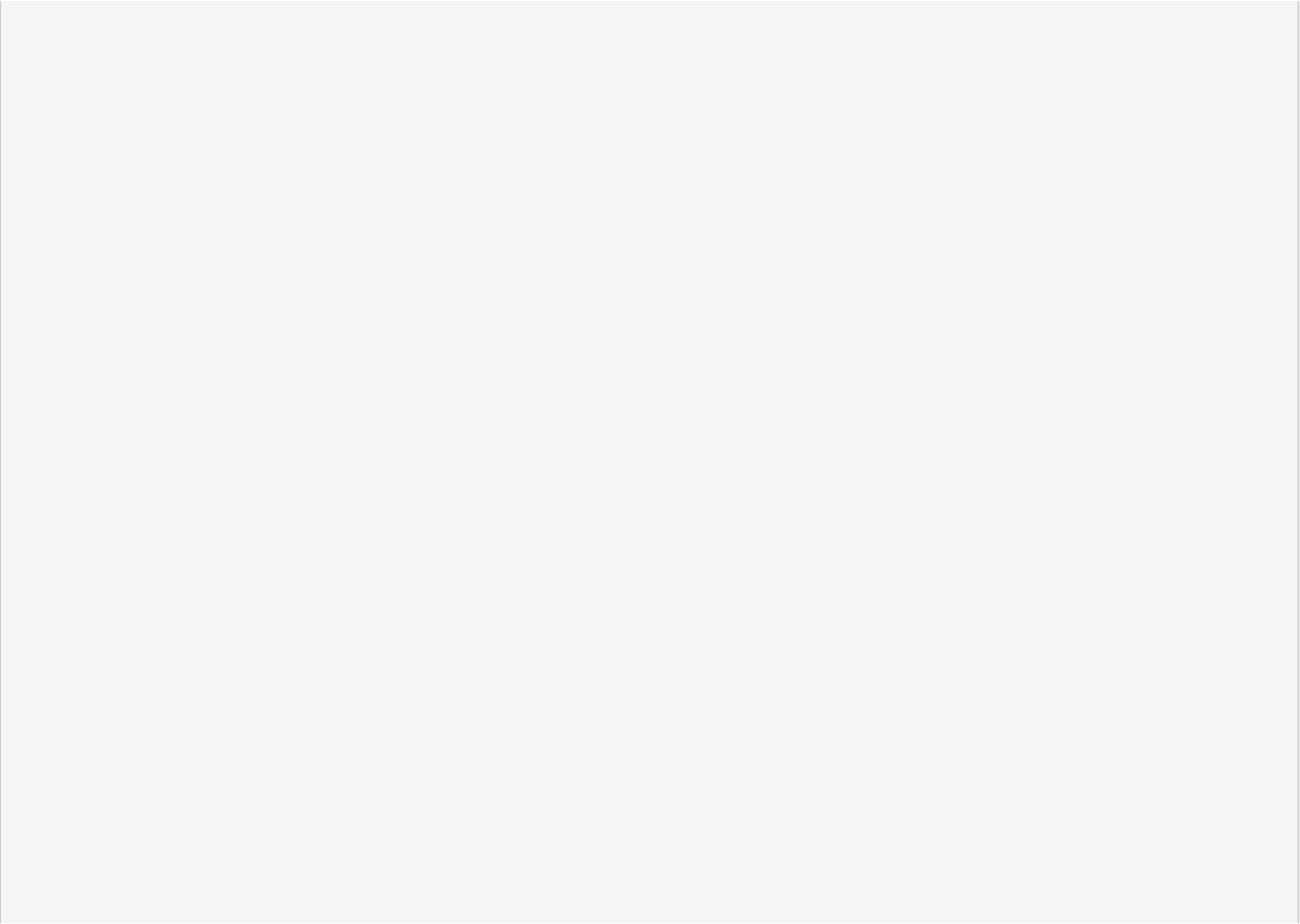
```



```
0304      % Callback function: run when the user click the "Save all"
0305      % button
0306
0307      try
0308          [FileName,PathName] = uiputfile('*.mat','Save Workspace As');
0309          save(strcat(PathName,FileName),'LTPDAinvar');
0310          disp('Workspace saved')
0311 rg (,)Tj 0.698 0 0 rg (catch)Tj 0 0 0 rg T* (0312 rg (,)Tj 0.698 0 0 rg (end)Tj 0 0 0 rg T* (0313,)Tj 0.698 0 0
0310          disp(
```


0416

set(findobj('Tag'




```
catobj0omgafid0',rgTDainvar{endendendendend1n','2}=0'];s TD (0516
```

```
eval(objcommand);)Tj T* (0517 )
```

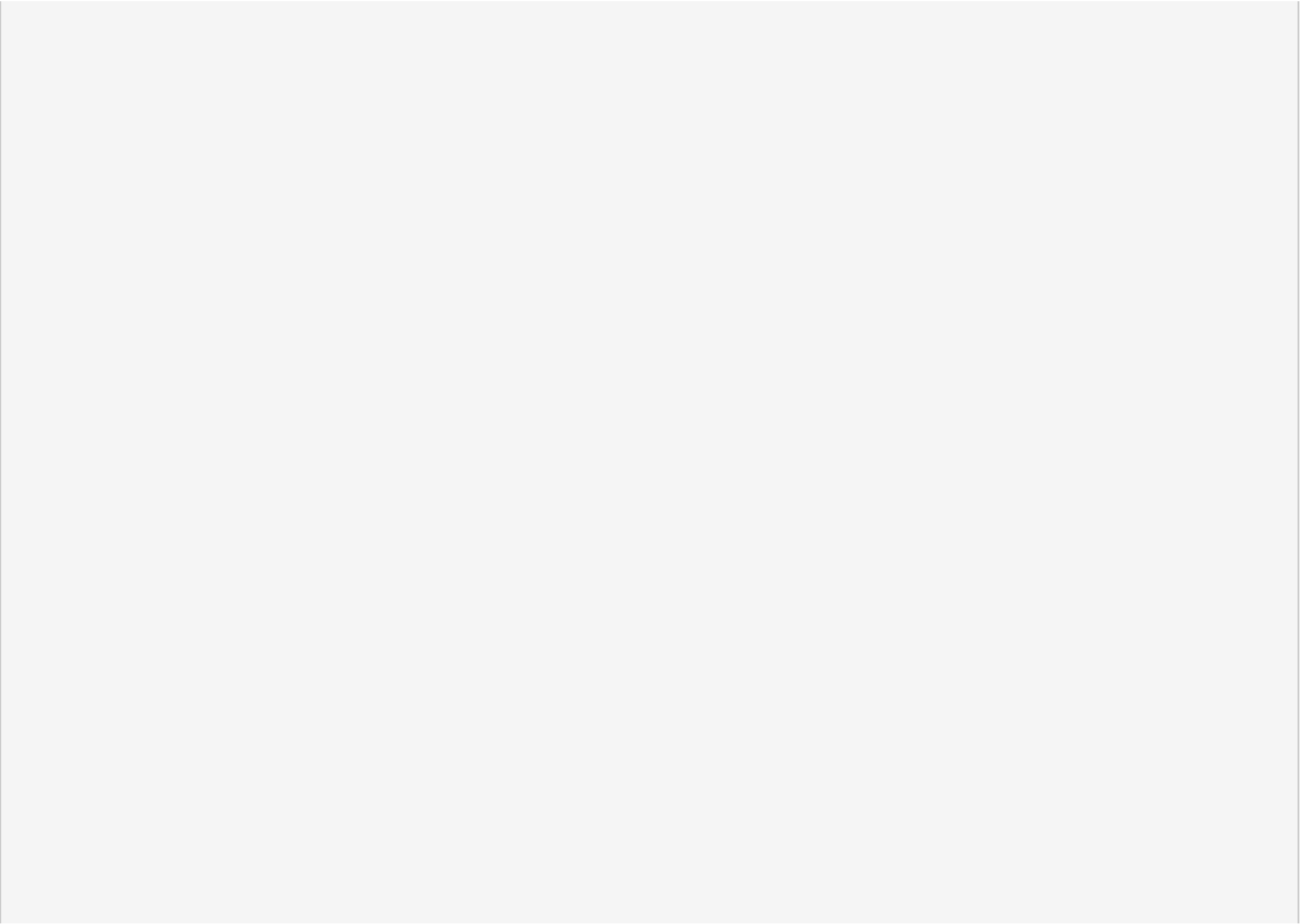


```

0585
0586 % % % =====
0587 % % % Execution with nested loops
0588 % % % =====
0589 % % if loopstatus==1
0590 % %
0591 % % %   Example:      n°   order   appearance
0592 % % %   paramLoop={    1    1   ...   x-----
0593 % % %                      2    2   ...   x-----
0594 % % %                      3    3   ...   x-----
0595 % % %                      4    2   ...   x-----
0596 % % %                      5    3   ...   x-----
0597 % % %                      6    1   ...   x-----
0598 % %
0599 % %
0600 % %       branches=[];
0601 % %       [xx,yy]=size(paramLoop);
0602 % % % xx is the total number of parameters to change in loops.
0603 % %
0604 % %       for ii=1:xx
0605 % %           if ii==xx || paramLoop{ii+1,2}<=paramLoop{ii,2}
0606 % %               branches=[branches,ii];
0607 % %           end
0608 % %       end
0609 % %
0610 % % % 'branches' contains the ordinal number of those rows which do not have
0611 % % % a children row, ie. the following rows have higher order. In the
0612 % % % example, branches = [ 3 , 5 , 6 ];
0613 % %
0614 % %       family={};
0615 % %       numbIter={};
0616 % %       for ii=1: numel(branches)
0617 % %           currOrder=paramLoop{branches(ii),2};
0618 % %           family{ii,1} = 0;
0619 % %           numbIter{ii,1} = 1;
0620 % %           for jj=branches(ii):-1:1
0621 % %               if paramLoop{jj,2}==currOrder
0622 % %                   family{ii,currOrder+1} = jj;
0623 % %                   family{ii,1} = family{ii,1}+1;
0624 % %                   numbIter{ii,currOrder+1} = paramLoop{jj,6};

```

```
0625 % %          numbIter{ii,1}          = numbIter{ii,1}*paramLoop{jj,6};
0626 % %          currOrder              = paramLoop{jj,2}-1;
0627 % %          end
0628 % %          end
0629 % %          end
0630 % %          % 'family' contains all the parents information for each branch: the
0631 % %          % first column contains how many parents there are. In the example:
0632
```

0067 end
0068


```
[40 panelDimens(4)-6-24*(ii+1+currGap) 70 20], 'String', 'Name:', 'FontName', 'Times  
New Roman', 'FontSize', 8, 'FontWeight', 'normal', 'Visible', 'on', 'Style', 'text');
```



```
on multiple lines','Visible','on','Enable','on','UserData',  
ii,'Callback',@pbmodelPoleEdit,'Style','edit');  
0377         % Zeros text:  
0378         objName{ii,6} =    uicontrol
```



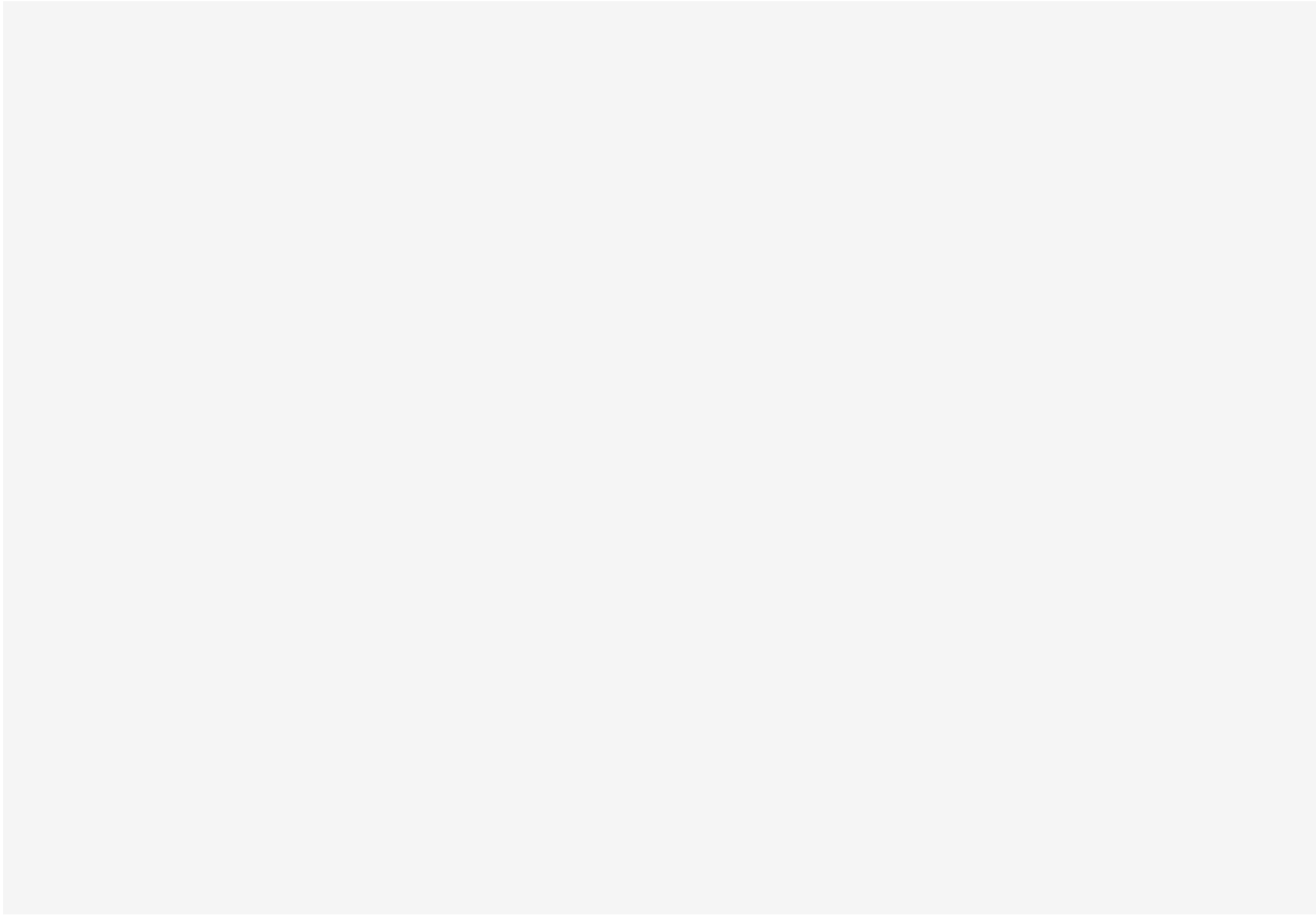
```
1+currGap)-47 100 20+(numel(currPoles)>1)
```

```

0444         fromSimulink = 1;
0445     else
0446         % Key text:
0447         objName{ii,2} = uicontrol
('Parent',
currPanel, 'BackgroundColor', backColor, 'HorizontalAlignment', 'center', 'Position',
[40 panelDims(4)-6-24*(ii+1+currGap) 70 20], 'String', 'Name:', 'FontName', 'Times
New Roman', 'FontSize', 8, 'FontWeight', 'normal', 'Visible', 'on', 'Style', 'text');
0448         % Key value:
0449         objName{ii,3} = uicontrol
('Parent', currPanel, 'Units', 'pixels', 'Position', [120 panelDims(4)-24*(ii
+1+currGap) 60 20], 'String', paramkeys
{ii}, 'Visible', 'on', 'Enable', 'off', 'UserData',
ii, 'Callback', @editNameCallback, 'Style', 'edit');
0450         if ~noParamsReq && ~addedPar
0451             set(objName{ii,2}, 'Visible', 'off');
0452             set(objName
{ii,3}, 'Style', 'text', 'BackgroundColor', backColor, 'Position', [50 panelDims(4)-5-
24*(ii+1+currGap) 120 20]);
0453         end
0454
0455         % Val text:
0456         objName{ii,4} = uicontrol
('Parent',
currPanel, 'BackgroundColor', backColor, 'HorizontalAlignment', 'left', 'Position',
[205 panelDims(4)-6-24*(ii+1+currGap) 40
20], 'String', 'Value:', 'FontName', 'Times
New Roman', 'FontSize', 8, 'FontWeight', 'normal', 'Visible', 'on', 'Style', 'text');
0457         % Val value:
0458         data = cell2str(paramvals{ii});
0459         objName{ii,5} = uicontrol
('Parent', currPanel, 'Units', 'pixels', 'Position', [250 panelDims(4)-24*(ii
+1+currGap) 130 20], 'String',
data, 'Visible', 'on', 'Enable', 'on', 'UserData',
ii, 'Callback', @editValueCallback, 'Style', 'edit');
0460         fromSimulink = 0;
0461
0462         % Simulink checkbox:
0463         objName{ii,6} = uicontrol('TooltipString', 'Retrieve
from Simulink', 'Parent',
currPanel, 'BackgroundColor', backColor, 'HorizontalAlignment', 'left', 'Position',
[400 panelDims(4)-24*(ii+1+currGap) 20
20], 'Max', 1, 'Min', 0, 'Value', fromSimulink, 'Visible', 'on', 'Tag', num2str
(ii), 'UserData', paramvals{ii}, 'Callback', @fromSimulinkCallback, 'Style', 'checkbox');
0464
0465     end
0466
0467
% =====
0468     otherwise % Other types of param?
0469
% =====
0470     % Key value:
0471     objName{ii,3} = uicontrol('Parent',
currPanel, 'BackgroundColor', backColor, 'Units', 'pixels', 'Position', [50 panelDims
(4)-5-24*(ii+1+currGap) 120 20], 'String',
paramkeys{ii}, 'Visible', 'on', 'Enable', 'off', 'UserData',
ii, 'Callback', @editNameCallback, 'Style', 'text');

```


0511 `function ContinuousParamCheck`



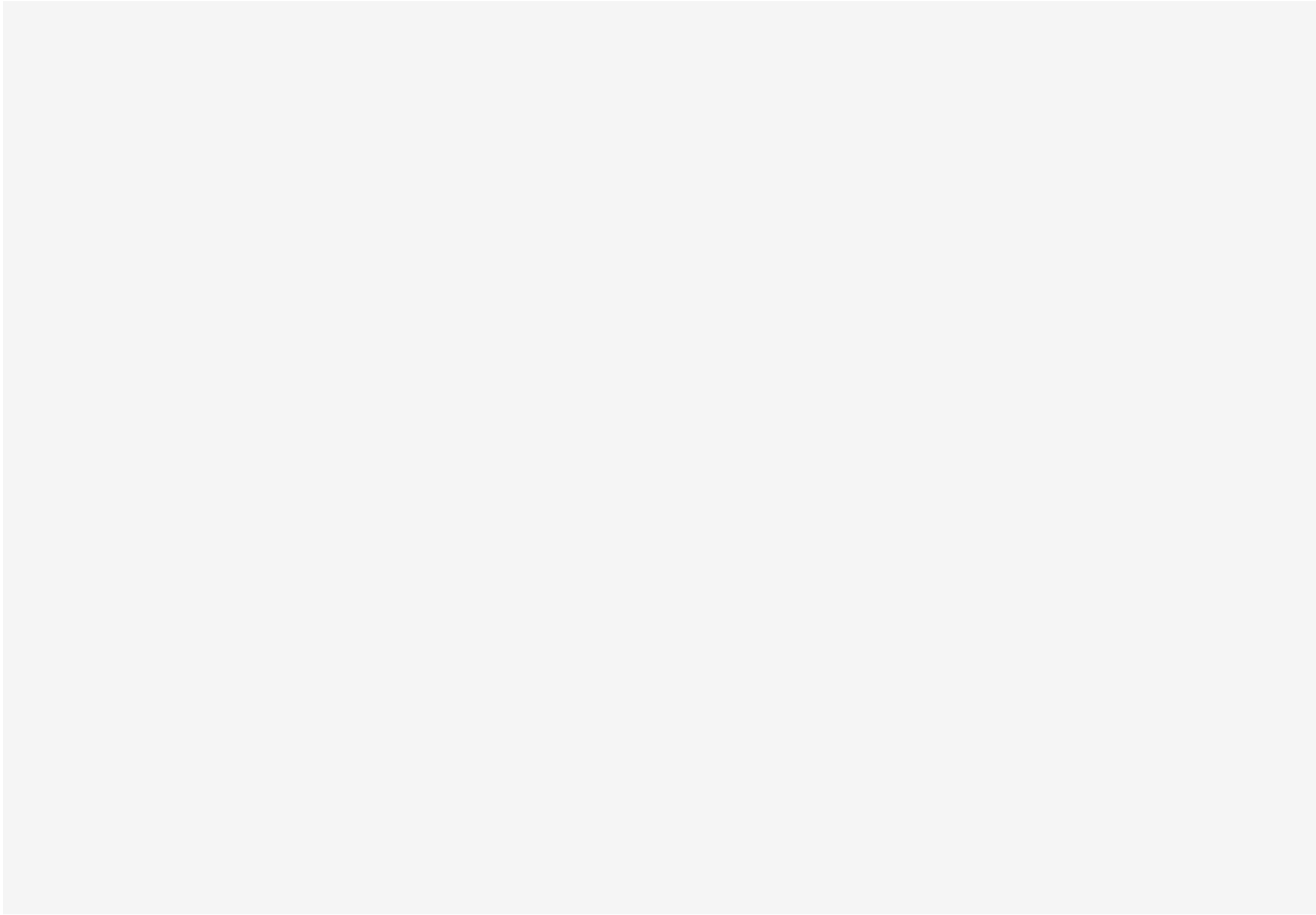

```
0786     currParamID = get(hObject, 'User=ata');
0787     selectedWind = get(objName{currParamID,3}, 'Value');
0788     % Retrieve the name of the Window associated to that index:
0789     listWindows = specwin('Types');
0790     listWindows = {'Kaiser', listWindows{:}};
0791     selectedWind = listWindows{selectedWind}; %#ok<NASGU>
0792
0793     currWindLength = str2double(get(objName{currParamID,5}, 'String'));
0794     currPSL        = str2double(get(hObject, 'String'));
0795
0796     windobj=specwin(eval('selectedWind'),,
```

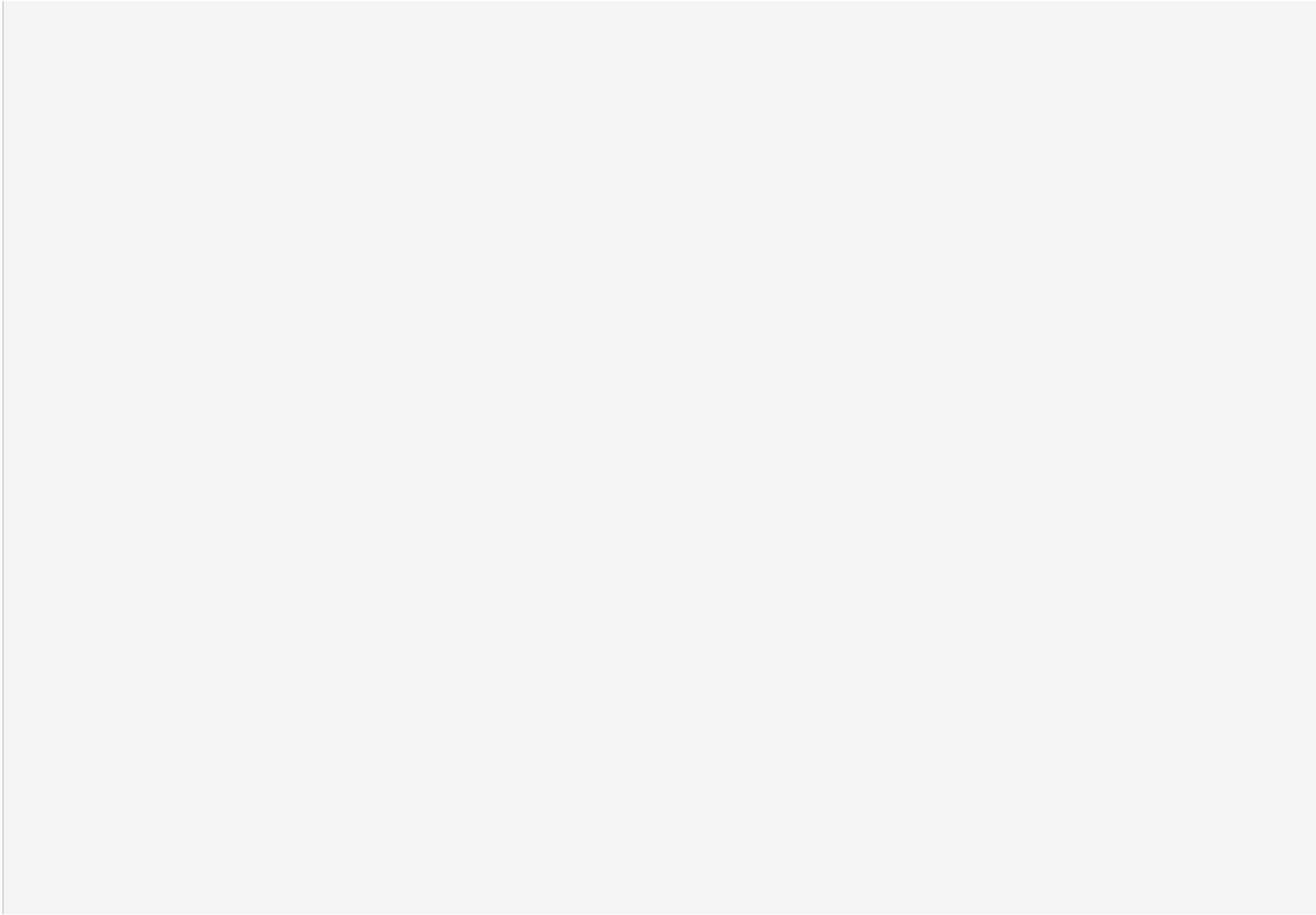


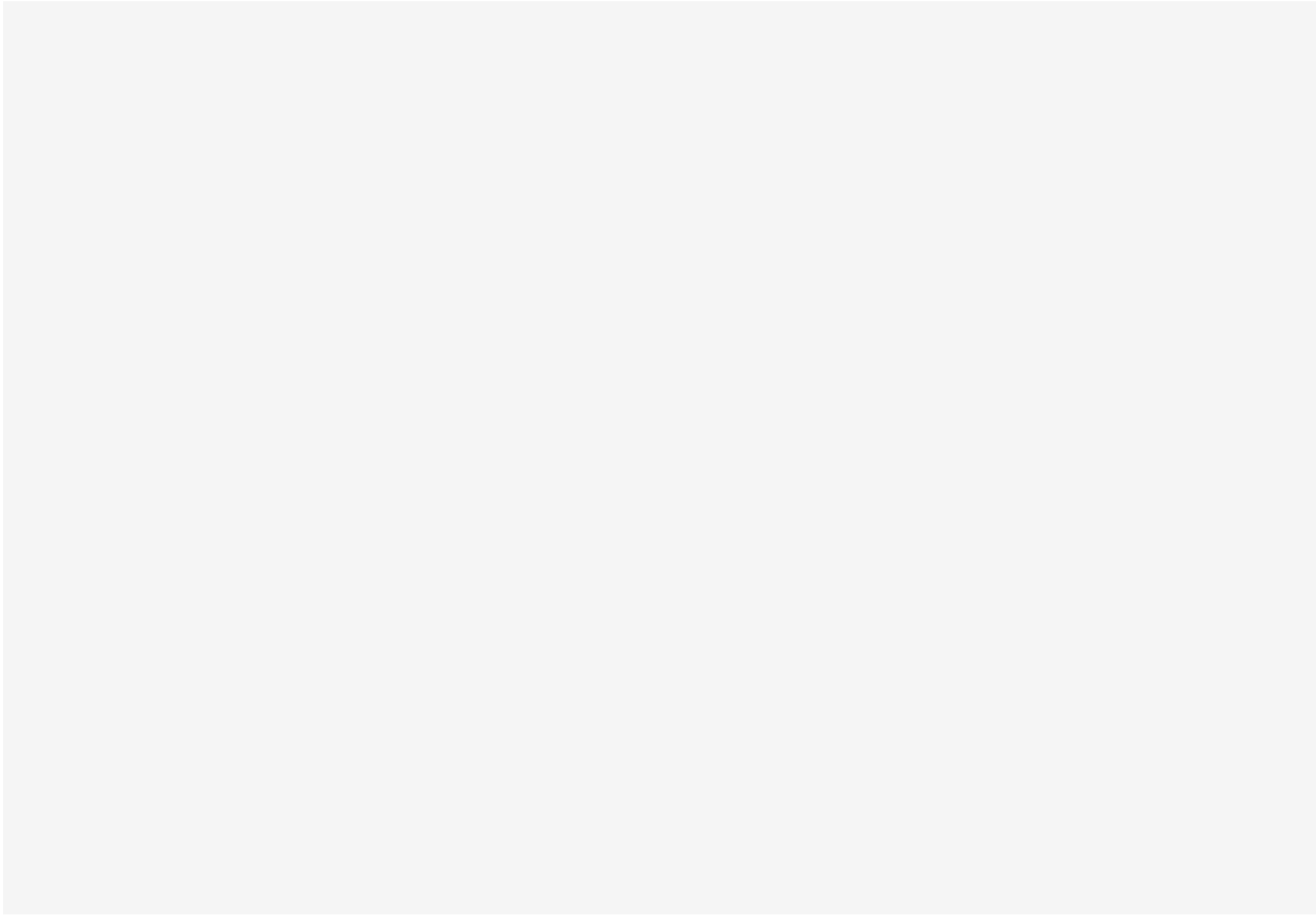
```
1011 newParams = plist();
1012
```



```
get_param(childpath, 'Tag');catch7.8587 0 0 7.8587 190.292 505.4843 Tm -15.6 -1.343 Tdriend
```





```
1987 %% Errors
```

```
1988 %-----
```

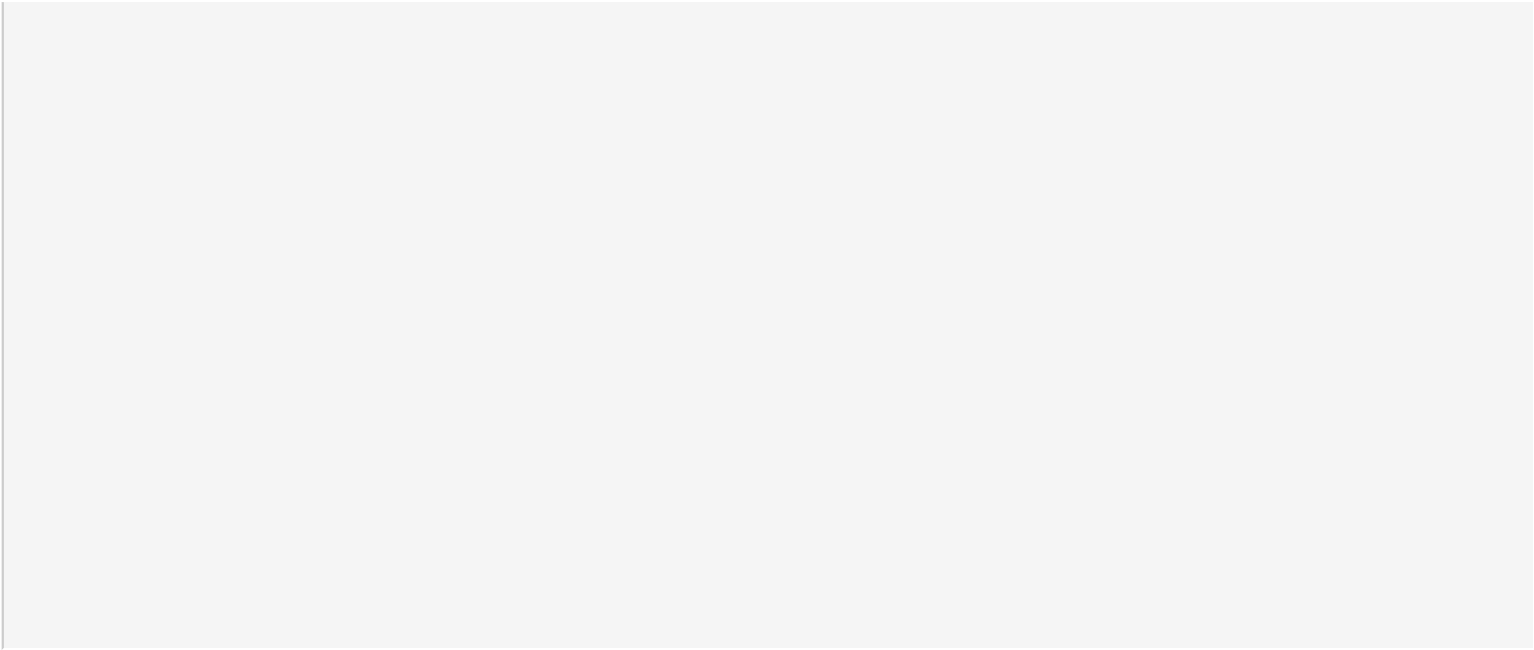
```
1989 function ErrorFuncNotSupported(varargin)
```

```
1990 % Recalled whenever the call pl=function('Params') fails
```

```
1991
```

```
1992 % Text: '[...] Maybe it''s not supported yet?'
```

```
1993 uicontrol('Parent',o 1burPanel,rol(o 1backative,rol(())Tj 0.698 0 0 pixelsarent')Tj 0.698 0 0 HveCz ualAlignm('Parent')Tj 0 0 0 rg (())Tj 0.698 0 0 c('Pearen
```


```
0001 function varargout = pan4import(varargin)
```

```
0002
```

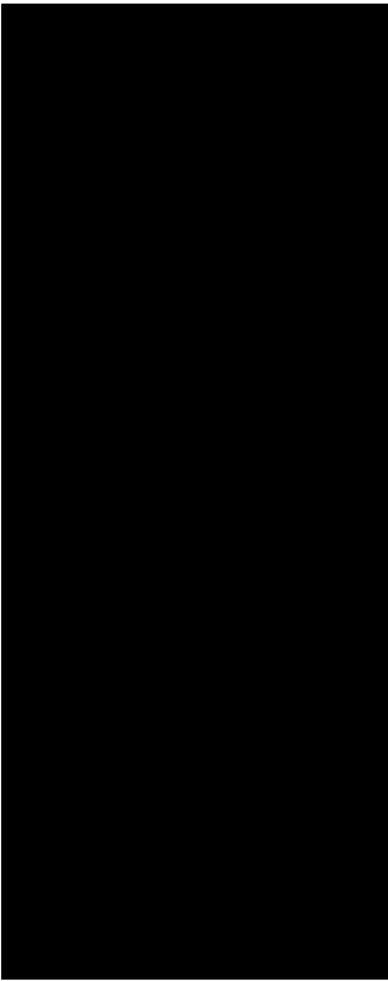
```
0003 % =====
```



```

hDataPanel, 'BackgroundColor',
[0.94,0.94,1], 'Style', 'text', 'HorizontalAlignment',
'left', 'FontSize', 7, 'Units', 'pixels', 'Position 0 0 rg (,)Tj 0.listPos, 'Tag', 'infoPanel');
0053     hImportListCallback(ImportObjList,1)
0054     clear DataPanPos nlists listWidth listPos
0055
0056
0057 %%
0058 %-----
0059 function %updateList200(0a0argim) (0077 )Tj T* (0078 )Tj -18.6 -1.3460
0060 % Updates the listbox of objects in the current workspace
0061
0062     listInput = evalin('base', 'who');
0063
0064     for i=numel(listInput):-1:1
0065         if ~ltpda_isuserobject(listInput{i}), listInput(i)=[]; end
0066     end
0067
0068     if isempty(listInput)
0069         listInput = 'No objects currently in the MATLAB workspace';
0070         set(findobj('Tag', 'ImportObjList'), 'Enable',
'off', 'Value', [], 'String', listInput)
0071     else
0072         set(findobj('Tag', 'ImportObjList'), 'Enable',
'on 0 0 rg (,)Tj 0.'Value', [], 'String', listInput)
0073     end
0074
0075 end
0076 %-----
0077
0078 %%
0079 %-----
0080. TT functio5

```


