Chapter 2
Numerical Integration

2.1 Introduction

An important aspect in the isogeometric approach, the vector and matrix formulations are requiring numerical integration scheme because the high-order polynomial presentations of the NURBS functions. The NURBS functions will be adopted as the shape functions in the formulations of a beam element which are hardly can be represented by using formulas explicitly. Our discussion of the numerical integration and their related methods is only related to the beam element formulation; for other methods, the reader should consult other references (Gradshteyn and Ryzhik 2000; Hildebrand 1956; Kaplan 1984; Ralston and Rabinowitz 1978).

2.2 Gauss-Legendre Quadrature

The Gauss-Legendre numerical integration can achieve the exact of fitness of $2n - 1$ or less by using the weighted sum of function values at specified points within the domain of integration. It approximates the following definite integration:

$$\int_{-1}^{+1} f(\xi)\ d\xi \approx \sum\nolimits_{i=1}^{n} f(\xi_i)\ w_i \tag{2.1}$$

where n is the order of degree of polynomial inside the integral, ξ_i is the coordinate of Gauss point i, and w_i is the weight of Gauss point i.

The Legendre polynomials can be defined by a recursive relation as follows:

B.S. Gan, *An Isogeometric Approach to Beam Structures*,
DOI 10.1007/978-3-319-56493-7_2

$$P_n(\xi) = \frac{1}{n}\{(2n-1)\ \xi\ P_{n-1}(\xi) - (n-1)\ P_{n-2}(\xi)\} \qquad n \geq 2$$
$$\text{where} \quad P_0(\xi) = 1, \quad P_1(\xi) = \xi \tag{2.2}$$

The points ξ_i, ($i = 1,...,n$) are the roots of (2.2) inside the range of [−1 1] by equating the polynomial to zero as follows:

$$P_n(\xi) = 0 \qquad \text{for } n \geq 1 \tag{2.3}$$

The weights w_i are calculated from corresponding ξ_i points as

$$w_i = \frac{2\left(1-\xi_i^2\right)}{n^2\ P_{n-1}^2(\xi_i)} \qquad \text{for } n \geq 1 \tag{2.4}$$

Table 2.1 lists the points and weights for order-n Legendre polynomial up to $n = 5$.

As an example, the calculation of areas under a given function: $f(\xi) = \xi^8 + \xi^7 + \xi^6 + \xi^5 + 10$ in the interval of [−1 1], by using the Gauss-Legendre rule, is tabulated in Table 2.2. We can see that the maximum order of the function is of the order eight. Hence, the exactness of solution can be achieved by using $2n - 1 \geq 8$; $n \geq 4.5$ (since n must be an integer, we use $n = 5$) to obtain the exact solution.

2.3 Length, Jacobian Operator, and Radius of Curvature of a Curve

What if we have to integrate an equation with real coordinates $\mathbf{x}$, different with the $\boldsymbol{\xi}$ [−1 +1] interval? Apparently, we will find them in most of the cases we deal with the real world. Hence, we need an operator that can scale between both coordinates systems ($\mathbf{x}$, $\boldsymbol{\xi}$) the so-called coordinate transformation.

In fact, the numerical integration that we should do in the finite element of a beam is a line integration scheme, not integrating the area under the line itself. The beam element components such as stiffness matrix, mass matrix, and loading vector are formed by integrating the material properties, cross section, or distributed mass along the path of a line or a curve.

