2D Gaussian quadratures: Quadrilateral Example

Contents

Example of 2D integration

Let's consider the function

$F(x,y) = x^2 - y^2$ defined on the quadrilateral Ω_k with vertices

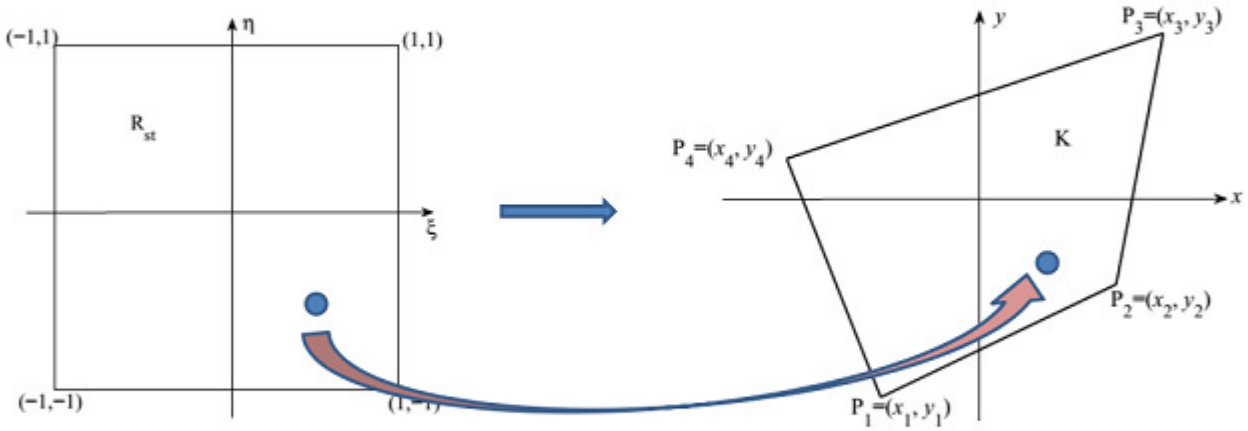
$$v_1 = (0,0), \quad v_2 = (5,-1), \quad v_3 = (4,5), \quad v_4 = (1,4).$$

We want to compute

$$\int_{\Omega_k} F(x,y)dxdy$$

Quadrilateral Elements

- **General Quadrilateral Elements**



Change of variables

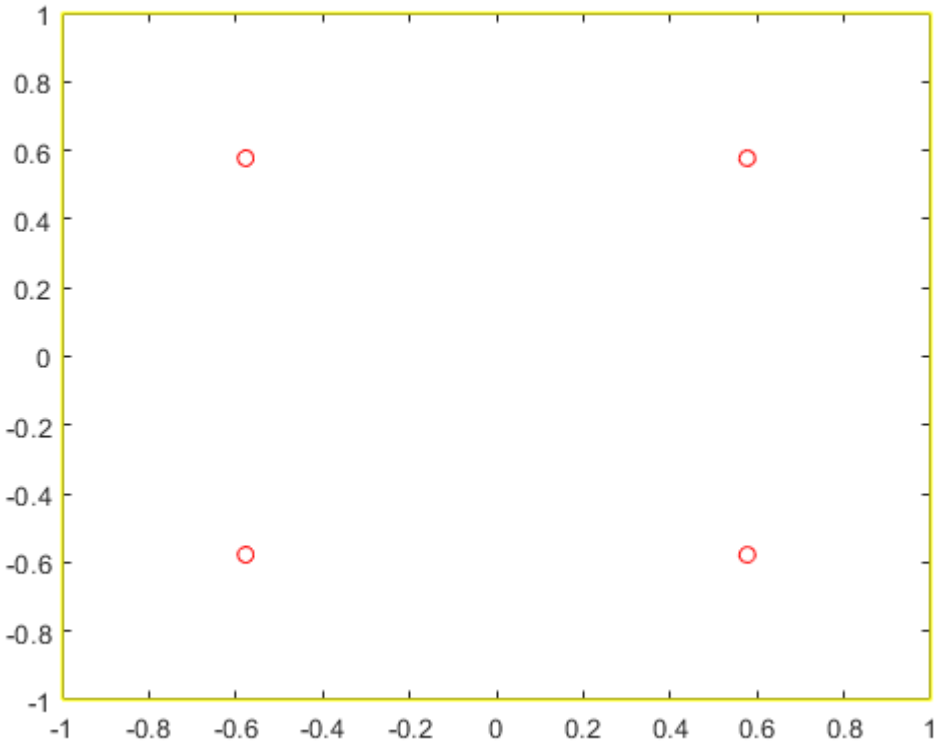
$$\begin{pmatrix} x \\ y \end{pmatrix} = \phi_k(\xi,\eta) = \begin{pmatrix} \psi_1^R(\xi,\eta)\,x_1^k + \psi_2^R(\xi,\eta)\,x_2^k + \psi_3^R(\xi,\eta)\,x_3^k + \psi_4^R(\xi,\eta)\,x_4^k \\ \psi_1^R(\xi,\eta)\,y_1^k + \psi_2^R(\xi,\eta)\,y_2^k + \psi_3^R(\xi,\eta)\,y_3^k + \psi_4^R(\xi,\eta)\,y_4^k \end{pmatrix}$$