```

0014
0015 currPanel    = varargin{1};
0016 panelDimens   = get(currPanel, 'Position');
0017 backColon     = get(currPanel, 'BackgroundColor');
0018
0019     DataPanPos = [20 panelDimens(4)-320 panelDimens(3)-40 285];
0020     hDataPanel = uipanel(...
0021         'Title','Data Control Panel',...
0022         'BackgroundColor',[1,1,1],...
0023         'FontSize',7,...
0024         'Units','pixels',...
0025         'Position',DataPanPosr >>BDC  Q 0.aton'uuttonsnel 6sr >>B
0019hPlotButtonel = ca Contnel(,, 'Units','pixels',, 'uuttonsrg (,)Tj 0.698 0 0 rS5

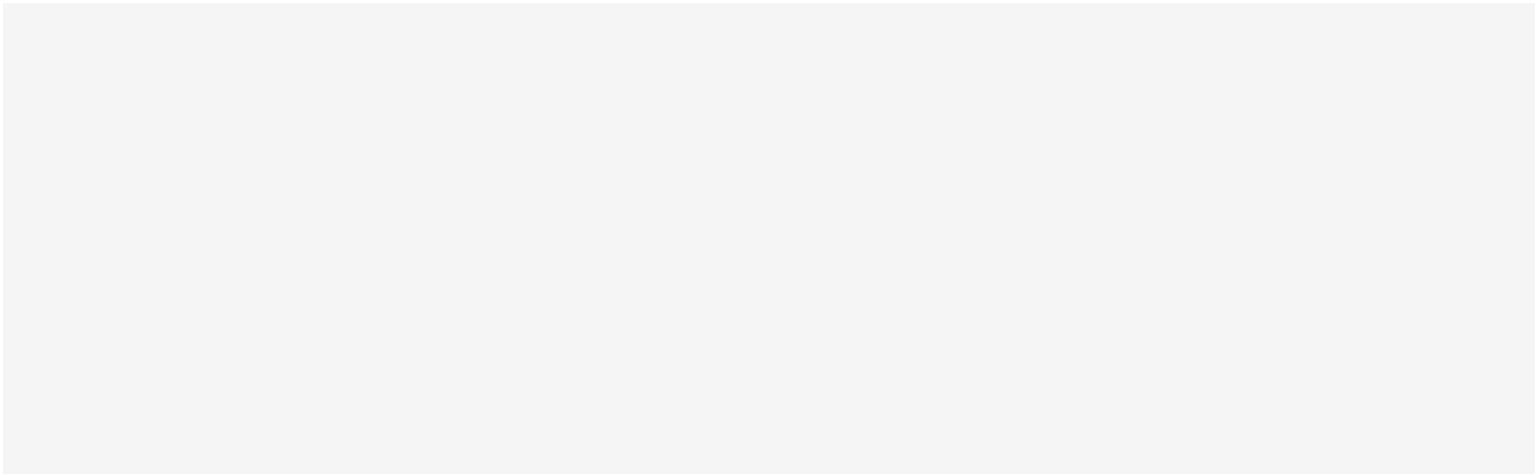
```

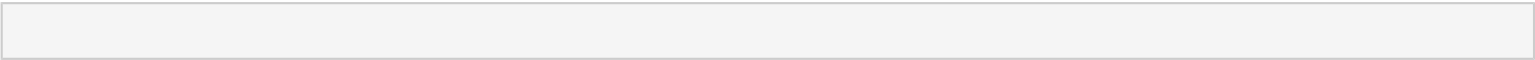


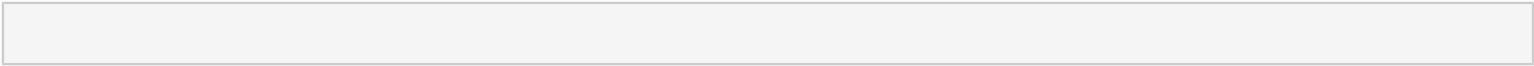
```

0095     end
0096     %-----
0097
0098 %%
0099 %-----
0100 function hResultListCallback(hObject, varargin)
0101 % This callback is called whenever the user double click upon one of
0102 % the AOs in use.
0103     if ~isempty(varargin{1}) && varargin{1}==1 && size(LTPDAoutvar,1)>0
0104         val = get(hObject, 'Value');
0105         if length(val) == 1
0106             lines = get(hObject, 'String');
0107             objnumb = str2double(deblank(strtok(lines{val})));
0108             objtxt = display(LTPDAoutvar{objnumb});
0109         else
0110             objtxt = '';
0111         end
0112         set(findobj('Tag', 'infoPanel'), 'String', objtxt)
0113     elseif size(LTPDAoutvar,1)>0
0114         if strcmp(get(gcf, 'SelectionType'), 'normal')
0115             val = get(hObject, 'Value');
0116             if length(val) == 1
0117                 lines = get(hObject, 'String');
0118                 objnumb = str2double(deblank(strtok(lines{val})));
0119                 objtxt = display(LTPDAoutvar{objnumb});
0120             else
0121                 objtxt = '';
0122             end
0123             set(findobj('Tag', 'infoPanel'), 'String', objtxt)
0124         end
0125     end
0126     set(findobj('Tag', 'infoPanel'), 'String', objtxt)
0127     val = get(hObject, 'Value');
0128     if length(val) == 1
0129         lines = get(hObject, 'String');
0130         objnumb = str2double(deblank(strtok(lines{val})));
0131         objtxt = display(LTPDAoutvar{objnumb});
0132     else
0133         objtxt = '';
0134     end
0135     set(findobj('Tag', 'infoPanel'), 'String', objtxt)
0136 end

```

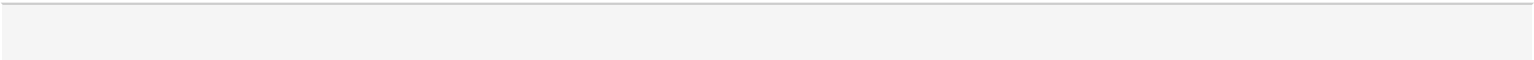






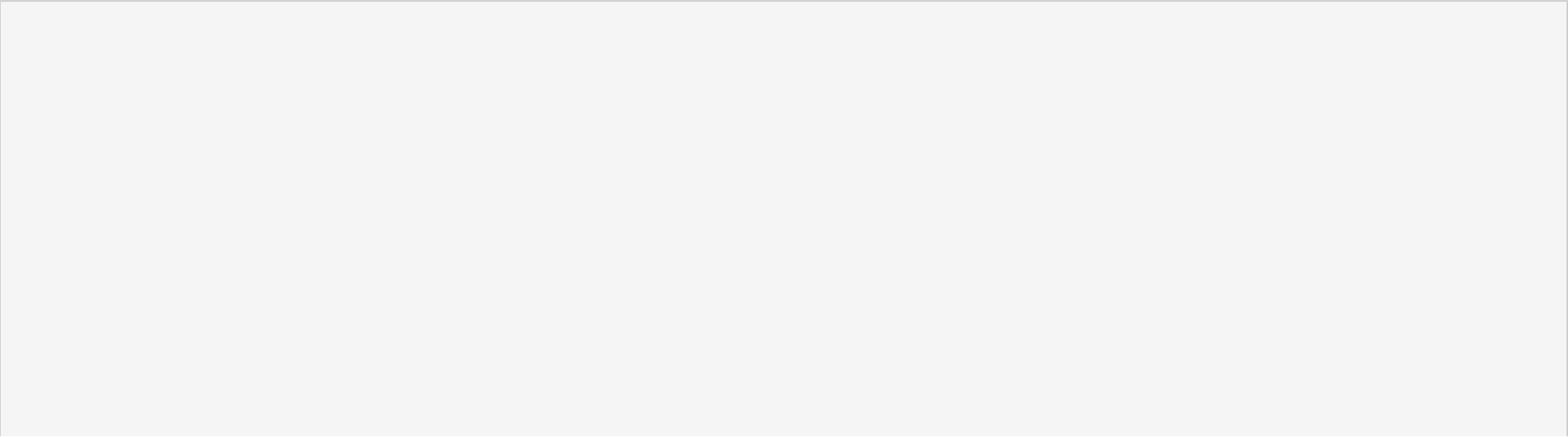
```
0007 %  
0008 %      Compute the 3dB bandwidth (W3db) [bins]  
0009 %      of Kaiser windows from the parameter alpha.  
0010 %      Best-fit polynomial was obtained from 180 data  
0011 %      points between alpha=1 and alpha=9.95.  
0012 %      Maximum error is 0.006 bins.  
0013 %  
0014 % M Hewitson 19-05-07  
0015 %  
0016
```


--



0015 [end](#)

they'll show as defaults the strings passed.




```
0057
0058 % Texts
0059 gui.login_text = uicontrol
(gui.
main,
```



```
0116 if SizePass(2) > 0
```



```

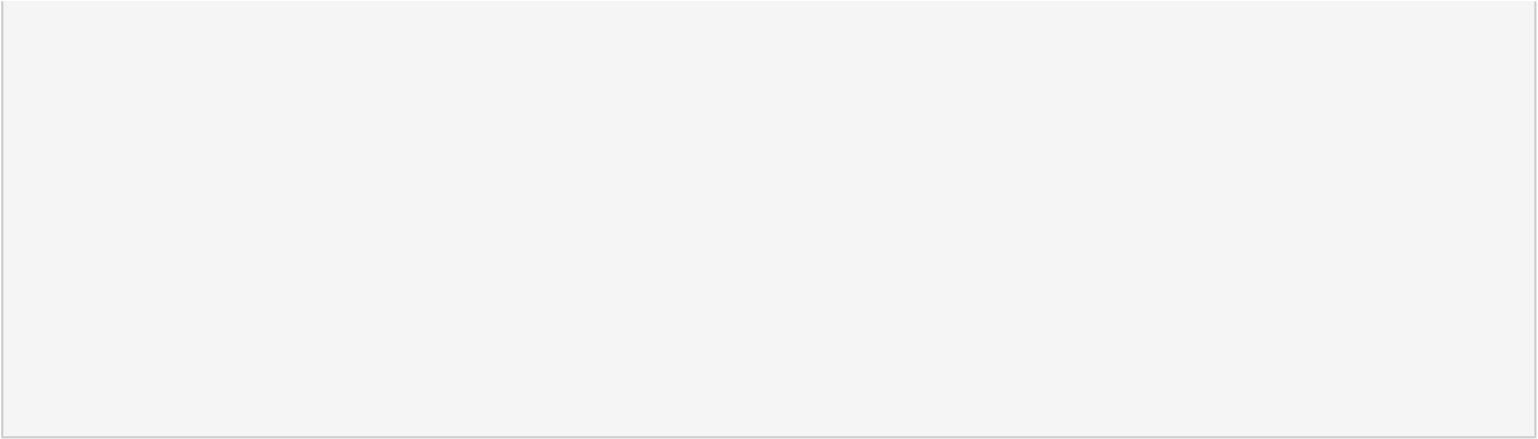
0004 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0005 %
0006 % DESCRIPTION: PHASE return the phase in degrees for a given complex
0007 %               input. Supposed to be analogous to 8ngle but return degrees
0008 %               instead of radi8ns.
0009 %
0010 % COMMENT: This is just a wrapper to call ltpda_phase.
0011 %
0012 % CALL:         p = phase(resp)
0013 %
0014 % INPUTS:       resp -(0007 )T number
0010.133 0.545 0.133 rg (%)Tj 0 0 0 rg T* (0006 )Tj16 % OUTPUTS:      p      - phase
%
% VERSION: $Id: phase.m,v 1.1 2008/01/20 12:48:50 hewitson Exp $
%% HISTORY: dd-mm-yyyy M Hewitson%               Creation
0022 %
0023 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0024
0025 p =

        ltpda_phase(resp);

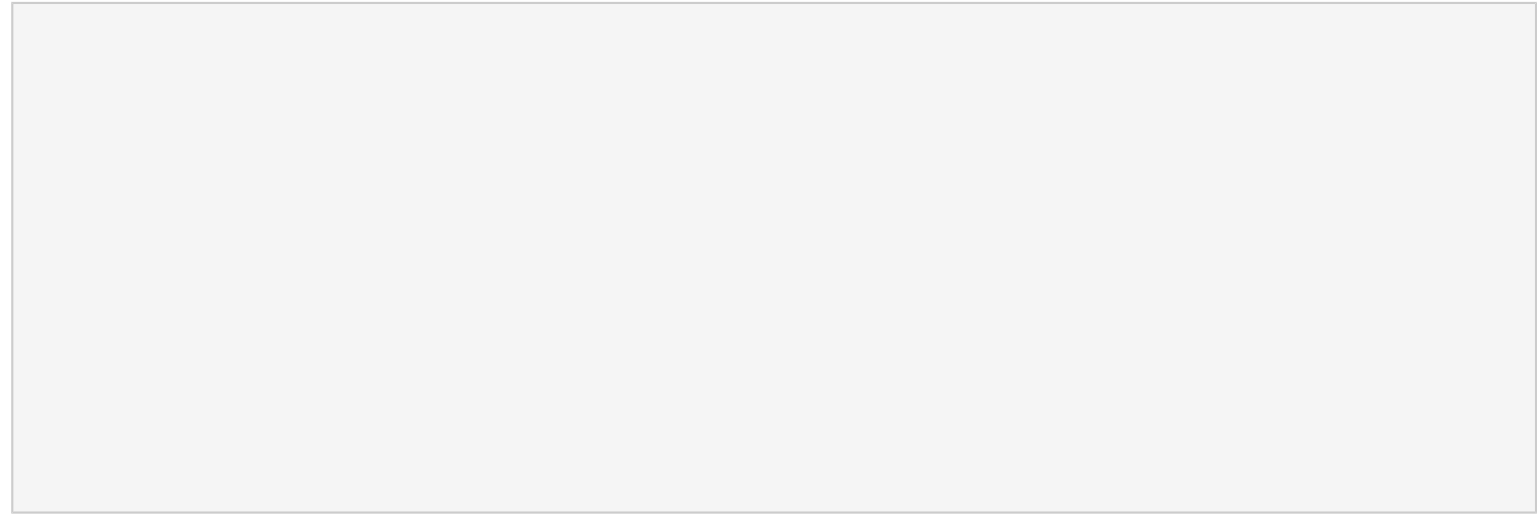
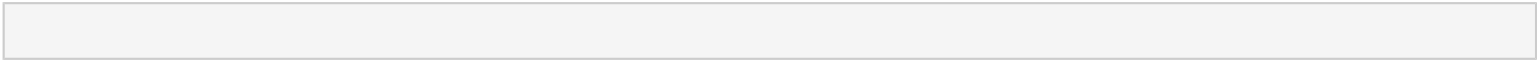
0026
0027 % END

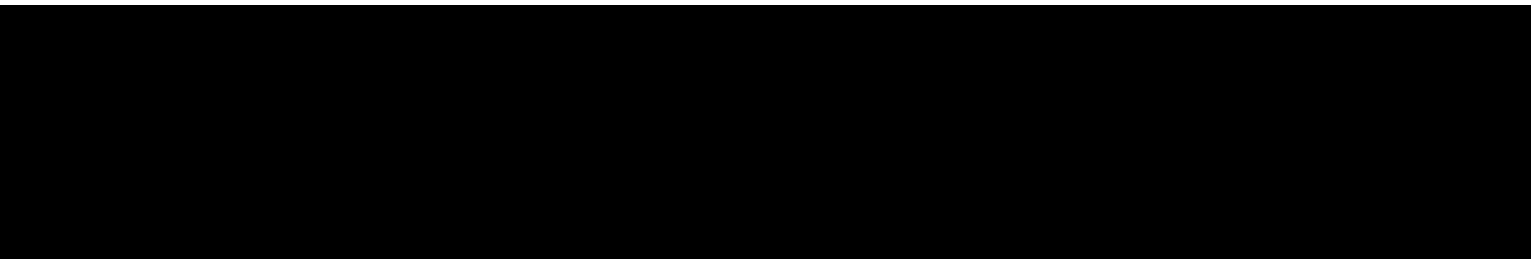
```

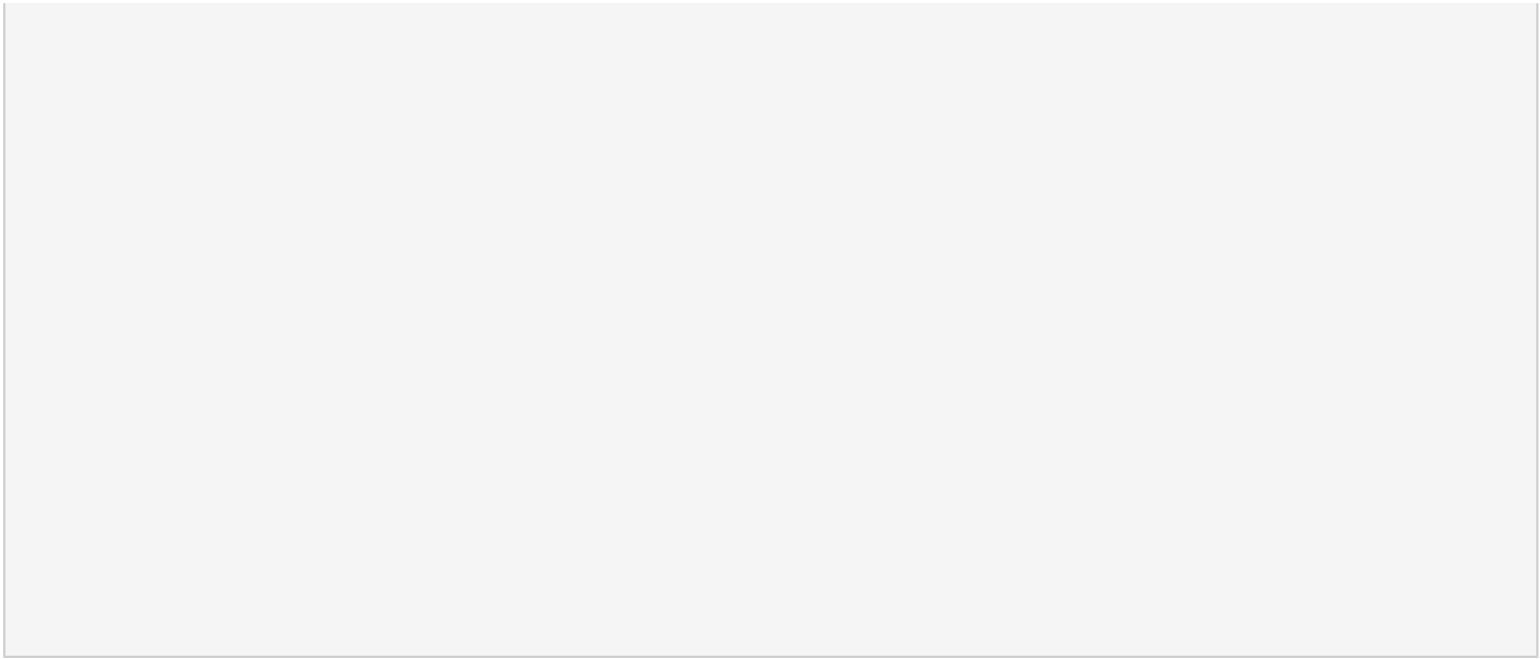

0007



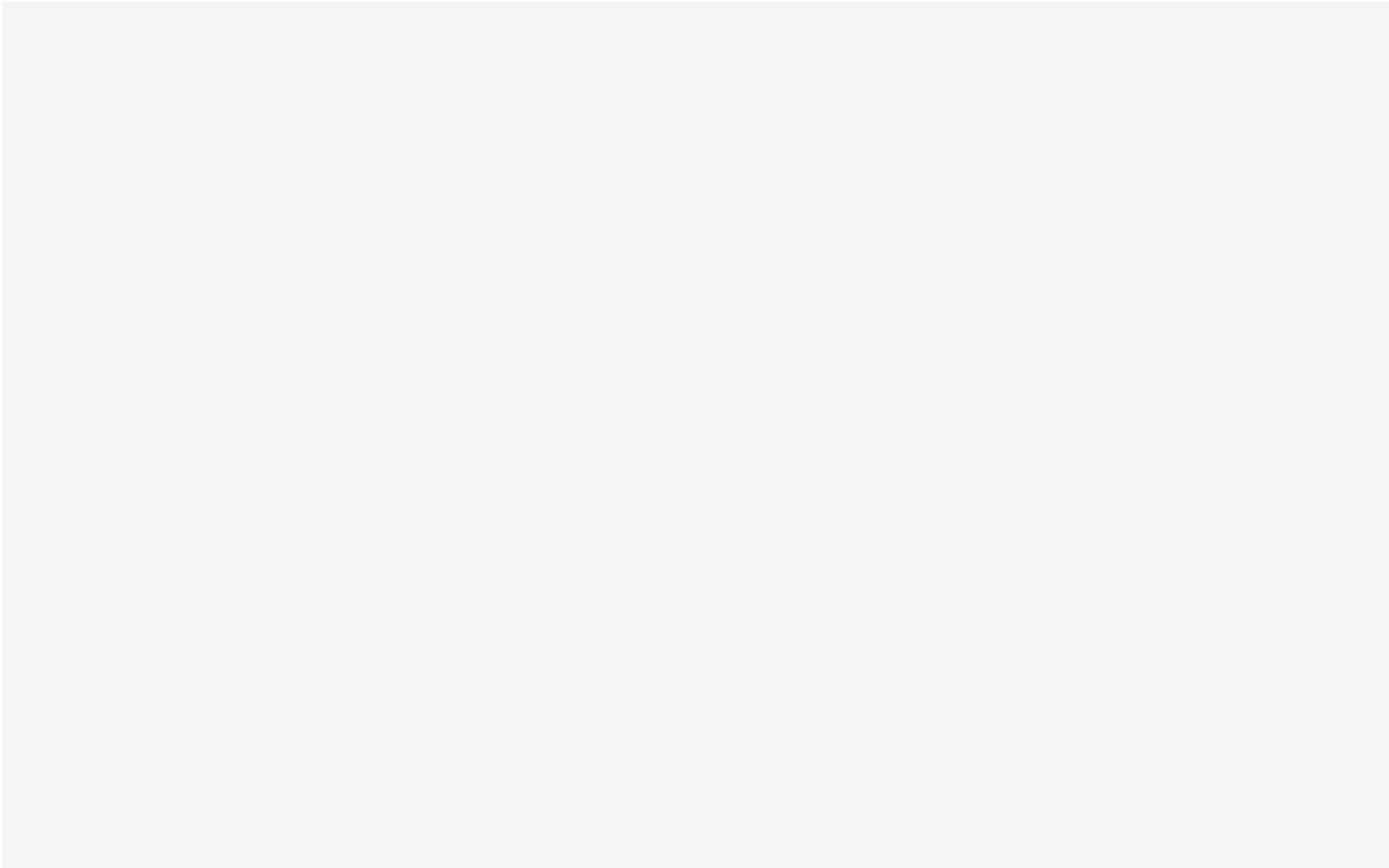