In calculus, a scaling factor that relates the incremental variable of both coordinates is called the *Jacobian* operator. We denote it by J. Invoking the chain rule,

$$\frac{d}{d\boldsymbol{\xi}} = \frac{d\mathbf{x}}{d\boldsymbol{\xi}}\frac{d}{d\mathbf{x}} = J\frac{d}{d\mathbf{x}} \quad \text{where} \quad \frac{d\mathbf{x}}{d\boldsymbol{\xi}} = J \quad \text{or} \quad \frac{d\boldsymbol{\xi}}{d\mathbf{x}} = \frac{1}{J} \tag{2.5}$$

which requires that J has a non-zero determinant value. The integration becomes

Table 2.1 Tabulated list of the order-n Legendre polynomial points and weights up to $n = 5$

n	Point, ξ_i Weight, w_i
1	0 2
2	$-\sqrt{\frac{1}{3}}, +\sqrt{\frac{1}{3}}$
	1, 1
3	$-\sqrt{\frac{3}{5}}, 0, +\sqrt{\frac{3}{5}}$ $\frac{5}{9}, \frac{8}{9}, \frac{5}{9}$
4	$-\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}, -\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}, +\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}, +\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$ $\frac{1}{2}+\frac{1}{6}\sqrt{\frac{5}{6}}, \frac{1}{2}-\frac{1}{6}\sqrt{\frac{5}{6}}, \frac{1}{2}-\frac{1}{6}\sqrt{\frac{5}{6}}, \frac{1}{2}+\frac{1}{6}\sqrt{\frac{5}{6}}$
5	$-\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}, -\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}, +\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}, +\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$ $\frac{322+13\sqrt{70}}{900}, \frac{322-13\sqrt{70}}{900}, \frac{322-13\sqrt{70}}{900}, \frac{322+13\sqrt{70}}{900}$

Table 2.2 Calculation of area by using different order of Legendre polynomial

n	$area = \sum_{i=1}^{n} f(\xi_i)\ w_i$	Exact to 16 decimal places
1	20.00000000000000	20.50793650793650
2	20.09876543209880	20.50793650793650
3	20.38400000000000	20.50793650793650
4	20.49632653061220	20.50793650793650
5	20.50793650793650	20.50793650793650

$$\int_{x_1}^{x_2} f(x)\ dx = \int_{-1}^{+1} f(\xi)\quad J\ d\xi \approx \sum_{i=1}^{n} f(\xi_i)\ J\ w_i \tag{2.6}$$

For one-dimensional integration of a straight line, we can easily relate the $x = \left(\frac{1-\xi}{2}\right)x_1 + \left(\frac{1+\xi}{2}\right)x_2$ relationship. Here $[x_1\ x_2]$ is the interval of the integration along the x coordinate. We can obtain the Jacobian operator from (2.5) as

$$J = \frac{dx}{d\xi} = -\frac{1}{2}x_1 + \frac{1}{2}x_2 = \frac{x_2 - x_1}{2} \tag{2.7}$$

Where the $[x_1\ x_2]$ interval shows the length of the beam. The length and total mass of the beam can be calculated from the following equation.

$$\begin{aligned} \text{Length} &: \int_{x_1}^{x_2} dx = \int_{-1}^{+1} J\ d\xi \approx \sum_{i=1}^{n} \left(\frac{x_2 - x_1}{2}\right)\ w_i \\ \text{Mass} &: \int_{x_1}^{x_2} \rho(x)\ dx = \int_{-1}^{+1} \rho(\xi)\ J\ d\xi \approx \sum_{i=1}^{n} \rho(\xi)\left(\frac{x_2 - x_1}{2}\right)\ w_i \end{aligned} \tag{2.8}$$

In the finite element context, the length of a curve segment, Jacobian operator, and curvature, respectively, can be given as

$$\begin{aligned} C(\xi) &= \int_{\xi_0=-1}^{\xi_0=\xi} \sqrt{\left(\frac{dx}{d\xi_0}\right)^2 + \left(\frac{dy}{d\xi_0}\right)^2} d\xi_0 \\ J &= \frac{dC}{d\xi} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} \\ \kappa(\xi) &= \frac{\left|\frac{dx}{d\xi}\frac{d^2y}{d\xi^2} - \frac{dy}{d\xi}\frac{d^2x}{d\xi^2}\right|}{J^3} \end{aligned} \tag{2.9}$$

