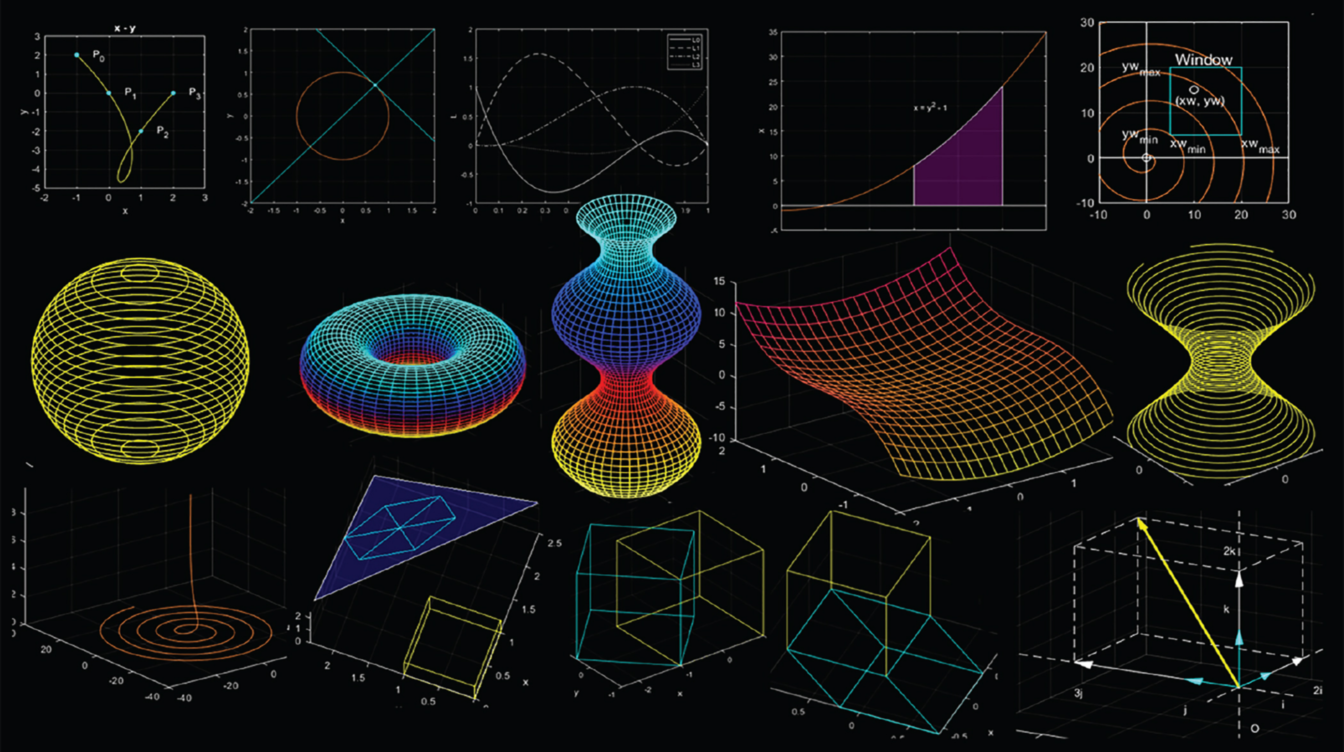

FUNDAMENTALS OF GRAPHICS USING MATLAB®

Ranjan Parekh

CRC Press
Taylor & Francis Group

Fundamentals of Graphics Using MATLAB®

Fundamentals of Graphics Using MATLAB®

Ranjan Parekh

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper

International Standard Book Number-13: 978-0-367-18482-7 (Hardback)

Contents

Preface

This book introduces fundamental concepts and principles of 2D and 3D graphics and is written for under- and postgraduate students studying graphics- and/or multimedia-related subjects. Most of the books on graphics use C programming environments to illustrate practical implementations. This book deviates from this common practice and illustrates the use of MATLAB® for the purpose. MATLAB by MathWorks, Inc. is a data analysis and visualization tool suitable for algorithmic development and simulation applications. One of the advantages of MATLAB is that it contains large libraries of in-built functions which can be utilized to reduce program-development time as compared to other contemporary programming environments. It is assumed that the student has basic knowledge of MATLAB, especially various matrix operations and plotting functions. The MATLAB codes have been provided as answers to specific examples, and the reader can simply copy and paste the codes to execute them. In general, the codes display answers to expected results like equation of curves, blending functions, and transformation matrices as well as plot the final results to provide a visual representation of the solution. The objectives of this book are, first, to demonstrate how MATLAB can be used to solve problems in graphics and, second, to help the student gain an in-depth knowledge about the subject matter through visual representations and practical examples.

This book is roughly divided into two parts: 2D graphics and 3D graphics, although in some places both of these concepts overlap mainly to highlight the differences between them or for using simpler concepts to prepare the reader for more complex ones.

The first part of this book mainly deals with concepts and problems related to 2D graphics, and spans over five chapters: (1) Interpolating Splines, (2) Blending Functions and Hybrid Splines, (3) Approximating Splines, (4) 2D Transformations, and (5) Spline Properties.

Chapter 1 provides an introduction regarding the various types of interpolating splines and their representations using polynomials. The theoretical concepts about how spline equations are derived and the matrix algebra involved are discussed in detail followed by numerical examples and MATLAB codes to illustrate the processes. Most of the examples are followed by graphical plots to enable the reader visualize how the equations get translated into corresponding curves given their start points, end points, and other related parameters. The chapter also highlights the differences in these procedures for both standard or spatial form and parametric form of the spline equations using linear, quadratic, and cubic variants.

Chapter 2 introduces the concept of blending functions and how these functions are used to derive equations for hybrid splines which pass through only a subset of their control points or where conditions other than control points are used for deriving their equations. Specifically, the chapter deals with the Hermite spline, Cardinal spline, Catmull–Rom spline, and Bezier spline. For Bezier splines, both the quadratic and cubic variants are discussed along with Bernstein polynomials used to formulate their blending functions. As in other chapters, the theoretical concepts are followed by numerical examples, MATLAB codes and graphical plots for visualization. The chapter ends with a discussion about how one spline type can be converted to another.

Chapter 3 discusses how polynomial equations are derived for approximating splines that do not pass through any of their control points and how their blending functions are computed. Specifically, the chapter provides detailed discussions about the Cox de Boor algorithm and how it can used to derive equations for linear, quadratic, and cubic B-splines. Essentially, B-splines consist of multiple curve segments with continuity at join points. Values of the parametric variable at the join points are stored in a vector called the knot vector. If the knot values are equally spaced, then the resulting spline is called uniform B-spline; otherwise, it is referred to as non-uniform. B-splines are called open-uniform when knot vector values are repeated. The chapter provides representations of the knot vector and illustrates how the spacing in the vector generates the above-mentioned variants. As before, the theoretical concepts are followed by numerical examples, MATLAB codes, and graphical plots for visualization.

Chapter 4 formally introduces a 2D coordinate system and then lays the foundations of a homogeneous coordinate system using which all the transformations can be represented in a uniform manner. Two-dimensional transformations are used to change the location, orientation, and shapes of splines in 2D plane. These transformations are translation, rotation, scaling, reflection, and shear applied individually or in combination of two or more; hence, they are known as composite transformations. Given known coordinates of a point, each of these transformations is represented by a matrix which when multiplied to the original coordinates produces a new set of transformed coordinates. The transformation matrices are first derived, and then their applications are illustrated using examples, MATLAB codes, and graphical plots. Both affine and perspective transformation types are discussed. The chapter ends with a discussion on viewing transformations used for mapping a window to a viewport, and coordinate system transformation used for mapping between multiple coordinate systems.