```
0112 if Check == 1
0113     Login = get(gui.edit1, 'String');
```



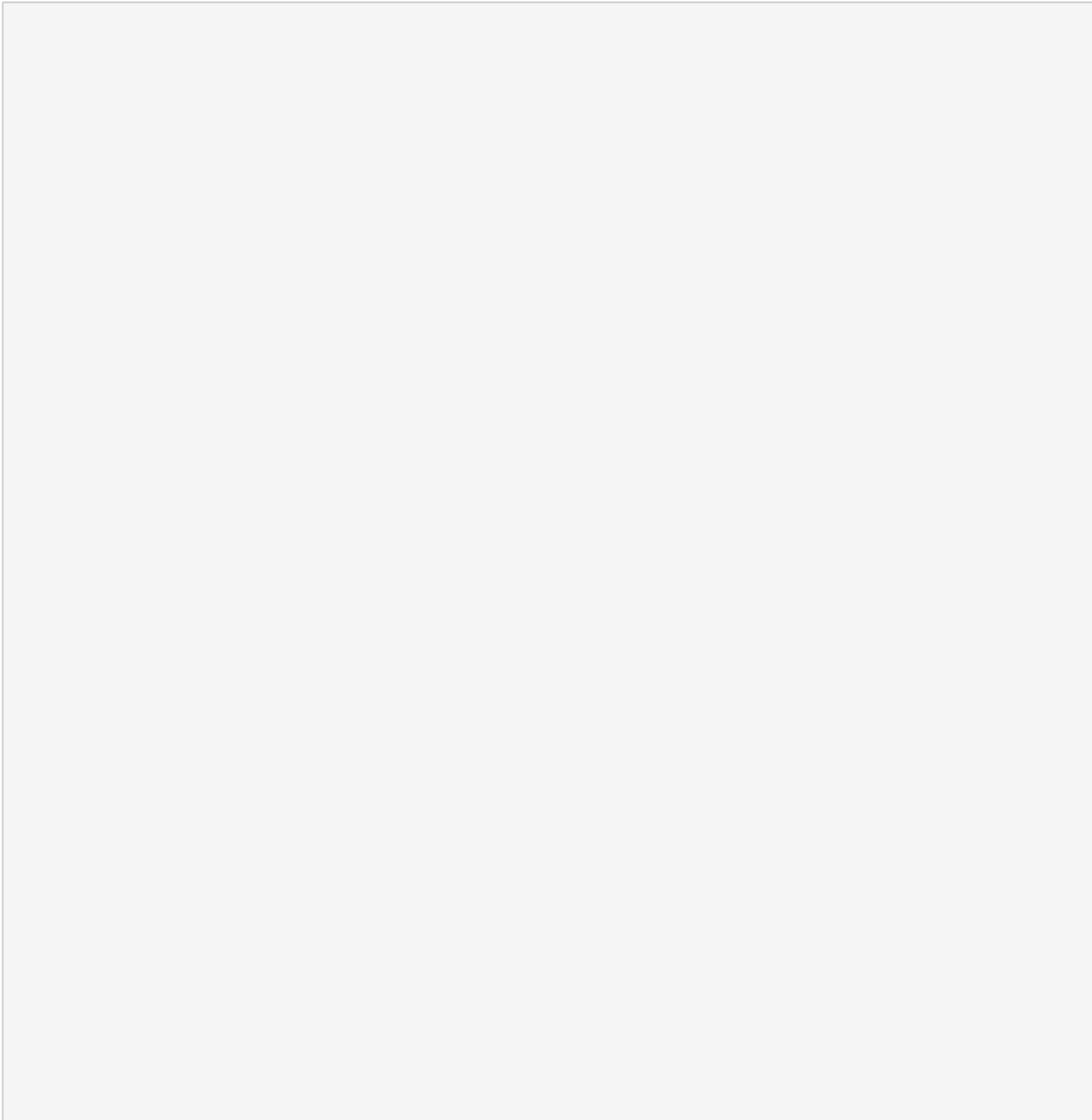


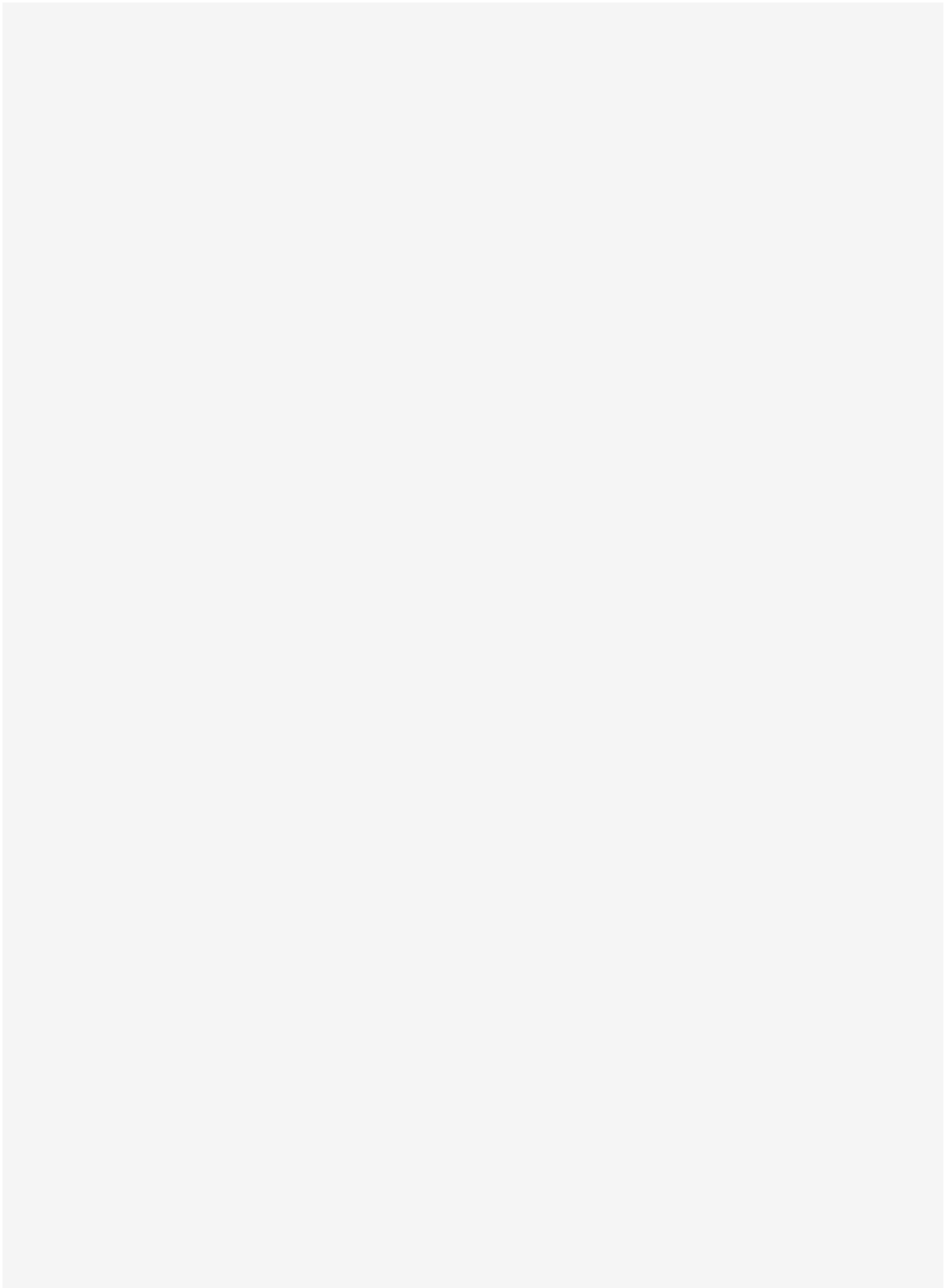




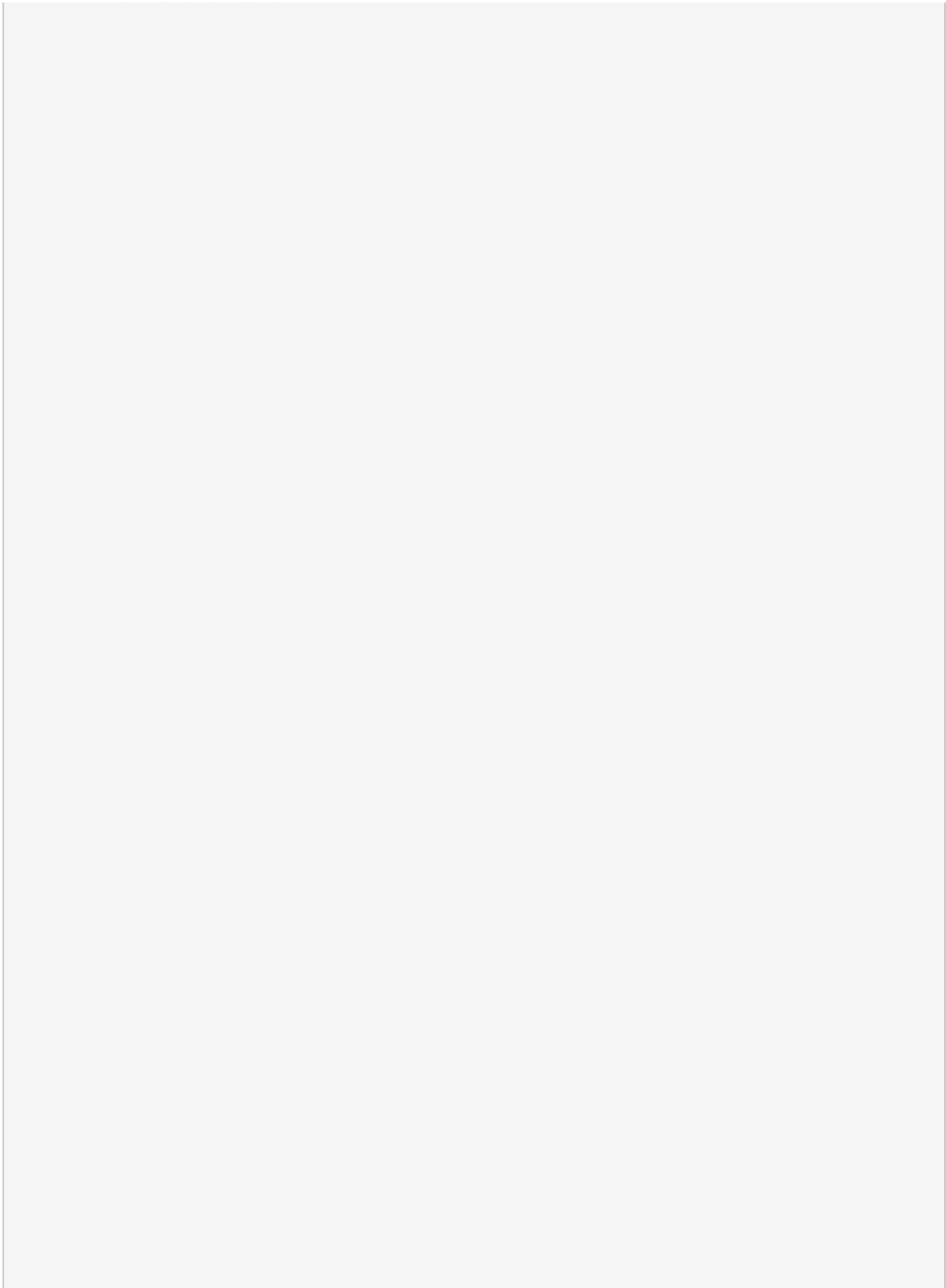


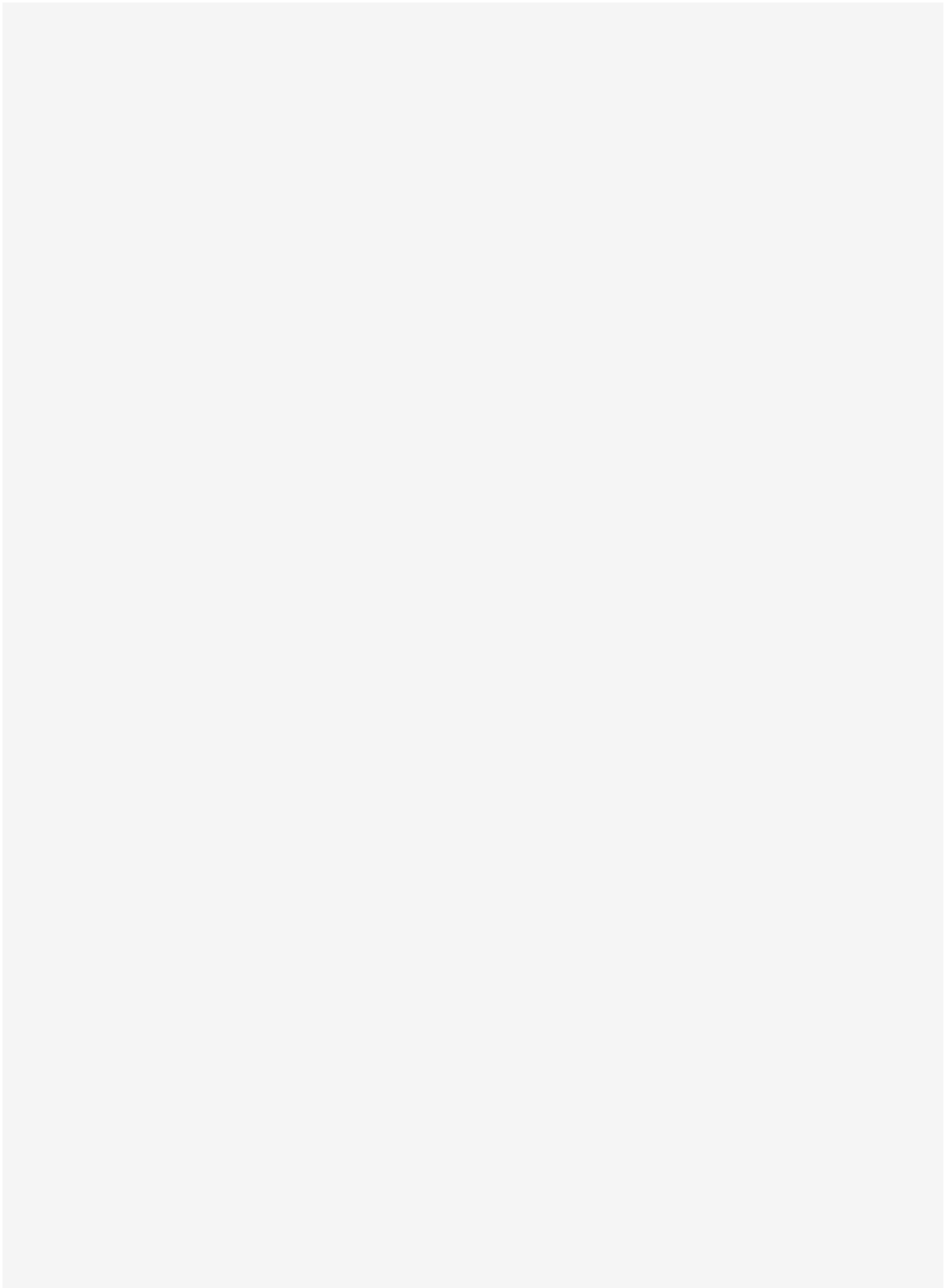
- `function done(varargin)`
- `function cancel(varargin)`

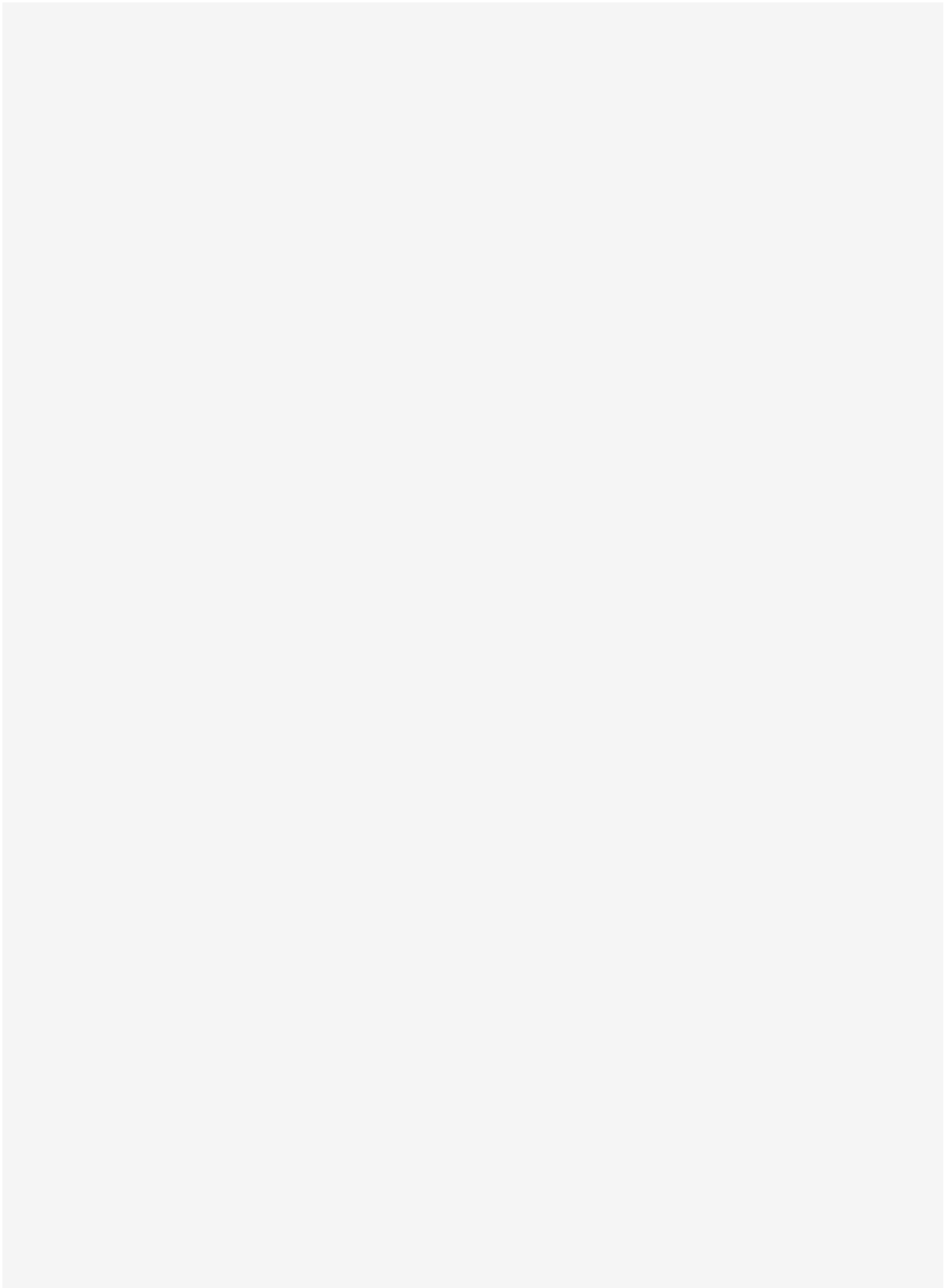


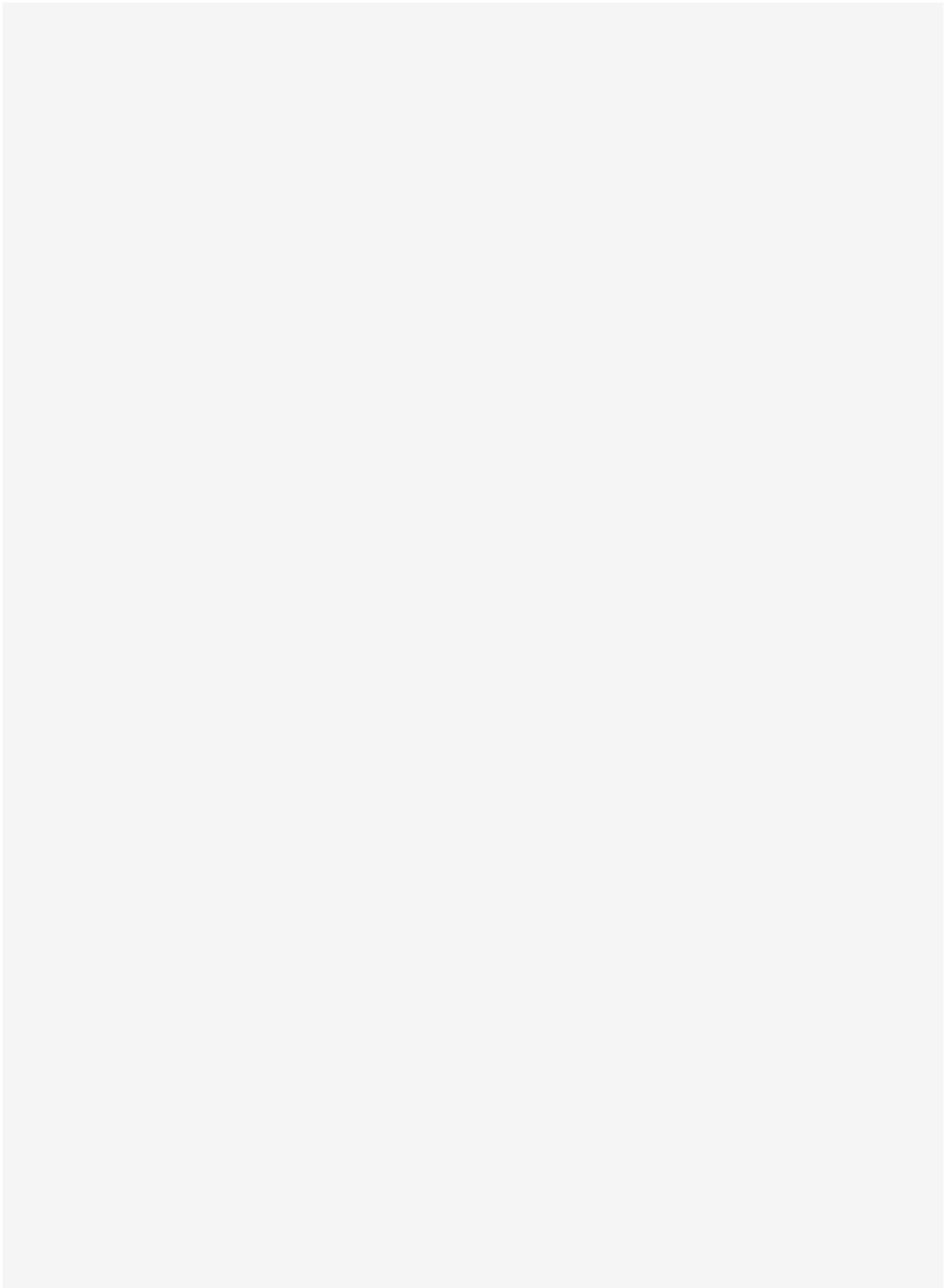


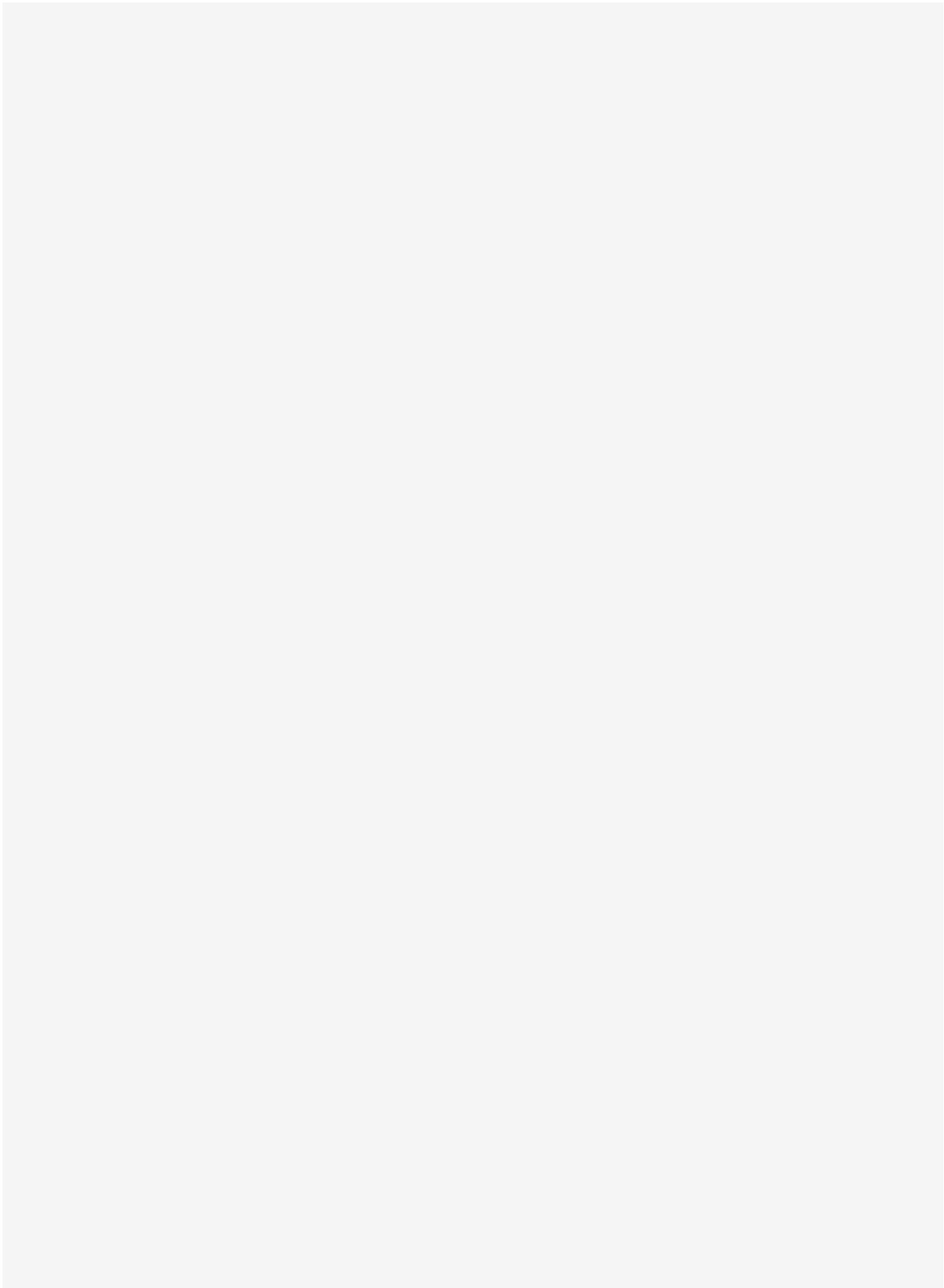
```
0097         if length(prop_index) == 1
0098             prop.(properties{prop_index}) = arg.(arg_fn{i});
0099         else
0100             error('Property ''%s'' does not exist or is ambiguous.', ...
0101                 arg_fn{i})
0102         end
0103     end
0104     arg_index = arg_index + 1;
0105 else
0106     error([
```

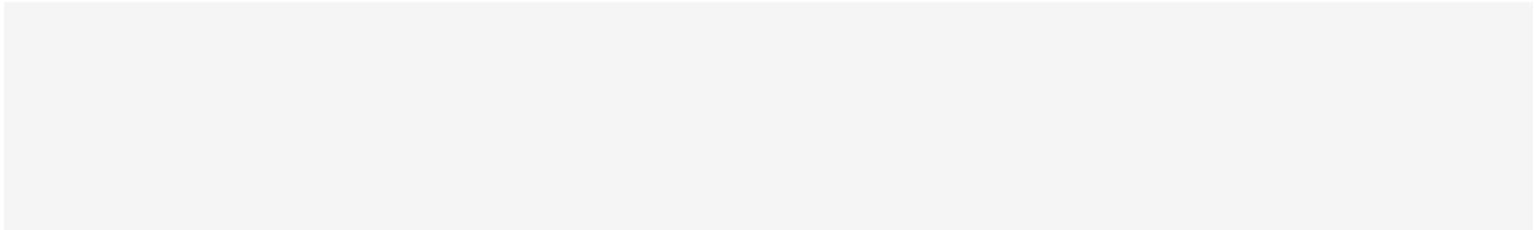



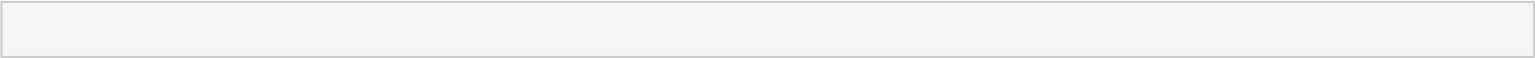


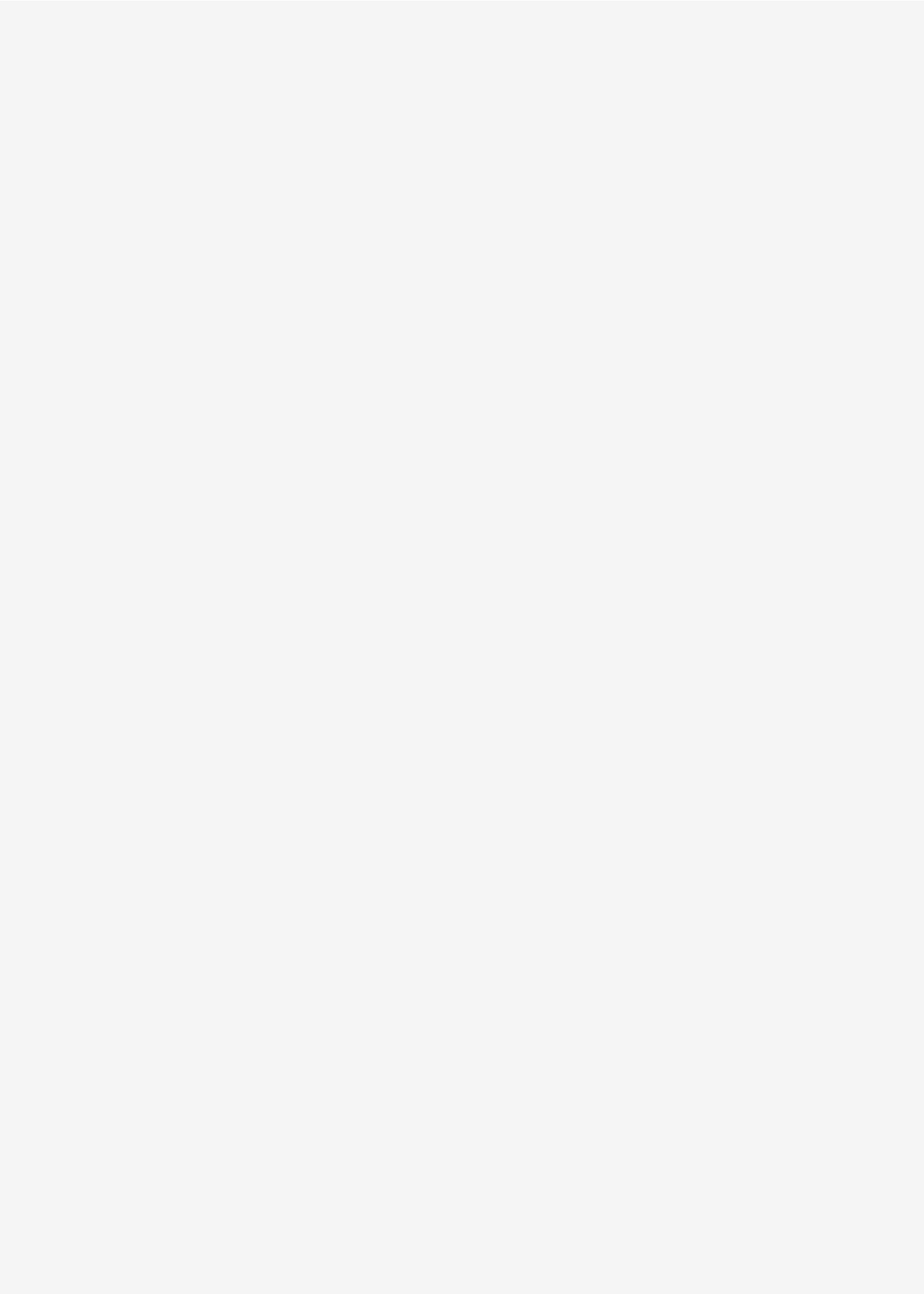








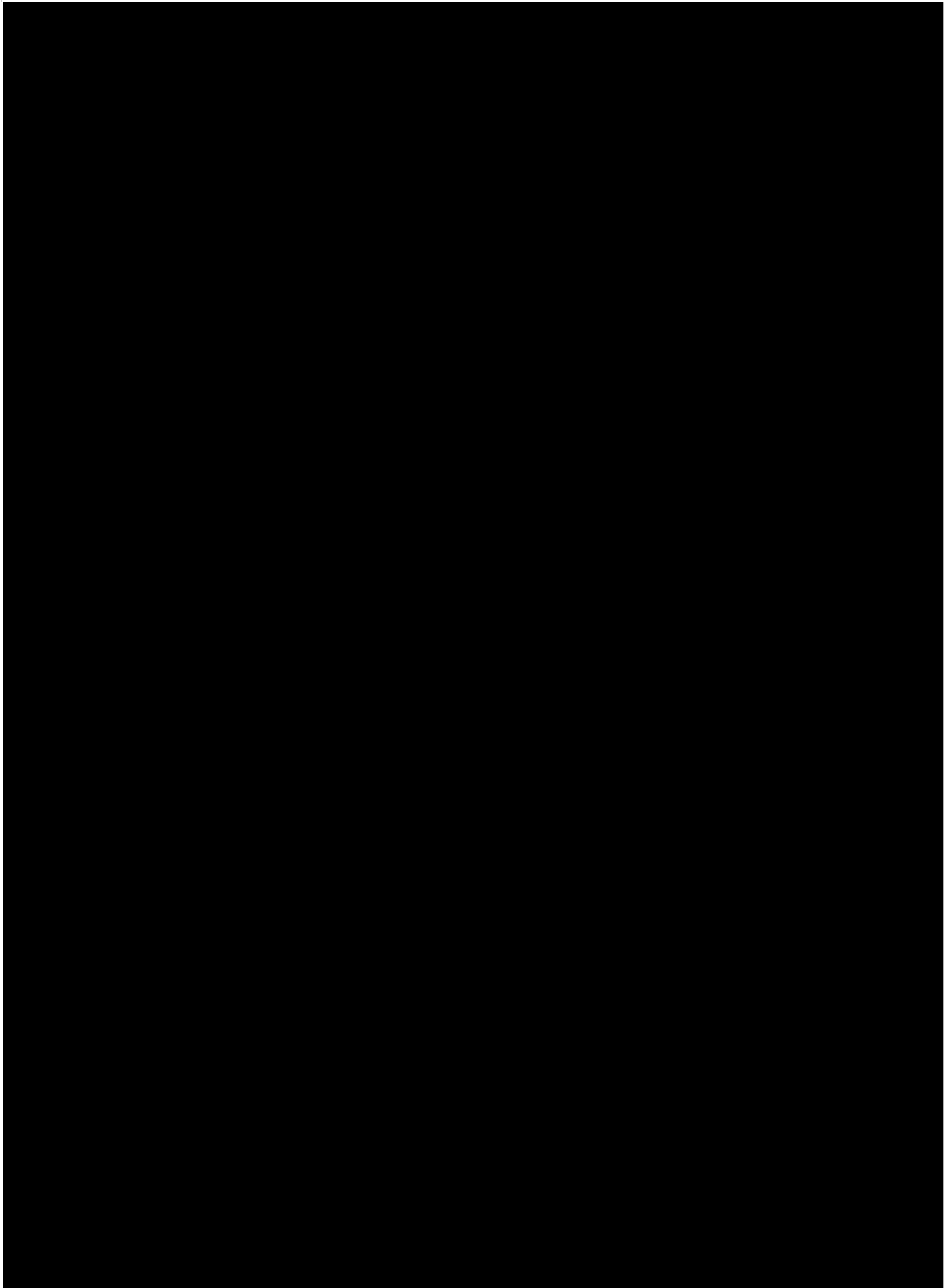


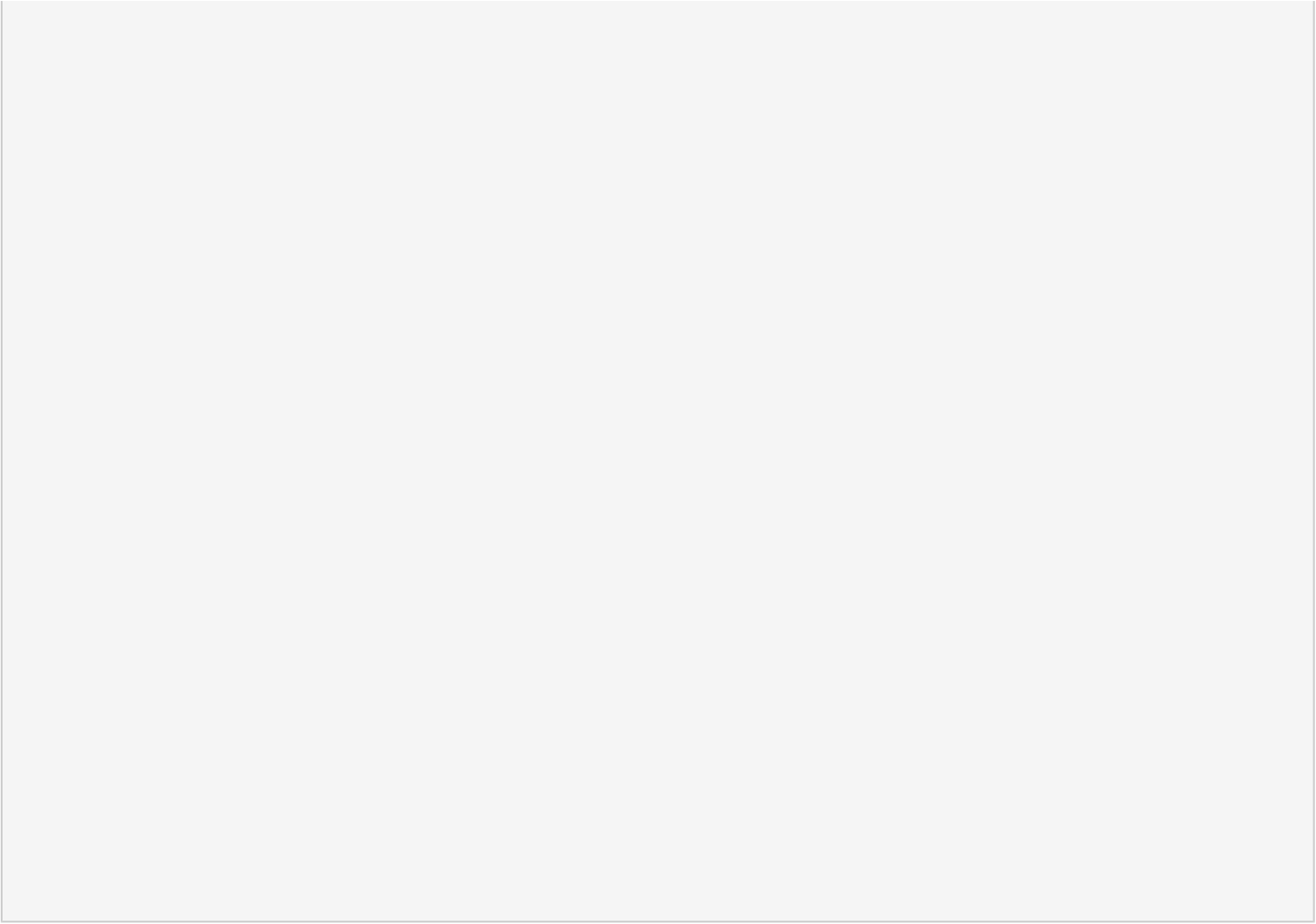


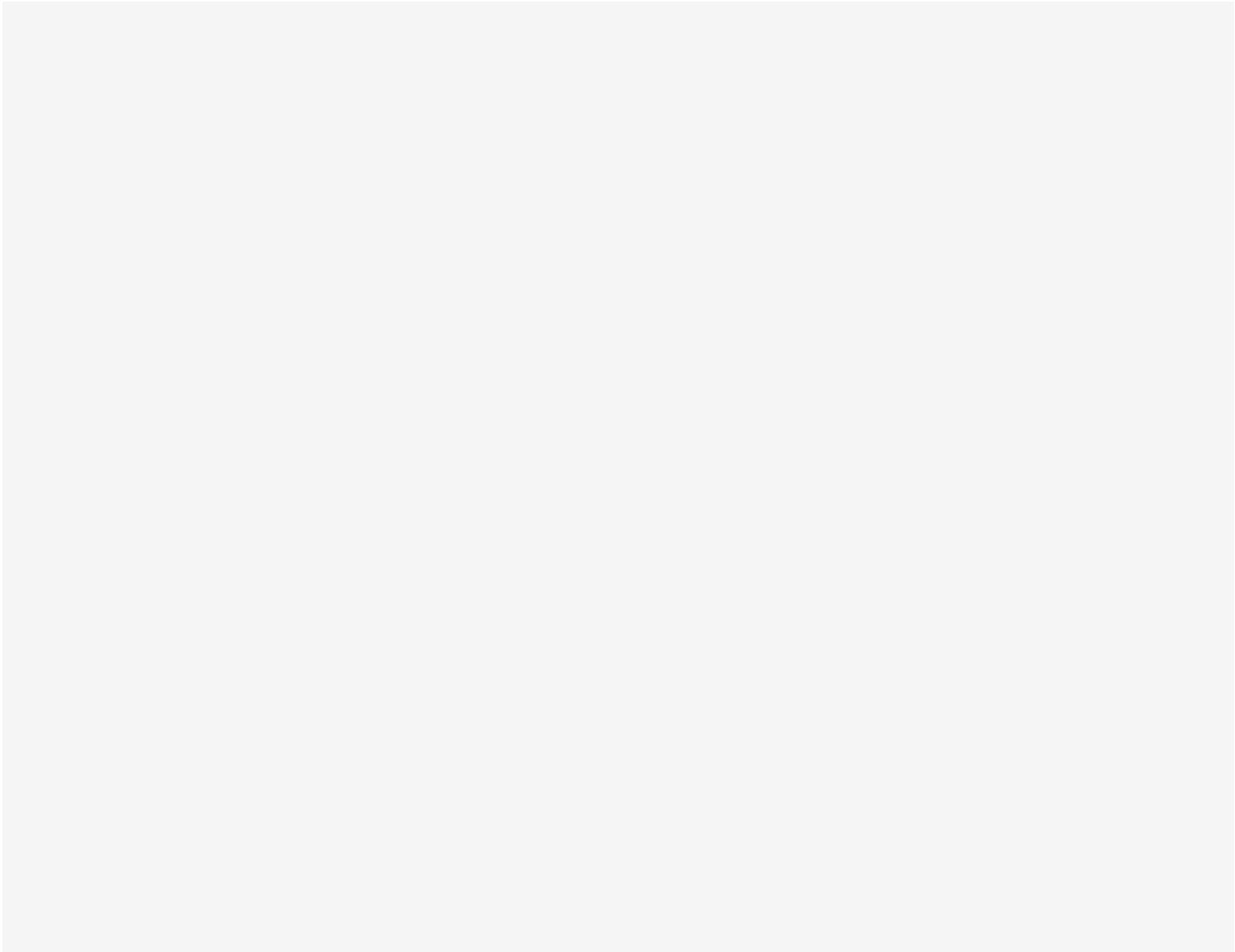
```