In case of a circular curved line, we can define the x-y coordinates by the trigonometric functions $x = R\ Cos(\varphi), y = R\ Sin(\varphi)$ and apply the chain rule from the relationship $\varphi = \frac{\varphi_1+\varphi_2}{2} + \frac{\varphi_1-\varphi_2}{2}\xi$, we can obtain the Jacobian operator from (2.9) as follows:

$$\begin{aligned} &\frac{dx}{d\xi} = -\left(\frac{\varphi_1 - \varphi_2}{2}\right) R\ Sin(\varphi)\ ;\ \frac{dy}{d\xi} = \left(\frac{\varphi_1 - \varphi_2}{2}\right) R\ Cos(\varphi) \\ &J = \left(\frac{\varphi_1 - \varphi_2}{2}\right) R \end{aligned} \tag{2.10}$$

Where the $[\varphi_1\ \varphi_2]$ interval shows the arc segment length of the beam. The length and total mass of the beam can be calculated from the following equation.

$$\begin{aligned} \text{Length} &: \int_{x_1}^{x_2} dx = \int_{-1}^{+1} J\ d\xi \approx \sum_{i=1}^{n} \left(\frac{\varphi_1 - \varphi_2}{2}\right) R\ w_i \\ \text{Mass} &: \int_{x_1}^{x_2} \rho(x)\ dx = \int_{-1}^{+1} \rho(\xi)\ J\ d\xi \approx \sum_{i=1}^{n} \rho(\xi) \left(\frac{\varphi_1 - \varphi_2}{2}\right) R\ w_i \end{aligned} \tag{2.11}$$

For a circle, where the radius of curvature is constant along the path, we have $\kappa(\xi) = {}^{1}/_{R}$. In fact, (2.9) can generally be used for all kind of curves, including the straight line by setting a zero value of curvature.

For a third-order polynomial curved line $y = x^3 - 1$, we can approximate the x coordinate related by a linear function $x = \left(\frac{1-\xi}{2}\right)x_1 + \left(\frac{1+\xi}{2}\right)x_2$ and by applying the chain rule for the y coordinate,

$$\frac{dy}{dx} = 3x^2 \quad ; \quad \frac{dy}{d\xi} = \frac{dy}{dx} \times \frac{dx}{d\xi} = 3x^2 \left(\frac{x_2 - x_1}{2}\right) \tag{2.12}$$

where the $[x_1\ x_2]$ interval shows the arc segment length of the beam. The Jacobian operator can be obtained from (2.9).

The length and total mass of the beam can be calculated from the following equation.

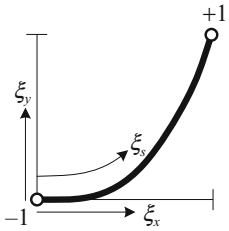

Fig. 2.1 Several possible coordinate transformations along the path of the curve in the Gauss-Legendre quadrature

$$\begin{aligned} \text{Length} \quad &: \quad \int_{x_1}^{x_2} dx = \int_{-1}^{+1} J \; d\xi \approx \sum_{i=1}^{n} J(\xi) R \; w_i \\ \text{Mass} \quad &: \quad \int_{x_1}^{x_2} \rho(x) \; dx = \int_{-1}^{+1} \rho(\xi) \; J(\xi) \; d\xi \approx \sum_{i=1}^{n} \rho(\xi) J(\xi) w_i \end{aligned} \tag{2.13}$$

In the above equation, the approximation of the x-abscissa to have a linear relationship with ξ is not accurate. To get an accurate result, the ξ parameter must be measured along the path of the curve, which can be done by doing the numerical integration scheme for computing the length of the curve in advance. As shown in Fig. 2.1, we can select either the x, y, or s axes optionally to be represented by the local coordinate ξ using the Gauss-Legendre quadrature integration. In a general curve, most probably the scaling factors among the axes are different. Hence, we get inaccurate results. The most convenient situation is when we have an explicit formula for the $s(x,y)$ as a function of x and y coordinates.

