A MATLAB digitizer-integrator

January 2006, updated August 2013

Adrian B. Biran, Technion – Faculty of Mechanical Engineering,

Companion software for the book *Ship Hydrostatics and Stability*, revised reprint, Oxford, UK, Elsevier Butterworth-Heinemann. Updated for second edition, August 2013, coauthor Rubén López-Pulido.

1 Introduction

Many engineering problems require the coordinates of points on a drawing, map or graph. Typical examples can be found in Naval Architecture. Before the advent of computers such tasks were carried on manually. When computers were introduced in engineering, the task was performed with the aid of instruments called *digitizers* or *digitizing tablets* and appropriate software. This software was part of many CAD packages. Today software such as MATLAB enables us to pick up the coordinates of points appearing in

graphic files. Graphic files can be produced by scanning drawings, maps or graphs; in MATLAB they can also be obtained from a plot by means of the command `print`. For example, with a figure window open we can store it as a JPG file by means of the command

```
print -djpeg myfile.eps
```

where instead of *myfile* we can write any name we choose.

Usually, the digitized points define a plane figure whose geometrical properties must be calculated. This task can be performed by numerical integration using, for example, the trapezoidal or Simpson's rule. Since the invention of the planimeter by Amsler, around 1854, Naval Architects used mechanical analog computers, specifically planimeters, and later integrators and integraphs to calculate the geometrical properties by following a closed contour on the lines drawing or on intermediate, specially-prepared graphs.

The web describes or advertises software designed for digitizing features appearing in graphic files and for calculating some properties of plane figures defined by the digitized points. To mention just a few examples, we can refer to the *Engauge Digitizer - Digitizing software* (http://digitizer/sourceforge.net/), *Digitize* (http://www.rockware.com/catalog/pages/digitize.html).

In this document we describe a MATLAB programme designed to digitize points appearing in a graphic file of a type that can be imported into MATLAB, such as `jpg`, `bmp`, or `tif`. The programme uses MATLAB built-in functions, among them `imread` to import the graphic file, `image` to display

the image to be digitized, and `ginput` to pick up point coordinates. For interactive input the programme uses dialogue boxes predefined in MATLAB. The output consists of two MATLAB arrays. The first array, called `REAL` within the programme, contains the real-world coordinates of the digitized points. The second array, named `PROPERTIES` within the programme, contains the following geometrical properties of the polygonal figure

1. polygonal area;
2. x-coordinate of centroid;
3. y-coordinate of centroid
4. second moment about barycentric y-axis
5. second moment about barycentric y-axis
6. product of inertia about barycentric axes
7. radius of gyration about barycentric y-axis
8. radius of gyration about barycentric x-axis
9. maximum principal moment of inertia
10. minimum principal moment of inertia
11. maximum radius of gyration about principal axes
12. minimum radius of gyration about principal axes

These, and additional data are also printed on a file called 'digitizer.out'. The output stored in the array `PROPERTIES` can be further processed by any of the available MATLAB tools. A synonym of the term '*barycentric*' is '*centroidal*'.

To calculate the area of the polygonal figure, the programme calls the MATLAB built-in function `polyarea`. The calculation of the perimeter is based on repeated invocations of the MATLAB function `norm`. The first and second moments of the polygonal area are calculated by means of Green's theorem. A statement of this theorem is

$$\oint_L M dx + N dy = \iint_S \left(\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y} \right) dx dy$$

where L is the boundary of the area S (see, for example, Finney, 1988, pp. 1027–39, or O'Neil, 2003, p. 565–9). Functions M and N suitable for the calculation of the geometrical properties can be found, for example, in a paper by Ken Turkowksi available on the web. The boundary L must be followed in the counterclockwise sense, that is so as to have the area A always on the left of the cursor.

If a figure has an axis of symmetry, that axis is also a principal axis. Small errors of calibration and/or digitization can yield wrong principal axes. The error can be important if the two moments of inertia about the barycentric axes are nearly equal.

The function `digitizer` calls the function `point`. Therefore, the file of the latter function must be included in the same directory as the function

`digitizer`.

2 Instructions of use

The programme can be called with the name and type of the graphic file as argument. The string composed of these data must be included between quotes, for example

```
digitizer('myfile.jpg')
```

If the programme is called simply as

```
digitizer
```

a dialogue box appears and the user has to choose the graphic file among those available in a list dialogue box that pops up. In the first step the user has to calibrate the figure. The programme prompts the user to pick up three non-collinear points by bringing the crosshair over each point and pressing the mouse left button. Then the programme prompts the user to enter the real-world coordinates of the same point. To shift from the window of the $x-$ coordinate to that of the $y-$ coordinate the user must bring the mouse pointer over the latter window. Alternatively, it is possible to press simultaneously the keys `Tab` and `Ctrl`. The user has to confirm the data by pressing the `OK` button.

Next, the programme prompts the user to start digitizing. After picking up a point the user can choose between continuing to another point or ending

the digitization process. To help the user in following the correct order, digitized points are marked by red points. The boundary of the figure must be followed so as to have the area on the left of the cursor. Digitizing in the opposite direction leads to wrong results!

When the digitizing process is ended, the programme displays a dialogue box from which it is possible to choose one of the following options;

1. show axes of coordinates;
2. find the real-world coordinates of a chosen point;
3. find the distance between two chosen points;
4. find the polygonal area;
5. find the geometric properties of the digitized plane figure;
6. exit the programme.