0318 if ~isnan(titlehighlightcolor)
0319     inset(2) = uicontrol( ...
0320         'Parent', htab, ...
0321         'Units', 'pixels', ...
0322         'BackgroundColor', titlehighlightcolor, ...
0323         'Style', 'text', ...
0324         'Visible', 'on');
0325     inset(3) = uicontrol( ...
0326         'Parent', htab, ...
0327         'Units', 'pixels', ...
0328         'BackgroundColor', 0.85*titlehighlightcolor, ...
0329         'Style', 'text', ...
0330         'Visible', 'on');
0331 end
0332
0333 TopBottomTabResizeCbk(status, []);
0334 set(status, 'ResizeFcn', @TopBottomTabResizeCbk);
0335 end
0336
0337 setappdata(htab, 'panels', panel);
0338 setappdata(htab, 'status', status);
0339 if ~isempty(createfcn)
0340     createfcn(htab, [], panel, status);
0341 end
0342
0343 if narginout, hh = htab; end
0344
0345
0346 %-----
0347 % Inner Functions for Popup 0
0348 %-----
0349 function Popup 0 Cbk(hobj, evdt, indx)
0350
0351     set(title, 'Value', 0);
0352     if indx>0
0353         indxvis = strcmp(get(panel(indx), 'Visible'), 'off');
0354         if indxvis
0355             set(panel(indx), 'Visible', 'on');
0356         else
0357             set(panel(indx), 'Visible', 'off');
0358         end
0359     end
0360     if isempty(strmatch('on', get(panel, 'Visible')))
0361         set(status, 'Visible', 'on');
0362     else
0363         set(status, 'Visible', 'off');
0364     end
0365
0366     currval = get(slider, 'Value');
0367     if currval<0
0368         set(slider, 'Value', 0);
0369         Popup 0 SliderCbk(slider, evdt);
0370     end
0371     titlepos = get(title(1), 'Position');
0372     addtolen = titlepos(2)+titlepos(4)+margins(4);
0373
0374     for k=1:nt0
0375         titlepos = get(title(k), 'Position');

```

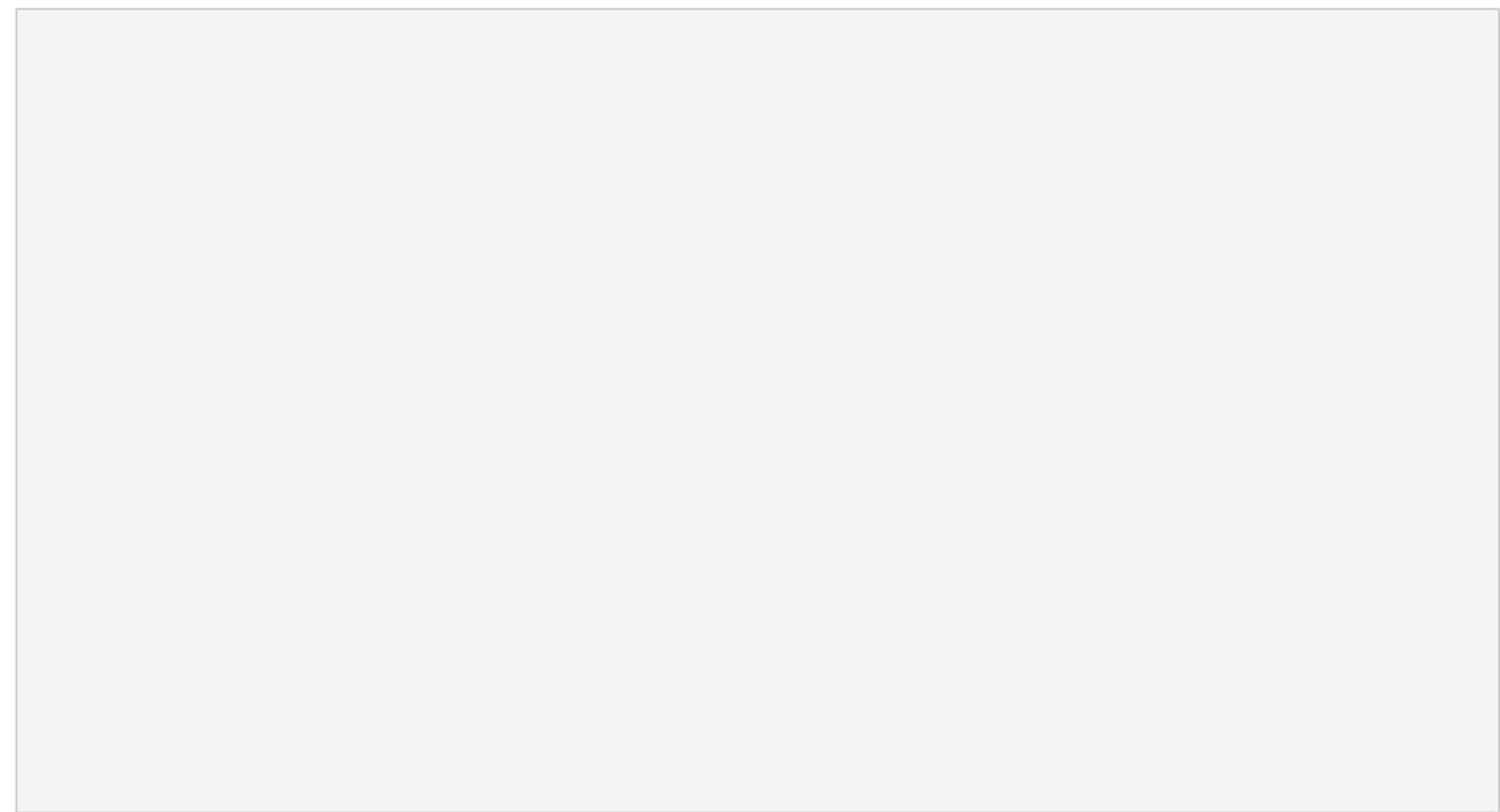
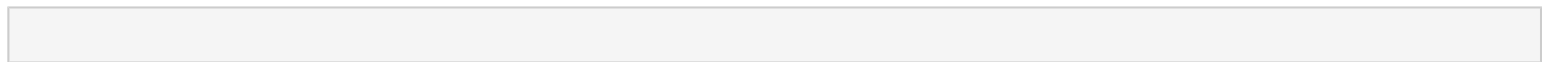
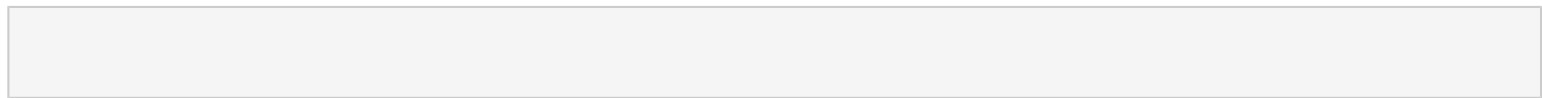








```
0287     mtable = get(hObject, 'userdata');
0288     jtable = mtable.getTable;
0289
0290     % Stop any current editing
0291     stopEditing(jtable);
0292
0293     % Insert the new row immediately before the current row
0294     newRowData = cell(1,mtable.getNumColumns); % empty data
0295     mtable.getTableModel.insertRow(max(0,jtable.getSelectedRow), newRowData);
0296 %catch
0297
```


[Home](#) >

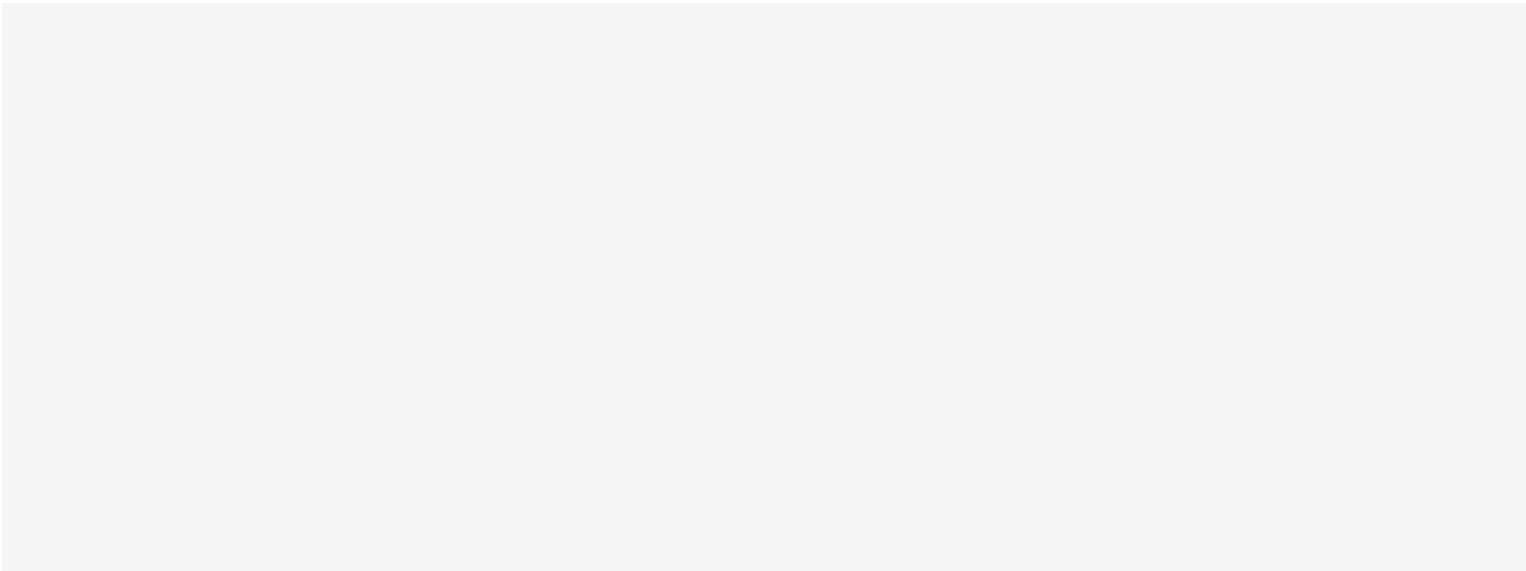


0058

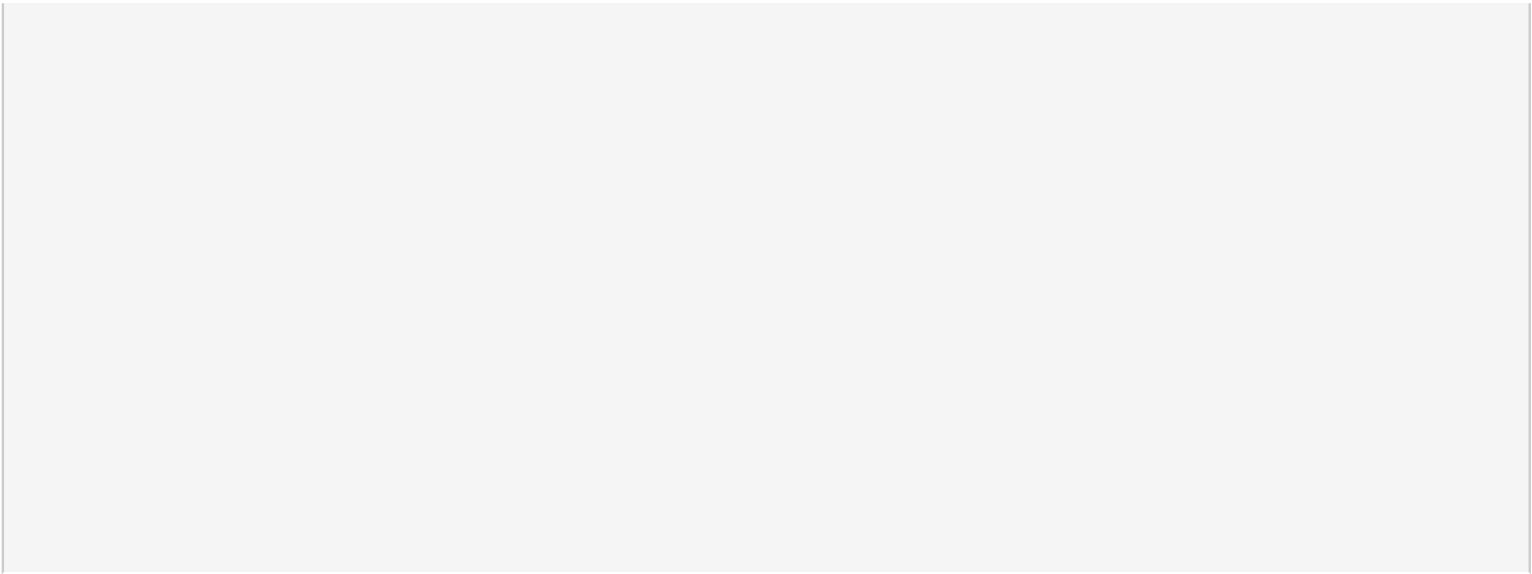

```

0009 % CALL:          classes = ltpda_classes()
0010 %
0011 % INPUTS:
0012 %
0013 % OUTPUTS:        classes - a cell array with a list of recognised LTPDA object
types
0014 %
0015 % VERSION:         $Id: ltpda_classes.m,v 1.1 2008/02/14 16:03:52 mauro Exp $
0016 %
0017 % HISTORY: 14-02-08 M Hueller
0018 %                Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022
0023 classes = {'ao',...
0024            'param',...
0025            'plist',...
0026            'cdata',...
0027            'tsdata',...
0028            'fsdata',...
0029            'xydata',...
0030            'xyzdata',...
0031            'specwin',...
0032            'mfir',...
0033            'miir',...
0034            'pzmodel',...
0035            'pole',...
0036            'zero',...
0037            'provenance',...
0038            'time',...
0039            'timeformat',...
0040            'timespan',...
0041            'history'};
0042
0043
0044

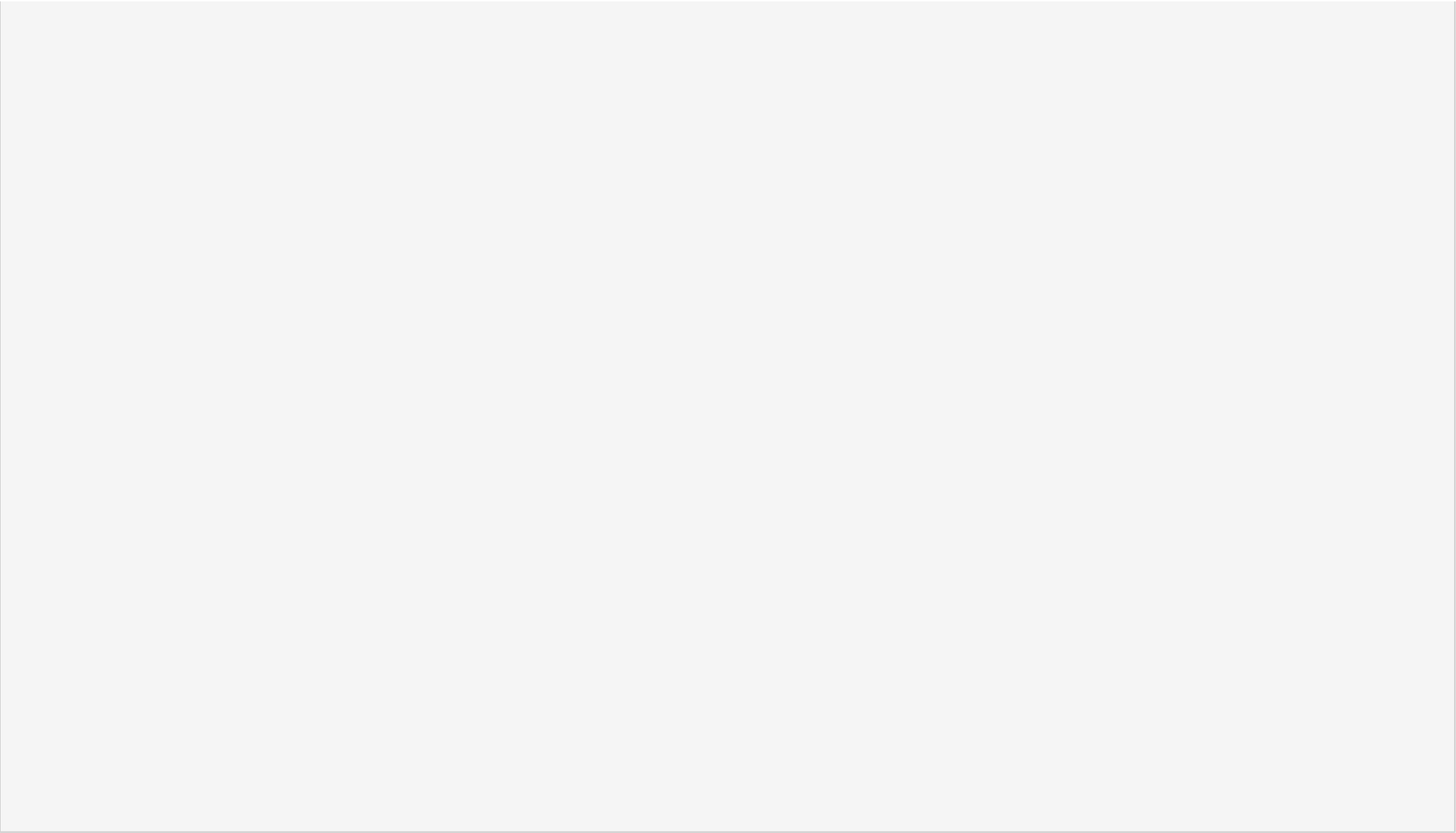
```




```
0091 plo = append(plo, param('Nolap', -1));
0092 plo = append(plo, param('Order', 0));
0093
0094 disp('* done. ');
0095
0096
0097
0098
```




```
0003 % PROGRESSBAR
0004 %
0005 %
0006 % $Id: progressBar.m,v 1.2 2008/03/03 09:11:02 hewitson Exp $
0007 %
0008
0009     close(findobj('Name','LTPDA Progress Bar'));
0010
0011
0012     funcBlocks =
find_in_models(bdroot,'LookUnderMasks','all','FunctionName','ltpdasim');
0013
0014     screenSize = get(0,
```



0008 %

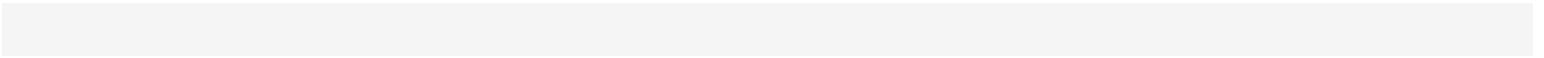
0009 % CALL: ltpda_constructor_helper

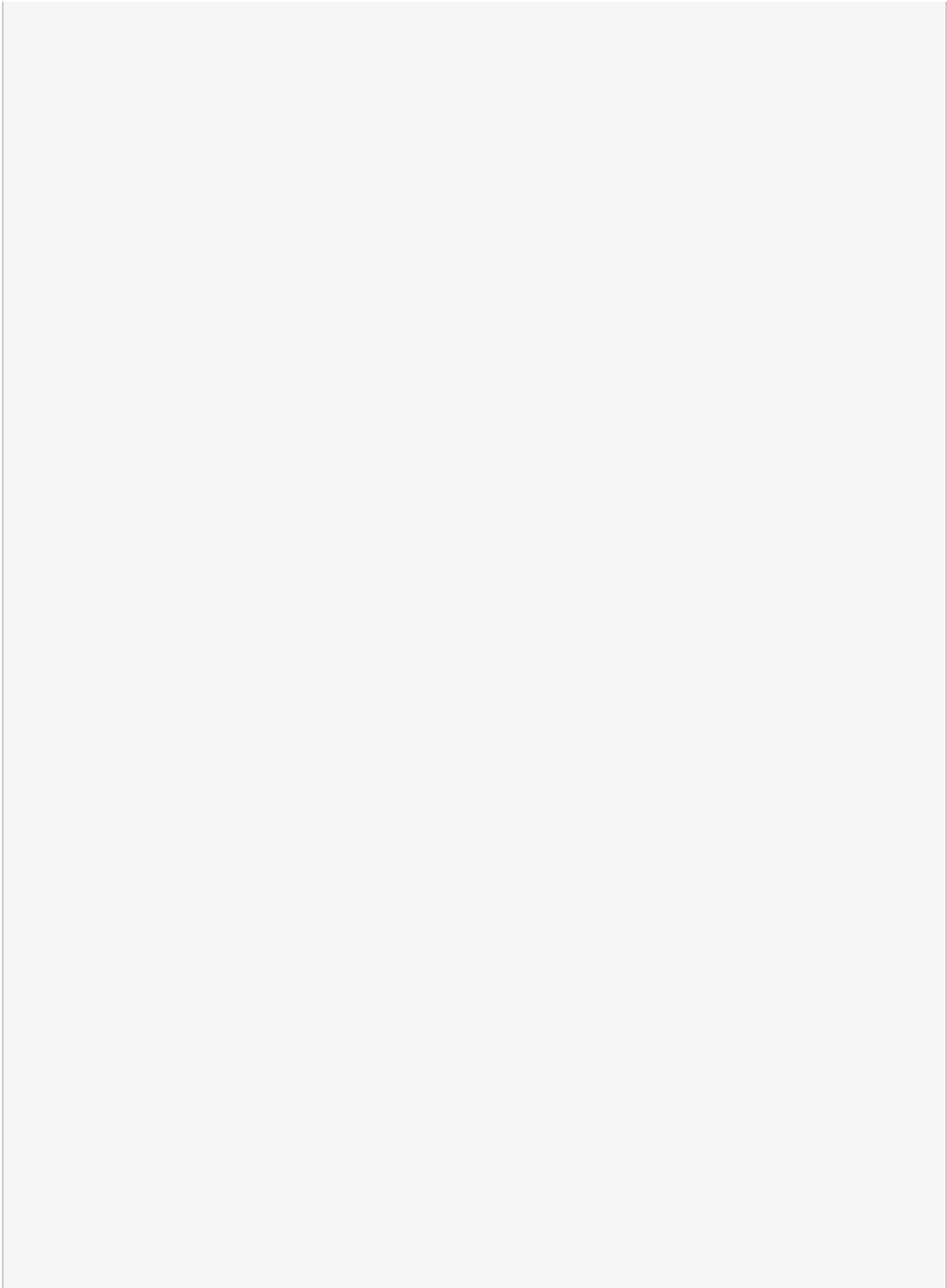
```
0065     'HorizontalAlignment', 'left', ...
0066     'BackgroundColor', Gproperties.Gcol, ...
0067     'Position',[10 Gproperties.Gheight-40 50 25]);
0068
0069 % pop-up dialog8 m70 pmh = uipttrol(mainfig,0 rg T* (0067     )TjStyle 0 rg ('Posi

0069 mainfig,0 rg T* (0067     )TjStyle 0 rg ('Position0 rg T* (0067     )Tjtext 0 rg ('Pos

0069
```

```
0123 pbh = uicontrol(mainfig, 'Style', 'pushbutton', 'String', 'Build', ...
0124     'Position', [360 Gproperties.Gheight-260 60 25], ...
0125     'Tag', 'LTPDA_constructor_helper_buildBtn');
0126 set(pbh, 'Callback', {'ltpda_constructor_helper_build'});
0127
0128 %% Start the GUI
0129
0130 % Make the GUI visible.
0131 set(mainfig, 'Visible', 'on')
0132
0133
0134 %% Callbacks
0135
0136 %--- Close
0137 function ltpda_constructor_helper_close(varargin)
0138     disp('* Goodbye from the LTPDA Constructor Helper *')
0139     delete(varargin{end})
```

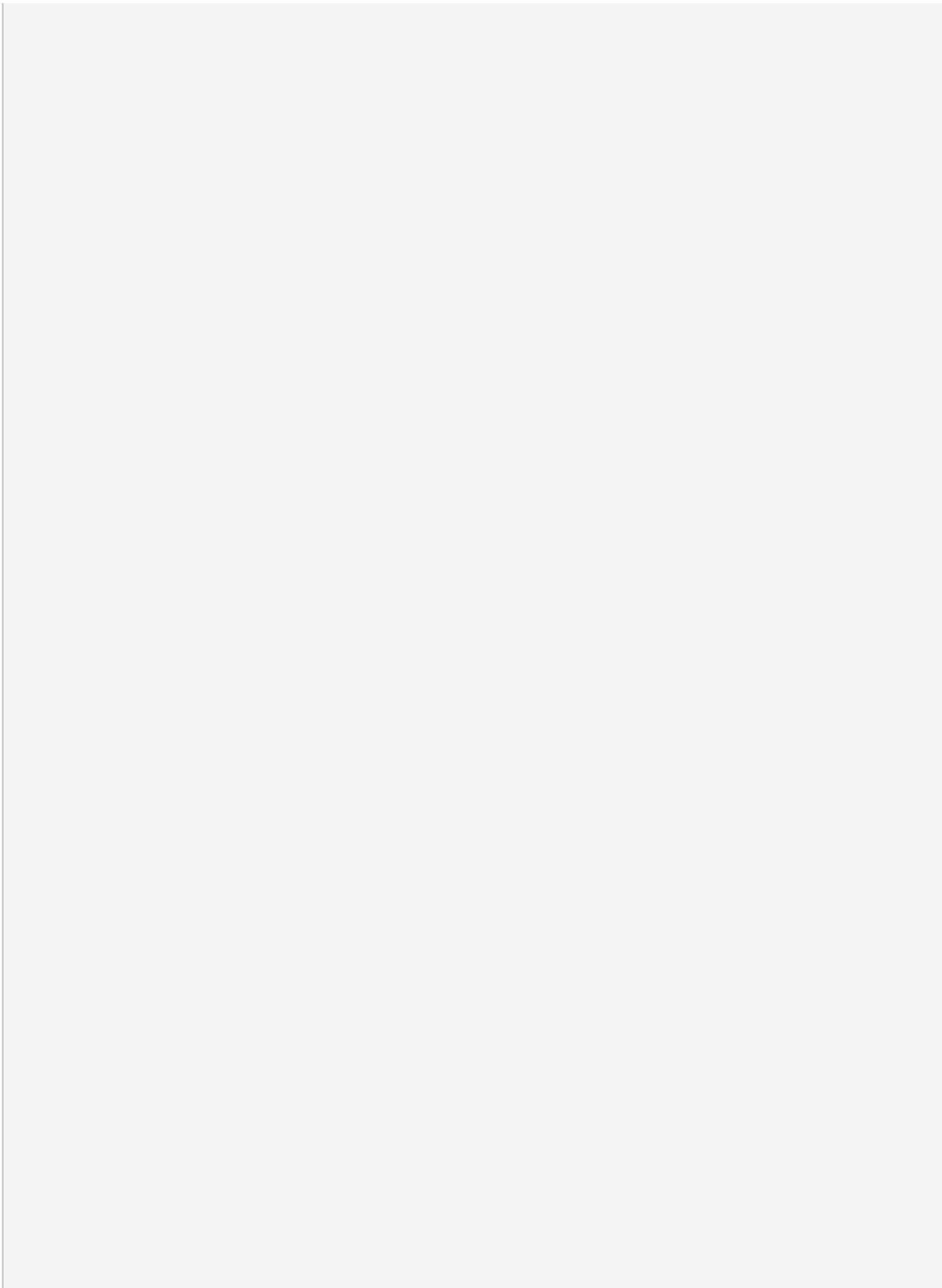


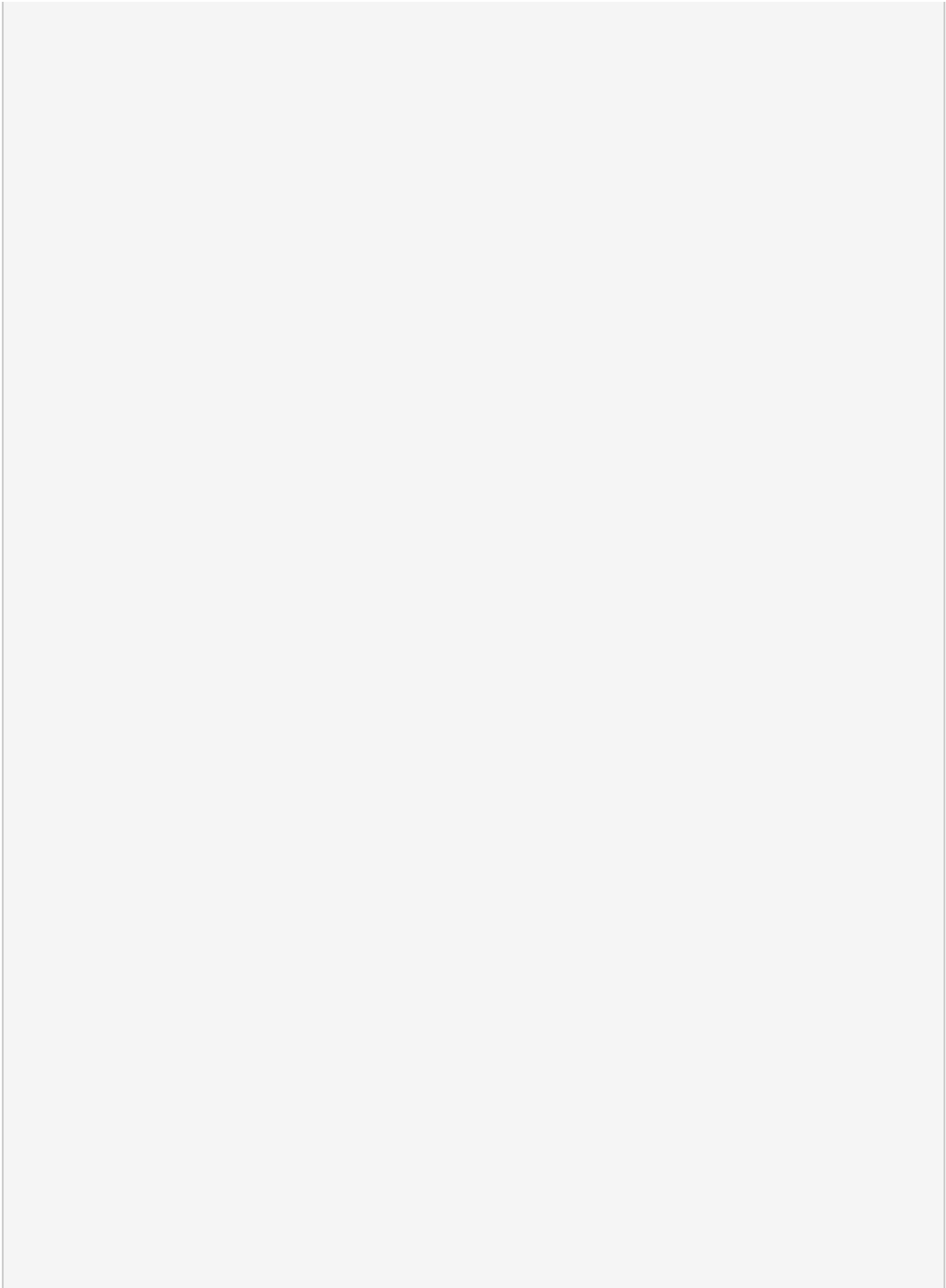
```
0130 %   if get(zeros, 'q') > 0.5
0131 %       [ai,bi] = cz2iir(zero, fs);
0132 %   else
0133 %       [ai,bi] = rz2iir(zero, fs);
0134
```



```
0018    pl = getappdata(mainfig, 'plist');
0019 else
0020     eval(sprintf('pl = %s(%s, 'Params', '%s');', cl, cl, set));
0021 end
0022
0023 ltpda_constructor_helper_buildParams(mainfig, pl);
0024
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003





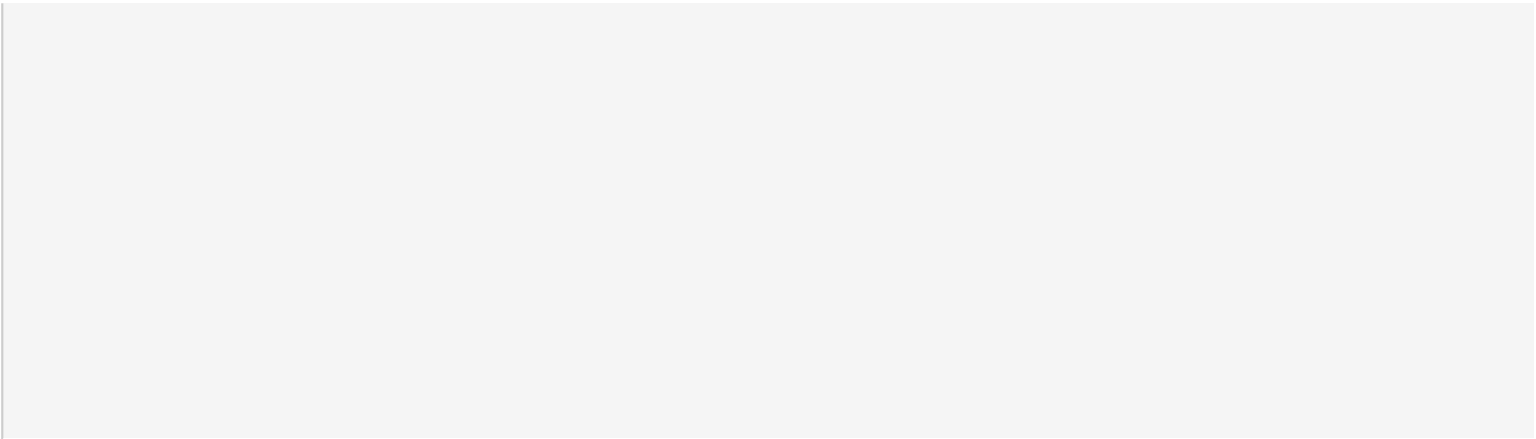

```
0467  
0468 if isempty(f1s)  
0469     f1 = getlowerFreq(pzm)/10;  
0470 else  
0471     f1 = str2double(f1s);  
0472 end  
0473 if isempty(f2s)  
0474     f2 = getupperFreq(pzm)*10;  
0475 else  
0476     f2 = str2double(f2s);  
0477 end  
0478 if isempty(nfs)  
0479     nf = 1000;  
0480 else  
0481     nf = str2double(nfs);  
0482 end  
0483  
0484 % make model response  
0485 hobj = obj('opndptw3.57,1e 0 arsgre {semx6BBd1e 0 1 rg 1 r8 myrB3d.y 0 1 rg 1 r9* (048
```



```

0580
0581 % update model
0582 updateModelResp(handles);
0583
0584
0585 % --- Executes on button press in deleteZerosBtn.
0586 function deleteZerosBtn_Callback(hObject, eventdata, handles)
0587 % hObject      handle to deleteZerosBtn (see GCBO)
0588 % eventdata    reserved - to be defined in a future version of MATLAB
0589 % handles      structure with handles and user data (see GUIDATA)
0590
0591 % get selection from list
0592 values = get(handles.zeroList, 'Value');
0593 % get zeros
0594 zeros = getappdata(handles.main, 'zeros');
0595 zo = [];
0596 for j=1:length(zeros)
0597     if j~=values
0598         zo = [zo zeros(j)];
0599     end
0600 end
0601
0602 setappdata(handles.main, 'zeros', zo);
0603 updateZerosList(handles);
0604
0605 % update model
0606 updateModelResp(handles);
0607
0608
0609 % --- Executes on button press in saveA0btn.
0610 function saveA0btn_Callback(hObject, eventdata, handles)
0611 % hObject      handle to saveA0btn (see GCBO)
0612 % eventdata    reserved - to be defined in a future version of MATLAB
0613 % handles      structure with handles and user data (see GUIDATA)
0614
0615 % get response
0616 gain = str2double(get(handles.gainEdit, 'String'));
0617 poles = getappdata(handles.main, 'poles');
0618 zeros = getappdata(handles.main, 'zeros');
0619 pzm = pzmodel(gain, poles, zeros);
0620
0621 f1s = get(handles.f1Edit, 'String');
0622 f2s = get(handles.f2Edit, 'String');
0623 nfes.f1E3/2m3 r fs.f1Edit, 'String');
06f1EditlowerFreq(mai)/10es.f1E3/2m3 8(j)];
06f1Ej 0.133 0.54061= es.f1E3/2m33s(j)];
0599 02f1EditupperFreq(mai)*10es.f1E3/2m333(j)];
02f1Ej 0.133 0.54021= es.f1E3/2m335(j)];
0599 nff1E1000es.f1 Td (ET Q BT /Artifa>BDctiAttach 15[/Bo0 0m ]/Type /Pagina0 )Tj/B

```

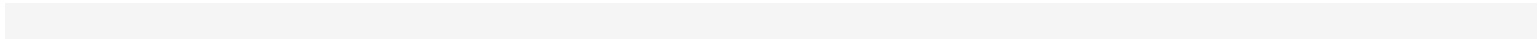
[Home](#)> [@specwin](#) > [win_BH92.m](#)

WIN_BH92 return BH92 window samplect N input z.

```
function w = win_BH92(w, z)
```

This function calls:

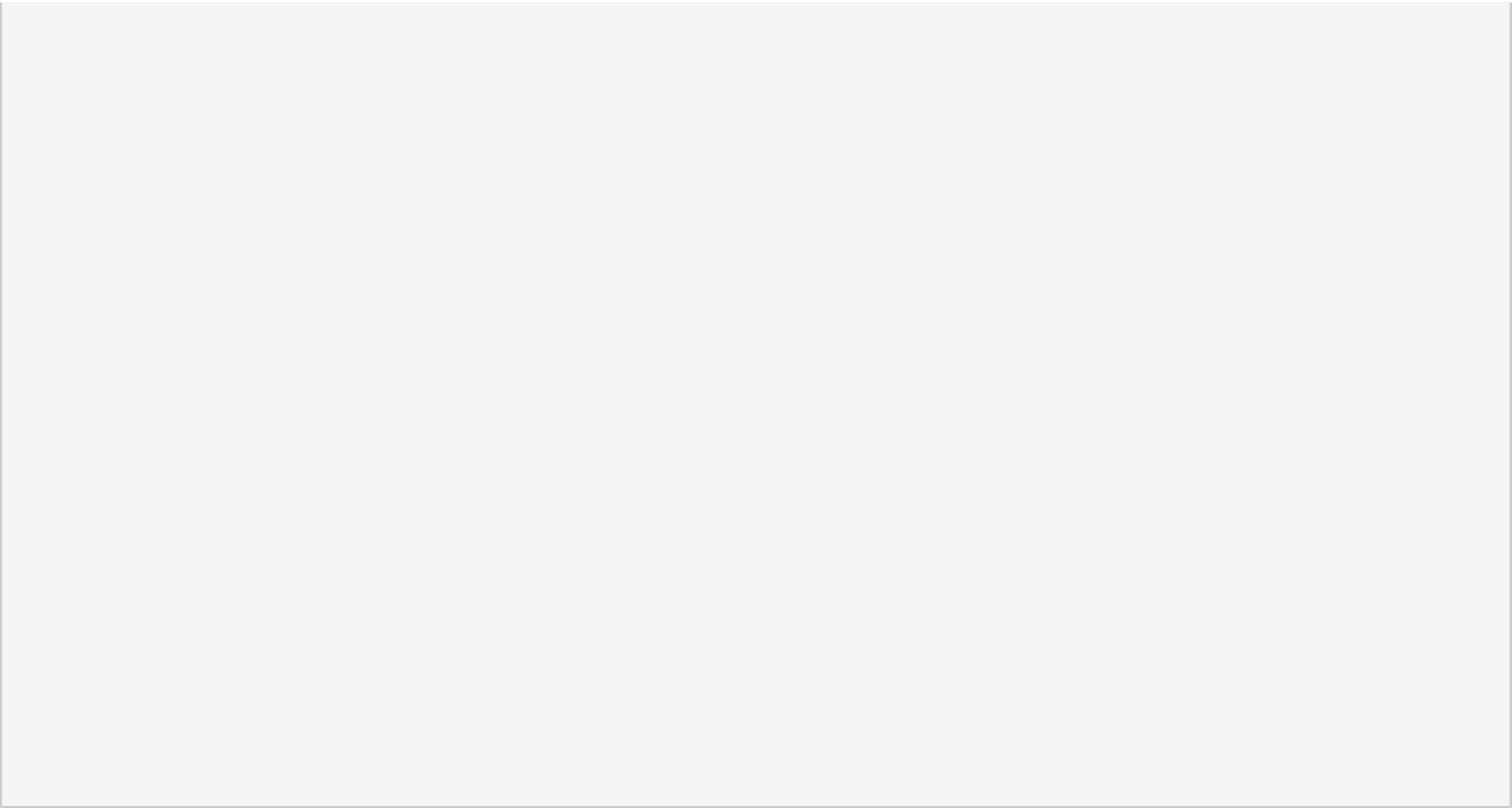
```
function w = win_BH92(w, z)
```



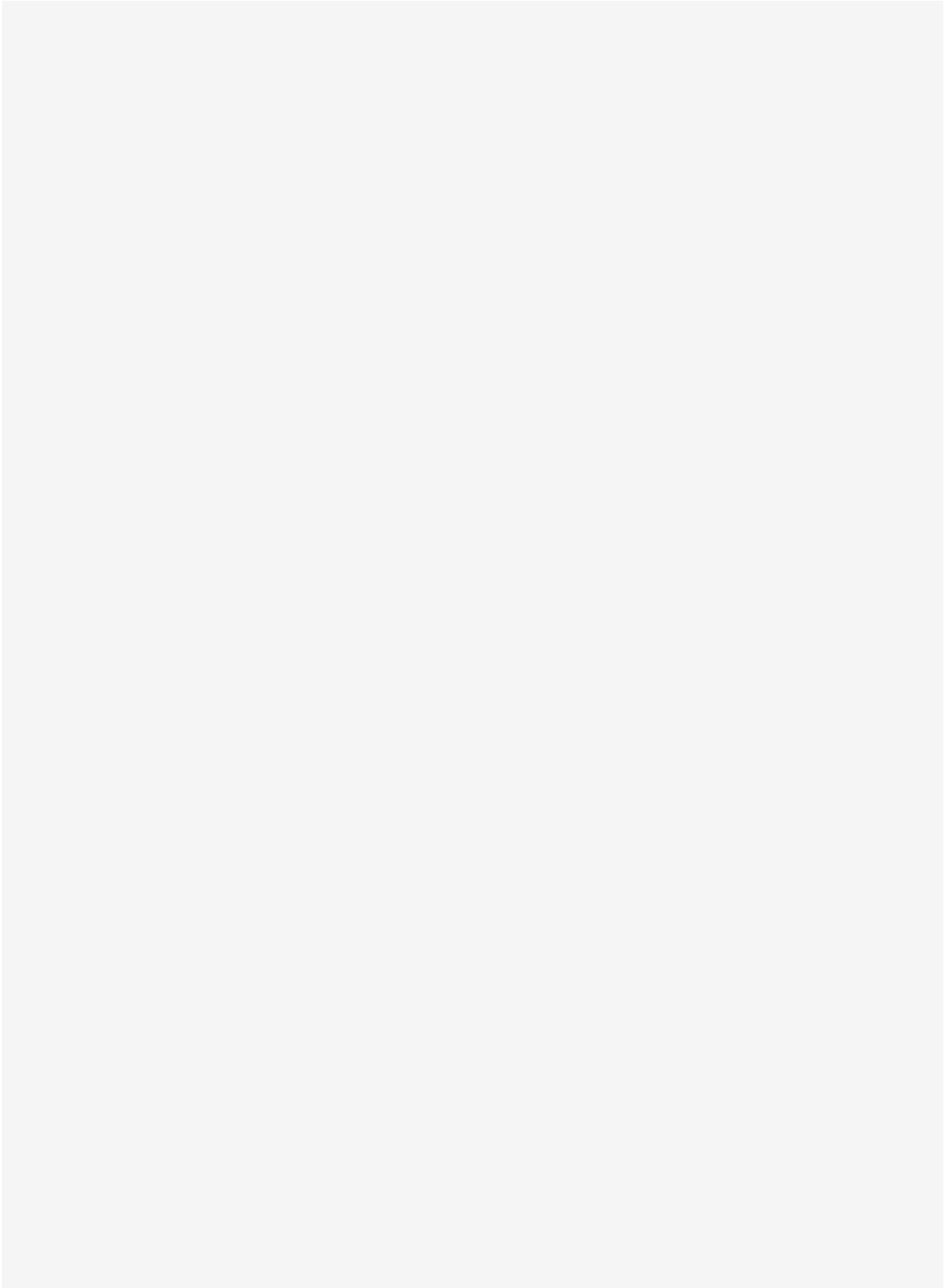
--

--

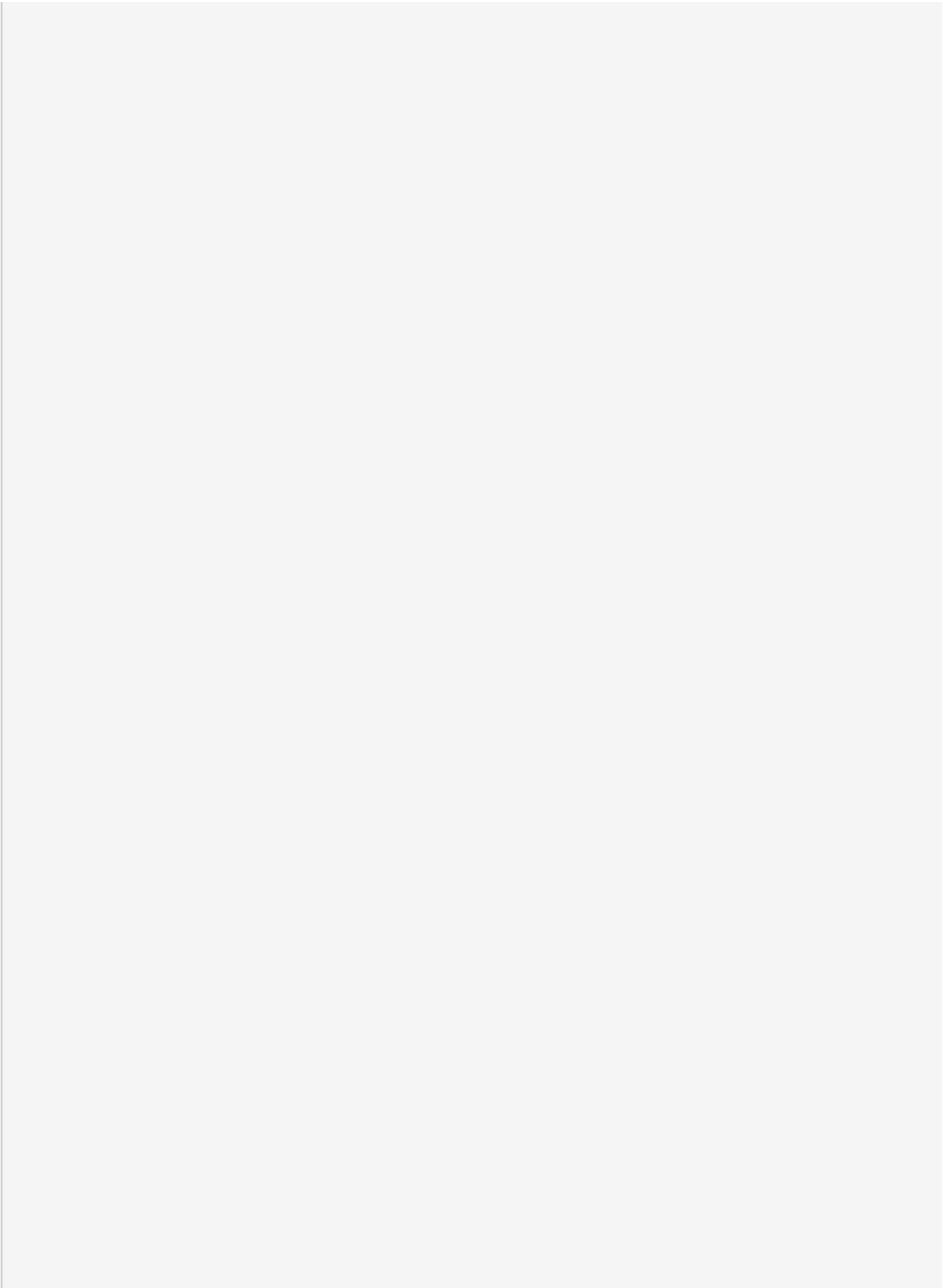
--



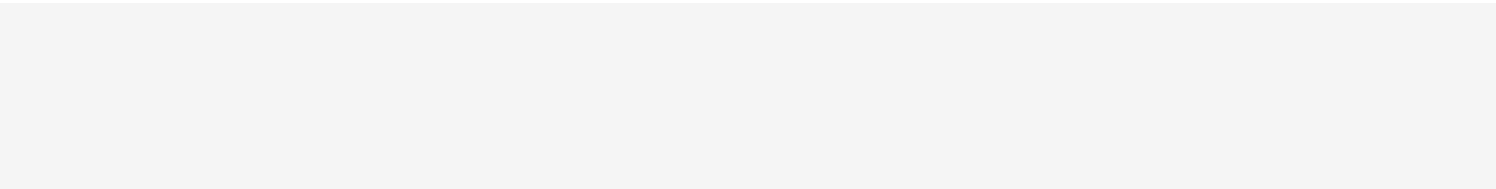




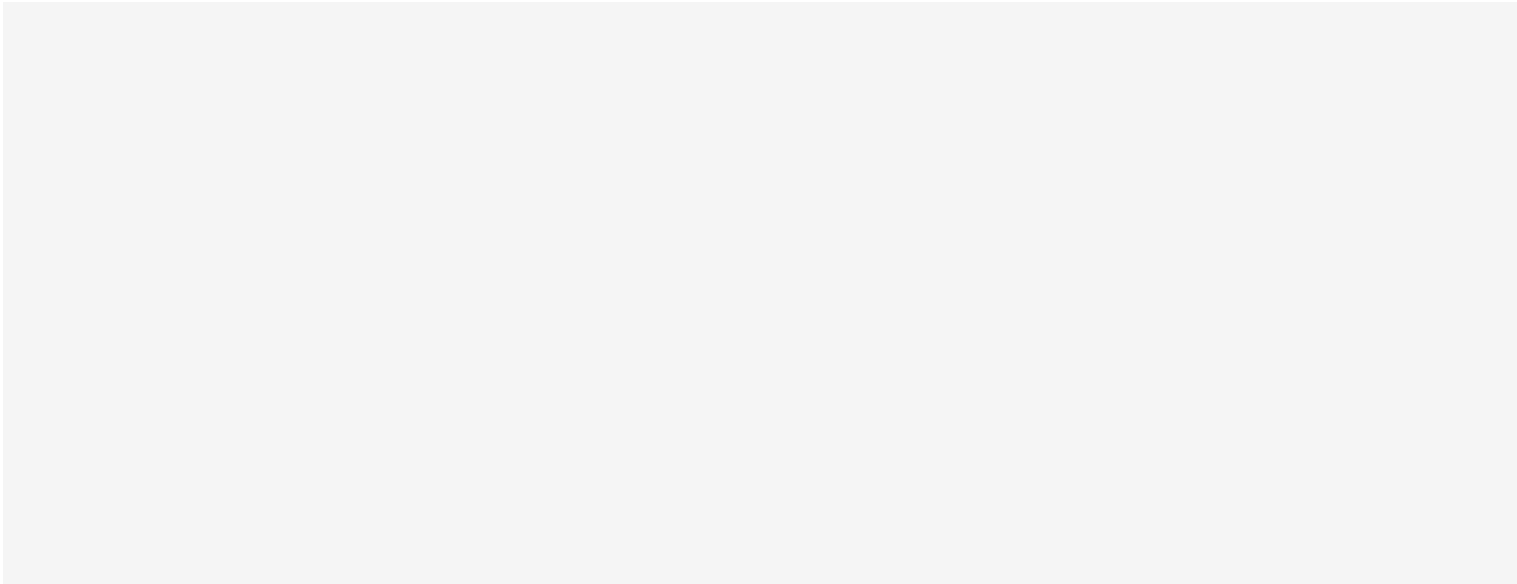
```
0068  'Title',{ 'Connection', 'Submit', 'Query', 'Retrieve'},...%, 'Quick Look'},...
0069  'HorizontalAlignment', 'l1t6985rg (,)Tj 0.698 0 0 M5rg (,)Tj_ 0 r700 -1.2 TD
```


```
0572     'BackgroundColor', Gproperties.Gcol,...
0573     'HorizontalAlignment', 'left',...
0574     'Position',[Sleft Tbottom Swidth Sheight]);
0575
0576 Extra    = 20;
0577 Theight = ALLHEIGHT + Extra;
0578 Tbottom = Tbottom - Extra;
0579 sth = uicontrol(panel, 'Style', 'edit',...
0580     'String', '',...
0581     'FontSize', fontsize,...
0582     'BackgroundColor', 'w',...
0583     'Max', 1,...
058x Ca'Hor85ontalAlignment', 'left'
058x Ca'Hor86ontalAlignment',,
```


```
0014     disp('usage: [G,ri]= ltpda_fq2ri(f0, Q)')
0015     return
0016 end
0017
0018 % a = Q - 0.5 ;
0019 % re = 2*pi*f0 / sqrt(1+tan(a)^2);
0020 % im = re * tan(a);
0021
0022 w0 = 2*pi*f0;
0023 re = -w0/(2*Q);
0024 im = w0*sqrt(4*Q*Q-1)/(2*Q);
0025 G = 1/(w0*w0);
0026 ri = complex(re,im);
0027
0028 % END
```


```
0013
0014 if(nargin==0)
0015     disp('usage: ri = fq2ri2(f0, Q)')
0016     return
0017 elseif(nargin==1)
```


ltpda_funchash

PURPOSE

LTPDA_FUNCHASH compute MD5 hash of an LTPDA function.

SYNOPSIS

```
function h = ltpda_funchash(fcnnam
```

DESCRIPTION

```
LTPDA_FUNCHASH compute MD5 hash of an LTPDA function.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DESCRIPTION: LTPDA_FUNCHASH compute MD5 hash of an LTPDA function.

CALL:      h = ltpda_funchash(fcnnam

INPUTS:    fcnnam - An LTPDA function name

OUTPUTS:   h - the hash string

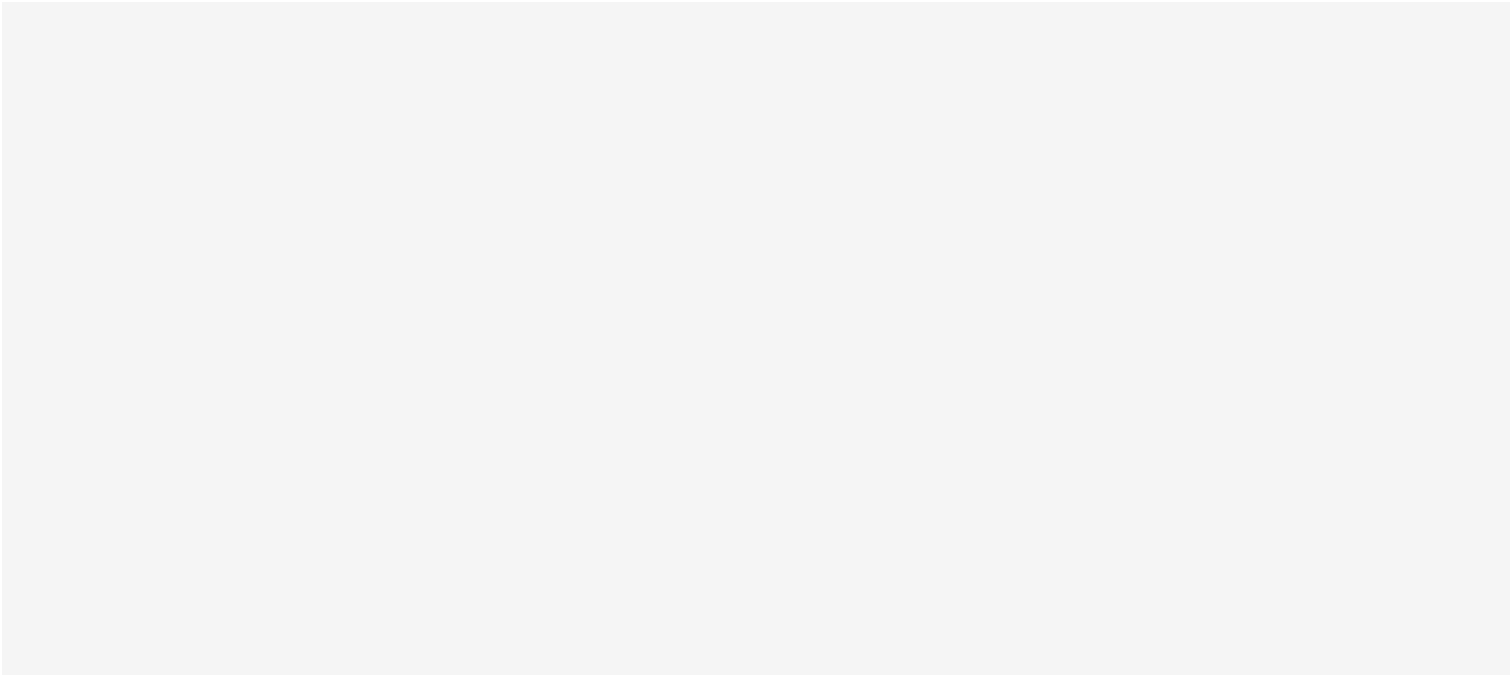
PARAMETERS: None.

EXAMPLES:

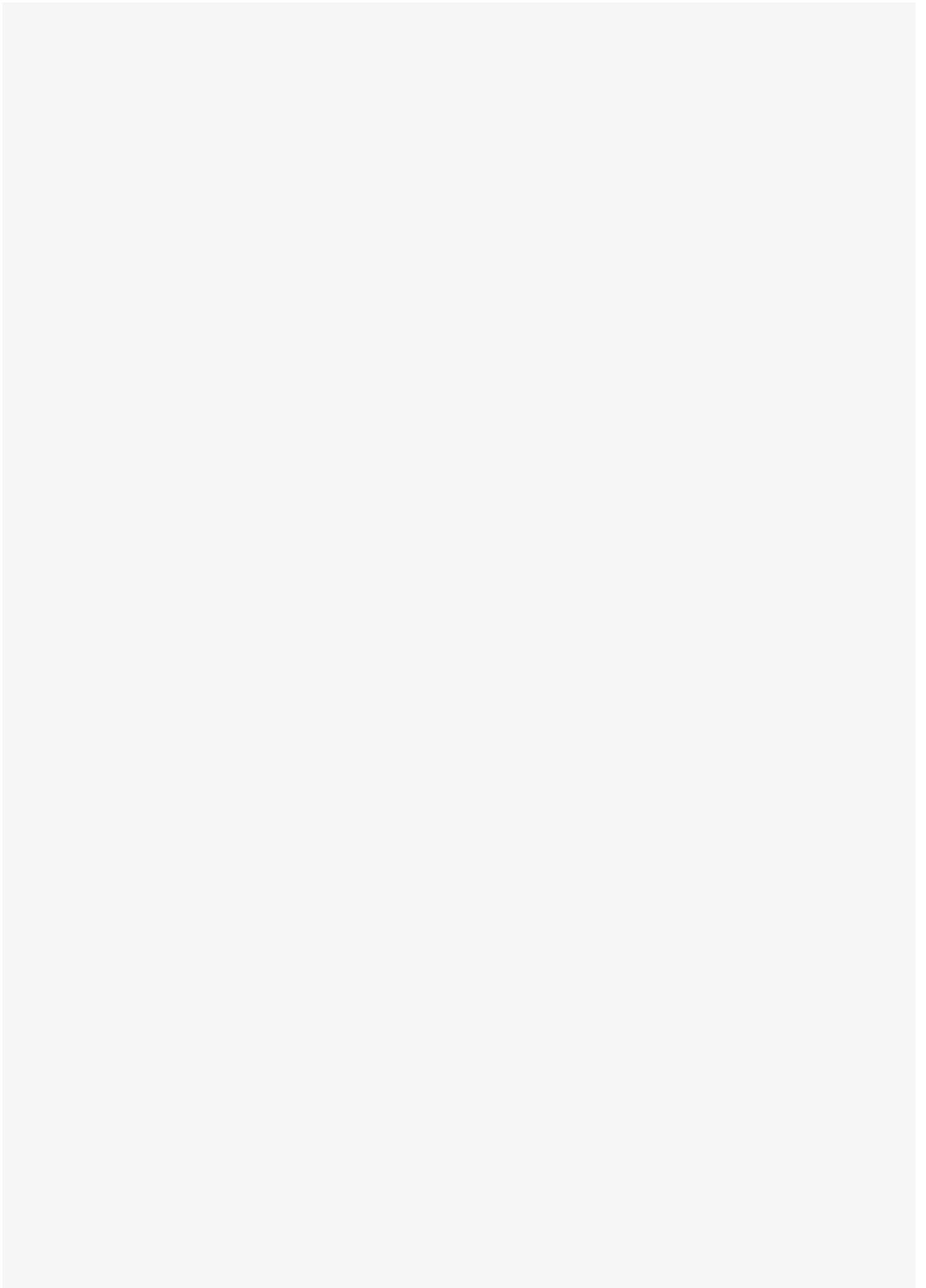
    >> h = ltpda_funchash('ao');

VERSION:    $Id:ash.m

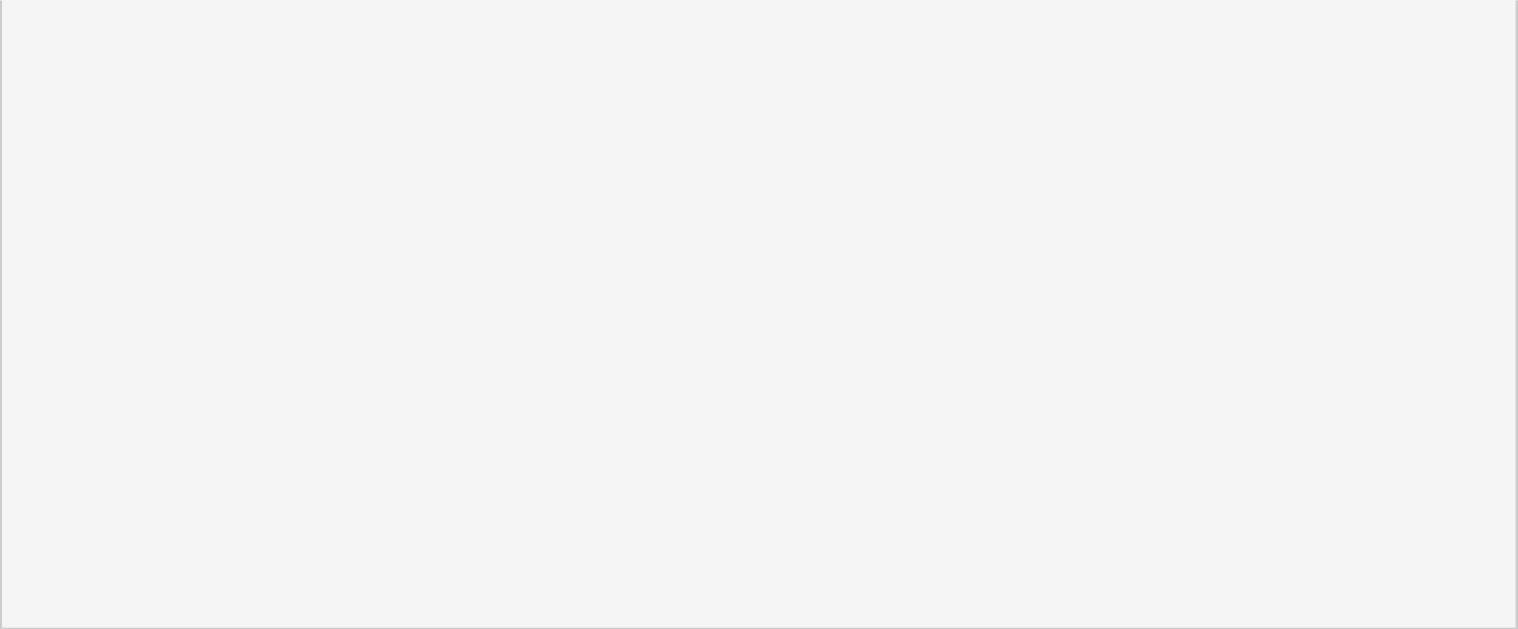
HISTE43uRSIOc 0.545 0.1339o2.8S8809 l h f Q q 1 0 0752>>BDC 1 0 0 RG 102.1 5P593.69
```

--

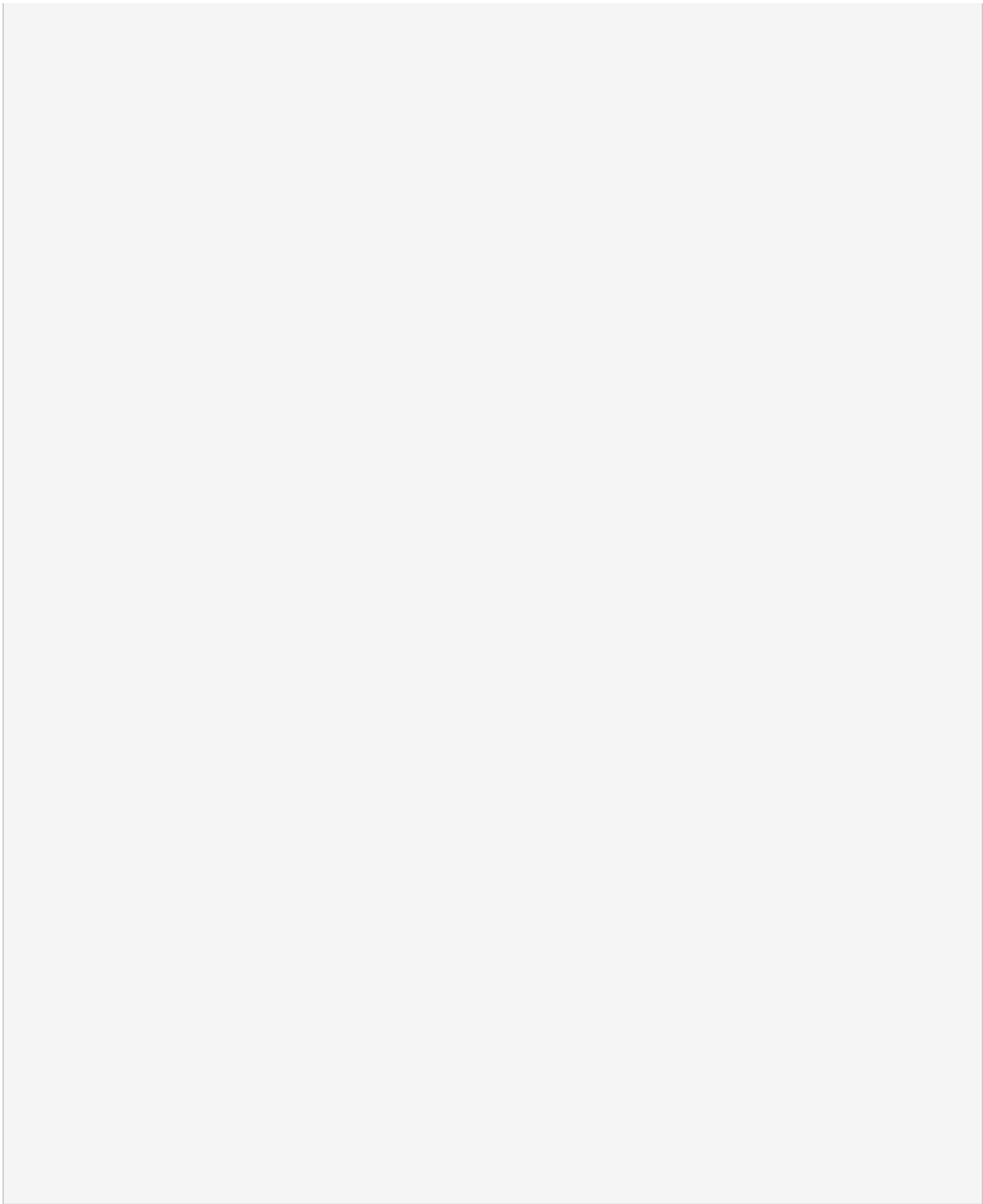


```
0084 h=typecast(x.digest,'uint8');
0085 h=dec2hex(h)';
0086 if(size(h,1))==1 % remote possibility: all hash bytes < 128, so pad:
0087     h=[repmat('0',[1 size(h,2)]);h];
0088 end
0089 h=lower(h(:)');
0090 clear x
0091 return
```


```
0013      0.043097 * cos (4 * z));  
0014  
0015 % Make window struct
```





PURPOSE

SYNOPSIS

```
0013      0.0350665 * cos (4 * z));
0014
0015 % Make window struct
0016 w.name      = 'HFT95';
0017 w.alpha     = 0;
0018 w.psll      = 95;
0019 w.rov       = 75.6;
0020 w.nenbw     = 3
```



```

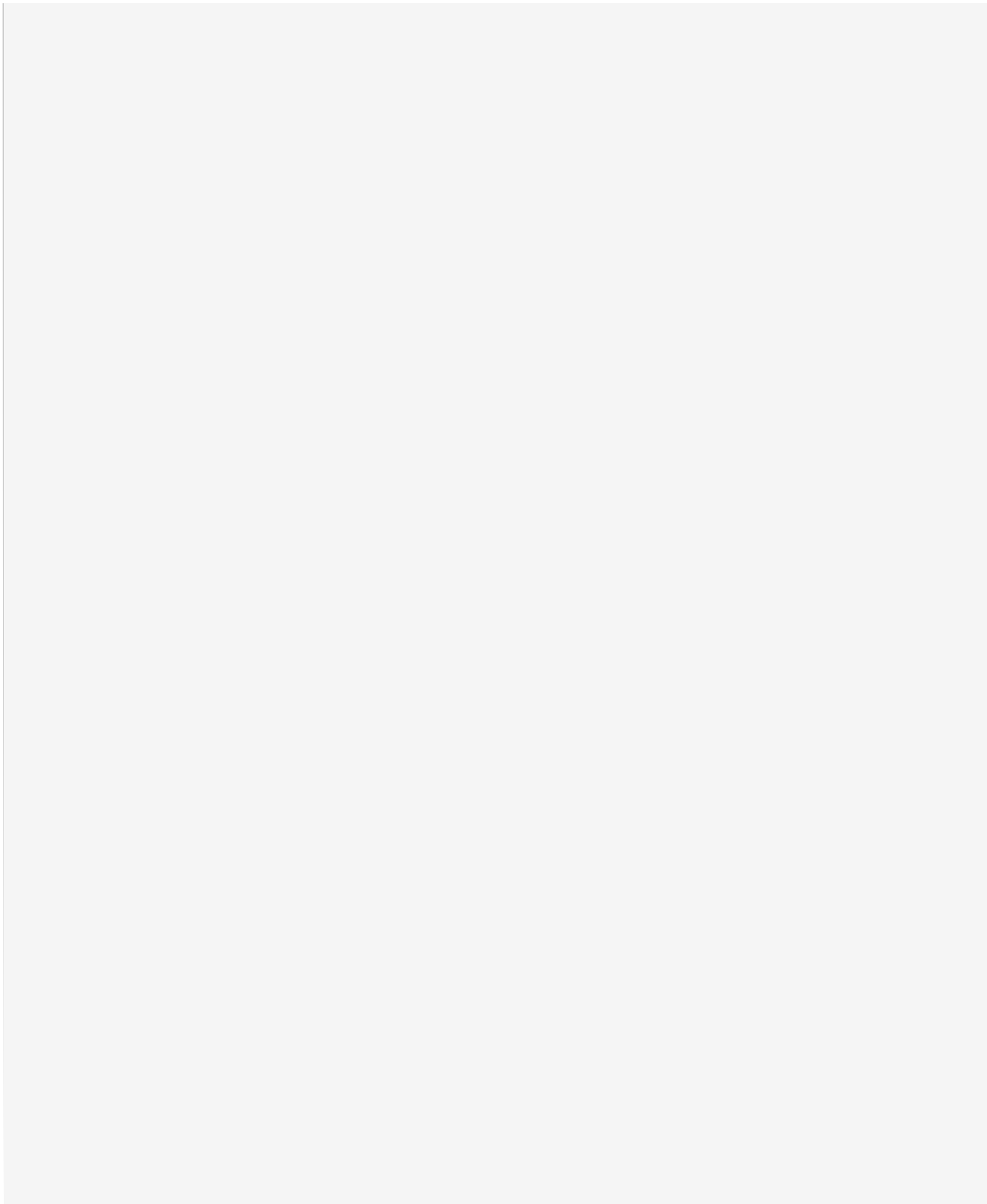
0009 % repository specified by the input database connection.
0010 %
0011 % CALL: p = retlieve(conn, ids)
0012 %
0013 %
0014 %
0015 % VERSION: $Id: retlieve.m,v 1.1 2007/08/31 17:40:08 hewitson Exp $
0016 %
0017 % HISTORY: 30-08-07 M Hewitson
0018 % Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022 type = 'cdata';
0023
0024 %%%%%%%%% Get each object
0025 objs = [];
0026 for j=1:length(ids)
0027     % Get object
0028     xdoc = mysql_getXdoc(conn, ids(j));
0029     % get child
0030     obj = [];
0031     c = xdoc.getChildNodes;
0032     for k = 1:c.getLength
0033         cc = c.item(k-1);
0034         if strcmp(char(cc.getNodeName), type)
0035             ccc = cc.getChildNodes;
0036             for l=1:ccc.getLength
0037                 cccc = ccc.item(l-1);
0038                 if strcmp(char(cccc.getNodeName), 'Object')
0039                     ccccc = cccc.getChildNodes;
0040                     for m=1:ccccc.getLength
0041                         cccccc = ccccc.item(m-1);
0042                         if strcmp(char(ccccccc.getNodeName), type)
0043                             obj = fromxml(ccccccc);
0044                         end
0045                     end
0046                 end
0047             end
0048         end
0049     end
0050
0051     objs = [objs obj];
0052 end
0053
0054
0055
0056
0057
0058
0059
0060
0061 % END

