draw2d – the MATLAB computer drafting library

Milan Batista

The University of Ljubljana, Faculty of Maritime Studies and Transport, Slovenia

milan.batista@fpp.uni-lj.si

1 Introduction

draw2d is a collection of the MATLAB function that allows one to draw various geometric entities in the plane. The produced picture is passive, i.e., the user » checks the results of the program and change the program when he wishes a different result (Enderle, Kansy, & Pfaff, 1984)«. *draw2d* is designed with the intent to programmatically produce simple engineering pictures. (see Fig 1)

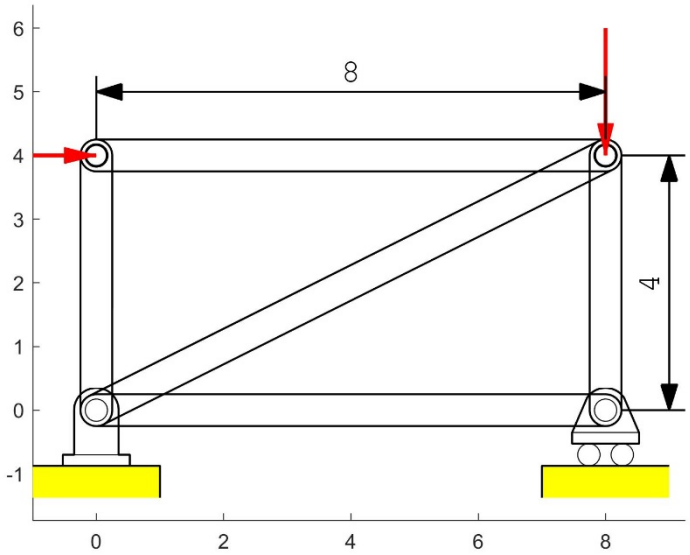

Figure 1. The picture produced by *draw2d* (see Appendix A)

2 Collection description

Currently, the library contains 50+ graphics functions an, three control functions. These are:

- drawing function which includes annotation, dimensions, curves, symbols, and shapes
- control functions

- evaluation functions
- utility functions

The complete set is given in Appendix B.

By *draw2d* functions the user can relatively easely create a picture. The position of graphic entities with respect to each other are facilities by using key-points. These key points are different for each drawing entity and are provided as optional output of drawing function. For example, to connect two blocks with spring one may use the following program

```
drawInit
b1 = drawRect(1,1,0,0,'-pos',4);
b2 = drawRect(1,1,4,0,'-pos',4);
drawSpring1(2,0.4,5,b1.xk(6),b1.yk(6),b2.xk(4),b2.yk(4))
drawSave
```

The produced picture is shown in Fig 2.

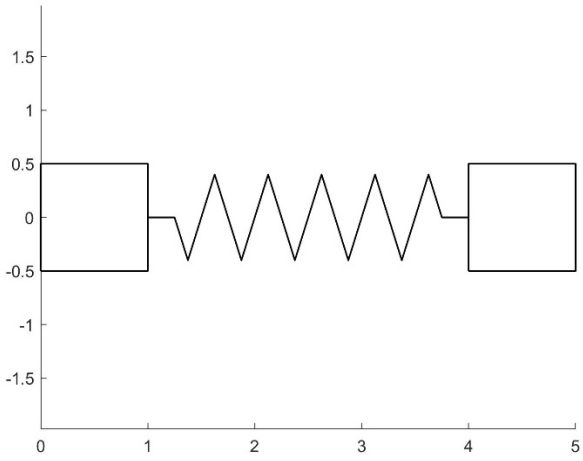

Figure 2. Block connected with spring

There is one design function: catenary, which calculates catenary when the length is given or when sag is given**.**

The typical program structure is

```
drawInit
```

```
% add drawing elements
drawShow
drawSave % if one wants hardcopy of the figure
```

For production an animation one may use the following structure

```
fn = drawInit
for t = tstart:dt:tend
    drawInit(fg)
    % add drawing elements
    drawShow
end
```

To produce animation file one may use the following structure

```
fg=drawInit;
k = 0;
for t=start:dt:tend
    drawInit(fg);
    % add drawing elements
    drawnow
    k = k + 1;
    M(k) = getframe;
end
v = VideoWriter('myfile.avi');
open(v)
writeVideo(v,M)
close(v)
```

For example, if one wants to draw a block on spring

3 Technical details

3.1 Output attributes

draw2d manage its own set of output attributes. However, one can bypass it by specifying the MATLAB attributes. For example:

```
drawSet('LineColor','k')
drawLine(x1,y1,x2,y2)
```

will draw a black line however

```
drawLine(x1,y1,x2,y2,'r:')
```

will draw doted red line.