Chapter 5 enumerates some of the common properties of splines and how these can be calculated from spline equations. First, it discusses the critical points namely minimum and maximum of spline curves. Additionally for splines of degree 3 or more, the point of inflection (POI) is of interest. Next, it discusses how the tangent and normal to a spline curve can be calculated. The tangent to a curve is the derivative of the curve equation, while the normal is the line perpendicular to the tangent. The third property is calculation of length of a spline curve between any two given points, both for spatial and parametric equations. The fourth property is to calculate the area under a curve, which is bounded by a primary axis and two horizontal or vertical lines. An extension to this is calculation of area bounded by two curves. The fifth property is calculation of centroid of an area, the point of the center of gravity for plates of uniform density. The chapter ends with a discussion on interpolation and curve fitting for data points and a list of some commonly used built-in MATLAB functions for plotting 2D graphs and plots.

The second part of this book focuses on concepts and problems related to 3D graphics and spans over the remaining four chapters namely (6) Vectors, (7) 3D Transformations, (8) Surfaces, and (9) Projections.

Chapter 6 introduces the concept of vectors and their mathematical representations in 2D and 3D spaces. Vectors involve both magnitude and direction. They are represented in terms of orthogonal reference components of unit magnitudes along the primary axes together with a set of scaling factors. The chapter discusses how vectors can be added and multiplied together. Vector products can either be scalar, called a dot product, or vector, called a cross product. Using these concepts, the chapter then provides details of how vector equations of lines and planes can be derived. Next, the chapter discusses how vectors can be aligned to specific directions and finally how vector equations can be represented using homogeneous coordinates. The chapter ends with a section on how the tangent vector and the normal vector can be calculated for a curve. As before, the theoretical concepts are followed by numerical examples, MATLAB codes, and graphical plots for visualization.

Chapter 7 demonstrates how 3D transformations can be treated as extensions of 2D transformations. These are used to change the location, orientation, and shapes of splines in 3D space. These transformations are translation, rotation, scaling, reflection, shear applied individually or in combination of two or more, known as composite transformation. This chapter formally introduces a 3D coordinate system and then uses homogeneous coordinates to derive transformation matrices for the above operations. Their applications are then illustrated using examples, MATLAB codes and graphical plots for visualization. The latter part of the chapter deals with vector alignment in 3D space and uses these concepts to derive rotation matrices in 3D space around vectors and arbitrary lines.

Chapter 8 takes a look at how surfaces can be created and represented using parametric and implicit equations, and how the nature of the surface depends on the parameters of the equations. Depending on creation process, surfaces can be categorized as extruded and surfaces of revolution, both of which are discussed with examples and graphical plots. The chapter then takes a look at how tangent planes of surfaces can be computed and provides methods for computing area and volume of surfaces. The latter part of the chapter deals with surface appearances namely how textures can be mapped on surfaces and how illumination models can be used to determine brightness intensities at a point on the surface. The chapter ends with a discussion on some commonly used built-in MATLAB functions for plotting 3D graphs.

Chapter 9 studies various types of projections and derives matrices for each. Projection is used to map a higher-dimensional object to a lower-dimensional view. Projection can be of two types: parallel and perspective. In parallel projection, projection lines are parallel to each other, while in perspective projection, projection lines appear to converge to a reference point. Parallel projection can again be of two types: orthographic and oblique. In parallel orthographic projection, the projection lines are perpendicular to the view plane, while in parallel oblique projection, the projection lines can be oriented at any arbitrary angle to the view plane. Usually for 3D projection, parallel orthographic projection can also be sub-divided into two types: multi-view and axonometric. In multi-view projection, the projection occurs on the primary planes i.e. *XY*-, *YZ*-, or *XZ*-planes, while in axonometric projection, the projection occurs on any arbitrary plane. The chapter illustrates each type of projection using examples, MATLAB codes, and graphical plots for visualization.

Each chapter is followed by a summarized list of salient points discussed in the chapter. A set of review questions and a list of practice problems are provided at the end of each chapter for self-evaluation. This book contains more than 90 solved numerical examples with their corresponding MATLAB codes and an additional 90 problems given for practice. Readers are encouraged to execute the codes given in the examples and also write their own codes to solve the practice problems. Most of the MATLAB codes given in this book will require MATLAB version 2015 or later to execute properly. Some of the functions mentioned have been specifically introduced from version 2016 and these have been mentioned at the appropriate places. The usage of about 70 different MATLAB functions related to graphics and plotting have been demonstrated in this book and a list of these functions with a short description is provided at Appendix I. Readers are asked to use MATLAB help utilities to get further information on these. The MATLAB codes are written in a verbose manner for a better understanding of the readers who are new to the subject matter. Some of the codes could have been written in a more compact manner but that might have reduced their comprehensibility. Around 170 figures have been included in this book to help the readers get proper visualization cues of the problems especially for 3D environments. Answers to the practice problems are provided in Appendix II.

All readers are encouraged to provide feedback about the content matter of the book as well as any omissions or typing errors. The author can be contacted at ranjan_parekh@yahoo.com.

Ranjan Parekh
Jadavpur University
Calcutta 700032, India
2019

Author

Dr. Ranjan Parekh, PhD (Engineering), is Professor at the School of Education Technology, Jadavpur University, Calcutta, India, and is involved with teaching subjects related to Graphics and Multimedia at the post graduate level. His research interests include multimedia information processing, pattern recognition, and computer vision. He is also the author of *Principles of Multimedia* (McGraw Hill, 2012; http://www.mhhe.com/parekh/multimedia2).

CHAPTER 1

Interpolating Splines

1.1 INTRODUCTION

Splines are irregular curve segments with known mathematical properties. Splines are frequently encountered in vector graphics when graphic objects are required to have a defined shape in 2D planes (Figure 1.1a) or 3D space (Figure 1.1b) or moved along a specified path (Figure 1.1c). Based on the coordinates of some of the points on the curves, or slopes of lines along the curves, the graphics system needs to calculate a mathematical representation of the curve before storing them onto a disk. This representation usually takes the form of "vectors" or a series of values stored in matrices. The values are calculated using an orthogonal 2D coordinate system consisting of the origin, X-axis, and Y-axis. These coordinate axes are often called the primary or principal axes.

The term "spline" has been derived from the ship building industry where it is used to refer to wooden planks bent between wooden posts for building the curved hull of ships (O'Rourke, 2003). The location of the fixed posts controlled the shape of the plank. In graphics, we use specific points along the spline curve to control the shape of the spline and hence they are aptly referred to as "control points," shortened as CPs. Depending on the relationship between the CPs and the actual curves, the splines can be broadly categorized into three types: (1) interpolating splines, where the spline actually goes through the CPs; (2) approximating splines, where the spline goes near the CPs but not actually through them; and (3) hybrid splines, where the spline goes through some of the CPs but not through all (Hearn and Baker, 1996) (see Figure 1.2).

Splines are mathematically modeled using polynomials. Polynomials are expressions constructed from variables and constants, and involve addition, subtraction, multiplication, and non-negative integer exponents. A polynomial can be 0 (zero) or a sum of non-zero terms. Each term consists of a constant, called coefficient, multiplied by a variable. The exponent of the variable is called its degree. The first example is a valid polynomial, but the second example is not, because the variable is associated with a division operation and also because of the fractional exponent. The general nth degree polynomial is shown in the third example.