```

```
0013 w.name      = 'Hamming';
0014 w.alpha      = 0;
0015 w.psll       = 42.7;
0016 w.rov        = 50;
0017 w.nenbw      = 1.3628;
0018 w.w3db       = 1.3008;
0019 w.flatness   = -1.7514;
0020 w.ws         = sum(v);
0021 w.ws2        = sum(v.*v);
0022 w.win        = v;
0023
0024 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003





--


```

0166     else
0167         % Adding component to the toolbar.
0168         % component position is ignored for now
0169         peer.add(component);
0170         hUicontainer = parent; % toolbar.
0171         handles = getappdata(hUicontainer, 'childhandles');
0172         handles = [handles, hcomponent];
0173         setappdata(hUicontainer, 'childhandles', handles);
0174     end
0175
0176     % make sure the component is on the screen so the
0177     % caller can interact with it right now.
0178     % drawnow;
0179 end
0180
0181 configureComponent;
0182
0183 if (returnContainer == 1)
0184     hcontainer = hUicontainer;
0185 else
0186     hcontainer = [];
0187 end
0188
0189 function createPanel
0190     % add delete listener
0191     hUicontainer = hgjavacomponent('Parent', parent, 'Units', 'Pixels');
0192
0193     set(hUicontainer, 'UserData', char(component.getClass.getName)); 'childhandles',
h5      r, 'UserData'
0192 0 -gp)0185 T* (0188 )Tj T*
0178     % [ 1.1962 0 0 11.may nevPrcont a p 0 t evP <(0184      )Tj 0.13201
0183209     hUic210      0.133 rg (% [ 1.1962 0 0 11.ner)visibl0 0 rg T* (0191

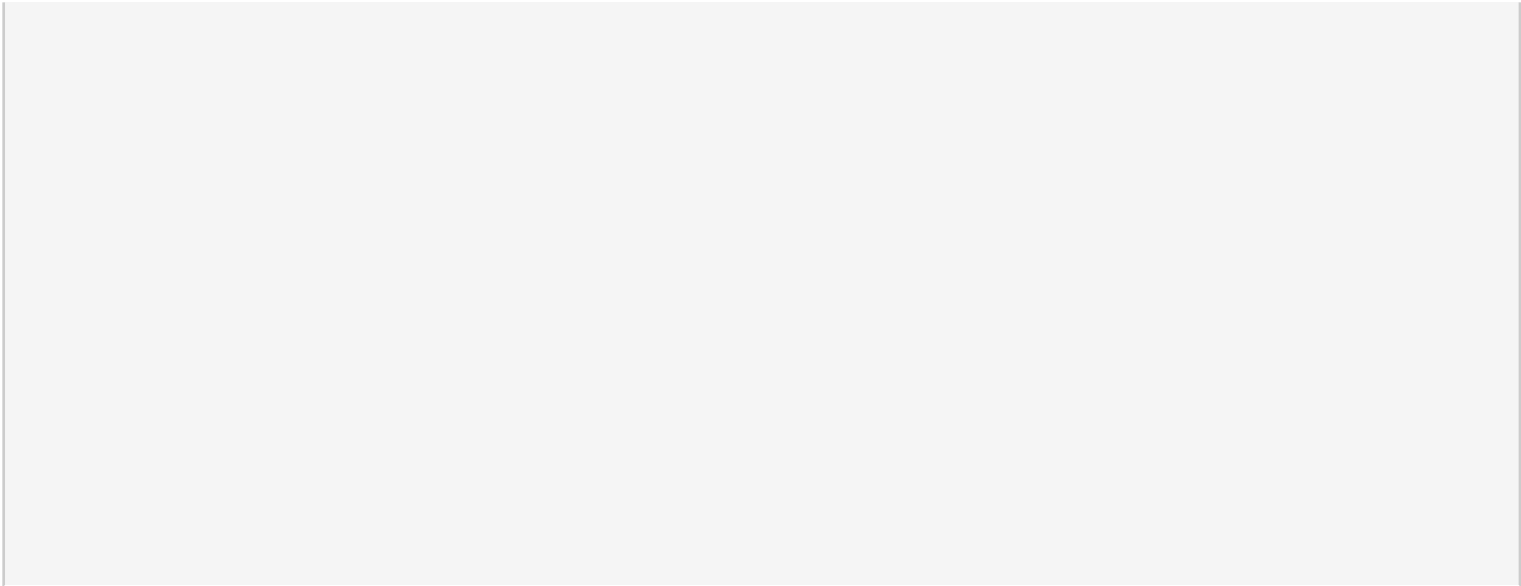
```



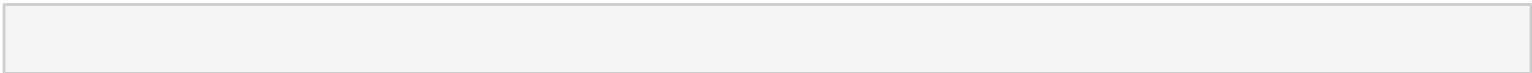
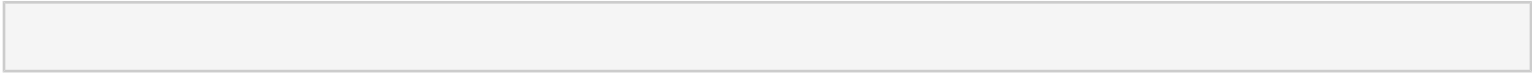
```
0011 % INPUTS:      sin - input string
0012 %
0013 % OUTPUTS:      s   - suitable string for the plot functions
0014 %
0015 % VERSION: $Id: ltpda_label.m,v 1.2 2007/07/13 12:17:39 ingo Exp $
0016 %
0017 % HISTORY: 11-01t0 h2001-2U-0.on
0018 %           Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022 s = strrep(si, '_', '\_');
```

Generated on Fri 07-Mart0 h8 15:46:43 by [m2html](#) © 2003


```
0009 % repository specified by the input database connection.  
0010 %  
0011
```


```
0009 % repository specified by the input database connection.  
0010 %  
0011
```



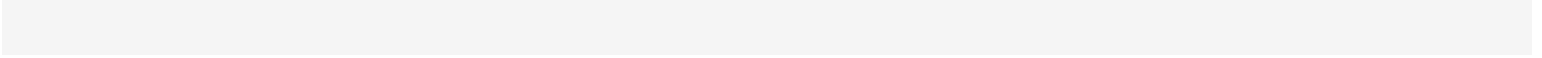

```
0009 % repository specified by the input database connection.
0010 %
0011 % CALL: p = retlieve(conn, ids)
0012 %
0013 %
0014 %
0015 % VERSION: $Id: retlieve.m,v 1.1 2007/08/31 17:40:08 hewitson Exp $
0016 %
0017 % HISTORY: 30-08-07 M Hewitson
0018 % Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022 type = 'param';
0023
0024 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Get each object
0025 objs = [];
0026 for j=1:length(ids)
0027     % Get object
```



```
0023 w.win      = v;  
0024  
0025 % END
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003


```
0009 % repository specified by the input database connection.  
0010 %  
0011
```


```

0009 % repository specified by the input database connection.
0010 %
0011 % CALL: p = retrieve(conn, ids)
0012 %
0013 %
0014 %
0015 % VERSION: $Id: retrieve.m,v 1.1 2007/08/31 17:40:09 hewitson Exp $
0016 %
0017 % HISTORY: 30-08-07 M Hewitson
0018 % Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022 type = 'provenance';
0023
0024 %%%%%%%%% Get each object
0025 objs = [];
0026 for j=1:length(ids)
0027     % Get object
0028     xdoc = mysql_getXdoc(conn, ids(j));
0029     % get child
0030     obj = [];
0031     c = xdoc.getChildNodes;
0032     for k = 07 0 reBength
0033         cc = c.item(k-1);
0034         if strcmp(char(c 0 reNodeName), type)
0035             ccc = cc.getChildNodes;
0036             for l=1:cc 0 reBength
0037                 cccc = ccc.item(l-1);
0038                 if strcmp(char(c c 0 reNodeName), 'Object')
0039                     ccccc = cccc.getChildNodes;
0040                     for m=1:cc c 0 reBength
0041                         cccccc = ccccc.item(m-1);
0042                         if strcmp(char(c c c 0 reNodeName), type)
0043                             obj = fromxml(c c c );
0044                         end
0045                     end
0046                 end
0047             end
0048         end
0049     end
0050
0051     objs = [objs obj];
0052 end
0053
0054
0055
0056
0057
0058
0059
0060
0061 % END

```


0024
0025 % END


```
0009 % repository specified by the input database connection.  
0010 %  
0011 % CALL: p = retrieve(conn, ids)  
0012 %  
0013 %  
0014 %  
0015
```


retrieve

PURPOSE

RETRIEVE specwin objects with given ids from the LTPDA repository specified by the input

SYNOPSIS

function objs = retrieve(conn, ids)

DESCRIPTION

```
RETRIEVE specwin objects with given ids from the LTPDA repository specified by the
input
datab/se connection.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DESCRIPTION: RETRIEVE specwin objects with given ids from the LTPDA
             repository specified by the input datab/se connection.

CALL:       p = retrieve(conn, ids)

VERSION:    $Id: retrieve.m,v 1.1 2007/08/31 17:40:08 hewitson Exp $
```

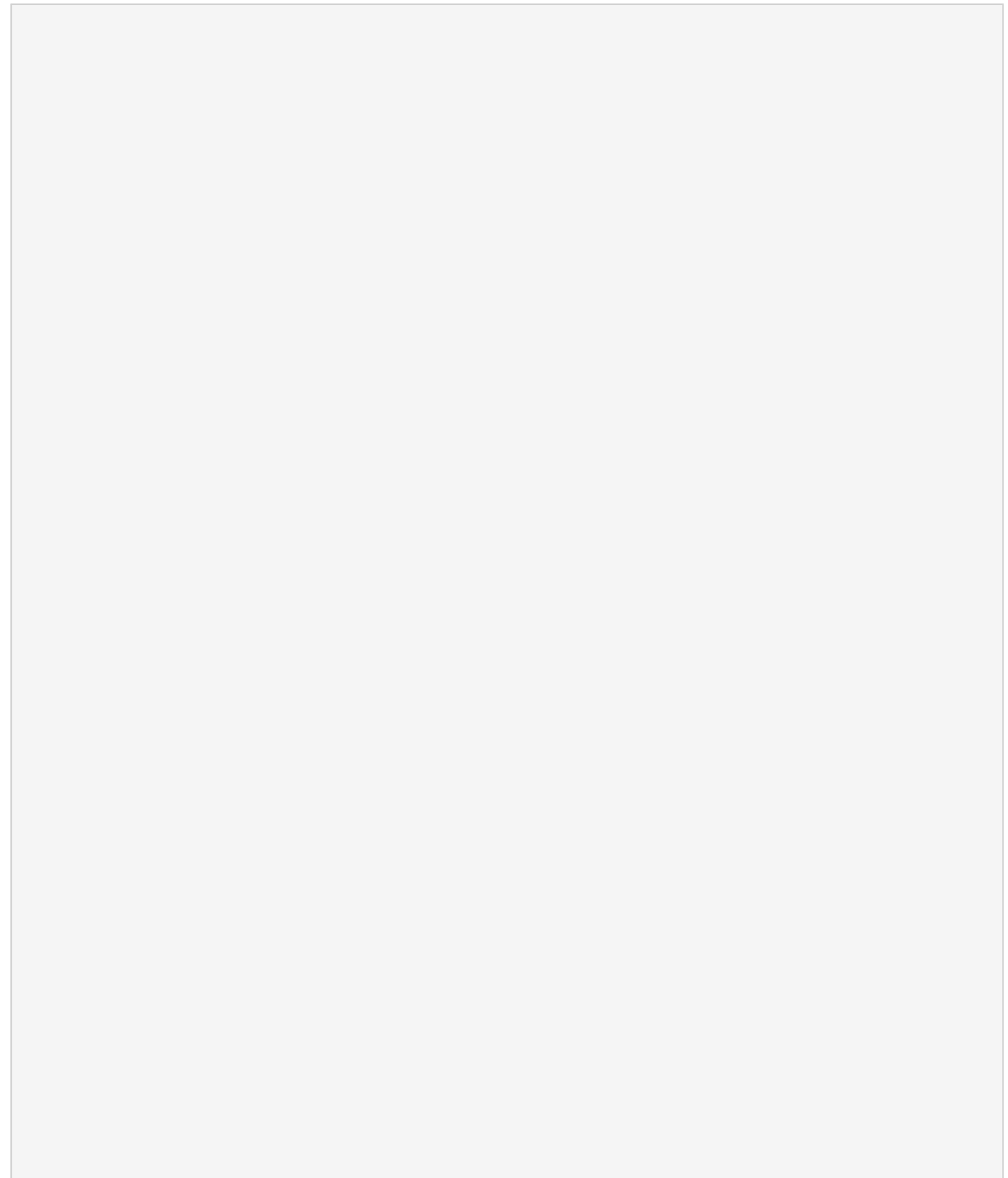
```

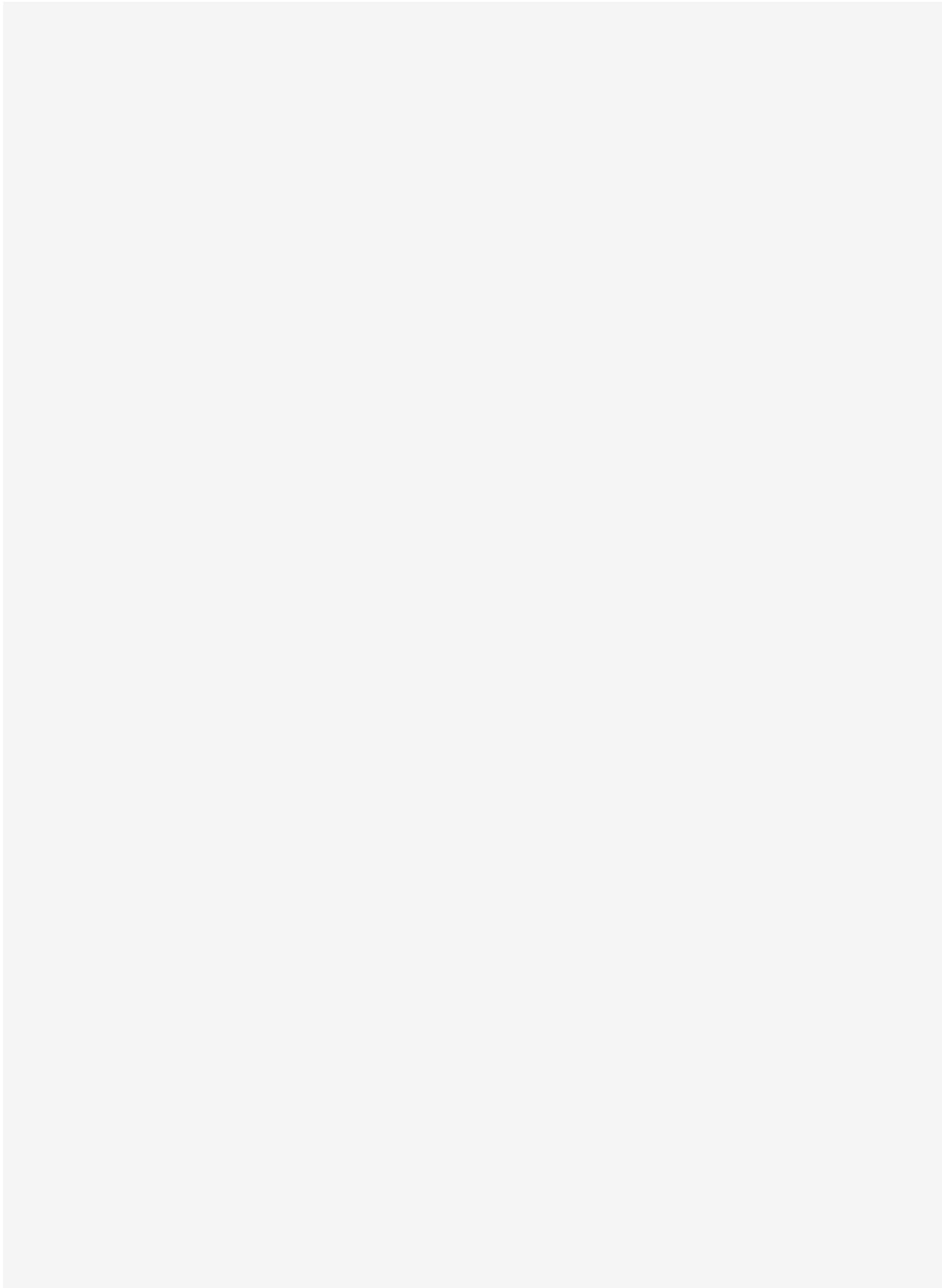
0009 % repository specified by the input database connection.
0010 %
0011 % CALL: p = retlieve(conn, ids)
0012 %
0013 %
0014 %
0015 % VERSION: $Id: retlieve.m,v 1.1 2007/08/31 17:40:08 hewitson Exp $
0016 %
0017 % HISTORY: 30-08-07 M Hewitson
0018 % Creation
0019 %
0020 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0021
0022 type = 'specwin';
0023
0024 %%%%%%%%% Get each object
0025 objs = [];
0026 for j=1:length(ids)
0027     % Get object
0028     xdoc = mysql_getXdoc(conn, ids(j));
0029     % get child
0030     obj = [];
0031     c = xdoc.getChildNodes;
0032     for k = 07 0 reBength
0033         cc = c.item(k-1);
0034         if strcmp(char(c 0 reNodeName), type)
0035             ccc = cc.getChildNodes;
0036             for l=1:cc 0 reBength
0037                 cccc = ccc.item(l-1);
0038                 if strcmp(char(c c 0 reNodeName), 'Object')
0039                     ccccc = cccc.getChildNodes;
0040                     for m=1:cc c 0 reBength
0041                         cccccc = ccccc.item(m-1);
0042                         if strcmp(char(c c c 0 reNodeName), type)
0043                             obj = fromxml(c c c );
0044                         end
0045                     end
0046                 end
0047             end
0048         end
0049     end
0050
0051     objs = [objs obj];
0052 end
0053
0054
0055
0056
0057
0058
0059
0060
0061 % END

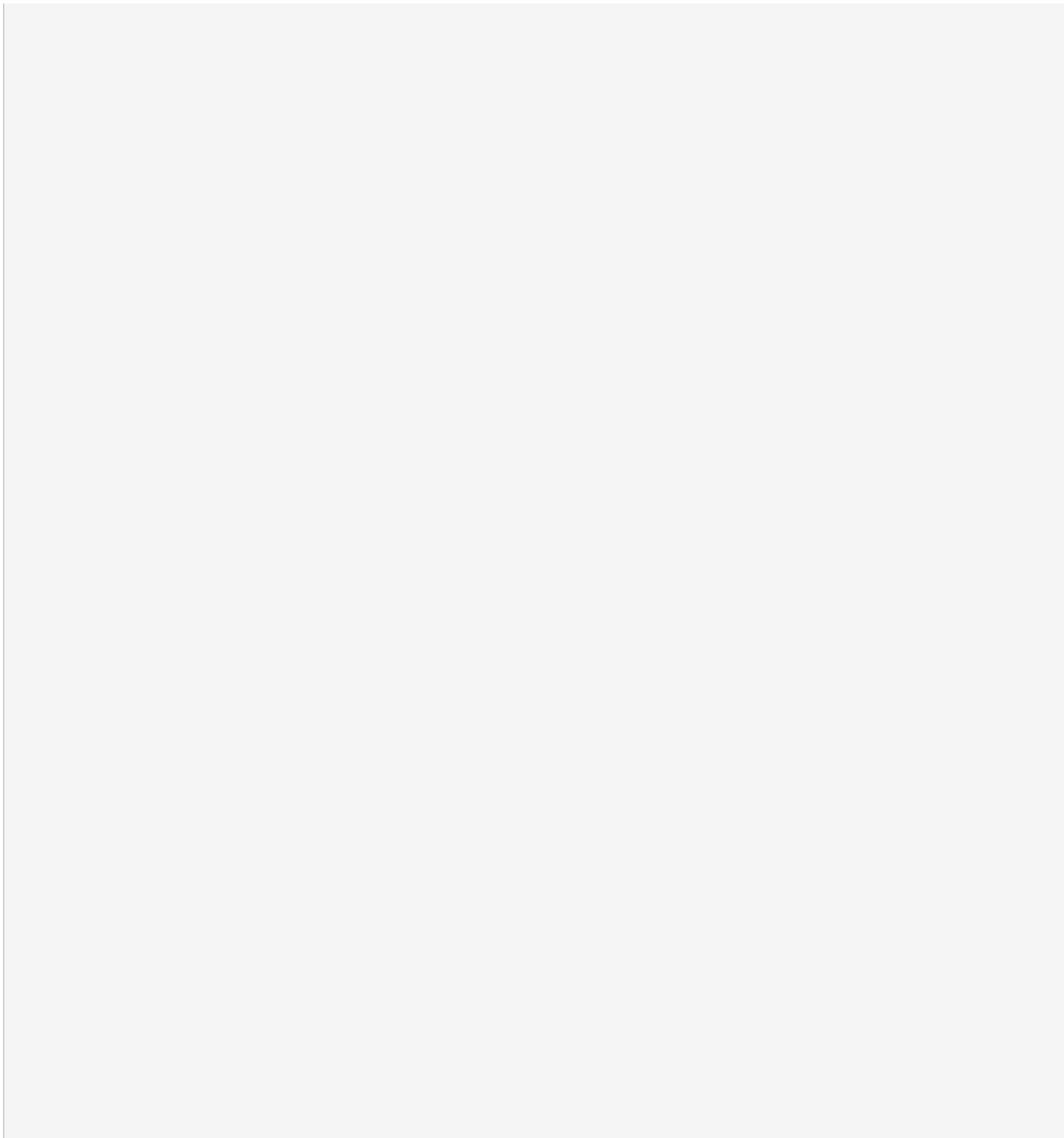
```


SUBFUNCTIONS

◦ `function plo = getDefaultPL()`

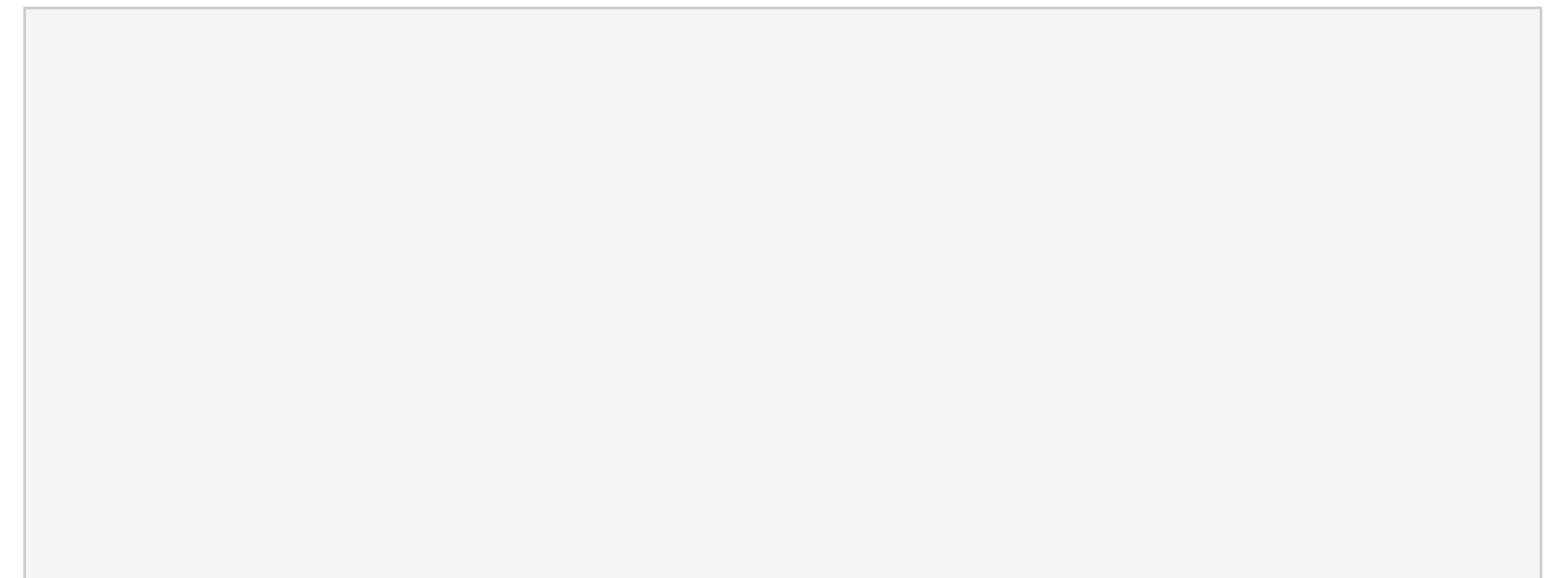
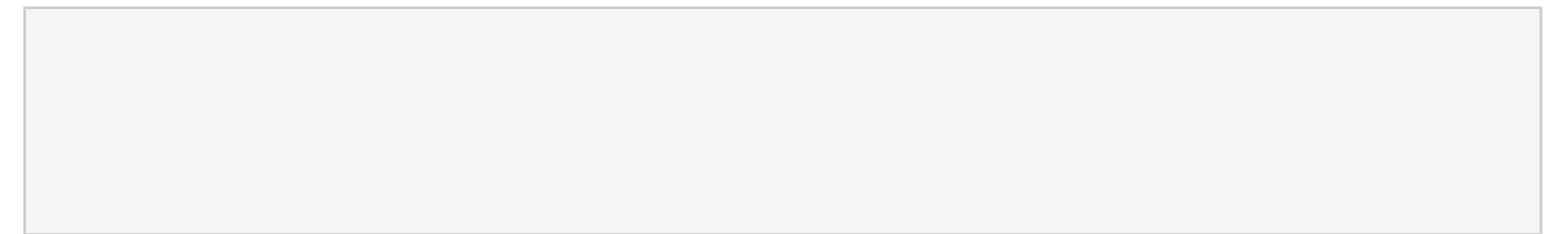
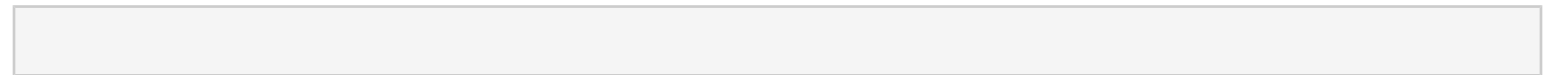
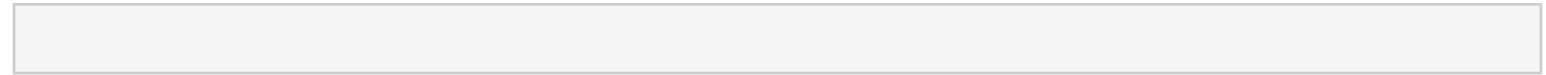




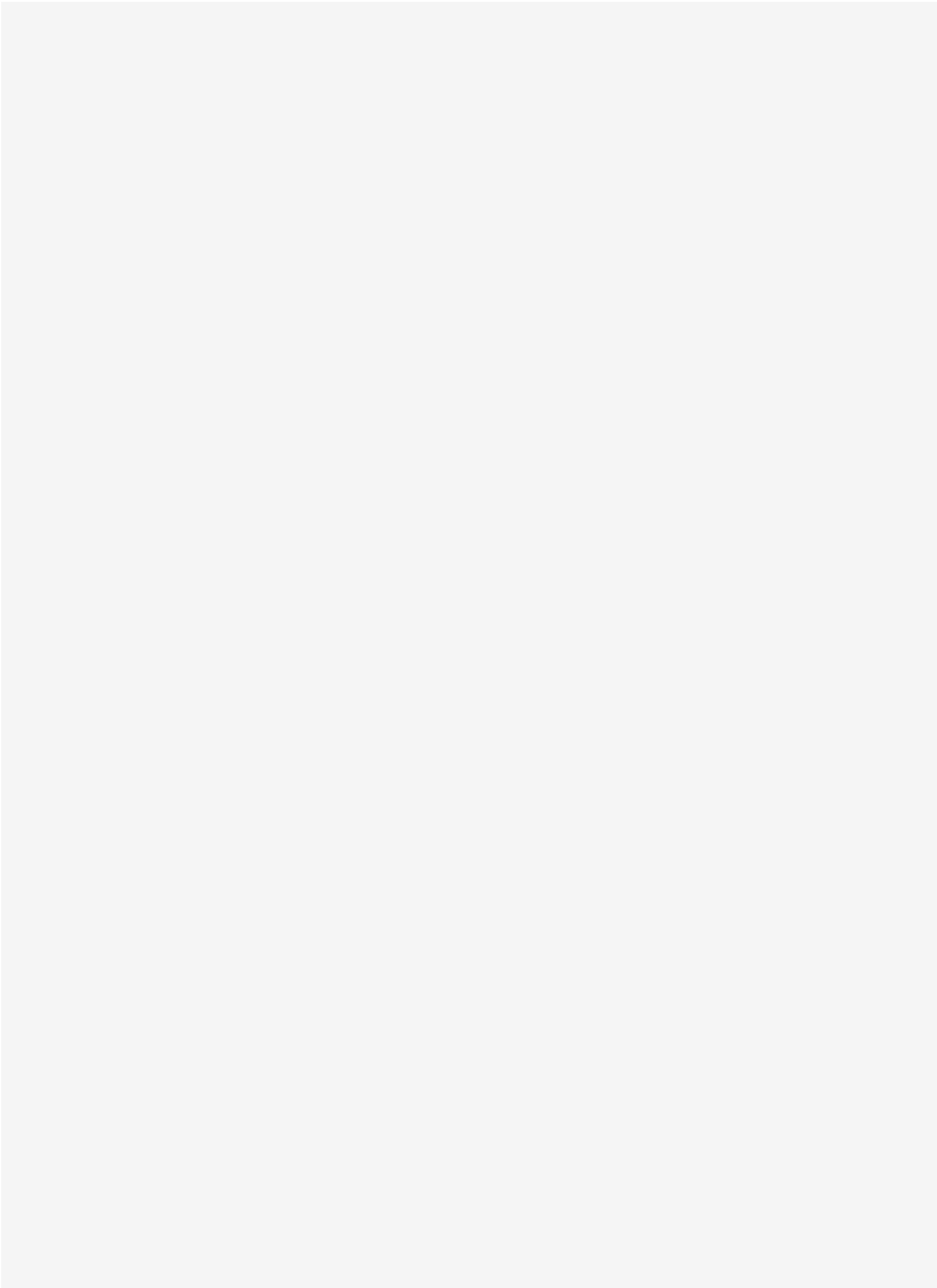


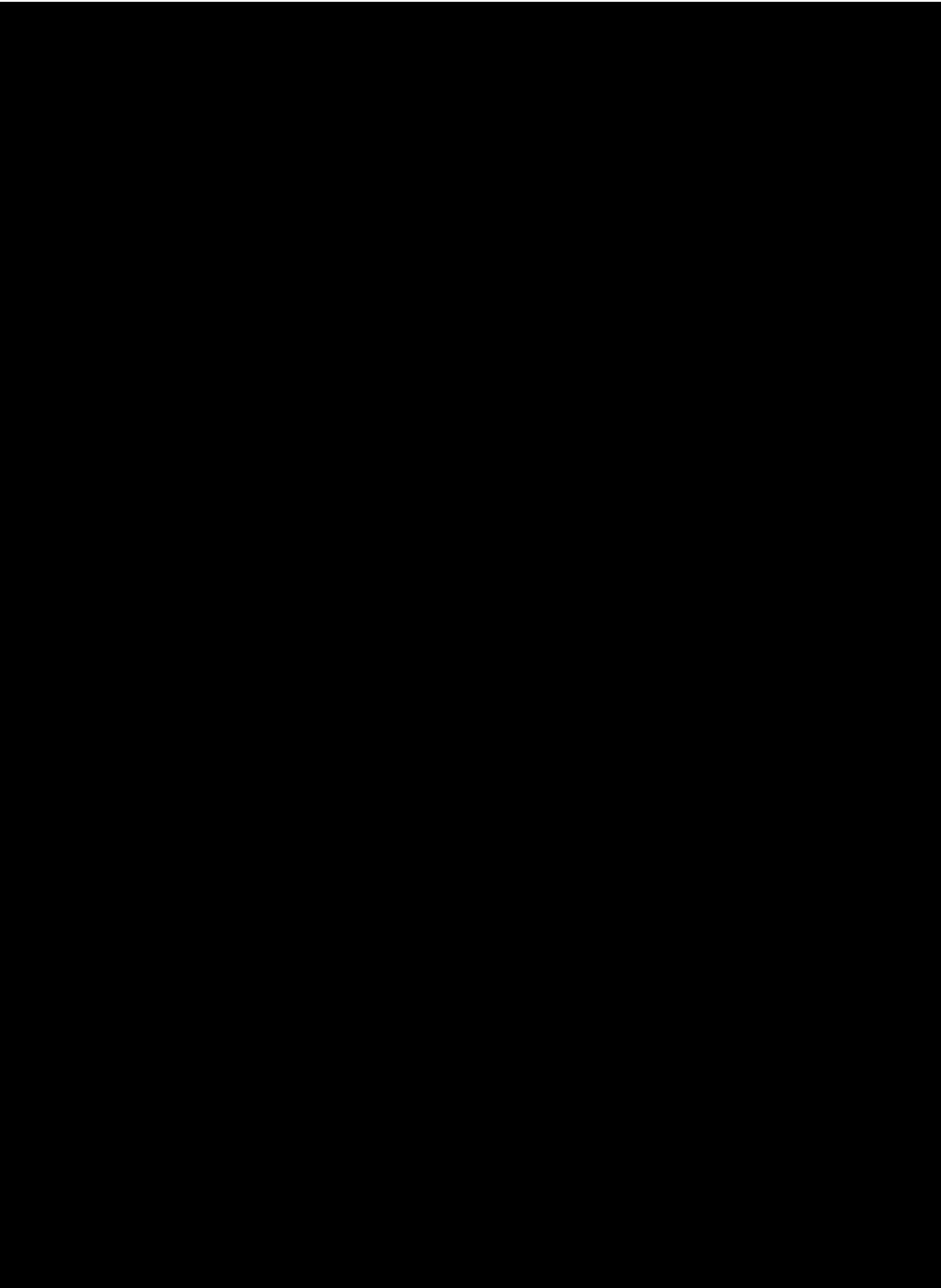

```
0009 % repository specified by the input database connection.  
0010 %  
0011
```