Compute the 2D Gauss points on the reference element

First we compute the appropriate Gauss points in the reference quadrilateral.

We can use a Gauss rule of only **N=2** in this case because $F(x,y)$ is a polynomial function of degree less than 3 in each variable.

```
N=2; %order of the Gaussian quadrature
[w,ptGaussRef]=gaussValues2DQuad(N);
% Draw Gauss points in the reference quadrilateral
plotRectangle([-1 -1],[1,-1],[1,1],[-1,1]);
hold on;
plot(ptGaussRef(:,1),ptGaussRef(:,2),'ro');
hold off;
%
```



Define the function $F(x,y)$ and the domain Ω_k

Now consider our function $F(x) = x^2 - y^2$ and we code in Matlab the above change of variable formulas:

```
% The present function
F =@(x,y) (x.^2)- y.^2;
% The domain vertices
v1=[0,0];
v2=[5,-1];
v3=[4,5];
v4=[1,4];
```

Compute the change of variables functions

The needed functions are the **shape functions**. These functions are associated to the respective vertex and satisfy

$\psi_i(v_j) = 0$ if $i \neq j$ and

$\psi_i(v_j) = 1$ if $i = j$

```
% We use Matlab *implicit function* definition:
%
% Shape functions
Psi1=@(x,y)(1-x).*(1-y)/4;
Psi2=@(x,y)(1+x).*(1-y)/4;
Psi3=@(x,y)(1+x).*(1+y)/4;
Psi4=@(x,y)(1-x).*(1+y)/4;
% Shape function derivatives
dPsi11=@(x,y) -(1-y)/4;
dPsi21=@(x,y) (1-y)/4;
dPsi31=@(x,y) (1+y)/4;
```

```
dPsi41=@(x,y) -(1+y)/4;
dPsi12=@(x,y) -(1-x)/4;
dPsi22=@(x,y) -(1+x)/4;
dPsi32=@(x,y) (1+x)/4;
dPsi42=@(x,y) (1-x)/4;
% Gradient matrix
Jacb =@(x,y) [dPsi11(x,y), dPsi21(x,y),dPsi31(x,y),dPsi41(x,y);...
              dPsi12(x,y), dPsi22(x,y),dPsi32(x,y),dPsi42(x,y)];
```