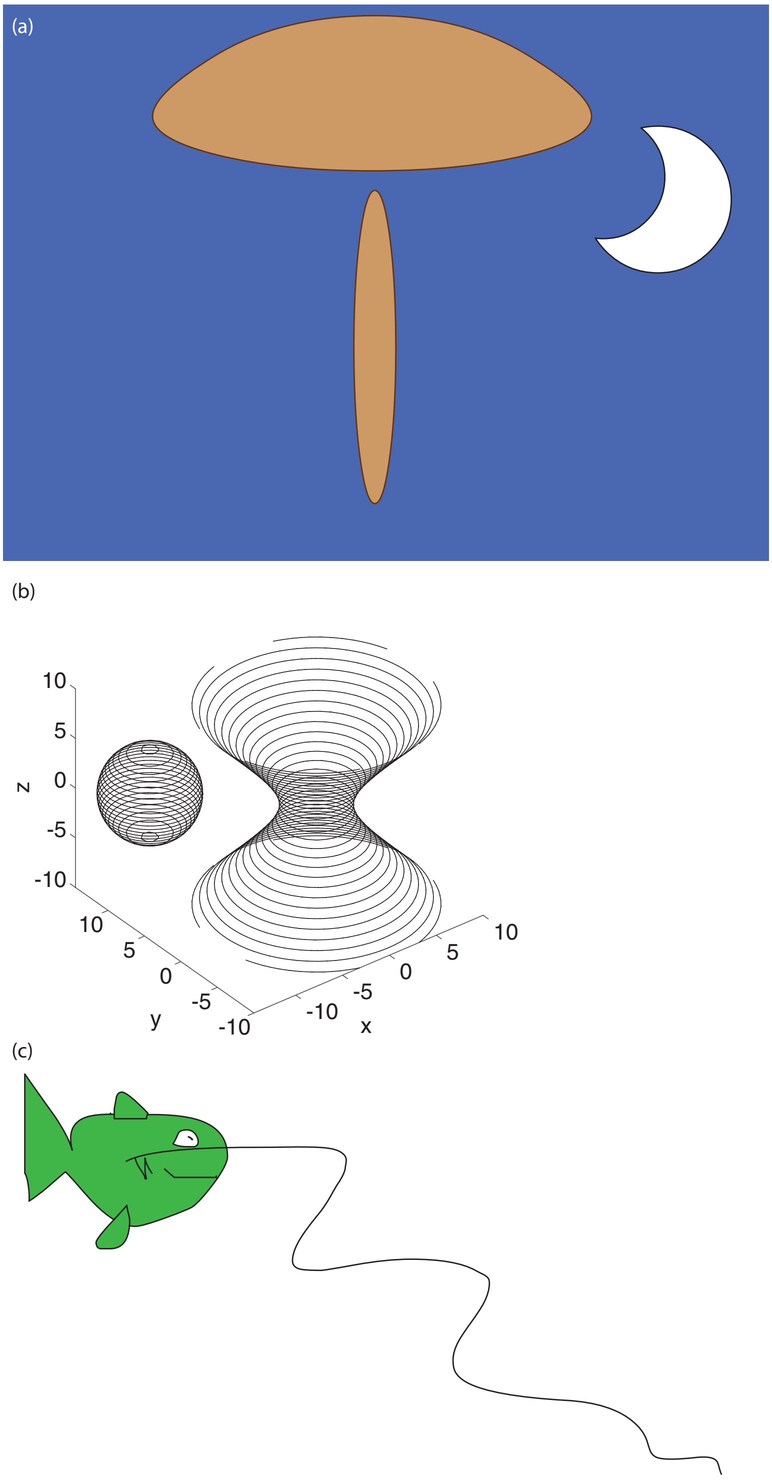

FIGURE 1.1 Use of splines in graphics for creating (a) 2D shapes (b) 3D surfaces (c) motion path trajectory

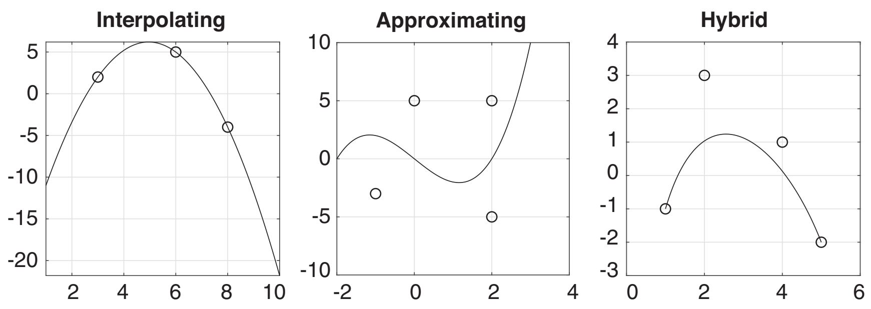

FIGURE 1.2 Types of splines.

$$1-2x+3x^2$$

$$1-\frac{2}{x}+3x^{2.5}$$

$$a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$$

A polynomial equation is written when one polynomial is set equal to another. It can either be in explicit form e.g. $y=f(x)$ when either side of the equation contains variables of explicit type, or it can be in implicit form e.g. $f(x,y)=0$ where multiple types of variables can be on the same side. Examples are shown below:

$$y=x^2$$

$$x^3+y^3-5xy=0$$

Polynomial equations can also be represented in parametric form where the variables are expressed as functions of another variable t e.g. $x=f(t), y=g(t)$. The advantage of parametric equations is that the variables x and y do not need to be constrained by a single equation and can be changed independently of each other, which offers more flexibility for representing complex curves. As a convention, the value of t is usually taken to lie between 0 and 1 unless otherwise specified. The value of $t=0$ corresponds to the start point and $t=1$ to the end point of the spline curve. Examples are shown below:

$$x=t, y=t^2$$

$$x=r\cdot\cos t, y=r\cdot\sin t$$

A polynomial equation is frequently represented using graphs, which are useful in visually depicting how one variable changes with another. The graph of a zero polynomial i.e. $f(x)=0$ is the X-axis. The graph of a zero degree polynomial represented by $f(x)=a$, where a is a constant, is a line parallel to the X-axis at a distance a from it. The graph of a degree 1 polynomial, represented by $f(x)=a+bx$, is a straight line with a slope b and intercept a. The graph

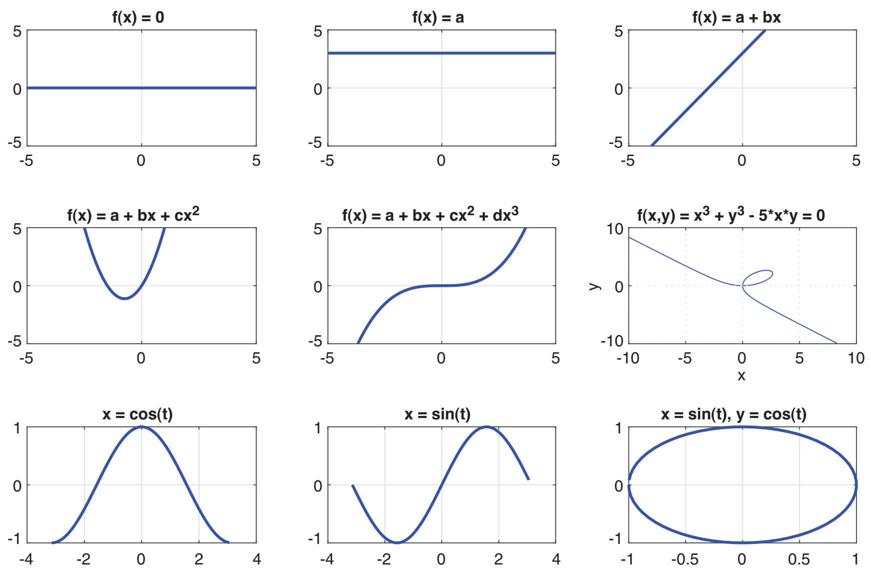

FIGURE 1.3 Graphs of polynomial equations.

of a degree 2 polynomial, represented by $f(x)=a+bx+cx^2$, is a parabolic curve and can be specified if at least three points are known on the curve. The graph of a degree 3 polynomial, represented by $f(x)=a+bx+cx^2+dx^3$, is a cubic curve and can be specified if at least four points are known on the curve. Instead of explicit equations, implicit equations $f(x,y)=0$ can also be plotted by varying the independent variable by fixed intervals and computing the corresponding values of the dependent variables. Plots of parametric equations consist of three different graphs: the first is the t vs. x graph generated from the function $x=f(t)$, the second is the t vs. y graph generated from the function $y=g(t)$, while the third graph is x vs. y generated by plotting the x and y values from the same values of t obtained from the previous plots. Thus, even if we always do not generate an equation between x and y by eliminating t, we can always plot a graph of x vs. y. Figure 1.3 shows graphs of various polynomial equations.

In the following sections, we take a look at few types of interpolating splines and how their equations are derived.