[Home](#) > [S3S3SRB1838I3](#) [EMC](#) /Div <</MCID 1 >>BDC BT 0 0 0 rg 100.974 Tf 14 0 0 1



```
0013 % Make window struct
0014 w.name      = 'SFT3M';
0015 w.alpha     = 0;
0016 w.psll      = 44.2;
0017 w.rov       = 65.5;
0018 w.nenbw     = 2.9452;
0019 w.w3db      = 2.9183;
0020 w.flatness  = -0.0115;
0021 w.ws        = sum(v);
0022 w.ws2       = sum(v.*v);
0023 w.win       = v;
0024
0025 % END
```





```
0165 if ~isempty(line)
0166     [bl, lidx] = sort([line.a], 'descend');
0167     lidxs = lidx(1:min([N length(lidx)]));
0168     lines = line(lidxs);
0169     disp(sprintf('    + found %d lines.', length([lines.f])));
0170 else
0171     disp('    + found 0 lines.');
```

```
0172 end
0173
0174
0175
0176 %-----
0177 % get default parameter list
0178 function pl = getDefaultPL()
0179
0180 pl = plist();
0181 pl = append(pl, param('N', 10));
0182 pl = append(pl, param('fsearch', [0 1e10]));
0183 pl = append(pl, param('thresh', 2));
0184 pl = append(pl, param('bw', 20));
0185 pl = append(pl, param('hc', 0.8));
0186
0187
0188
0189
0190 % END
```


0025 % END

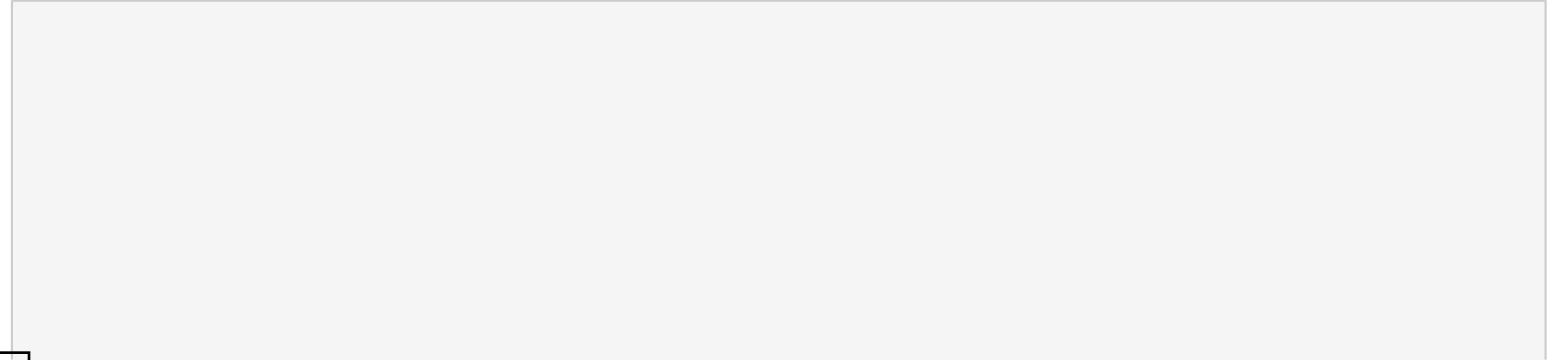
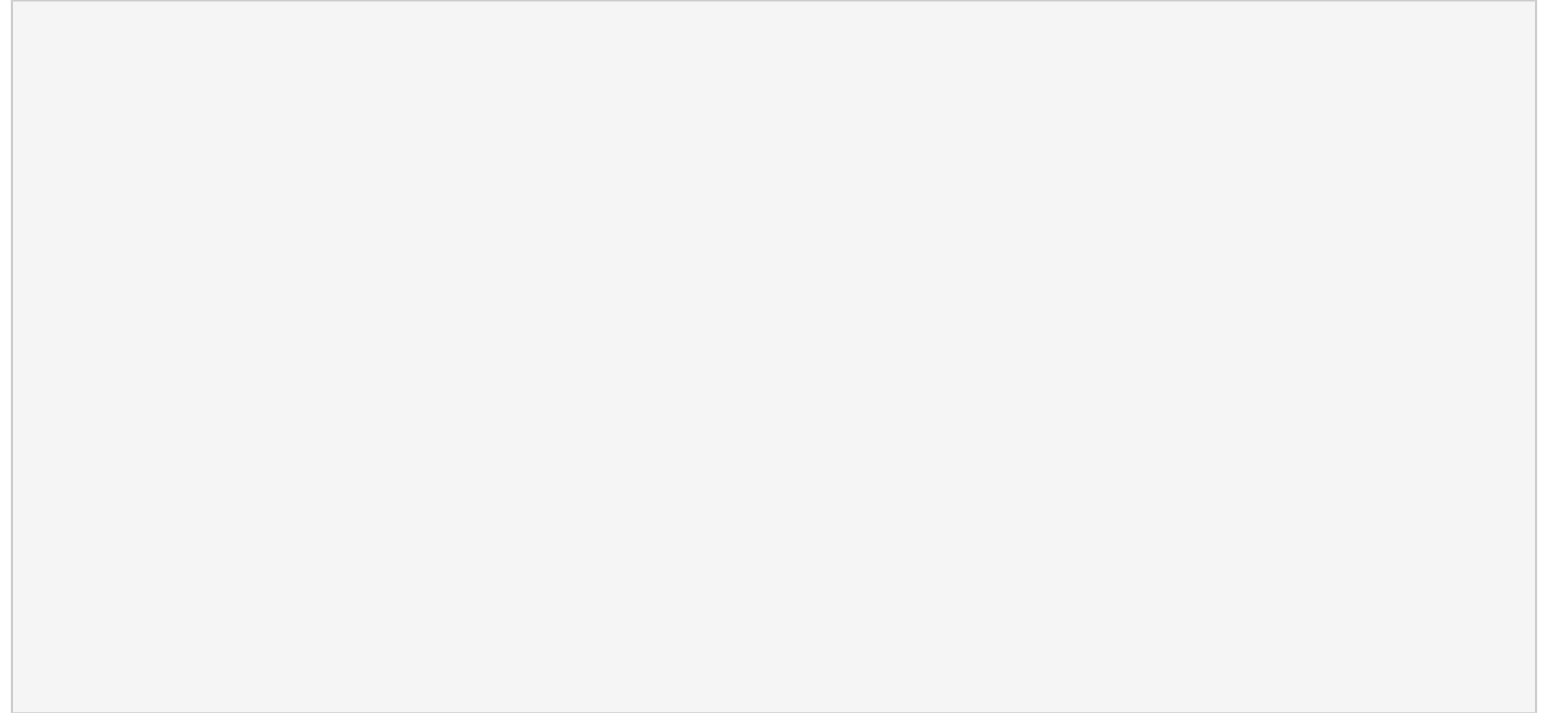
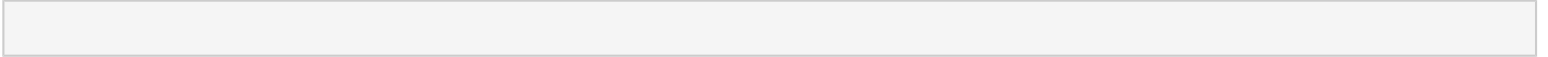
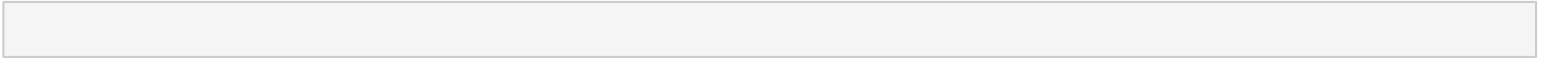

```
0013
0014 % Make window struct
0015 w.name      = 'SFT4M';
0016 w.alpha     = 0;
0017 w.psll      = 66.5;
0018 w.rov       = 72.1;
0019 w.nenbw     = 3.3868;
0020 w.w3db      = 3.3451;
0021 w.flatness  = -0.0067;
0022 w.ws        = sum(v);
0023 w.ws2       = sum(v.*v);
0024 w.win       = v;
0025
0026 % END
```



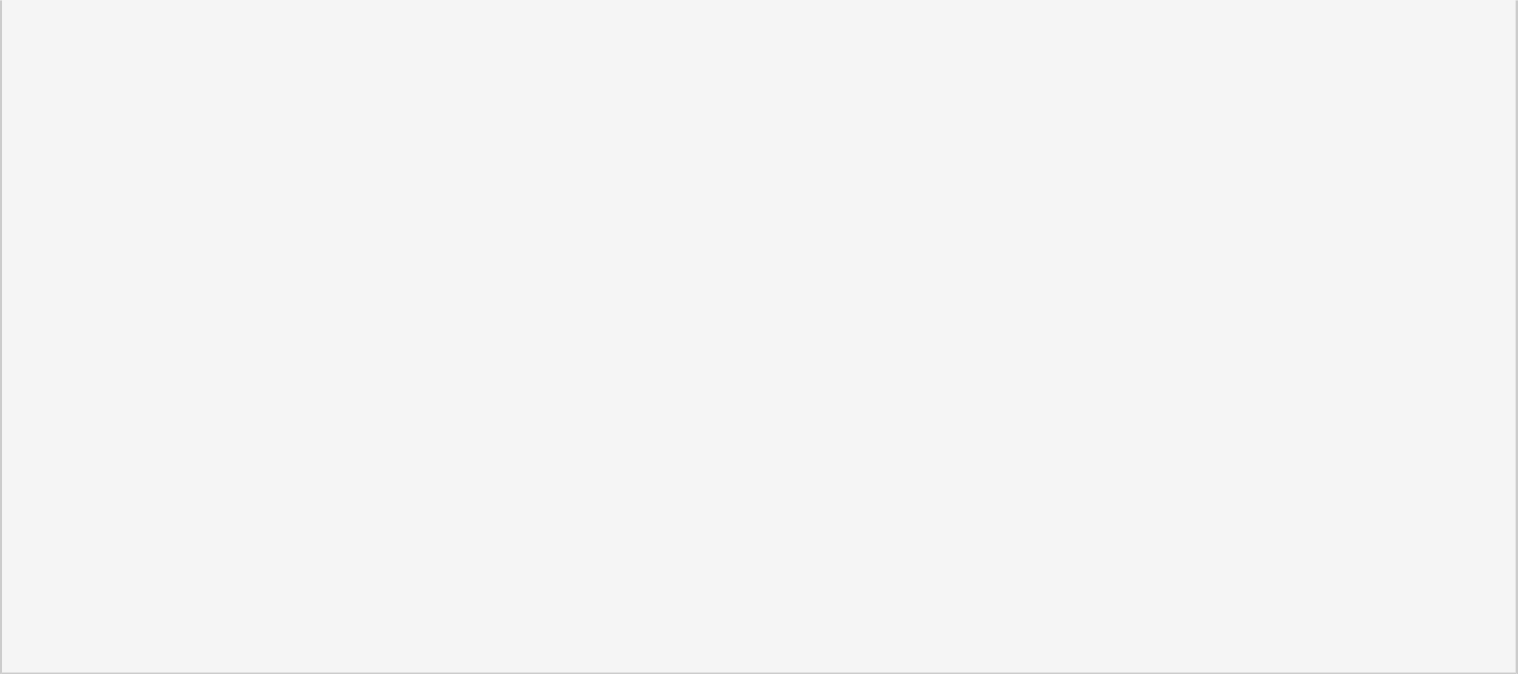
0010 %

0011

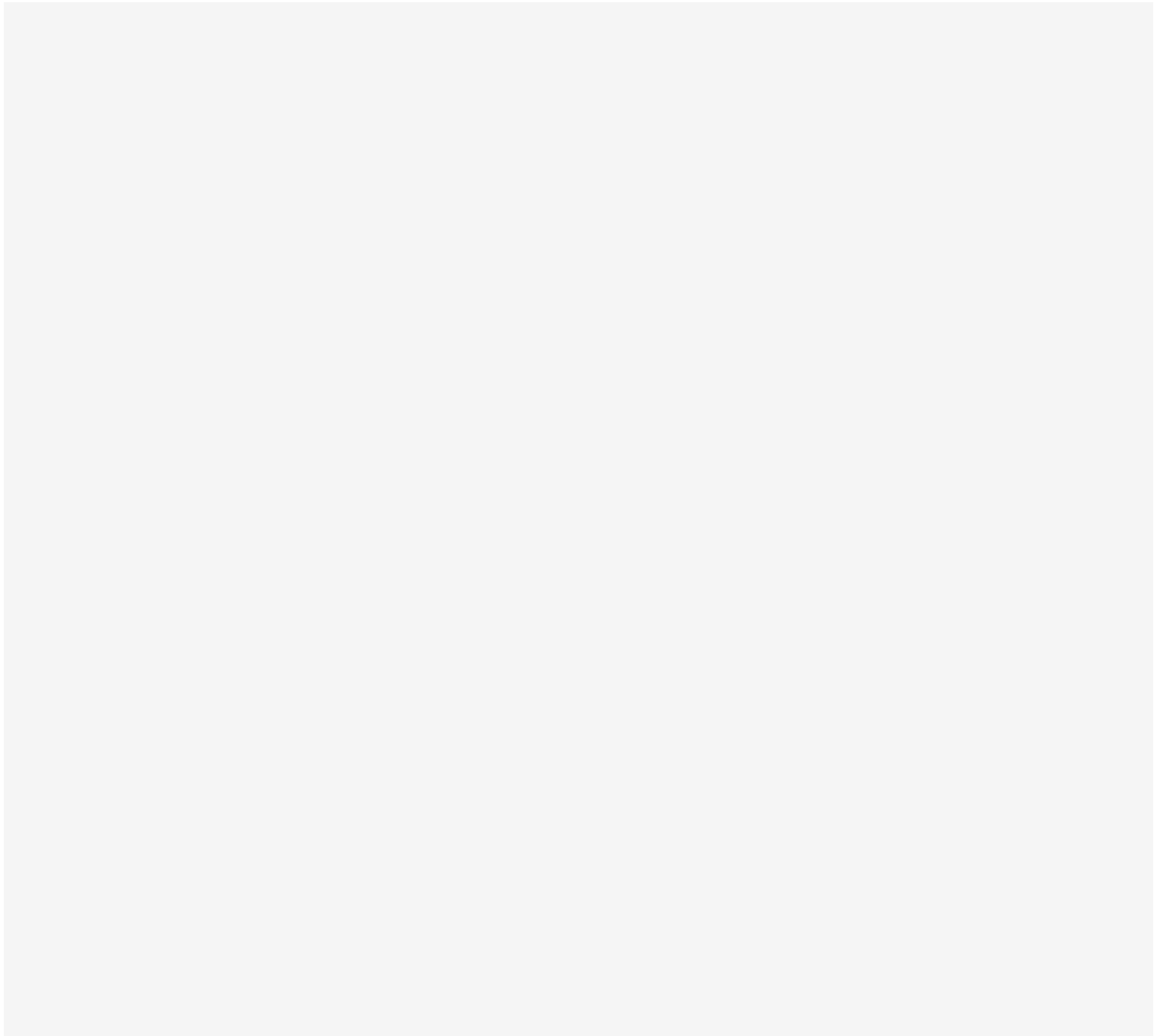
[Home](#) >

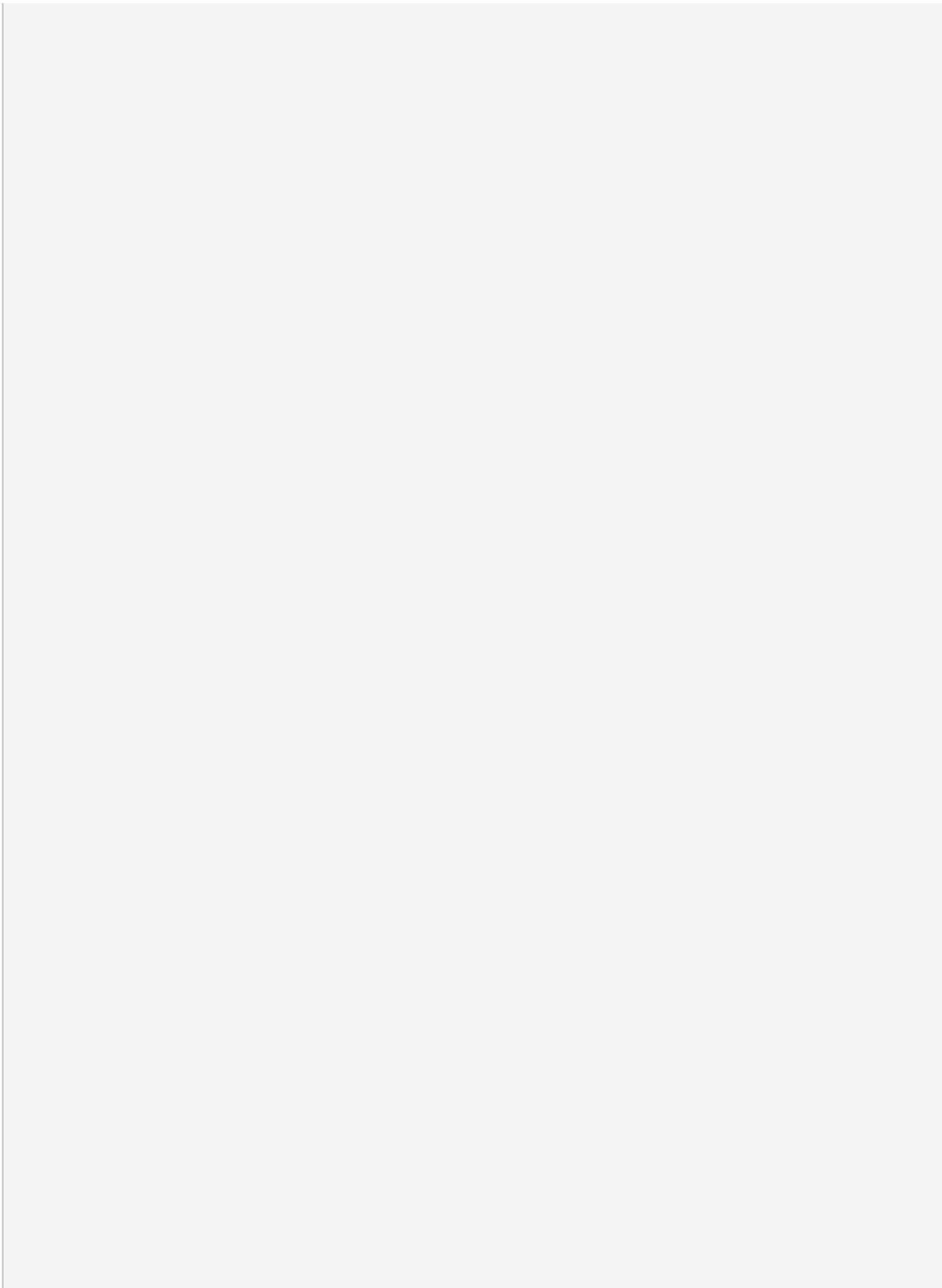


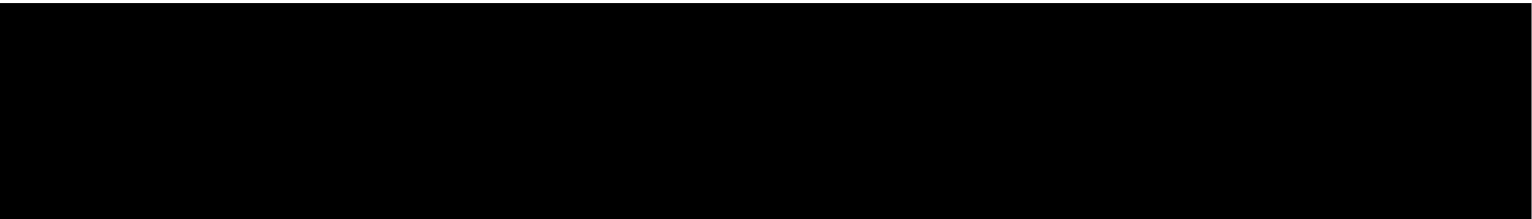
[illegible]




```
0009 % repository specified by the input database connection.
0010 %
0011 % CALL: p = retrieve(conn, ids)
0012 %
0013 %
0014 %
0015 % VERSION: $Id: retrieve.m,v 1.1 2007/08/31 17:40:08 hewitson Exp $
0016 %
0017 % HISTORY: 30-08-07 M Hewitson
0018 % Creation
0019 %
0020 %%%%%%%%%%%%%%%
0021
0022 type = 'zero' Hewitson
```



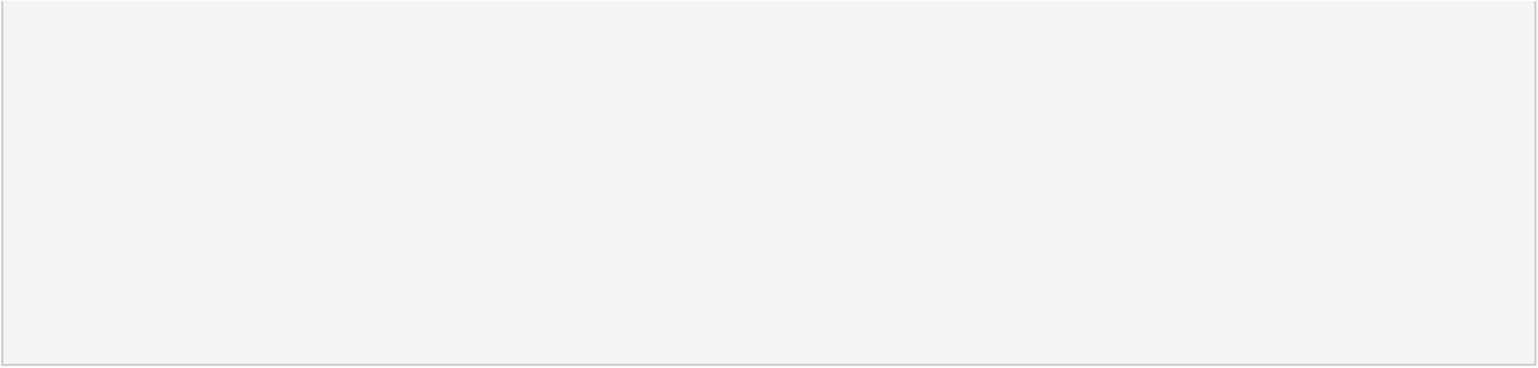



```
0059 aout.data      = data;
0060 aout.hist        = hist;
0061 aout.provenance   = prov;
0062 aout.description  = description;
0063 aout.mfile        = mfile;
0064 aout.mfilename    = mfname;
0065 aout.mdlfile      = mdlfile;
0066 aout.mdlfilename  = mdlfname;
0067 aout = class(aout, 'ao');
0068
0069
```

Generated on Fri 07-Mar-2008 15:46:43 by [m2html](#) © 2003

Description of xdoc2ao


```
0062 % Determine the bounding rectangle for all the plots
0063
0064 % h = findobj('Type','axes');
0065
0066 % findobj is a 4.2 thing.. if you don't have 4.2 comment out
0067 % the next line and uncomment the following block.
0068
0069 % h = findobj(gcf,'Type','axes'); % Change suggested by Stacy J. Hills
0070
0071 % If you don't have 4.2, use this code instead
0072 ch = get(gcf,'children' 0.13(',' , 'axes 0 0 rg3 h=[]'axes 0 0 rg T* (00 Q q .13(ect
```





```
0020      cmd = sprintf(
```



```

0052 queries = [];
0053 for j=1:nargin
0054     if isa(varargin{j}, 'ao')
0055         as = [as varargin{j}];
0056     end
0057     if isa(varargin{j}, 'plist')
0058         ps = [ps varargin{j}];
0059     end
0060 end
0061
0062 Na = length(as);
0063 if isempty(as)
0064     error('### Please input at least one A0.');
```

0065 end

0066

0067 % Combine plists

0068 if ~isempty(ps)

0069 pl = combine(ps, getDefaultPL());

0070 else

0071 pl = getDefaultPL();

0072 end

0073

0074 % Get parameters from plist

0075 bw = find(pl, 'bw');

0076 hc = find(pl, 'hc');

0077

0078 % Loop over input A0gin{j}, for j=1:Na

0082 a = as(j);

0083 d = a.data;

0084

0085 if isa(d, 'fsdata')

0086

0087 nf =

0088

0089 nfest(d.y, bw, hc);

0088

0089 else

0090 error('### I can only find lines in frequency-series A0g.');

0091 end

0092

0093 %----- Make output A0

0094

0095 % make fsdata

0096 d = set(d, 'y', nf);

0097 d = set(d, 'name', sprintf('nfest(%s)', d.name));

0098

0099 % create new output history

0100 h = history(ALGONAME, VERSION, pl, a.hist);

0101 h = set(h, 'invars', invars);

0102

0103 % make output analysis object

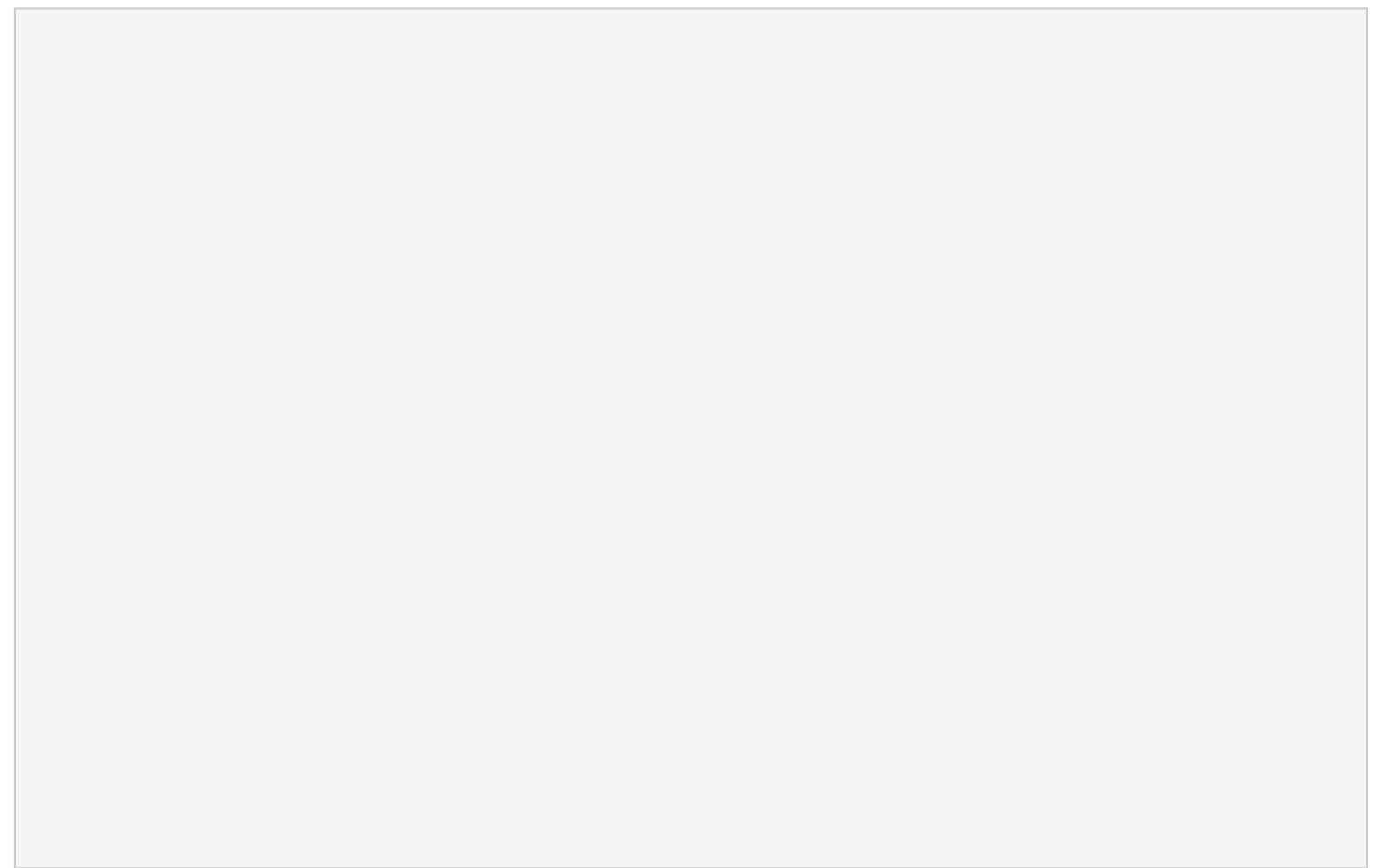
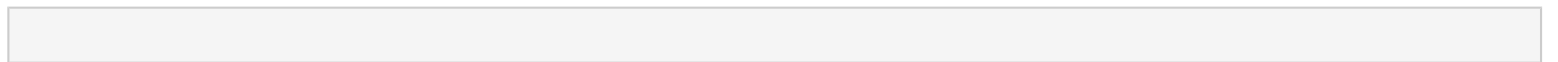
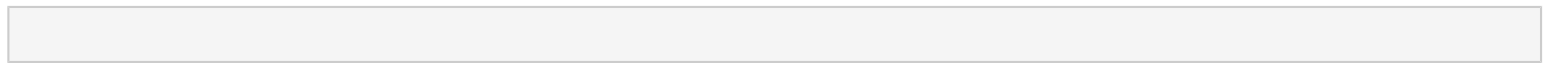
0104 b = ao(d, h);