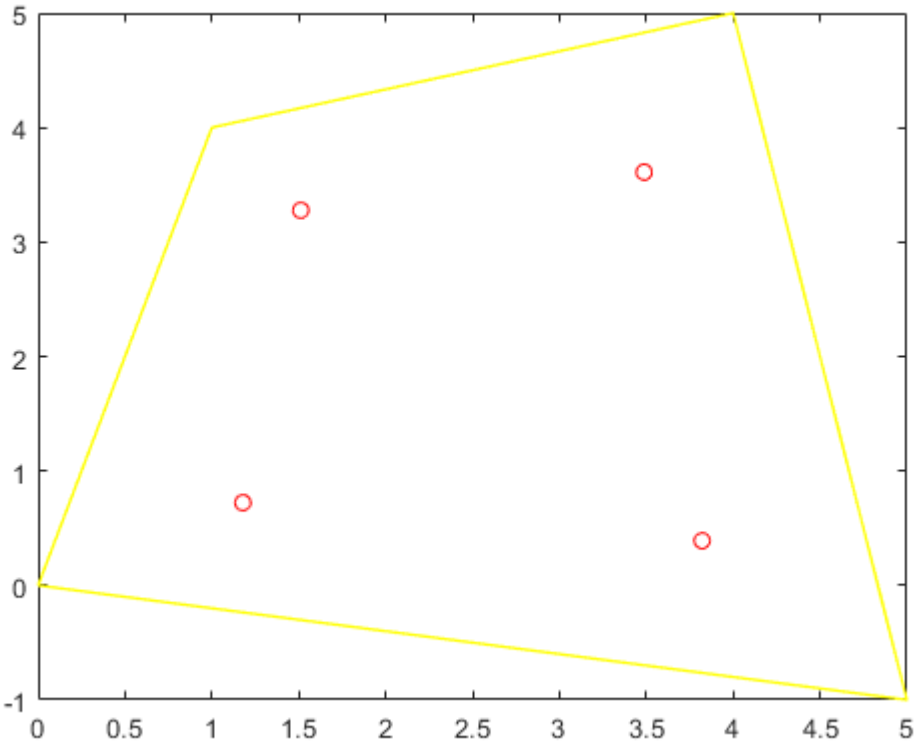
Compute the corresponding Gaussian points on the domain

The change of variables from the reference quadrilateral to a genral one is:

$$(x,y) = \psi_1(\xi,\eta)V_1 + \psi_2(\xi,\eta)V_2 + \psi_3(\xi,\eta)V_3 + \psi_4(\xi,\eta)V_4$$

evaluate Shape functions on Gaussian reference points

```
xx = ptGaussRef(:,1);
yy = ptGaussRef(:,2);
evalPsi1 = Psi1(xx,yy);
evalPsi2 = Psi2(xx,yy);
evalPsi3 = Psi3(xx,yy);
evalPsi4 = Psi4(xx,yy);
% from the change of variables function
ptGaussDomain = evalPsi1*v1+evalPsi2*v2+evalPsi3*v3+evalPsi4*v4;
% Draw Gauss points in the present domain
figure()
plotRectangle(v1,v2,v3,v4);
hold on;
plot(ptGaussDomain(:,1),ptGaussDomain(:,2),'ro');
hold off;
```



Compute the Jacobian terms

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

Quadrilateral Elements

- **The Jacobian term**

$$J = \frac{\partial(x,y)}{\partial(\xi,\eta)} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{4} x_i^k \frac{\partial \psi_i^R}{\partial \xi} & \sum_{i=1}^{4} x_i^k \frac{\partial \psi_i^R}{\partial \eta} \\ \sum_{i=1}^{4} y_i^k \frac{\partial \psi_i^R}{\partial \xi} & \sum_{i=1}^{4} y_i^k \frac{\partial \psi_i^R}{\partial \eta} \end{pmatrix}$$

Taken into account the definition on the shape functions

$$J^T = \frac{1}{4}\begin{bmatrix} -(1-\eta) & 1-\eta & 1+\eta & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & 1+\xi & 1-\xi \end{bmatrix}\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

- **Numerical Gaussian 2D quadrature**

$$\iint_K F(x,y)\,dxdy \approx \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j F(P(\xi_i,\eta_j), Q(\xi_i,\eta_j))|J(\xi_i,\eta_j)|.$$

vertex matrix

```
v = [v1;v2;v3;v4];
```

```
% evaluate Jacobian contribution for each Gauss point
for i=1:size(xx,1)
    evalDetJacb(i) = det(Jacb(xx(i),yy(i))*v);
end
```