3.2 Functions structure

All drawing function share the same structure

```
drawXXXX( what, where,[ how])
```

what depends on entity type.

where ara reference point coordinates xc, yc and optionally coordinates rotation angle *rot*

how contains additional data and MATLAB line or fill specification

Among additional data are

'-seg,' '-sec', draw section or segment of circle, ellipse, hyperbola or parabola

'-v',v1,v2 – for rect, ngon: allow only part of rect or ngon to be drawn

'-np',np – controls the number of points that approximate the curve

'-delta',dx,dy or '-polar',r,th define a line, spring to be drawn using relative or polar coordinates.

3.3 Notes:

- The direction of vertices in Ngon, rectangle, polygon are in CCLW direction
- angles are in degrees.
- Arrowhead forms are taken from IGES standard.
- Function for calculation of B-spline curve and Bezier curve are based on subroutines from (Rogers & Adams, 1976)

Appendix A

```
drawInit
ht = 0.5;hs = 0.7*ht;F=1;d1=8;d2=4;
% draw truses
drawCanoe1( ht, 0,0,d1,0 )
drawCanoe1( ht, d1,0,d1,d2 )
drawCanoe1( ht, 0,0,d1,d2 )
drawCanoe1( ht, 0,d2,d1,d2 )
drawCanoe1( ht, 0,0,0,d2 )
% draw pins
drawDonut(hs,0.9*hs,0,d2)
drawDonut(hs,0.9*hs,d1,d2)
% draw supports
s1 = drawSupport(1,hs,0,0);
s2 = drawSupport(2,hs,d1,0);
% draw floor
fillRect('y',d2*ht,ht,s1.xk(3),s1.yk(3),'-pos',d1)
fillRect('y',d2*ht,ht,s2.xk(3),s2.yk(3),'-pos',d1)
drawRect(d2*ht,ht,s1.xk(3),s1.yk(3),'-pos',d1,'-v',2,4)
drawRect(d2*ht,ht,s2.xk(3),s2.yk(3),'-pos',d1,'-v',3,1)
% draw forces
drawForce(F,180,0,0,d2,'-ad',0.5,'r','LineWidth',2)
drawForce(2*F,90,0,d1,d2,'-ad',0.5,'r','LineWidth',2)
% dimension
drawVDim(3,0.8*ht,ht/2,d1,0,d1,d2,d1+1,d2/2);
drawHDim(3,0.8*ht,ht/2,0,d2,d1,d2,d1/2,d2+1);
drawShow
drawSave
```