3 An example

The file `TtestZ.m` (stored on the same site with this document) draws a Z-shaped figure as exemplified in Beer and Johnston (1998). The user may print the figure and scan it to produce a graphic file. However, on the same site the reader can find the file `Ztest.jpg` ready for use with the function `digitizer`. See Figure 1. Assuming that the dimensions are given in m, the properties of the above figure are

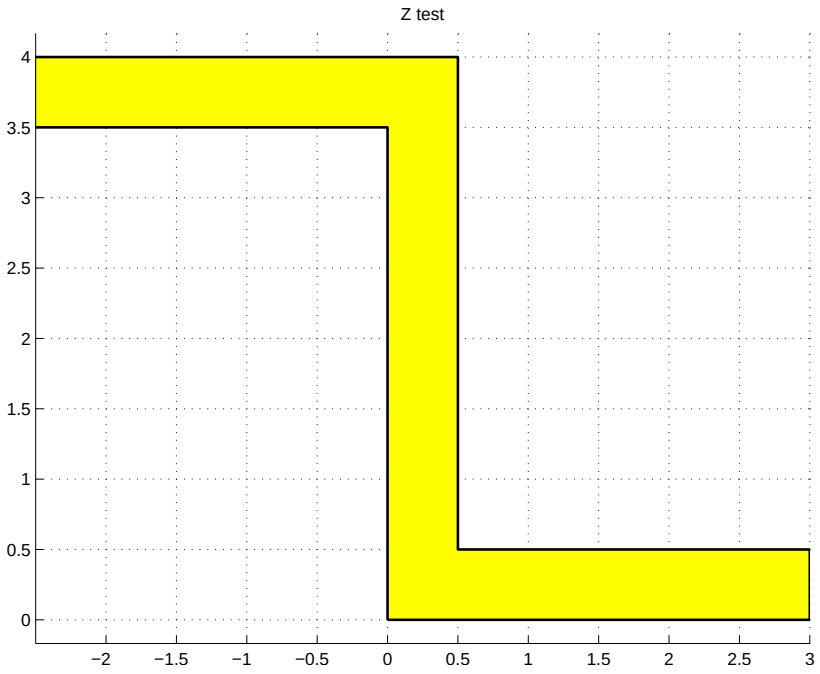

Figure 1: A Z-shaped figure

area	4.500	m^2
x-coordinate of centroid	0.250	m
y-coordinate of centroid	2.000	m
moment of inertia about given y-axis	7.250	m^4
moment of inertia about given x-axis	28.375	m^4
product of inertia about given axes	-4.313	m^4
moment of inertia about barycentric y-axis	6.969	m^4
moment of inertia about barycentric x-axis	10.375	m^4
product of inertia about barycentric axes	-6.563	m^4
angle of principal axes	37.726	degrees
maximum moment of inertia about principal axes	15.452	m^4
minimum moment of inertia about principal axes	1.892	m^4

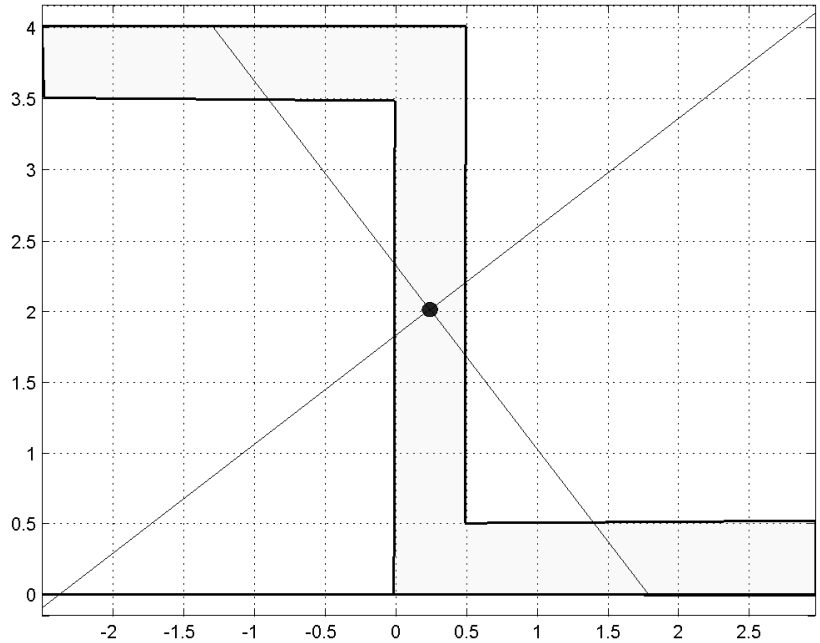

Figure 2: The digitized figure and its principal axes

The figure defined by the digitized points and its principal axes are plotted as in Figure 2. The MATLAB script `momentsZ.m` calculates the properties of the Z-shaped figure by decomposing it into three rectangular areas. The output is displayed on the screen and can be used to validate the results yielded by the MATLAB digitizer-integrator.

Above we call *barycentric axes* the axes with the origin of coordinates in the centroid of the figure. Beer and Johnston (1998) call these axes *centroidal*. To exercise, the reader can call the function with the command `digitize('Ztest.jpg')`, digitize it and choose the fifth post-processing option — *find geometric properties* — and compare the results with the data

presented above. Usually, there will be some discrepancies due to unavoidable errors in picking up the exact points. However, results acceptable in practical engineering work can be obtained when working attentively.

4 Bibliography

Beer, F.P., and Johnston, E.R. Jr (1998), *Vector mechanics for engineers*, 3d SI metric edition, Toronto, McGraw-Hill Ryerson.

Finney, R.L. (1988), *Calculus and analytic geometry*, 7th ed., Reading-Massachusetts, Addison Wesley publishing Company.

O'Neil, P.V. (2003), *Advanced engineering mathematics*, 5th ed., Pacific Grove - CA, Thomson Brooks/Cole.

Turkowski, K. (1997), *Computing 2D polygon moments using Green's Theorem*, http://www.worldserver.com/turk/computergraphics/Moments.pdf.