MATLAB code Gaussquadrature.m (Sect. 2.3.1) uses the function Legendre.m (Sect. 2.3.2), to integrate numerically the length of curves (Fig. 2.2) and their total mass from distributed density function which is defined by a polynomial equation by using (2.6) and (2.9). The computed lengths and total mass are given in Table 2.3. In the program, the calculation of length can be done by letting the variable ρ (ξ) is equal to one.

It can be noted that from the beginning of this book, the interval of the local coordinate ξ is kept unchanged between the range of [−1 + 1] to have a smooth transition between equations and formulations across the chapters.

2.3.1 Gaussquadrature Program List

```
% Gaussquadrature.m
% Length and Mass calculation of curves
% Given density of mass per unit length
clear all;clc;
ng=3;
Gpw = Legendre(ng);
%
% Distributed material on a straight line
x=[-2 1];
```

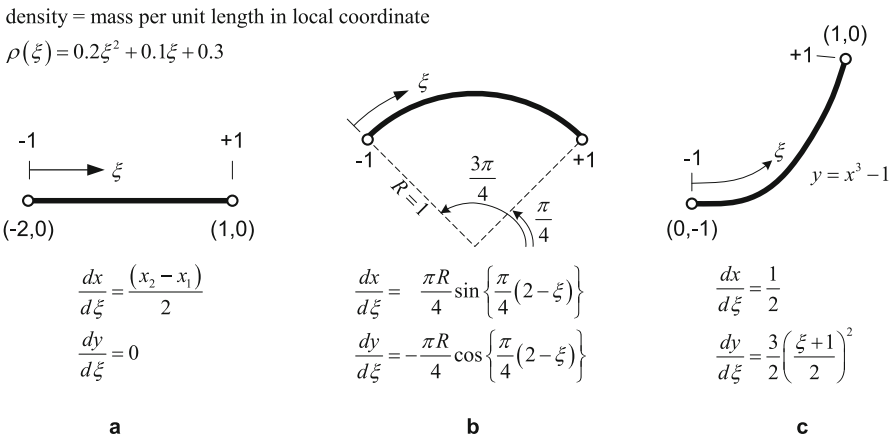

Fig. 2.2 Numerical integration of several curves and their total mass calculation: (**a**) straight line; (**b**) arc of a circle; (**c**) third-order polynomial curve

Table 2.3 Numerical integration of length of the curves and their total mass

Type of curve	Length	Total mass	Gauss point used
(a) Straight line	3.0000	1.1000	2
(b) Arc of a circle	1.5708	0.5760	2
(c) Third-order polynomial	1.5478	0.6162	5

```
dxdxi=(x(2)-x(1))/2; dydxi=0;
Jac=sqrt(dxdxi^2+dydxi^2);
C1=0;
for i=1:ng
 rho=0.2*Gpw(i,1)^2+0.1*Gpw(i,1)+0.3;
 %rho=1; % to calculate the length
 C1=C1+rho*Gpw(i,2)*Jac; % rho(xi) along the straight line
end
%
% Distributed material on a 90 deg arc of a circle
r=1;
Jac=pi*r/4;
C2=0;
for i=1:ng
 rho=0.2*Gpw(i,1)^2+0.1*Gpw(i,1)+0.3;
 %rho=1; % to calculate the length
 C2=C2+rho*Gpw(i,2)*Jac; % rho(xi) along the arc of a circle
end
%
% Distributed material on a 3rd order polynomial curve
C3=0;
```

```
for i=1:ng
 Jac=sqrt((1/2)^2+9/4*((Gpw(i,1)+1)/2)^4);
 rho=0.2*Gpw(i,1)^2+0.1*Gpw(i,1)+0.3;
 %rho=1; % to calculate the length
 C3=C3+rho*Gpw(i,2)*Jac; % rho(xi) along the 3rd order polynomial
curve
end
% p = order
```