1.2 LINEAR SPLINE (STANDARD FORMS)

A linear spline is a straight line represented by a first-degree polynomial and can be generated given two points are known along it. Standard form of a linear spline implies the spline equation is computed in the spatial domain i.e. the x-y plane. Let the given points be $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. Choose a starting linear equation that is written in matrix form

$$y=a+bx=\begin{bmatrix} 1 & x \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} \tag{1.1}$$

Substitute the given points in the starting equation to generate two equations. Two equations are sufficient to solve for the two unknown coefficients a and b

$$\begin{aligned} y_1 &= a + bx_1 \\ y_2 &= a + bx_2 \end{aligned} \tag{1.2}$$

The two equations are written in matrix form $Y = C{\cdot}A$, where C is the constraint matrix and A is the coefficient matrix:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{1.3}$$

The equations are solved to find the values of the unknown coefficients. Thus, we have $A = \text{inv}(C){\cdot}Y$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{1.4}$$

The values of the coefficients are substituted in the starting equation to arrive at the equation of the spline.

$$y = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{1.5}$$

Example 1.1

Find the equation of a line through points $P_1(3, 2)$ and $P_2(8, -4)$.

Choose a starting equation

$$y = a + bx = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Substitute given points in the equation

$$\begin{aligned} 2 &= a + b(3) \\ -4 &= a + b(8) \end{aligned}$$

Write in matrix form $Y = C{\cdot}A$

$$\begin{bmatrix} 2 \\ -4 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 8 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

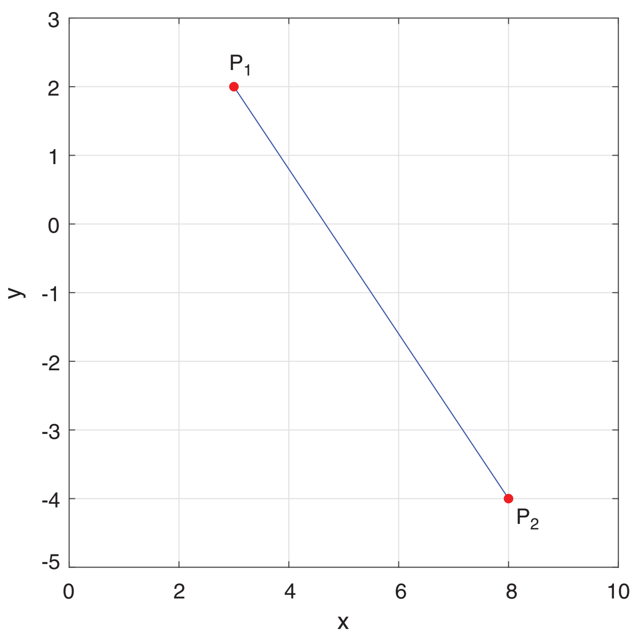

FIGURE 1.4 Plot for Example 1.1.

Solve the matrix equation $A = C^{-1} \cdot Y$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1.6 & -0.6 \\ -0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 2 \\ -4 \end{bmatrix} = \begin{bmatrix} 5.6 \\ -1.2 \end{bmatrix}$$

Substitute the coefficient values in the starting equation

$$y = 5.6 - 1.2x$$

Verification: In most graphics problems, it is usually possible to verify the results obtained by substituting the given data in it. Putting $x = 3$, we get $y = 2$, and putting $x = 8$, we get $y = -4$. Hence, the line does indeed pass through the given points (Figure 1.4).

MATLAB® Code 1.1

```
clear all; clc;
syms x;
x1 = 3; y1 = 2;
x2 = 8; y2 = -4;
X = [x1 x2]; Y = [y1 y2];
C = [1 x1; 1 x2];
A = inv(C)*Y';
a = A(1);
```

```
b = A(2);
fprintf('Required equation : \n');
y = a + b*x;
y = vpa(y)

%plotting
xx = linspace(x1, x2);
yy = subs(y, x, xx);
plot(xx, yy, 'b-'); hold on;
scatter(X, Y, 20, 'r', 'filled');
xlabel('x'); ylabel('y');
grid; axis square;
axis([0 10 -5 3]);
d = 0.5;
text(x1+d, y1, 'P_1');
text(x2+d, y2, 'P_2');
hold off;
```

NOTE

`%`: signifies a comment line
`axis`: controls appearance of the axes of the plot, specifies ordered range of values to display
`clc`: clears workspace of previous text
`clear`: clears memory of all stored variables
`fprintf`: prints out strings and values using formatting options
`grid`: turns on display of grid lines in a plot
`hold`: holds the current graph state so that subsequent commands can add to the same graph
`inv`: computes inverse of a matrix
`linspace`: creates 100 linearly spaced values between the two end-points specified
`plot`: creates a graphical plot from a set of values
`scatter`: type of plot where the data is represented by colored circles
`subs`: substitutes symbolic variable with a matrix of values for generating a plot
`syms`: declares the arguments following as symbolic variables
`text`: inserts textual strings at specified locations in the graph
`title`: displays a title on top of the graph
`vpa`: displays symbolic values as variable precision floating point values
`xlabel, ylabel`: puts text labels along the corresponding primary axes

An alternate form of the standard line equation can be formed where, instead of two given points, only one point and the slope of the line are given. This aspect is discussed below.

Let the given points be $P_1(x_1, y_1)$ and s be the slope of the line. Choose a starting linear equation that is written in matrix form as before in Equation (1.1).

$$y = a + bx = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Calculate derivative of the starting equation

$$y' = \frac{dy}{dx} = b \tag{1.6}$$

Substitute the given values in the starting equation to generate two equations

$$\begin{aligned} y_1 &= a + bx_1 \\ s &= b \end{aligned} \tag{1.7}$$

The two equations are written in matrix form $Y = C{\cdot}A$ as before

$$\begin{bmatrix} y_1 \\ s \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{1.8}$$

The equations are solved to find the values of the coefficients: $A = C^{-1}{\cdot}Y$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ s \end{bmatrix} \tag{1.9}$$

The values of the coefficients are substituted in the starting equation to arrive at the equation of the spline.

$$y = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ s \end{bmatrix} \tag{1.10}$$

Example 1.2

Find the equation of a line through the point P(–1, 1) and having slope 2.

Choose a starting equation

$$y = a + bx = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Calculate derivative of the starting equation

$$y' = \frac{dy}{dx} = b$$

Substitute given values in the equation

$$\begin{aligned} 1 &= a + b(-1) \\ 2 &= b \end{aligned}$$

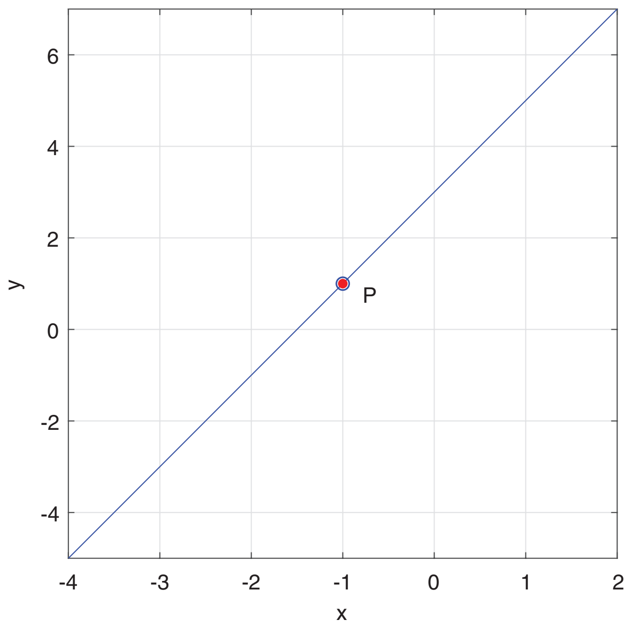

FIGURE 1.5 Plot for Example 1.2.

