## Purpose:

How to train an MLP neural network in MATLAB environment!

**that is**

For good computations,
we need good formulae
for good algorithms;
and good visualization
for good illustration
and proper testing
of good methods
and succesfull applications!

# About *API*s:

- How to link other programs and MATLAB
- Needed for
  - using existing subroutines or libraries from MATLAB
  - speeding up MATLAB especially for `for`-loops

- API supports, e.g.,
  - usage of C- and Fortran-routines as M-files (MEX)
  - import and export of variables (data) (MAT)
  - Java-support (from version 6 JVM included in environment)
  - communication with PC applications using DDE (*Dynamic Data Exchage*), usage of MATLAB as an AciveX-server

- `mxArray`
  - intrinsic representation of varibles in MATLAB
  - Example: setting `x = 2` creates a structure

```
-----------------------------------------------------
Name: x
Dimensions: 1x1
Class Name: double
-----------------------------------------------------
                            (1,1) = 2
```

- MEX-program gets information from MATLAB workspace:
  - prefix `mx`
  - Example: `mxGetPr(plhs[0])`
- MEX-program puts information to MATLAB workspace:
  - prefix `mex`
  - Example: `mexErrMsgTxt(''Error ...'')`.

```
function [j1,grad] = rosenbrock(x,cof)
%Modification of Erkki Heikkola's func2d.m (TK 11.11.02)
j1 = cof * (x(2) - x(1)^2)^2 + (1 - x(1))^2;
grad = zeros(size(x));
grad(1) = -4 * cof * x(1) * (x(2) - x(1)^2) - 2 * (1 - x(1));
grad(2) = 2 * cof * (x(2) - x(1)^2);
```

```
#include "mex.h"
 void rosenb(double x[],double cof,double *j1,double grad[])
 {
   double t1, t2;
   t1 = x[1] - x[0]*x[0]; t2 = 1e0 - x[0];
   *j1 = cof*t1*t1 + t2*t2;
   grad[0] = -4e0*cof*x[0]*t1 - 2e0*t2; grad[1] = 2e0*cof*t1;
   return;
 }
/* The gateway function. */
 void mexFunction(
   int nlhs, mxArray *plhs[],
   int nrhs, const mxArray *prhs[])
   {
     int n, m;
     double *x, cof, *j1, *grad;
/* Check for proper number of arguments */
     if (nrhs !=2)
       mexErrMsgTxt("Two inputs required.");
     else if ...
/* Treatment of inputs, vector x and scalar cof. */
     x = mxGetPr(prhs[0]);
     n = mxGetN(prhs[0]); m = mxGetM(prhs[0]);
     cof = mxGetScalar(prhs[1]);
/* Allocate and attach the outputs. */
     plhs[0] = mxCreateDoubleMatrix(1,1,mxREAL);
     j1 = mxGetPr(plhs[0]);
     plhs[1] = mxCreateDoubleMatrix(m,n,mxREAL);
     grad = mxGetPr(plhs[1]);
/* Compute the result using C subroutine. */
     rosenb(x,cof,j1,grad);
   }
```

# *Example (cont.)*:

- compilation of c-routine from workspace:

```
>> mex -O rosenbrock_c.c
```

- testing as usual:

```
x = rand(1,2); cof = 1e3;
[j1_m,grad_m] = rosenbrock(x,cof);
[j1_c,grad_c] = rosenbrock_c(x,cof);
norm(j1_m-j1_c), norm(grad_m-grad_c)
```

- calling MATLAB from a mex file (core of *median filter*):

```
void medflt(double u[],double z[],int n,int m,int v1,int v2)
{
int xs, ys, ws, i, j, ind, num_in = 1, num_out = 1;
double *wp, *op, *ip;
mxArray *w, *w1, *out_array[1], *in_array[1];
xs = v1/2; ys = v2/2; ws = v1*v2;
w1 = mxCreateDoubleMatrix(v1,1,mxREAL);
w = mxCreateDoubleMatrix(ws,1,mxREAL);
ind = xs;
for (j = xs; j < m-xs; j++) {
 wp = mxGetPr(w);
 in_array[0] = w;
 for (i = ys; i < n-ys; i++) {
  setw (wp,z,i,j,n,xs,ys); /* values of z within window */
  mexCallMATLAB(num_out,out_array,num_in,in_array,"median");
  op = mxGetPr(out_array[0]);
  u[ind++] = *op;
}
}
return;
}
```