Appendix B

```
% draw2d - 2d drawing library
%================================
% History:
%   MB, created, 7.5-3.6.2019
%
% Ver: 1.0
%
%==============================
% This is a collection of MATLAB functions that allow to programaticaly
% create simple 2d drawing that can be used in engineering.
%==============================
% NOTE 1:
%  For proper use one must set
%
%   axis equal
%
%   This is done by drawInit
%
% NOTE 2:
%   The library must be set on MATLAB search path. For example: if
%   file structure is:
%       myMatlab -> draw2d
%                -> examples/ test01.m
%
% Then first line of test01.m should be
%   addpath('..\draw2d');
%
%==============================
%
% Drawing functions:
% ------------------
%
% Anotations
%   p = drawArcArrow( form, ad1, ad2, xc, yc, r, sang, ang, varargin)
%   p = drawArrow( form, ad1, ad2, x1, y1, varargin)
%       drawArrowhead( form, ad1, ad2, xh, yh, rot, varargin)
%   p = drawAxes( form, ad, a, xc, yc, varargin)
%   p = drawCross(  d1, d2, xc, yc, varargin )
%   p = drawLeader( form, ad1, ad2, xh, yh, xt, yt, varargin)
%   drawText( x, y, str, varargin)
%
% Curves:
%   p = drawBezier( xv, yv, varargin )
%   p = drawBspline( c, xv, yv, varargin )
%   p = drawCatenary( x1, y1, dx, dy, varargin )
%   p = drawCircle( xc, yc, r, varargin )
%   p = drawEllipse( a, b, xc, yc, varargin )
%   p = drawHyperbola( t1, t2, a, b, xc, yc, varargin )
%   p = drawLine( x1, y1, varargin )
%   p = drawParabola( t1, t2, f, xc, yc, varargin )
%   p = drawPolygon( xv, yv, varargin)
%   p = drawPolyline( xv, yv, varargin)
%   p = drawSpiral( a, sang, eang, varargin)
%   p = drawSpline( ctype, xv, yv, varargin )
%
% Dimensions
%   drawAngDim( form, ad1, ad2, xc, yc, sang, ang, rt, at, varargin)
%   drawAngDim3p( form, ad1, ad2, xc, yc, x1, y1, x2, y2, rt, at, varargin)
%   drawDim( form, ad1, ad2, x1, y1, x2, y2, xt, yt, varargin)
```

```
%   drawHDim( form, ad1, ad2, x1, y1, x2, y2, xt, yt, varargin)
%   drawVDim( form, ad1, ad2, x1, y1, x2, y2, xt, yt, varargin)
%
% Shapes:
%   p = drawCanoe1( ht, x1, y1, varargin )
%   p = drawDonut(  d1, d2, xc, yc, varargin )
%   p = drawNgon( n, r, xc, yc, varargin )
%   p = drawRect( wd, ht, xr, yr, varargin )
%   p = drawSpring1(form,r,nc,x1,y1,varargin)
%   p = drawSupport(type, r,x0,y0,varargin)
%
% Symbols:
%   drawCOG( dia, xc, yc, varargin)
%   p = drawPoint( form, d, xp, yp, varargin )
%
% Force and load:
%   p = drawForce(F,th,d,xr,yr,varargin)
%   p = drawLoad(wd,h1,h2,nc,d,xr,yr,varargin)
%
% Fill area
%------------
%   p = fillCanoe( c, wd, ht, xr, yr, varargin )
%   p = fillCanoe1( c, ht, x1, y1, varargin )
%   p = fillCircle( c, xc, yc, r, varargin )
%   p = fillDonut(  c1, c2, d1, d2, xc, yc, varargin )
%   p = fillEllipse( c, a, b, xc, yc, varargin )
%   p = fillNgon( c, n, r, xc, yc, varargin )
%   p = fillPolygon( c, xv, yv, varargin)
%   p = fillRect( c, wd, ht, xr, yr, varargin )
%
% Output attributes
% -----------------
%   v = drawGet(name)
%   fn = drawInit(figNum)
%   drawSet(varargin)
%
% Figure window functions:
% ---------
%   fn = drawInit(figNum)
%   drawLimits( xmin, xmax, ymin, ymax)
%
% Hard copy:
% ----------
%   drawSave(fileName,varargin)
%
% Evaluation functions:
% ---------------------
%   [x, y] = evalBezier( xv, yv, tt)
%   [x, y] = evalBspline( c, xv, yv, np)
%   [x, y] = evalCatenary( lambda, x1, y1, s1, s)
%   [x, y] = evalCircle( xc, yc, r, th)
%   [x, y] = evalEllipse( a, b, xc, yc, rot, th))
%   [x, y] = evalHyperbola( a, b, xc, yc, rot, t)
%   [x, y] = evalLine( x1, y1, x2, y2, t)
%   [x, y] = evalNgon( nv, r, xc, yc, rot, v1, v2)
%   [x, y] = evalParabola( f, xc, yc, rot, t)
%   [x, y] = evalRect( wd, ht, xr, yr, rot, ip, v1, v2 )
%   [x, y] = evalSpiral( a, xc, yc, rot, th)
%   [x, y, th] = evalSpline(ctype, xv, yv, u, v, np)
%
% Transformation functions
```

```
%-------------------------
%   [xx,yy] = trRot2d( x, y, x0, y0, theta)
%   [xx,yy] = trScale2d( x, y, dx, dy)
%   [xx,yy] = trShift2d( x, y, dx, dy)
%
% Other functions
%------------------------
%
%   p = PCA2d( X, Y)
%   [x,y] = swap(x,y)
```

References

Enderle, G., Kansy, K., & Pfaff, G. (1984). *Computer graphics programming GKS - the graphics standard*. Berlin a.o.: Springer.

Rogers, D. F., & Adams, J. A. (1976). *Mathematical elements for computer graphics*. New York a.o.: McGraw-Hill.