Write in matrix form $Y = C{\cdot}A$

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Solve the matrix equation $A = C^{-1}{\cdot}Y$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Substitute the coefficient values in the starting equation

$$y = 2x + 3$$

Verification: Putting $x = -1$ in the above equation, we get $y = 1$. Also, slope $\dfrac{dy}{dx} = 2$ (Figure 1.5).

MATLAB Code 1.2

```
clear all; clc;
syms x;
x1 = -1; y1 = 1; s = 2;
Y = [y1 s];
C = [1 x1; 0 1];
```

```
A = inv(C)*Y';
a = A(1);
b = A(2);
fprintf('Required equation:\n');
y = a + b*x;
y = vpa(y)

%plotting
xx = linspace(x1-3, x1+3);
yy = subs(y, x, xx);
plot(xx, yy, 'b-', x1, y1, 'bo'); hold on;
scatter(x1, y1, 20, 'r', 'filled');
xlabel('x'); ylabel('y');
grid; axis square; axis tight;
text(x1+0.5, y1, 'P');
hold off;
```

1.3 LINEAR SPLINE (PARAMETRIC FORM)

A linear spline can also be represented by parametric equations. Let the given points through which the spline passes be P_0 and P_1. As per the convention mentioned before $P_0 \equiv P(0)$ i.e. the first point corresponds to the point where $t = 0$. Similarly, $P_1 \equiv P(1)$ i.e. the second point corresponds to the point where $t = 1$. Note that in some cases, the first point may not always correspond to $t = 0$ or the last point may not always correspond to $t = 1$. We will discuss these issues subsequently.

Choose a starting linear parametric equation written in matrix form

$$P(t) = a + bt = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{1.11}$$

Substitute given points in the starting equation by choosing $t = 0$ at start and $t = 1$ at end.

$$\begin{aligned} P_0 &= a + b(0) \\ P_1 &= a + b(1) \end{aligned} \tag{1.12}$$

Write equations in matrix form $G = C{\cdot}A$, where G is called the geometry matrix

$$\begin{bmatrix} P_0 \\ P_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{1.13}$$

Solve the equation for A i.e. $A = C^{-1}{\cdot}G = B{\cdot}G$, where B is called the basis matrix

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P_0 \\ P_1 \end{bmatrix} \tag{1.14}$$

Substitute the coefficient values in the starting equation

$$P(t)=\begin{bmatrix} 1 & t \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} P_0 \\ P_1 \end{bmatrix} \tag{1.15}$$

NOTE

In reality, the parametric equations should be written separately for x and y i.e. $x(t)=a_x+b_x\cdot t$ and $y(t)=a_y+b_y\cdot t$. However, we use a compact notation here by substituting $P(t)=[x(t),y(t)]$, $a=\left[a_x,a_y\right]$, $b=\left[b_x,b_y\right]$. After solving for a and b, we separate out the individual components and substitute them in the respective equations for x and y.

Example 1.3

Find the equation of a line through points $P_0(3, 2)$ and $P_1(8, -4)$ in parametric form.

Choose a starting equation.

$$P(t)=a+bt=\begin{bmatrix} 1 & t \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}$$

Write equations in matrix form $G=C\cdot A$, where G is called the geometry matrix

$$\begin{bmatrix} 3 & 2 \\ 8 & -4 \end{bmatrix}=\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}$$

Solve the equation for A i.e. $A=C^{-1}\cdot G=B\cdot G$

$$\begin{bmatrix} a \\ b \end{bmatrix}=\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 3 & 2 \\ 8 & -4 \end{bmatrix}=\begin{bmatrix} 3 & 2 \\ 5 & -6 \end{bmatrix}$$

Substitute the coefficient values in the starting equation

$$P(t)=\begin{bmatrix} 1 & t \end{bmatrix}\begin{bmatrix} 3 & 2 \\ 5 & -6 \end{bmatrix}$$

The required parametric equations are obtained by separating out the x and y components

$$x=3+5t$$

$$y=2-6t$$

Verification: $x(0)=3$, $x(1)=8$, $y(0)=2$, $y(1)=-4$ (Figure 1.6).

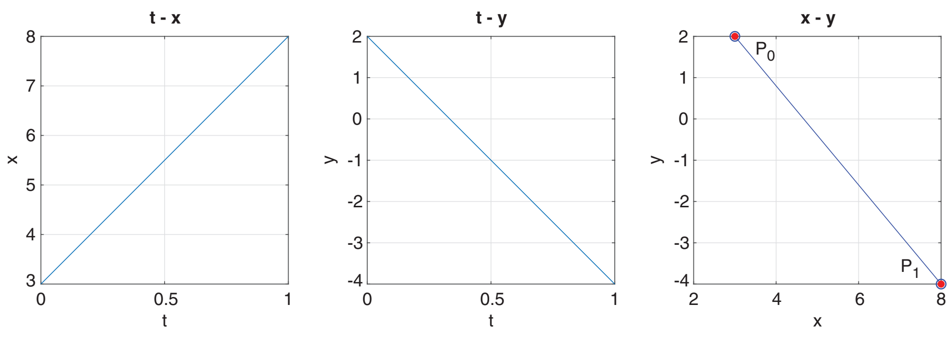

FIGURE 1.6 Plots for Example 1.3.

MATLAB Code 1.3

```
clear all; clc;
syms t;
x1 = 3; y1 = 2;
x2 = 8; y2 = -4;
X = [x1 x2]; Y = [y1 y2];
G = [X ; Y];
C = [1 0; 1 1];
A = inv(C)*G';
ax = A(1,1); ay = A(1,2);
bx = A(2,1); by = A(2,2);

fprintf('Required equations : \n');
x = ax + bx*t; x = vpa(x)
y = ay + by*t; y = vpa(y)

%plotting
tt = linspace(0,1);
xx = subs(x, t, tt);
yy = subs(y, t, tt);

subplot(131), plot(tt, xx); grid; axis square;
xlabel('t'); ylabel('x'); title('t - x');
subplot(132), plot(tt, yy); grid; axis square;
xlabel('t'); ylabel('y'); title('t - y');
subplot(133), plot(xx, yy, 'b-', X, Y, 'bo');
grid; axis square; hold on;
scatter(X, Y, 20, 'r', 'filled');
xlabel('x'); ylabel('y'); title('x - y');
text(x1+1, y1-0.5, 'P_0');
text(x2-1, y2+0.5, 'P_1');
hold off;
```

NOTE

`subplot`: displays multiple plots within a single figure window

1.4 QUADRATIC SPLINE (STANDARD FORM)

A quadratic spline is a parabolic curve represented by a second-degree polynomial equation and can be generated if at least three points are known along the curve. Standard form of a quadratic spline implies the spline equation is computed in the spatial domain i.e. the *x*-*y* plane. Let the given points be $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, and $P_3(x_3, y_3)$. Choose a starting quadratic equation, which is written in matrix form

$$y = a + bx + cx^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{1.16}$$

Substitute the given points in the starting equation to generate three equations. They are sufficient to solve for the three unknown coefficients *a*, *b*, and *c*.

$$\begin{aligned} y_1 &= a + bx_1 + cx_1^2 \\ y_2 &= a + bx_2 + cx_2^2 \\ y_3 &= a + bx_3 + cx_3^2 \end{aligned} \tag{1.17}$$

The three equations are written in matrix form $Y = C{\cdot}A$ as before

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{1.18}$$

The equations are solved to find the values of the coefficients: $A = C^{-1}{\cdot}Y$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{1.19}$$

The values of the coefficients are substituted in the starting equation to arrive at the equation of the spline

$$y = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{1.20}$$

Example 1.4

Find the equation of a quadratic spline through points $P_1(3, 2)$, $P_2(6, 5)$, and $P_3(8, -4)$.

Choose starting equation

$$y = a + bx + cx^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Write equations in matrix form $Y = C{\cdot}A$

$$\begin{bmatrix} 2 \\ -4 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 9 \\ 1 & 8 & 64 \\ 1 & 6 & 36 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Solve for A: $A = C^{-1}{\cdot}Y$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -20.8 \\ 10.9 \\ -1.1 \end{bmatrix}$$

Substitute in the starting equation

$$y = -20.8 + 10.9x - 1.1x^2$$

Verification: $y(3) = 2$, $y(6) = 5$, $y(8) = -4$ (Figure 1.7).