**User Interface** is a single *entity* that helps *user* to perform the desired *tasks*.

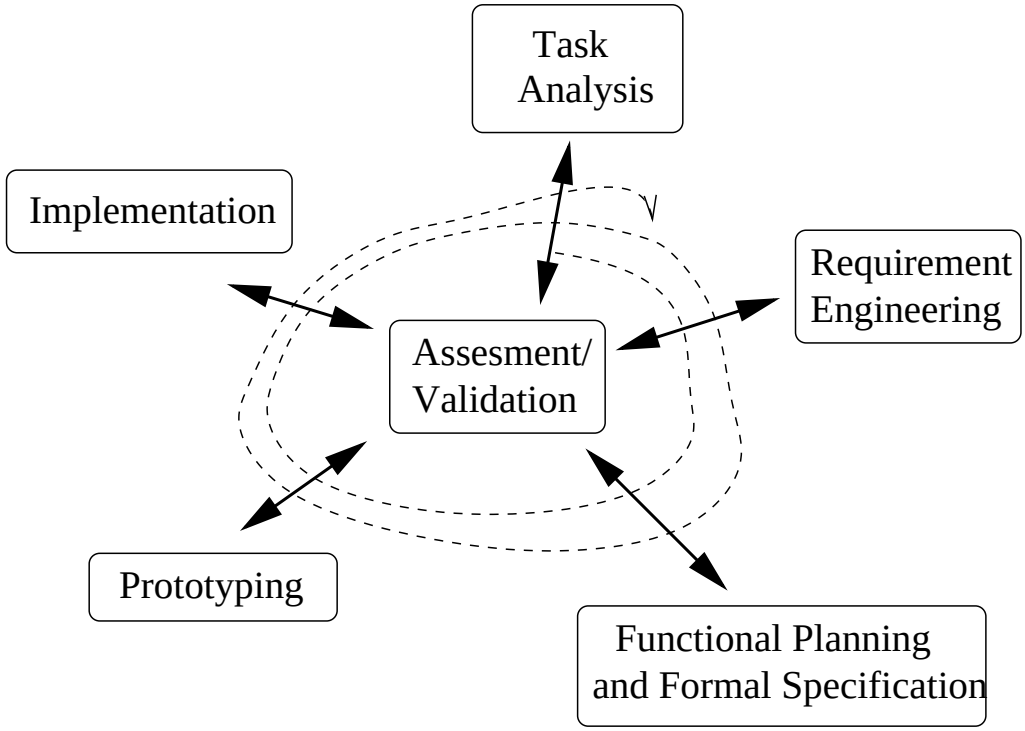
- Key questions:

  (0. Does the user really knows what he/she wants?)

  1. Does the user now where he/she is at every time?
  2. Does the user now what happens next at every time?

- Usability:
  - **learnability**: novice's time to be able to use the system
  - **efficiency**: performance for a skilful user
  - **memorability**: consistency of appearance
  - **faultlessness**: amount and type of errors (that always exist)
  - **satisfaction**: subjective feeling, approachbility
  - **flexibility**: behaviour for new tasks in new environments
  - **effectiveness**: accomplishment of the targets for usage

- Program development paradigm: *Star model* by Preece et al (1994)



*Menus*: familiar part of many window's toolbars

```
>> Hm_l = uimenu(Hx_parent,'Propertyname',Propertyvalue,...)
```

where `Hx_parent` is parent-GO's handle (`figure` or parent-`uimenu`)

• properties

```
Accelerator,BusyAction,ButtonDownFcn,Callback,Checked,
Children,Clipping,CreateFcn,DeleteFcn,Enable,ForeGroundColor
HandleVisibility,HitTest,Interruptible,Label,Parent,Position
Selected,SelectionHighlight,Separator,Tag,Type,UserData,Visible
```

Most important: `Label` visible text on top of one selection,
`CallBack` name of routine to be evaluated when selected

*Contextual menus*: like menus, but can be attached to any GO

```
>> Huix = uicontextmenu(Hf_parent,'Propertyname',PV,...)
```

`Hf_parent` `figure`-handle, attachment to GO by setting `Huix` to
`Contextmenu`-property, creation of actual menu using `uimenu`

*Controls*:

```
>> Hc_l = uicontrol(Hf_parent,'Propertyname',Propertyvalue,...)
```

• properties

```
BackgroundColor,BusyAction,ButtonDownFcn,Callback,Cdata,
Children,Clipping,CreateFcn,DeleteFcn,Enable,Extent,FontAngle,
FontName,FontSize,FontUnits,FontWeight,ForegroundColor,
HandleVisibility,HitTest,HorizontalAlignment,Interruptible,
ListboxTop,Max,Min,Parent,Position,Selected,SelectionHighlight,
SliderStep,String,Style,Tag,TooltipString,Type,UiContextMenu,
Units,UserData,Value,Visible
```

Most important: `Style` that determines control type to

`pushbutton`: mouse-activated action using `ButtonDownFcn`-property
`togglebutton`: like previous, but with only two possible states (`Max` and `Min`)
`radiobutton`: selection of one from a group of mutually exclusive choices
`checkbox`: setting state or attribute 'on' or 'off'
`edit`: editable text for user to modify

`text`: static text, e.g., for announcements
`slider`: (SCROLL BAR) selection of value from a given interval by sliding the mouse
`frame`: for visual separation of a group of `uicontrol`-objects
`listbox`: selection of one or many strings from given list of choices
`popupmenu`: presentation of mutually exclusive choices for the user

*Miscellaneous*-stuff

- mouse capturing and event activation: GO's `Callback`- and `Enable`-properties, `figure`s `CurrentPoint`-, `SelectionType`-, `WindowButtonMotionFcn`- and `WindowButtonDownFcn`-properties and `uicontrol`s `ButtonDownFcn`
- actual event selection and activation using *event queue*
- `gcbo` handle to present *callback*-object and `gcbf` handle to corresponding father-`figure`
- Built-in dialogs for application-user interaction:

| **Function** | **Description** |
|---|---|
| `axlimdlg` | Axes limits dialog box. |
| `dialog` | Create figure for dialog box or GUI. |
| `errordlg` | Error dialog box. |
| `helpdlg` | Help dialog box. |
| `inputdlg` | Input dialog box. |
| `listdlg` | List selection dialog box. |
| `menu` | Menu choice selection dialog box. |
| `msgbox` | Generic message dialog box. |
| `pagedlg` | Page position dialog box. |
| `pagesetupdlg` | Page setup dialog box. |
| `printdlg` | Print dialog box. |
| `printpreview` | Print preview dialog box. |
| `questdlg` | Question dialog box. |
| `uigetfile` | Standard open file dialog box. |
| `uiputfile` | Standard save file dialog box. |
| `uisetcolor` | Color selection dialog box. |
| `uisetfont` | Font and font attributes dialog box. |
| `waitbar` | Display wait bar. |
| `warndlg` | Warning dialog box. |

- GUI realization: Use GUIDE!