2.3.2 Legendre Function List

```
function Gpw = Legendre(ng)
% ng = number of Gauss points
% Gauss point is stored in Gpw(n,1)
% Gauss weight is stored in Gpw(n,2)
%----------------------------------
Gpw=zeros(ng,2);
switch ng
 case 1
   Gpw(1,1)=0;
   Gpw(1,2)=2;
 case 2
   Gpw(1,1)=-sqrt(1/3);
   Gpw(1,2)=1;
   Gpw(2,1)=+sqrt(1/3);
   Gpw(2,2)=1;
 case 3
   Gpw(1,1)=-sqrt(3/5);
   Gpw(1,2)=5/9;
   Gpw(2,1)=0;
   Gpw(2,2)=8/9;
   Gpw(3,1)=+sqrt(3/5);
   Gpw(3,2)=5/9;
 case 4
   Gpw(1,1)=-sqrt(3/7+2/7*sqrt(6/5));
   Gpw(1,2)=(18-sqrt(30))/36;
   Gpw(2,1)=-sqrt(3/7-2/7*sqrt(6/5));
   Gpw(2,2)=(18+sqrt(30))/36;
   Gpw(3,1)=+sqrt(3/7-2/7*sqrt(6/5));
   Gpw(3,2)=(18+sqrt(30))/36;
   Gpw(4,1)=+sqrt(3/7+2/7*sqrt(6/5));
   Gpw(4,2)=(18-sqrt(30))/36;
```

```
case 5
  Gpw(1,1)=-1/3*sqrt(5+2*sqrt(10/7));
  Gpw(1,2)=(322-13*sqrt(70))/900;
   Gpw(2,1)=-1/3*sqrt(5-2*sqrt(10/7));
   Gpw(2,2)=(322+13*sqrt(70))/900;
   Gpw(3,1)=0;
   Gpw(3,2)=128/225;
   Gpw(4,1)=+1/3*sqrt(5-2*sqrt(10/7));
   Gpw(4,2)=(322+13*sqrt(70))/900;
   Gpw(5,1)=+1/3*sqrt(5+2*sqrt(10/7));
   Gpw(5,2)=(322-13*sqrt(70))/900;
case 6
   Gpw(1,1) = -0.932469514203151938982;
   Gpw(2,1) = -0.661209386466264592509;
   Gpw(3,1) = -0.238619186083196932469;
   Gpw(4,1) = 0.238619186083196932469;
   Gpw(5,1) = 0.661209386466264592509;
   Gpw(6,1) = 0.932469514203151938982;
   Gpw(1,2) = 0.171324492379171867455;
   Gpw(2,2) = 0.360761573048138139704;
   Gpw(3,2) = 0.467913934572691007877;
   Gpw(4,2) = 0.467913934572691007877;
   Gpw(5,2) = 0.360761573048138139704;
   Gpw(6,2) = 0.171324492379171867455;
case 7
   Gpw(1,1) = -0.949107912342758486214;
   Gpw(2,1) = -0.741531185599394682074;
   Gpw(3,1) = -0.405845151377397184156;
   Gpw(4,1) = 0.000000000000000000000;
   Gpw(5,1) = 0.405845151377397184156;
   Gpw(6,1) = 0.741531185599394682074;
   Gpw(7,1) = 0.949107912342758486214;
   Gpw(1,2) = 0.129484966168870452732;
   Gpw(2,2) = 0.279705391489274882221;
   Gpw(3,2) = 0.381830050505118893749;
   Gpw(4,2) = 0.417959183673469387755;
   Gpw(5,2) = 0.381830050505118893749;
   Gpw(6,2) = 0.279705391489274882221;
   Gpw(7,2) = 0.129484966168870452732;
case 8
   Gpw(1,1) = -0.960289856497536509162;
   Gpw(2,1) = -0.796666477413626949956;
   Gpw(3,1) = -0.525532409916328990763;
   Gpw(4,1) = -0.183434642495649807836;
   Gpw(5,1) = 0.183434642495649807836;
```

```
    Gpw(6,1) = 0.525532409916328990763;
    Gpw(7,1) = 0.796666477413626949956;
    Gpw(8,1) = 0.960289856497536509162;
    Gpw(1,2) = 0.101228536290370022005;
    Gpw(2,2) = 0.222381034453372634246;
    Gpw(3,2) = 0.313706645877887267061;
    Gpw(4,2) = 0.362683783378361979375;
    Gpw(5,2) = 0.362683783378361979375;
    Gpw(6,2) = 0.313706645877887267061;
    Gpw(7,2) = 0.222381034453372634246;
    Gpw(8,2) = 0.101228536290370022005;
case 9
    Gpw(1,1) = -0.968160239507625086598;
    Gpw(2,1) = -0.836031107326636102552;
    Gpw(3,1) = -0.613371432700592578104;
    Gpw(4,1) = -0.324253423403808971326;
    Gpw(5,1) = 0.000000000000000000000;
    Gpw(6,1) = 0.324253423403808971326;
    Gpw(7,1) = 0.613371432700592578104;
    Gpw(8,1) = 0.836031107326636102552;
    Gpw(9,1) = 0.968160239507625086598;
    Gpw(1,2) = 0.081274388361599606396;
    Gpw(2,2) = 0.180648160694854311268;
    Gpw(3,2) = 0.260610696402924285138;
    Gpw(4,2) = 0.312347077040002744348;
    Gpw(5,2) = 0.330239355001259763165;
    Gpw(6,2) = 0.312347077040002744348;
    Gpw(7,2) = 0.260610696402924285138;
    Gpw(8,2) = 0.180648160694854311268;
    Gpw(9,2) = 0.081274388361599606396;
case 10
    Gpw(1,1) = -0.973906528517169189864;
    Gpw(2,1) = -0.865063366688986756738;
    Gpw(3,1) = -0.679409568299023991446;
    Gpw(4,1) = -0.433395394129247379933;
    Gpw(5,1) = -0.148874338981631215706;
    Gpw(6,1) = 0.148874338981631215706;
    Gpw(7,1) = 0.433395394129247379933;
    Gpw(8,1) = 0.679409568299023991446;
    Gpw(9,1) = 0.865063366688986756738;
    Gpw(10,1)= 0.973906528517169189864;
    Gpw(1,2) = 0.066671344308758311881;
    Gpw(2,2) = 0.149451349150555209279;
    Gpw(3,2) = 0.219086362515984566833;
    Gpw(4,2) = 0.269266719309995757214;
```

```
    Gpw(5,2) = 0.295524224714752865402;
    Gpw(6,2) = 0.295524224714752865402;
    Gpw(7,2) = 0.269266719309995757214;
    Gpw(8,2) = 0.219086362515984566833;
    Gpw(9,2) = 0.149451349150555209279;
    Gpw(10,2) = 0.066671344308758311881;
end
return
```