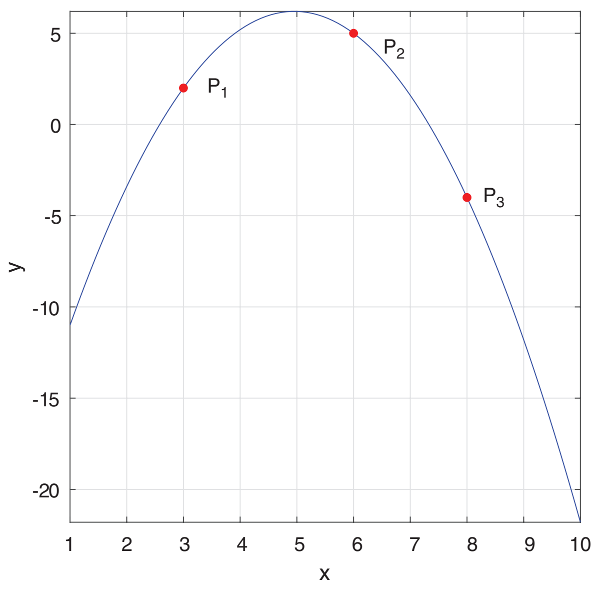

FIGURE 1.7 Plot for Example 1.4.

MATLAB Code 1.4

```
clear all; clc;
syms x;
x1 = 3; y1 = 2;
x2 = 6; y2 = 5;
x3 = 8; y3 = -4;
X = [x1, x2, x3];
Y = [y1, y2, y3];
C = [1, x1, x1^2; 1, x2, x2^2; 1, x3, x3^2];
A = inv(C)*Y';
a = A(1); b = A(2); c = A(3);
fprintf('Required equation : \n');
y = a + b*x + c*x^2; y = vpa(y)

%plotting
d = 0.5;
xx = linspace(x1-2, x3+2);
yy = subs(y, x, xx);
plot(xx,yy, 'b-');
hold on; grid;
scatter(X, Y, 20, 'r', 'filled');
xlabel('x'); ylabel('y');
axis square; axis tight;
text(x1+d, y1, 'P_1');
text(x2+d, y2, 'P_2');
text(x3+d, y3, 'P_3');
hold off;
```

1.5 QUADRATIC SPLINE (PARAMETRIC FORM)

A quadratic spline can also be represented by parametric equations. Let the given points be P_0, P_1, and P_2. Here, P_0 is the starting point i.e. $P_0 \equiv P(0)$ and P_2 is the end point i.e. $P_2 \equiv P(1)$. To determine the curve uniquely an additional piece of information is required regarding the value of parameter t at the middle point P_1. Let the value of t at P_1 be k, where $0 \le k \le 1$ i.e. $P_1 \equiv P(k)$. Different values of k referred to as the sub-division ratio, will give rise to curves with the same start and end points but having different shapes.

Choose a starting parametric quadratic equation written in matrix form

$$P(t) = a + bt + ct^2 = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{1.21}$$

Substitute the given points in the starting equation by choosing $t = 0$ at start, $t = k$ at the middle, and $t = 1$ at end.

$$\begin{aligned}P_0 &= a+b(0)+c(0)^2\\ P_1 &= a+b(k)+c(k)^2\\ P_2 &= a+b(1)+c(1)^2\end{aligned} \tag{1.22}$$

Write equations in matrix form $G = C{\cdot}A$

$$\begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & k & k^2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{1.23}$$

Solve the equation for A i.e. $A = C^{-1}{\cdot}G = B{\cdot}G$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & k & k^2 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} \tag{1.24}$$

Substitute the coefficient values in the starting equation

$$P(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & k & k^2 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} \tag{1.25}$$

Example 1.5

Find the equation of a quadratic spline through points $P_0(3, 2)$, $P_1(8, -4)$, and $P_2(6, 5)$ in parametric form with sub-division ratio $k = 0.8$.

Choose starting equation

$$P(t) = a + bt + ct^2 = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Write equations in matrix form $G = C{\cdot}A$

$$\begin{bmatrix} 3 & 2 \\ 8 & -4 \\ 6 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.8 & 0.64 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

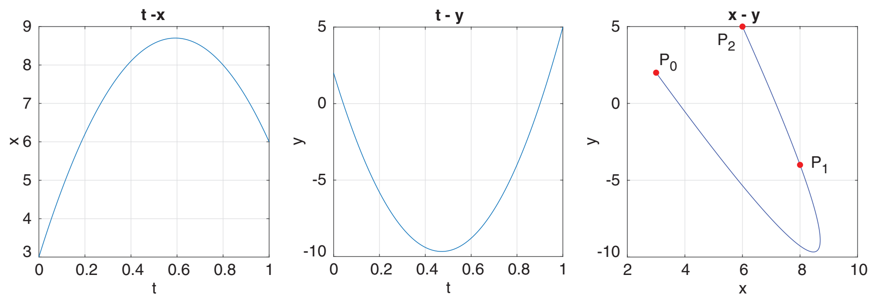

FIGURE 1.8 Plots for Example 1.5.

Solve the equation for A i.e. $A = C^{-1} \cdot G = B \cdot G$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 19.25 & -49.5 \\ -16.25 & 52.5 \end{bmatrix}$$

Substitute the coefficient values in the starting equation

$$x = 3 + 19.25t - 16.25t^2$$

$$y = 2 - 49.5t + 52.5t^2$$

Verification: $x(0)=3$, $y(0)=2$, $x(0.8)=8$, $y(0.8)=-0.4$, $x(1)=6$, $y(1)=5$ (Figure 1.8).

MATLAB Code 1.5

```
clear all; clc;
syms t;
x0 = 3; y0 = 2;
x1 = 8; y1 = -4;
x2 = 6; y2 = 5;
k = 0.8;
G = [x0, y0 ; x1, y1 ; x2, y2];
X = [x0 ; x1 ; x2]; Y = [y0 ; y1 ; y2];
C = [1, 0, 0; 1, k, k^2; 1, 1, 1];
A = inv(C)*G;
ax = A(1,1); ay = A(1,2);
bx = A(2,1); by = A(2,2);
cx = A(3,1); cy = A(3,2);
fprintf('Required equations: \n');
x = ax + bx*t + cx*t^2 ; x = vpa(x)
y = ay + by*t + cy*t^2 ; y = vpa(y)

%plotting
tt = linspace(0,1);
xx = subs(x, t, tt);
```

```
yy = subs(y, t, tt);
subplot(131), plot(tt, xx);
xlabel('t'); ylabel('x'); title('t - x');
grid; axis square;
subplot(132), plot(tt, yy);
xlabel('t'); ylabel('y'); title('t - y');
grid; axis square;
subplot(133), plot(xx, yy, 'b-');
hold on; grid; axis square;
scatter(X, Y, 20, 'r', 'filled');
xlabel('x'); ylabel('y'); title('x - y');
d = 0.5;
text(x0+d, y0, 'P_0');
text(x1+d, y1, 'P_1');
text(x2-1, y2-1, 'P_2');
hold off;
```

1.6 CUBIC SPLINE (STANDARD FORM)

A cubic spline is represented by a third-degree polynomial and can be generated if at least four points along the curve are known. Standard form of a cubic spline implies the spline equation is computed in the spatial domain i.e. the *x*-*y* plane. Let the given points be $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, and $P_4(x_4, y_4)$. Choose a starting cubic equation, which is written in matrix form

$$y = a + bx + cx^2 + dx^3 = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{1.26}$$

Substitute the given points in the starting equation to generate four equations. Four equations are sufficient to solve the four unknown coefficients *a*, *b*, *c*, and *d*

$$\begin{aligned} y_1 &= a + bx_1 + c{x_1}^2 + d{x_1}^3 \\ y_2 &= a + bx_2 + c{x_2}^2 + d{x_2}^3 \\ y_3 &= a + bx_3 + c{x_3}^2 + d{x_3}^3 \\ y_4 &= a + bx_4 + c{x_4}^2 + d{x_4}^3 \end{aligned} \tag{1.27}$$

The four equations are written in matrix form $Y = C{\cdot}A$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & {x_1}^2 & {x_1}^3 \\ 1 & x_2 & {x_2}^2 & {x_2}^3 \\ 1 & x_3 & {x_3}^2 & {x_3}^3 \\ 1 & x_4 & {x_4}^2 & {x_4}^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{1.28}$$

The equations are solved to find the values of the coefficients: $A = C^{-1} \cdot Y$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{1.29}$$

The values of the coefficients are substituted in the starting equation to arrive at the equation of the spline.

$$y = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{1.30}$$

Example 1.6

Find the equation of a cubic spline through points $P_1(-1, 2)$, $P_2(0, 0)$, $P_3(1, -2)$, and $P_4(2, 0)$.