Compute the integral value according Gauss formula

```
%evaluate the function on the domain points
evalF=F(ptGaussDomain(:,1),ptGaussDomain(:,2));

% Finally, apply Gauss integration formula
suma=0;
for i=1:size(ptGaussDomain,1)
    suma=suma+w(i)*evalF(i)*evalDetJacb(i);
end
integ = suma
```

```
integ =

   6.0000e+01
```

Exercise 1: Build the integQuad function

From the previous example, build a function that returns the value of the integral of an inline function defined on a quadrilateral domain (see at the end of this document).

function integralValue=integQuad(F,vertices,N)

where

- **F:** is an inline function previously defined
- **vertices:** are the coordinates of the quadrilateral vertices.
- **N:** order of Gauss integration

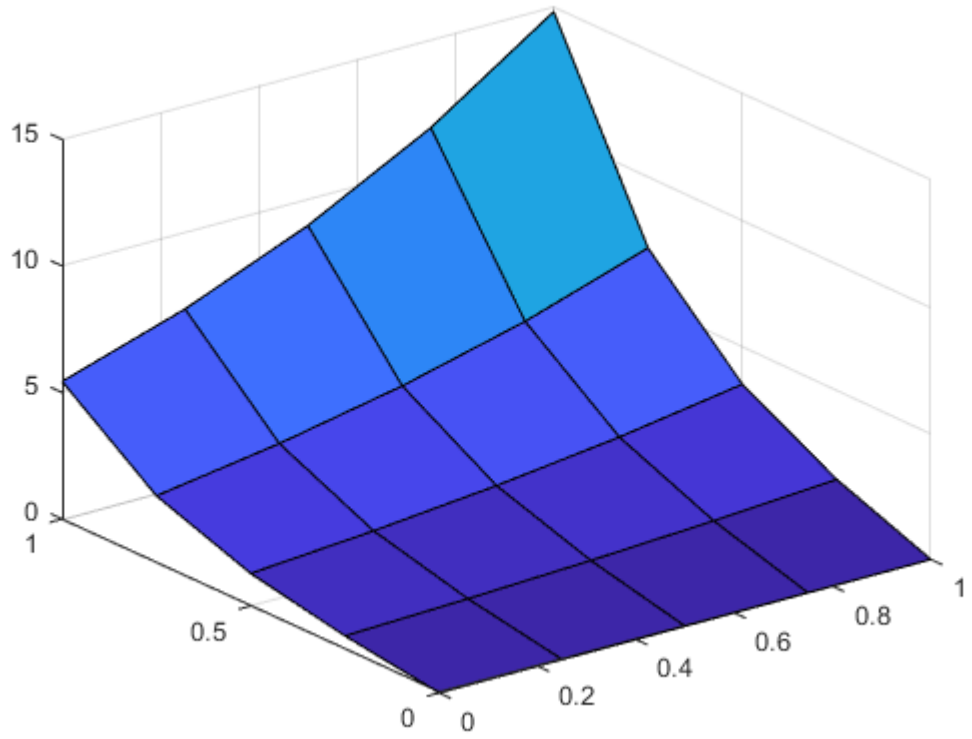
Application: Integration over a mesh

Let's consider the integral of $F(x,y) = 2\exp^{x+y^2} y$ in a rectangular domain [xmin,xmax]x[ymin,ymax]

We want to show how to approximate the integral value using a mesh subdivision of the domain.

```
F=@(x,y) 2*exp(x+y.^2).*y; %present function
xmin=0; % rectangle dimensions
xmax=1;
ymin=0;
ymax=1;
hx=1/4; %number of subdivisions
hy=hx;  % it can be different for each axis
xx=xmin:hx:xmax; %all points in the x-axis
yy=ymin:hy:ymax; %all points in the y-axis
[X,Y] = ndgrid(xx,yy); % not really a FEM grid
Z = F(X,Y); % function values
surf(X,Y,Z); % optional: is just to visualize the result
```

```
[elem,vertex] = surf2patch(X,Y,Z); % the main variables
% we must permute columns 2 and 4 to have a positive orientation
% (if the mesh is well orientated this is not needed)
perm=[1,4,3,2];
elem=elem(:,perm);
%
```