2.4 Sinusoidal Curve Example

MATLAB code Sinusoidalcurve.m (Sect. 2.4.1) uses the functions Nurbs.m (Sect. 1.3.10) and DNurbsLeibnitz.m (Sect. 1.3.14), to draw a sinusoidal curve (2.14), its derivative, Jacobian values, and curvature along the length of a curve.

$$y = 1.5 \sin [x] \quad , \quad 0 \leq x \leq 2\pi \tag{2.14}$$

Rewriting the above equation into the Gaussian coordinate form,

$$y = 1.5 \sin [\pi(\xi + 1)] \quad , \quad -1 \leq \xi \leq +1 \tag{2.15}$$

with

$x = \pi(\xi + 1)$

Jacobian, $J(\xi) = \frac{dy}{d\xi} = \frac{dy}{dx} \cdot \frac{dx}{d\xi} = 1.5\pi \cos [\pi(\xi + 1)]$.

Curvature, $\kappa(\xi) = \frac{\left| \frac{dx}{d\xi} \frac{d^2y}{d\xi^2} - \frac{dy}{d\xi} \frac{d^2x}{d\xi^2} \right|}{J(\xi)^3} = \frac{\left| -\pi^2 J(\xi) \right|}{J(\xi)^3} = \frac{\pi^2}{J(\xi)^2}$.

Figure 2.3 depicts the sinusoidal curve ($p = 8$), its derivative curve, Jacobian values, and curvature along the length of curve.

2.4.1 Sinusoidalcurve Program List

```
% Sinusoidalcurve.m
% Calculating the Jacobian and Radius of Curvature of a sinusoidal
  curve
% p = degree of Bspline curves
clear all;clc;
p=8;
```