Choose starting equation

$$y = a + bx + cx^2 + dx^3 = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Write equations in matrix form $Y = C \cdot A$

$$\begin{bmatrix} 2 \\ 0 \\ -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Solve for A: $A = C^{-1} \cdot Y$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ -2.67 \\ 0 \\ 0.67 \end{bmatrix}$$

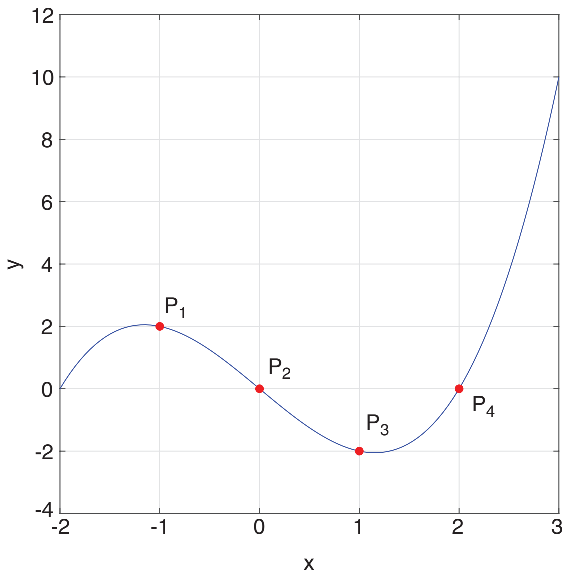

FIGURE 1.9 Plot for Example 1.6.

Substitute in the starting equation

$$y = -2.67x + 0.67x^3$$

Verification: $y(-1) = 2$, $y(0) = 0$, $y(1) = -2$, $y(2) = 0$ (Figure 1.9).

MATLAB Code 1.6

```
clear all; clc;
syms x;
x1 = -1; y1 = 2;
x2 = 0; y2 = 0;
x3 = 1; y3 = -2;
x4 = 2; y4 = 0;
X = [x1 ; x2 ; x3 ; x4];
Y = [y1 ; y2 ; y3 ; y4];
C = [1, x1, x1^2, x1^3; 1, x2, x2^2, x2^3; 1, x3, x3^2, x3^3; 1, x4, x4^2, x4^3];
A = inv(C)*Y;
a = A(1); b = A(2); c = A(3); d = A(4);
fprintf('Required equation  : \n');
y = a + b*x + c*x^2 + d*x^3; y = vpa(y, 3)

%plotting
X = [x1, x2, x3, x4]; m = min(X); n = max(X);
xx = linspace(m - 1, n + 1);
yy = subs(y, x, xx);
plot(xx,yy, 'b');
hold on; grid;
scatter(X, Y, 20, 'r', 'filled');
xlabel('x'); ylabel('y'); axis square;
e = 1;
text(x1, y1+e, 'P_1');
text(x2, y2+e, 'P_2');
text(x3, y3+e, 'P_3');
text(x4, y4+2*e, 'P_4');
hold off;
```

1.7 CUBIC SPLINE (PARAMETRIC FORM)

A cubic spline can also be represented by parametric equations. Let the given points be P_0, P_1, P_2, and P_3. Here, P_0 is the starting point i.e. $P_0 \equiv P(0)$ and P_3 is the end point of the curve i.e. $P_3 \equiv P(1)$. To determine the curve uniquely two additional pieces of information are required regarding the value of parameter t at the middle points P_1 and P_2. Let these values of t be m and n, where $0 \le m,\ n \le 1$ i.e. $P_1 \equiv P(m)$ and $P_2 \equiv P(n)$. Different values of m and n referred to as the sub-division ratios, will give rise to curves with the same start and end points but having different shapes.

Choose starting equation written in matrix form

$$P(t) = a + bt + ct^2 + dt^3 = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{1.31}$$

Substitute the given points in the starting equation by choosing $t = 0$ at start, $t = m$, n at the middle points, and $t = 1$ at end.

$$\begin{aligned} P_0 &= a + b(0) + c(0)^2 \\ P_1 &= a + b(m) + c(m)^2 \\ P_2 &= a + b(n) + c(n)^2 \\ P_3 &= a + b(1) + c(1)^2 \end{aligned} \tag{1.32}$$

Write equations in matrix form $G = C{\cdot}A$

$$\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & m & m^2 & m^3 \\ 1 & n & n^2 & n^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{1.33}$$

Solve the equation for A: $A = C^{-1}{\cdot}G = B{\cdot}G$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & m & m^2 & m^3 \\ 1 & n & n^2 & n^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{1.34}$$

Substitute the coefficient values in the starting equation

$$P(t)=\begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 1 & m & m^2 & m^3\\ 1 & n & n^2 & n^3\\ 1 & 1 & 1 & 1\end{bmatrix}^{-1}\begin{bmatrix}P_0\\ P_1\\ P_2\\ P_3\end{bmatrix} \tag{1.35}$$

Example 1.7

Find the equation of a cubic spline through points (–1, 2), (0, 0), (1, –2), and (2, 0) in parametric form with sub-division ratios m = 0.1 and n = 0.9.

Choose starting equation written in matrix form

$$P(t)=a+bt+ct^2+dt^3=\begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}a\\ b\\ c\\ d\end{bmatrix}$$

Write equations in matrix form $G = C{\cdot}A$

$$\begin{bmatrix}-1 & 2\\ 0 & 0\\ 1 & -2\\ 2 & 0\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 0.1 & 0.01 & 0.001\\ 1 & 0.9 & 0.81 & 0.729\\ 1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}a\\ b\\ c\\ d\end{bmatrix}$$

Solve the equation for A i.e. $A = C^{-1}{\cdot}G = B{\cdot}G$

$$\begin{bmatrix}a\\ b\\ c\\ d\end{bmatrix}=\begin{bmatrix}-1 & 2\\ 12.7222 & -21.4444\\ -29.1667 & 13.8889\\ 19.4444 & 5.5556\end{bmatrix}$$

Substitute the coefficient values in the starting equation

$$x=-1+12.7222t-29.1667t^2+19.4444t^3$$

$$y=2-21.4444t+13.8889t^2+5.5556t^3$$

Verification: $x(0) = -1, x(0.1) = 0, x(0.9) = 1, x(1) = 2, y(0) = 2, y(0.1) = 0, y(0.9) = -2, y(1)=0$

The actual values might differ slightly due to round-off errors (Figure 1.10).

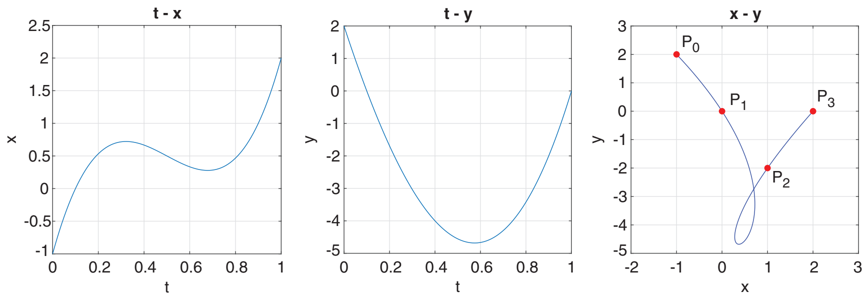

FIGURE 1.10 Plots for Example 1.7.