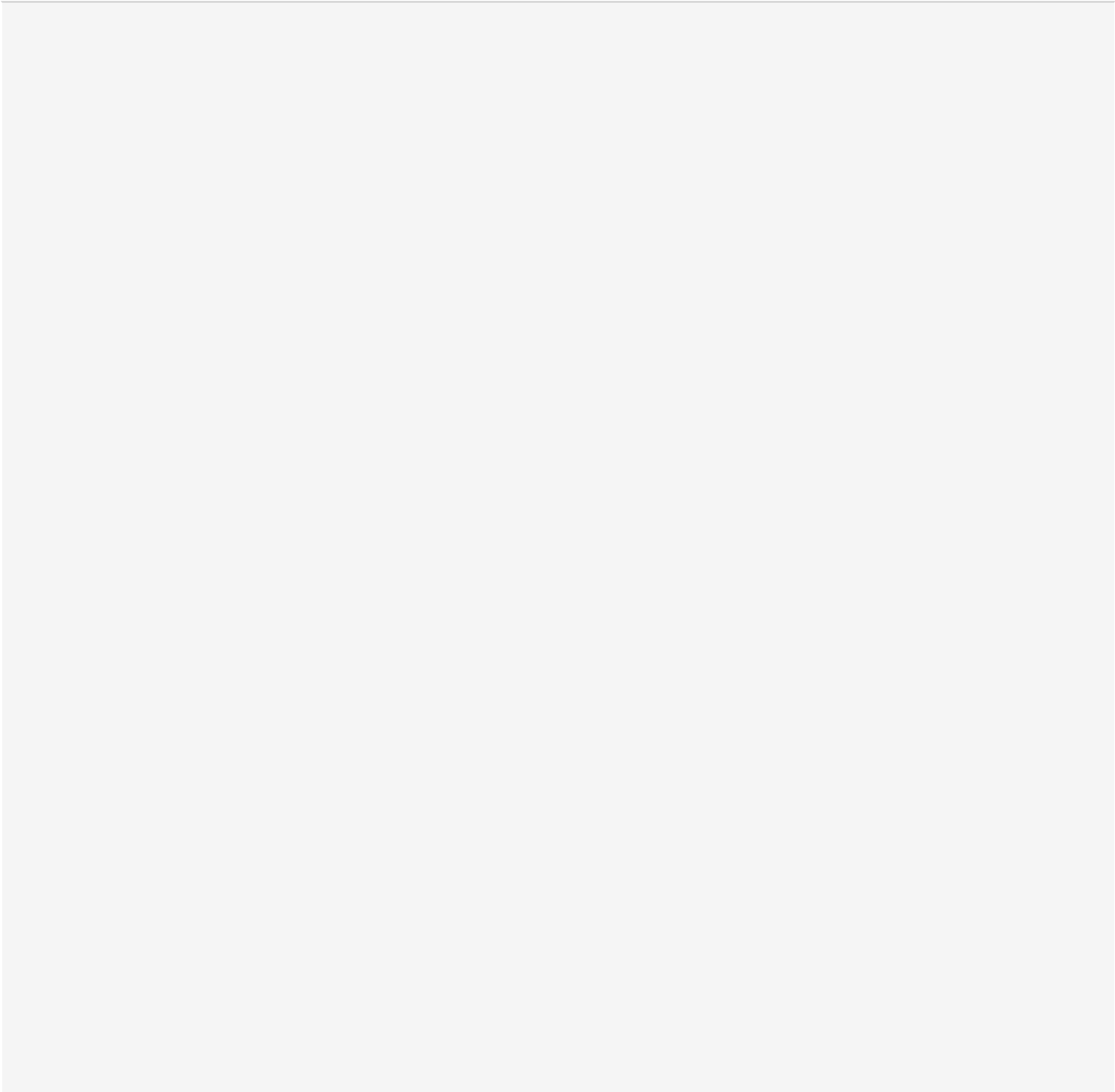
0105

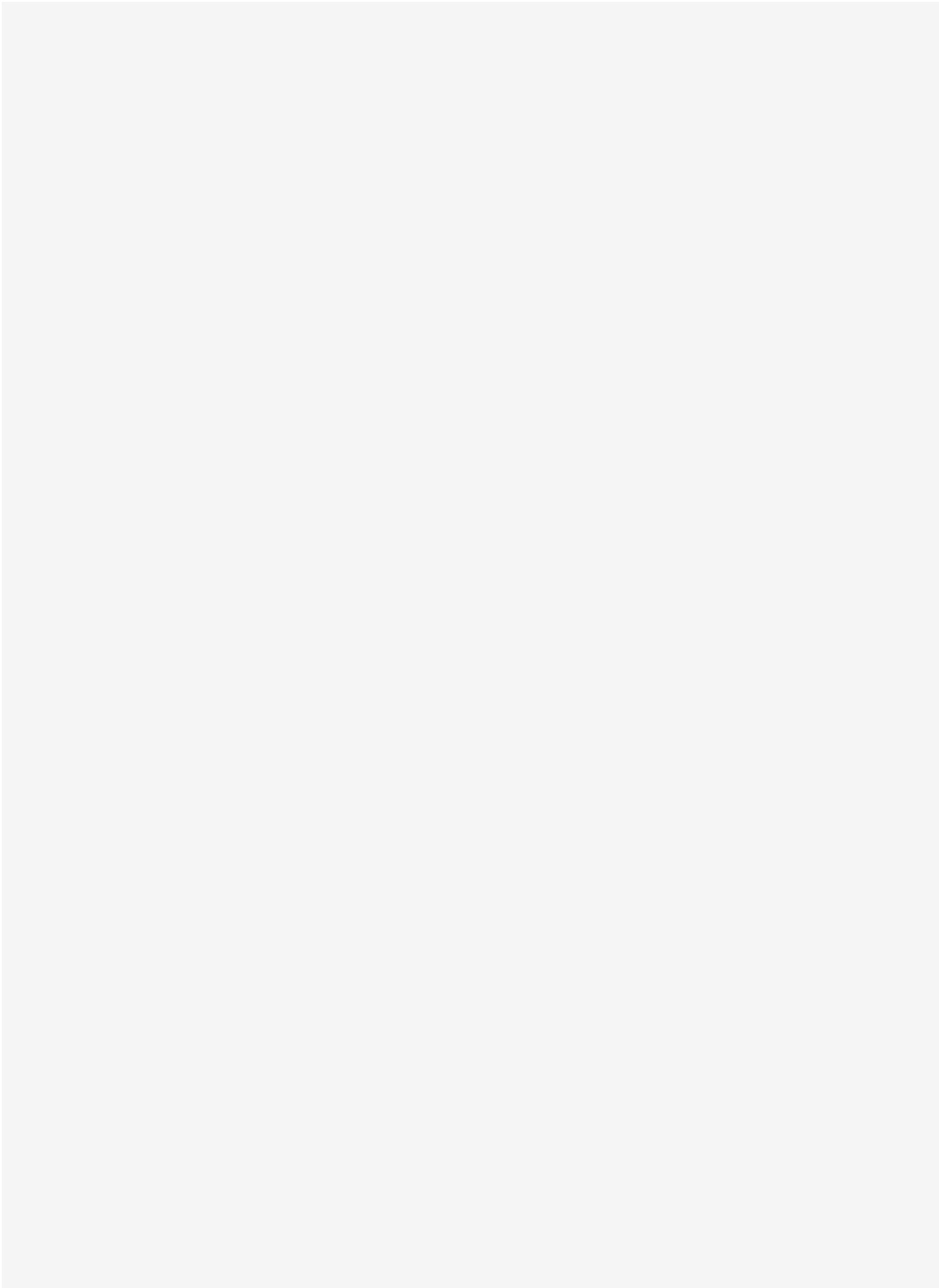
0106 % set name

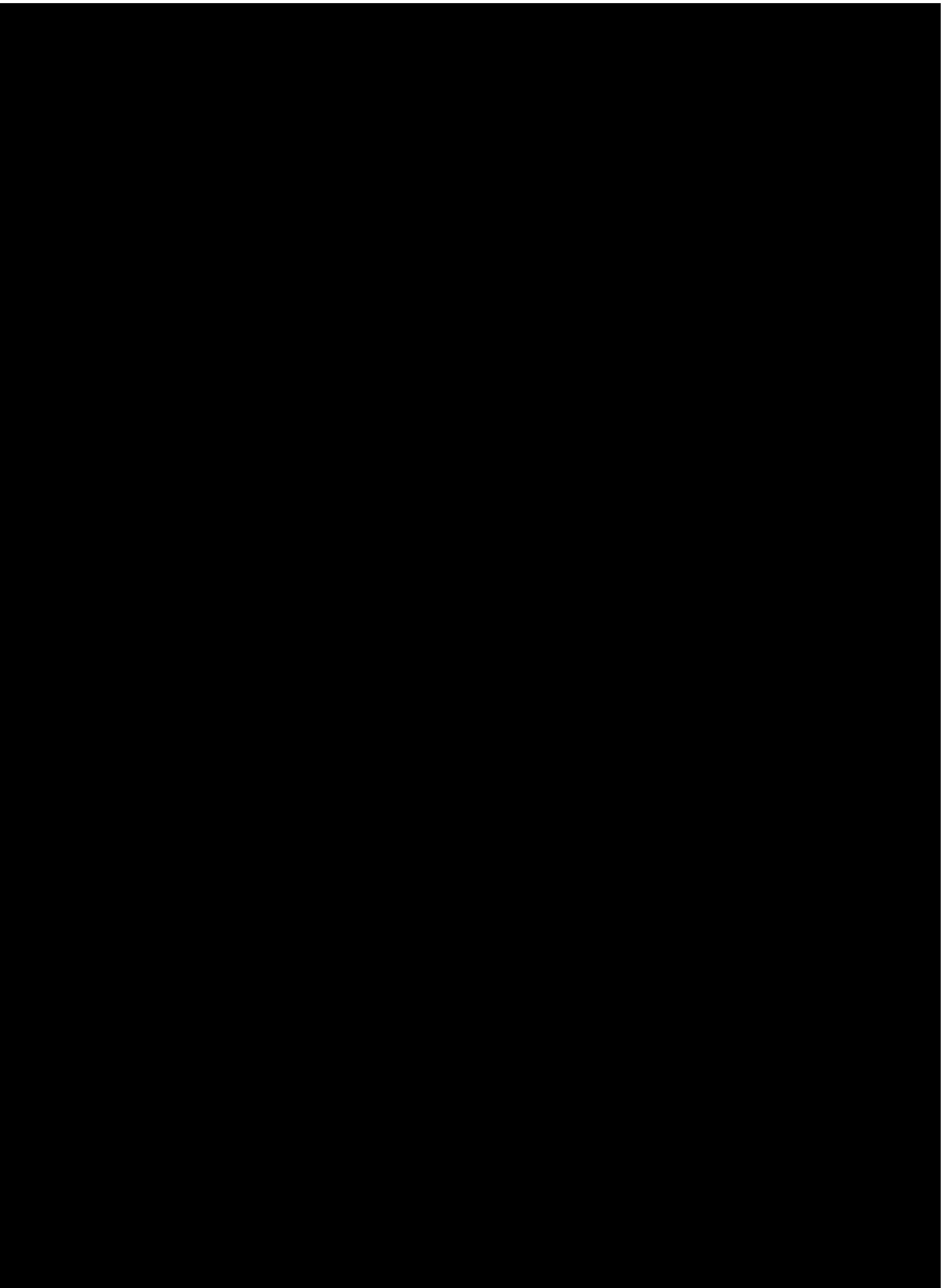
0107 % name for this object

0108 if isempty(invars{j})





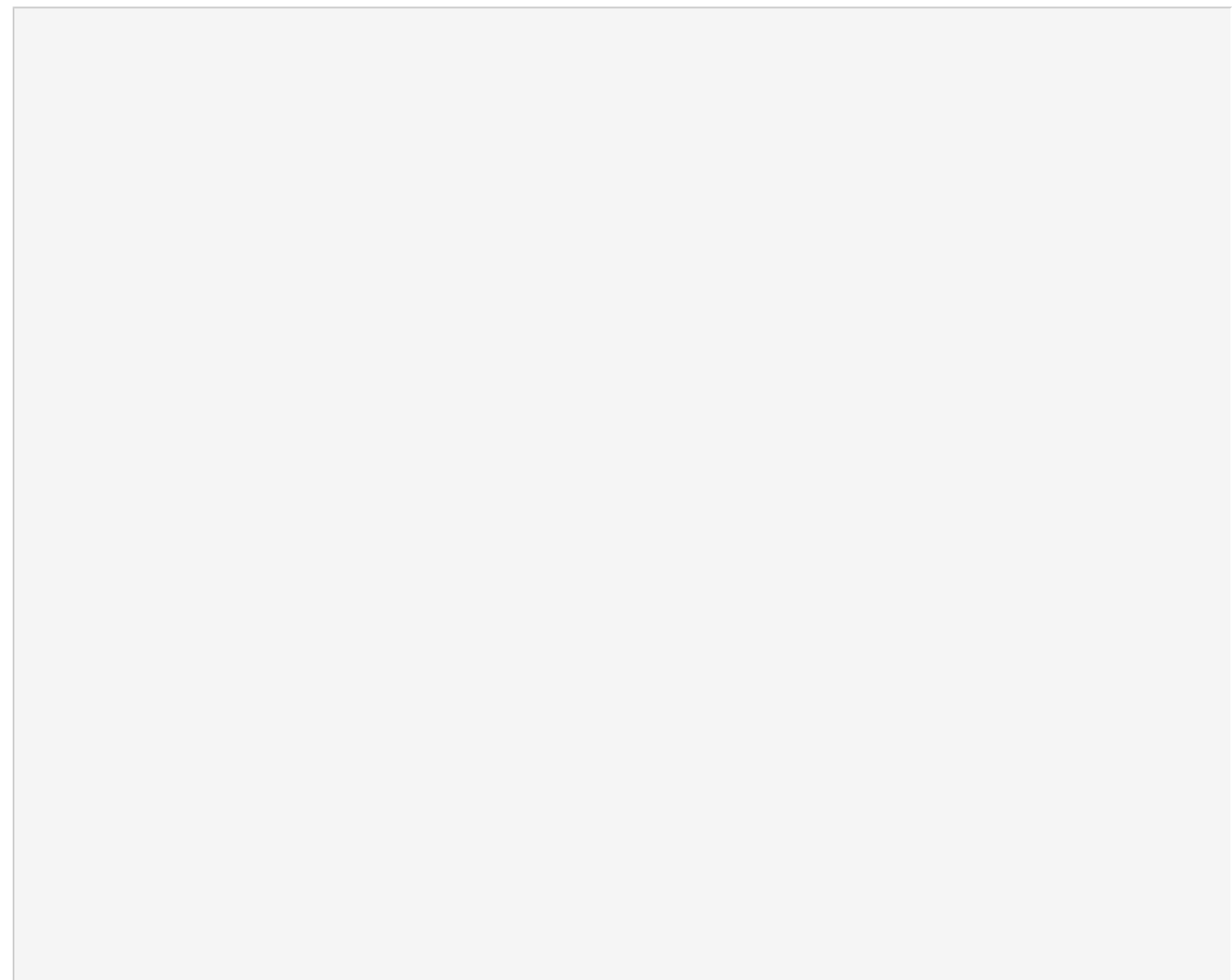
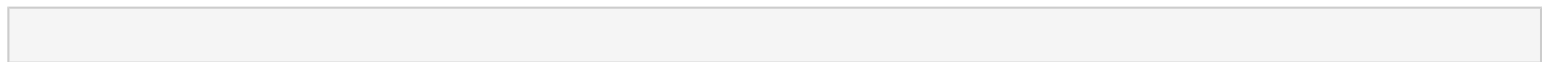
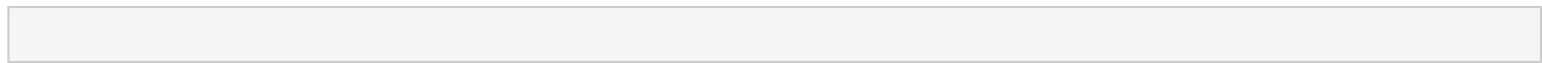


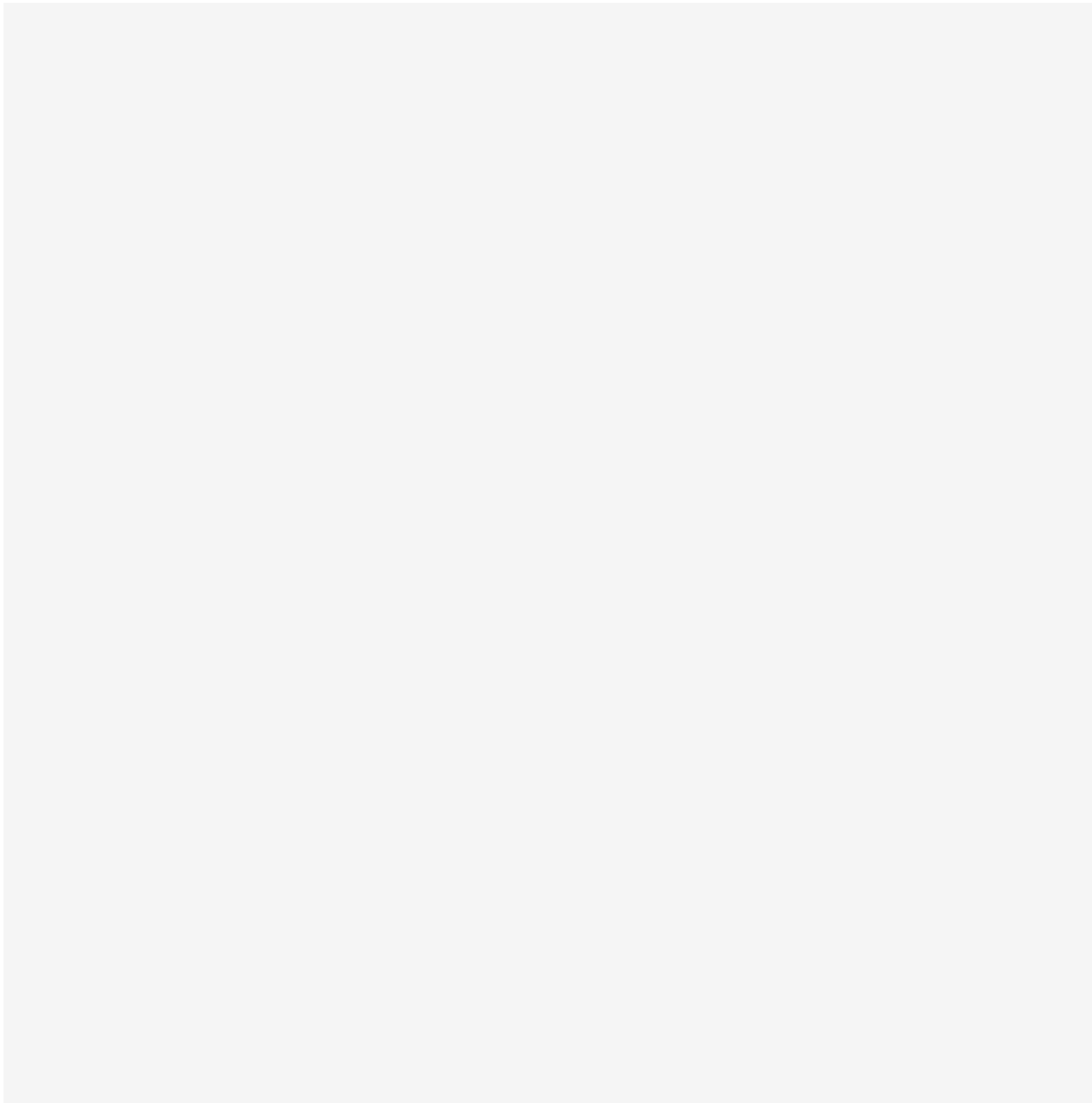


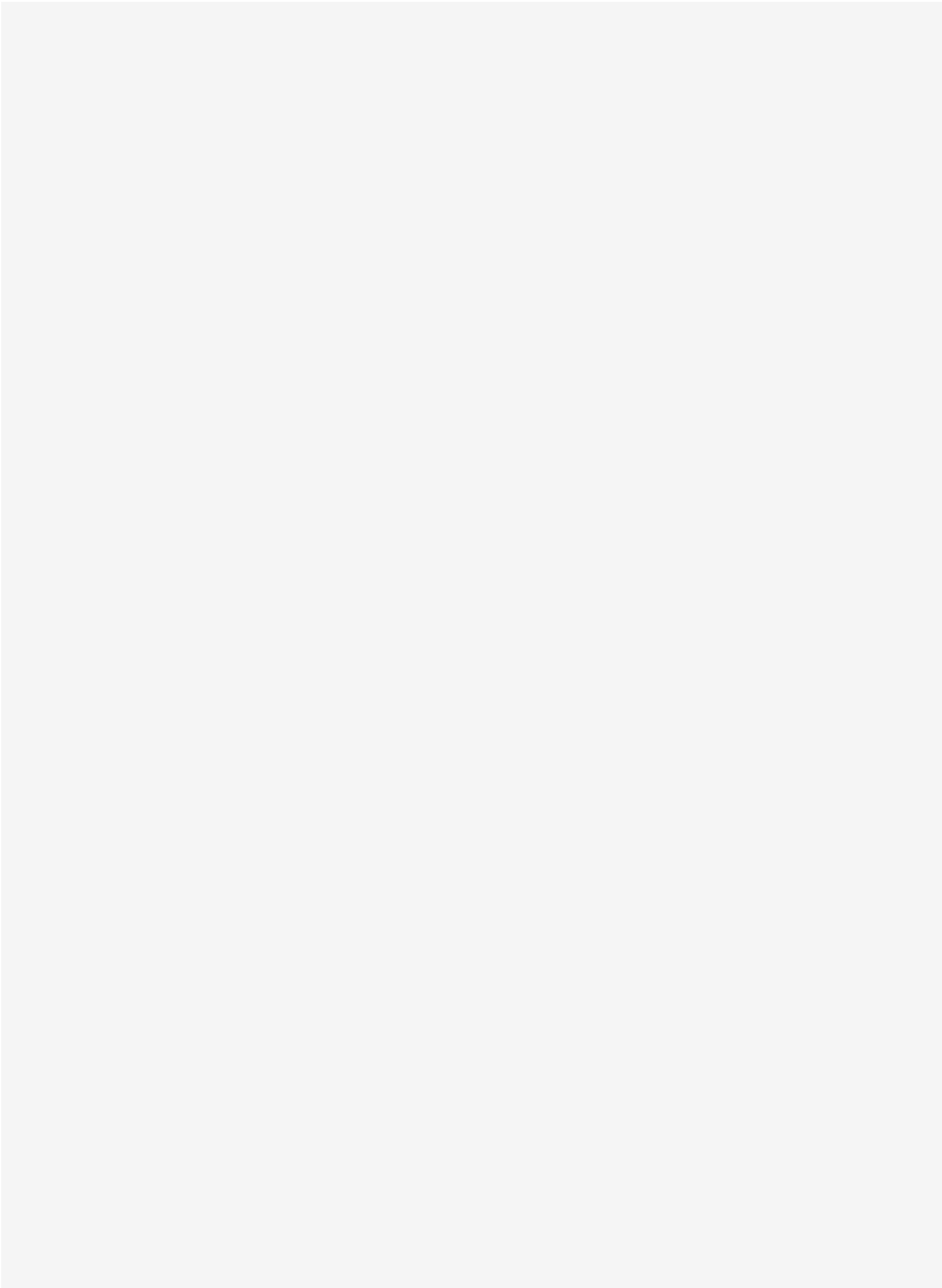

```
0559
0560         elseif iscell(vai0 -7j 0 0 T*60    1
```

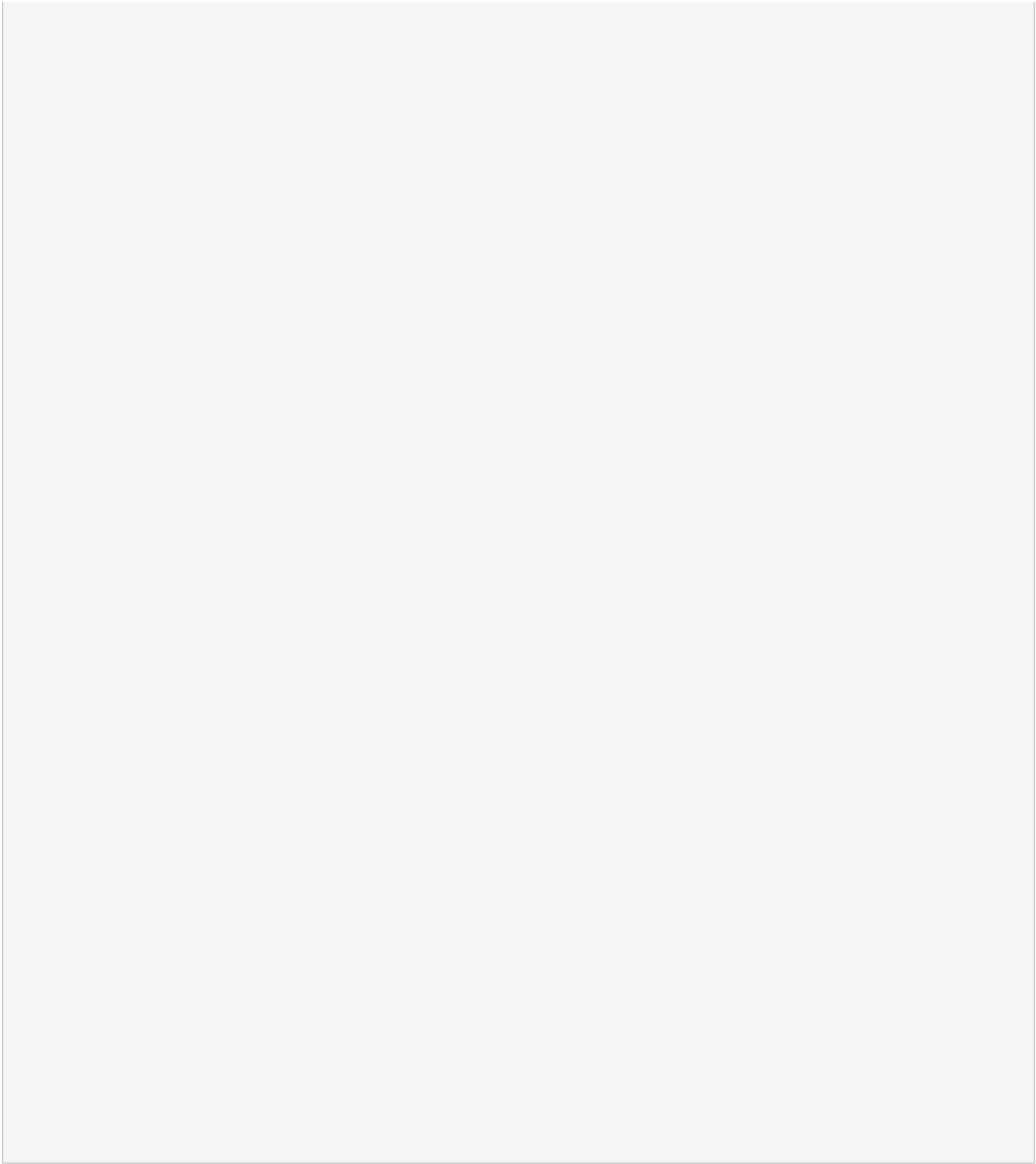


```
0790         elseif numel(val) ~= 1
0791             error('##### MAch mich neu');
0792         end
0793         val = getfield(val, field);
0794     end
0795 end
0796 else
0797     displayed = cNode.getValue;
0798     if iscell(val) && numel(val) == 1
0799         val = val{1};
0800     else
0801         end
0802     return;
0803 end
0804 end
0805
0806
```

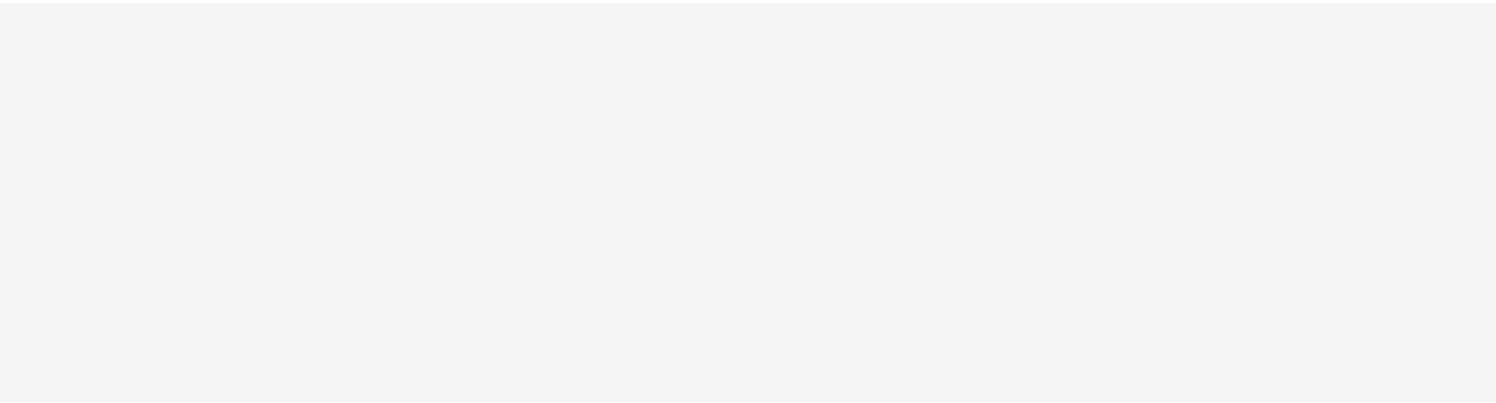


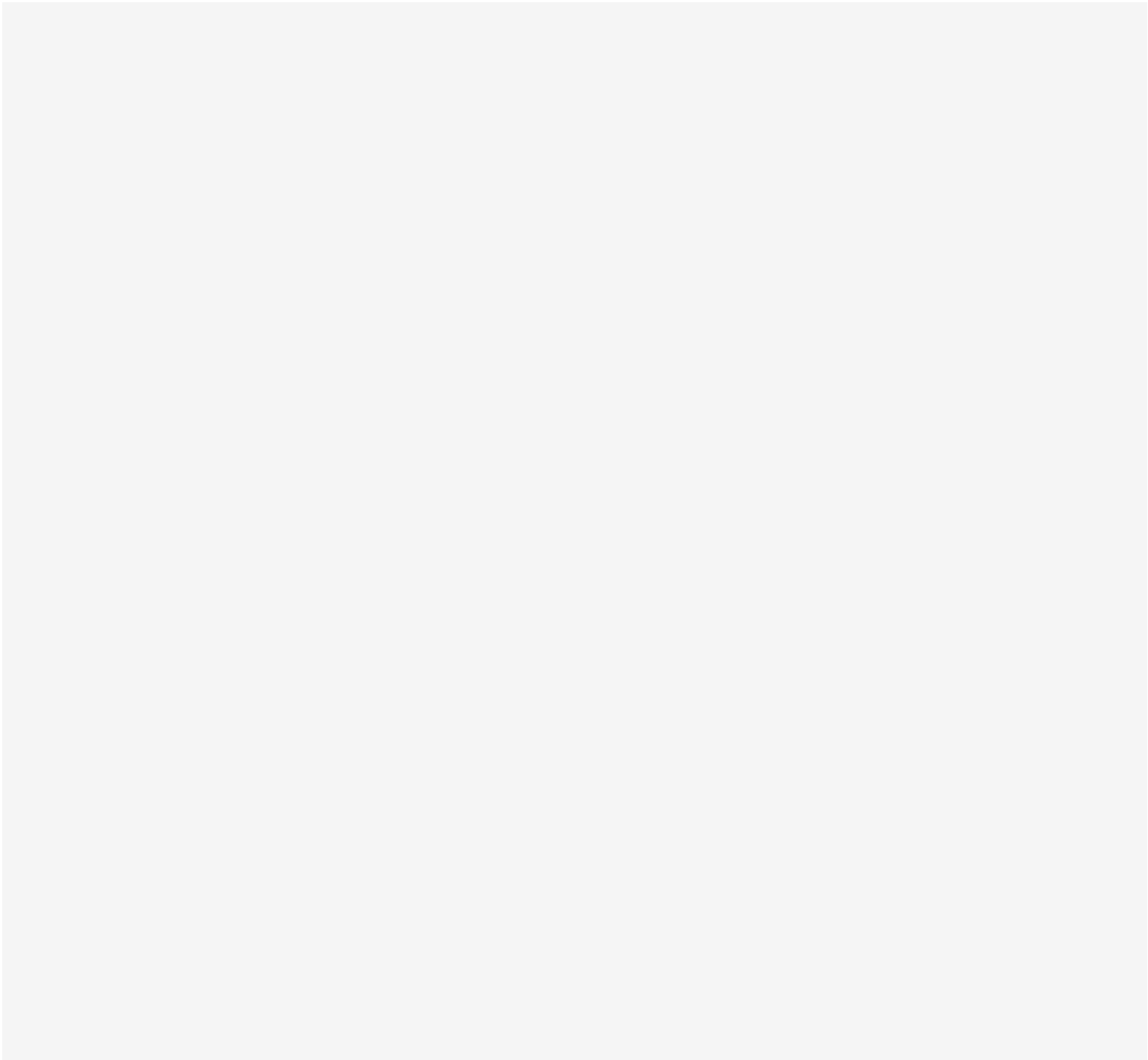






--





```
0039 %% Process inputs
0040
0041 objs = {};
0042 sinfo = [];
0043 for j=1:nargin
0044     if ltpda_isobject(varargin{j})
0045         ins = varargin{j};
0046         if iscell(ins)
0047             for k=1:length(ins)
0048                 objs = [objs {ins{k}}];
0049             end
0050         else
0051             for k=1:length(ins)
0052                 objs = [objs {ins(k)}];
0053             end
0054         end
0055     elseif isstruct(varargin{j})
0056         sinfo = varargin{j};
0057     else
0058         warning(sprintf('!!! input argument %d will be ignored.', j));
0059     end
0060 end
0061
0062 if isempty(objs)
0063     error('### Please input at least one LTPDA object.');
```

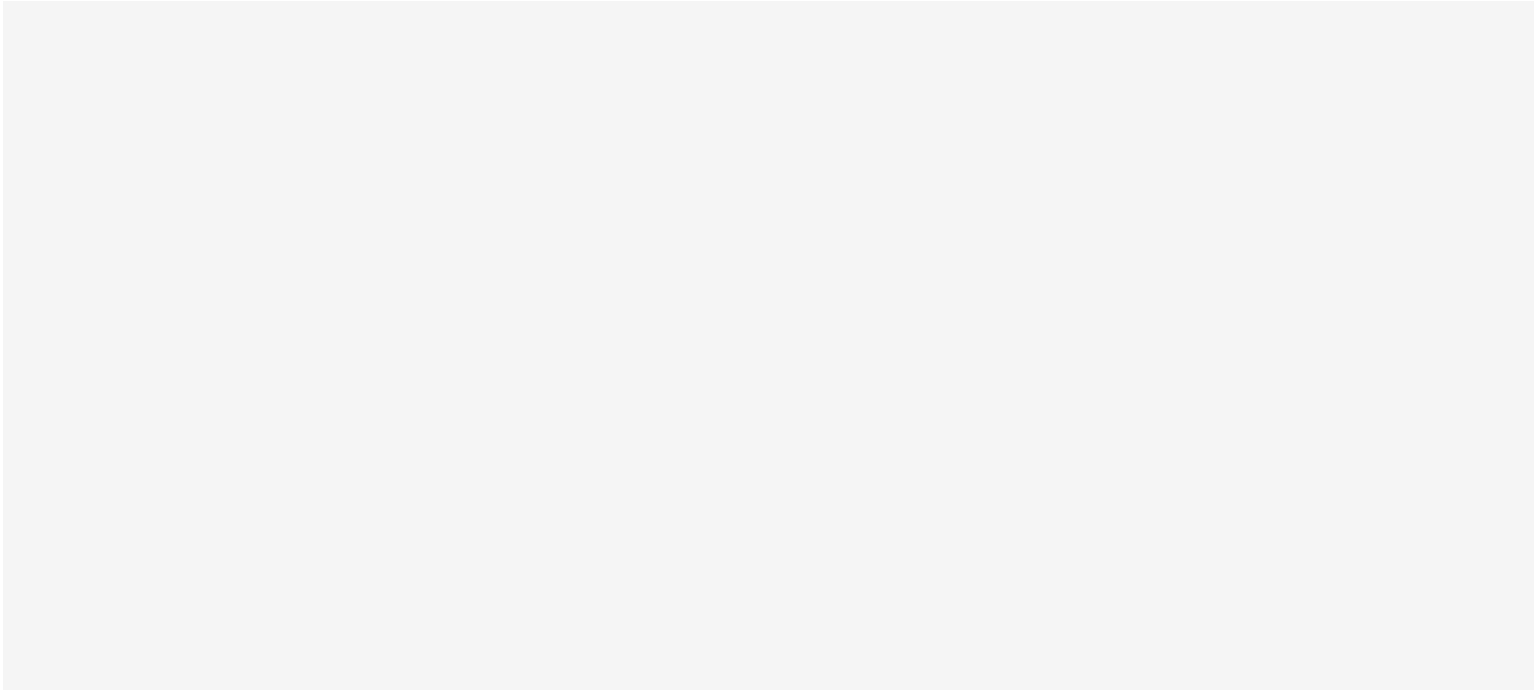

0212

catch

```
0270 % eid = update_object_tables(conn, obj)
0271 % eid = update_object_tables(conn, obj, objid)
0272 %
02730 m 0#0 EMC   BT/LinkP <</MCID10 >>BDC 0.6027 0.245 09410 rg /TT1 1 Tf 11.1962 0 0 1
```

```
0327         error('### failed to make meta-data entry for %s table', cl);
0328     end
0329     %----- FSDATA
0330     case 'fsdata'
0331
```

```
0384         );  
0385         disp(sprintf(' + made meta-data entry for %s', cl));  
0386
```



[< Master index](#)

[< Master index](#)

[Index for classes/@time/private >](#)

Index for classes/@time/private

Matlab files in this directory:

[Contents](#)





☐	win_Nuttall3b	WIN_NUTTALL3B return Nuttall3b window sample for input z.
☐	win_Nuttall4	WIN_NUTTALL4 return Nuttall4 window sample for input z.
☐	win_Nuttall4a	WIN_NUTTALL4A return Nuttall4a window sample for input z.
☐	win_Nuttall4b	WIN_NUTTALL4B return Nuttall4b window sample for input z.
☐	win_Nuttall4c	WIN_NUTTALL4C return Nuttall4c window sample for input z.
☐	win_Rectangular	WIN_RECTANGULAR return rectangular window sample for input z.
☐	win_SFT3F	WIN_SFT3F return SFT3F window sample for input z.
☐	win_SFT3M	WIN_SFT3M return SFT3M window sample for input z.
☐	0 0 1 rg /07 0 0 rg /TTb864m 661 rg /TT0 1 Tf 14 0 0 14 32.5 618.4709 T6.c7/p/ w R>.4549 c	

[!\[\]\(8af806fb1314382d09bc5ec5b767526c_img.jpg\) Master index](#)

 [utp_ltpda_ss_random_generate](#)

utp_ltpda_ss_random_generate subfunction generates a
-1.2 5system