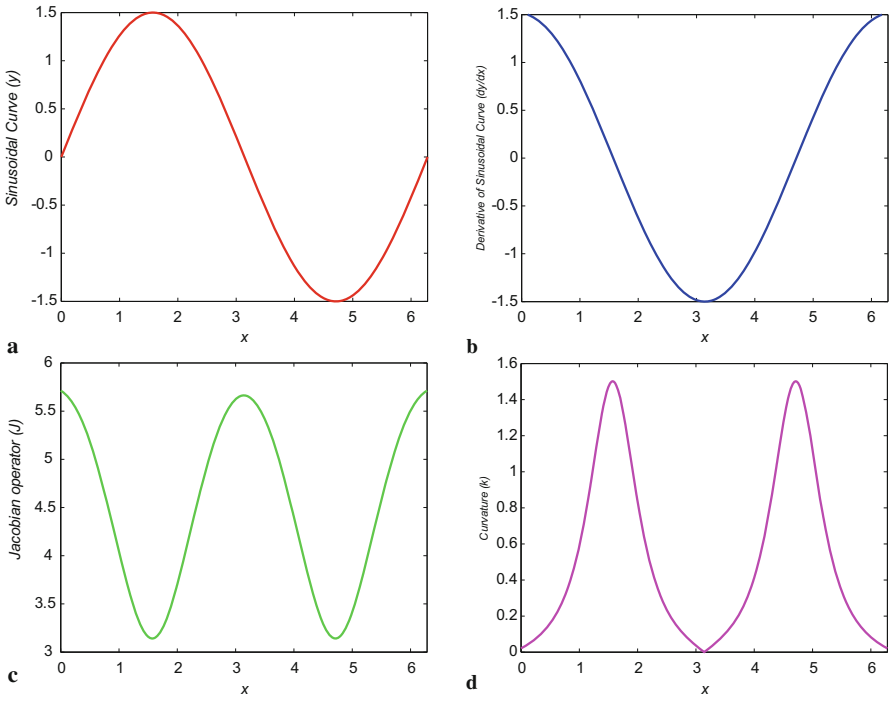

Fig. 2.3 Sinusoidal curve example: (**a**) curve; (**b**) derivative of curve; (**c**) Jacobian; (**d**) curvature

```
xi=zeros(1,p+1);
yi=zeros(1,p+1);
c1=(0.2*sqrt(2)/(sqrt(2)-1));
for ip=1:p+1
 xi(1,ip)=2*pi*(ip-1)/p;
 ksi=(xi(1,ip)-pi)/pi;
 yi(1,ip)=1.5*sin(pi*(ksi+1));
end
xyi=transpose([xi yi]);
% Knot vector
knot=[-1*ones(1,p+1) 1*ones(1,p+1)];
knot=[knot max(knot)*ones(1,p)];
nb=size(knot,2)-p-1;          % max number of basis
                              functions
% Parameter t
t=-1:2/p:1;
nt=size(t,2);
wt=ones(1,p+1);
Sb=zeros(nt,1);               % BsplineBasisPlot (order,basis)
tmi=zeros(nt,p+1);
```

```
for i=0:nb-p-1
 for dt=1:nt
   Nurbsbasis = Nurbs(p,knot,wt,t(dt));
   tmi(dt,i+1)=Nurbsbasis(i+1,p+2);
 end
end
% [tm] matrix
tm=[tmi zeros(nb-p,nb-p);zeros(nb-p,nb-p) tmi];
% Coefficients Px's and Py's calculation
pxy=tm\xyi;
px=pxy(1:nt);      % n+1 control points
py=pxy(nt+1:2*nt); % n+1 control points
%nb=size(knot,2)-p-1; % max number of basis functions
ndt=100;              % steps of parameter t
t1=min(knot); t2=max(knot);
t=t1:(t2-t1)/ndt:t2;
nt=size(t,2);
wt=ones(nb-p);         % Uniform weights
curvx=zeros(nt,1);     % Nurbs variables
curvy=zeros(nt,1); curvyd=zeros(nt,1);
xx=zeros(nt);
Jac=zeros(nt);
curva=zeros(nt,1);
% Calculating the Jacobian and Radius of Curvature
for it=1:nt
 Nurbsbasis = DNurbsLeibnitz(p,knot,wt,t(it));
 dx=0;   dy=0;
 ddy=0; ddx=0;
 for ib=1:nb-p                      % Number of basis function
 dx=dx+Nurbsbasis(ib,p+7)*px(ib);   % dx
 dy=dy+Nurbsbasis(ib,p+7)*py(ib);   % dy
 ddx=ddx+Nurbsbasis(ib,p+8)*px(ib); % dx
 ddy=ddy+Nurbsbasis(ib,p+8)*py(ib); % dy