MATLAB Code 1.7

```
clear all; clc;
syms t;
x0 = -1; y0 = 2;
x1 = 0; y1 = 0;
x2 = 1; y2 = -2;
x3 = 2; y3 = 0;
m = 0.1; n = 0.9;
P = [x0 y0 ; x1 y1 ; x2 y2 ; x3 y3];
X = [x0 ; x1 ; x2 ; x3]; Y = [y0 ; y1 ; y2 ; y3];
C = [1, 0, 0, 0; 1, m, m^2, m^3; 1, n, n^2, n^3; 1, 1, 1, 1];
A = inv(C)*P;
ax = A(1,1); ay = A(1,2); bx = A(2,1); by = A(2,2);
cx = A(3,1); cy = A(3,2); dx = A(4,1); dy = A(4,2);
fprintf('Required equations : \n');
x = ax + bx*t + cx*t^2 + dx*t^3; x = vpa(x, 3)
y = ay + by*t + cy*t^2 + dy*t^3; y = vpa(y, 3)

%plotting
tt = linspace(0,1);
xx = subs(x, t, tt);
yy = subs(y, t, tt);
subplot(131), plot(tt,xx); grid;
xlabel('t'); ylabel('x'); title('t - x'); axis square;
subplot(132), plot(tt,yy); grid;
xlabel('t'); ylabel('y'); title('t - y'); axis square;
subplot(133), plot(xx,yy,'b-'); grid;
xlabel('x'); ylabel('y'); title('x - y'); axis square;
hold on;
scatter(X, Y, 20, 'r', 'filled');
axis([-2 3 -5 3]);
e = 0.5;
text(x0+e, y0, 'P_0');
text(x1+e, y1, 'P_1');
text(x2+e, y2, 'P_2');
text(x3+e, y3, 'P_3');
hold off;
```

1.8 PIECEWISE SPLINES (STANDARD FORM)

Complex curves cannot be appropriately modeled using cubic splines. They are typically S-shaped curves while complex curves may contain a number of twists and turns. One option is to model the curves using higher order splines; however, they need higher degree equations to be solved, which increases the computational overhead and time delay of the system. Moreover, higher degree splines are too sensitive to slight changes in CPs, which is typically not desirable since we generally want slight changes of the splines to be made by small adjustments of their CPs and do not favor drastic changes in shape. Such curves are best modeled by using multiple cubic splines joined end to end. These are known as piecewise splines.

Consider four given points P_1, P_2, P_3, and P_4 and it is required to find equations of piecewise splines through them. Essentially, this means that instead of a single cubic spline passing through the four points it is required to find three separate splines passing through each pair of points as shown in Figure 1.11.

Let the coordinates of the given points be $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, and $P_4(x_4, y_4)$. Let the three cubic curve segments be designated as A, B, and C between points P_1 and P_2, P_2 and P_3, and P_3 and P_4, respectively. As before, let starting cubic equations be of the form $y = a + bx + cx^2 + dx^3$. Since now there are three curve segments, there needs to be three different sets of coefficients as follows:

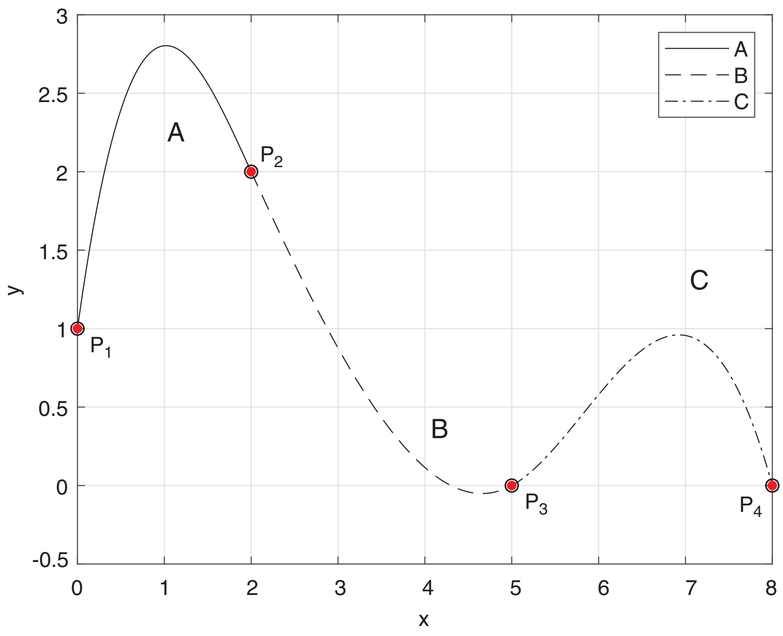

FIGURE 1.11 Piecewise splines.

$$\begin{aligned} &A\colon y = a_1 + b_1 x + c_1 x^2 + d_1 x^3 \\ &B\colon y = a_2 + b_2 x + c_2 x^2 + d_2 x^3 \\ &C\colon y = a_3 + b_3 x + c_3 x^2 + d_3 x^3 \end{aligned} \tag{1.36}$$

So altogether there are 12 different unknowns and at least 12 different equations are needed to solve them.

In order to formulate these 12 equations, various constraints are used to ensure that three separate spline segments join together to form a single smooth curve. The first constraint is known as C^0 continuity condition, which states that in order to form a smooth curve the three splines should physically meet at their joining points (Hearn and Baker, 1996). In other words, spline A should pass through points P_1 and P_2, spline B should pass through points P_2 and P_3, and spline C should pass through points P_3 and P_4. Substituting the point coordinates in the respective starting equations the following six equations are obtained. If $S(P_k)$ denotes segment S passing through point P_k, we can write:

$$\begin{aligned} &A(P_1)\colon y_1 = a_1 + b_1 x_1 + c_1 {x_1}^2 + d_1 {x_1}^3 \\ &A(P_2)\colon y_2 = a_1 + b_1 x_2 + c_1 {x_2}^2 + d_1 {x_2}^3 \\ &B(P_2)\colon y_2 = a_2 + b_2 x_2 + c_2 {x_2}^2 + d_2 {x_2}^3 \\ &B(P_3)\colon y_3 = a_2 + b_2 x_3 + c_2 {x_3}^2 + d_2 {x_3}^3 \\ &C(P_3)\colon y_3 = a_3 + b_3 x_3 + c_3 {x_3}^2 + d_3 {x_3}^3 \\ &C(P_4)\colon y_4 = a_3 + b_3 x_4 + c_3 {x_4}^2 + d_3 {x_4}^3 \end{aligned} \tag{1.37}$$

The second constraint to be obeyed is known as C^1 continuity condition, which states that to form a smooth curve the slopes of the individual spline segments should be equal at their meeting points (Hearn and Baker, 1996). Taking the derivative of the spline equations the following are obtained:

$$\begin{aligned} &A'\colon y' = b_1 + 2c_1 \cdot x + 3d_1 \cdot x^2 \\ &B'\colon y' = b_2 + 2c_2 \cdot x + 3d_2 \cdot x^2 \\ &C'\colon y' = b_3 + 2c_3 \cdot x + 3d_3 \cdot x^2 \end{aligned} \tag{1.38}$$

In this case: slope of A at P_2 = slope of B at P_2. If $S'(P_k)$ denotes slope of segment S at point P_k we have:

$$A'(P_2) = B'(P_2)\colon\ b_1 + 2c_1 \cdot x_2 + 3d_1 \cdot {x_2}^2 = b_2 + 2c_2 \cdot x_2 + 3d_2 \cdot {x_2}^2$$

Rearranging:

$$0 = -b_1 - 2c_1 \cdot x_2 - 3d_1 \cdot {x_2}^2 + b_2 + 2c_2 \cdot x_2 + 3d_2 \cdot {x_2}^2 \tag{1.39}$$