Mesh definition

vertex: is a numPx3 matrix, where numP is the number of grid points. For each row:

- the two first coordinates are the (x,y) grid values
- the third coordinate is the function value: z=F(x,y)

elem: is a numElemx4 matrix, where numElem is the number of elements For each row:

- the four numbers are the number of the vertices defining this element

```
numElem=size(elem,1); %total number of elements
N=2; %number of Gauss points = NxN
integralTot=0;
for i=1:numElem %compute the integral on each element
    v1=[vertex(elem(i,1),1),vertex(elem(i,1),2)];
    v2=[vertex(elem(i,2),1),vertex(elem(i,2),2)];
    v3=[vertex(elem(i,3),1),vertex(elem(i,3),2)];
    v4=[vertex(elem(i,4),1),vertex(elem(i,4),2)];
    vertices=[v1;v2;v3;v4];
    elemInteg=integQuad(F,vertices,N);
    integralTot=integralTot+elemInteg;
end
actualIntegVal= (exp(1)-1)^2 %exact known value
```

```
errorInt=abs(actualIntegVal-integralTot) %absolute error
```

Quadrilateral integration function: integQuad

Copy this function in a new file named **integQuad.m**

```
function valInteg = integQuad(F,vertices,N)
    [w,ptGaussRef]=gaussValues2DQuad(N);
    % Shape functions
    Psi1=@(x,y)(1-x).*(1-y)/4;
    Psi2=@(x,y)(1+x).*(1-y)/4;
    Psi3=@(x,y)(1+x).*(1+y)/4;
    Psi4=@(x,y)(1-x).*(1+y)/4;
    % Shape function derivatives
    dPsi11=@(x,y) -(1-y)/4;
    dPsi21=@(x,y) (1-y)/4;
    dPsi31=@(x,y) (1+y)/4;
    dPsi41=@(x,y) -(1+y)/4;
    dPsi12=@(x,y) -(1-x)/4;
    dPsi22=@(x,y) -(1+x)/4;
    dPsi32=@(x,y) (1+x)/4;
    dPsi42=@(x,y) (1-x)/4;
    % Gradient matrix
    Jacb =@(x,y) [dPsi11(x,y), dPsi21(x,y),dPsi31(x,y),dPsi41(x,y);...
                  dPsi12(x,y), dPsi22(x,y),dPsi32(x,y),dPsi42(x,y)];
    % evaluate Shape functions on Gaussian reference points
    xx = ptGaussRef(:,1);
    yy = ptGaussRef(:,2);
    evalPsi1 = Psi1(xx,yy);
    evalPsi2 = Psi2(xx,yy);
    evalPsi3 = Psi3(xx,yy);
    evalPsi4 = Psi4(xx,yy);
    % from the change of variables function
    ptGaussDomain = [evalPsi1,evalPsi2,evalPsi3,evalPsi4]*vertices;
    % evaluate Jacobian contribution for each point
    for i=1:size(xx,1)
        evalDetJacb(i) = det(Jacb(xx(i),yy(i))*vertices);
    end
    %evaluate the function on the domain points
    evalF=F(ptGaussDomain(:,1),ptGaussDomain(:,2));
    % Finally, apply Gauss formula
    suma=0;
    for i=1:size(ptGaussDomain,1)
        suma=suma+w(i)*evalF(i)*evalDetJacb(i);
    end
    valInteg = suma;
end
```

```
actualIntegVal =

   2.9525e+00


errorInt =

   2.9420e-04
```

Exercise 2:

Increase the number of subdivisions and see how the error evolves.

Published with MATLAB® R2017b