 end
 Jac(it) =sqrt(dx^2+dy^2);
 curva(it)=abs(dx*ddy-dy*ddx)/Jac(it)^3;
end
% Drawing the Curve and its Derivative
for it=1:nt                 % ksi
 dy=0; dx=0;
 for ib=1:nb-p              % Number of basis function
 Nurbsbasis = DNurbsLeibnitz(p,knot,wt,t(it));
 curvx(it) =curvx(it) +Nurbsbasis(ib,p+5)*px(ib);
 curvy(it)=curvy(it)+Nurbsbasis(ib,p+5)*py(ib); % NonDerivative
 dx=dx+Nurbsbasis(ib,p+7)/Jac(it)*px(ib);
```

```
 dy=dy+Nurbsbasis(ib,p+7)/Jac(it)*py(ib);
 end
 curvyd(it)=dy/dx; % 1st Derivative
end
figure(1); plot(curvx,curvy,'-k','LineWidth',2);hold all;
axis([0.0 2*pi -1.5 1.5]);
xlabel('\itx'); ylabel('\itSinusoidal Curve (y)');
figure(2); plot(curvx,curvyd,'-k','LineWidth',2);hold all;
axis([0.0 2*pi -1.5 1.5]);
xlabel('\itx'); ylabel('\itDerivative of Sinusoidal Curve
(dy/dx)');
figure(3); plot(curvx,Jac,'-k','LineWidth',2);hold all;
axis([0.0 2*pi 3.0 6.0]);
xlabel('\itx'); ylabel('\itJacobian operator (J)');
figure(4); plot(curvx,curva,'-k','LineWidth',2);hold all;
axis([0.0 2*pi 0.0 1.6]);
xlabel('\itx'); ylabel('\itCurvature (k)');
```

Further Reading

Gradshteyn IS, Ryzhik IM (2000) Table of integrals, series, and products, 6th edn. Academic Press, San Diego

Hildebrand FB (1956) Introduction to numerical analysis. McGraw-Hill, New York

Kaplan W (1984) Advanced calculus, 3rd edn. Addison-Wesley, Reading

Ralston A, Rabinowitz P (1978) A first course in numerical analysis. McGraw-Hill, London

http://www.springer.com/978-3-319-56492-0

An Isogeometric Approach to Beam Structures
Bridging the Classical to Modern Technique
Gan, B.S.
2018, XIV, 233 p. 65 illus., 44 illus. in color., Hardcover
ISBN: 978-3-319-56492-0