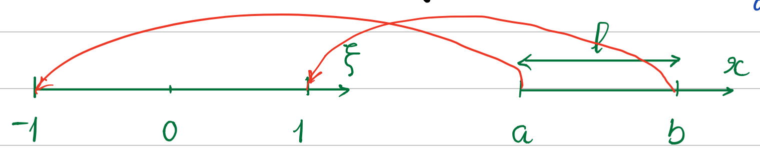
Topic: Gauss Integration — Notebook

In FEM, numerical integration is needed. Although there are many numerical integration techniques, Gauss quadrature, which is described in this section, is one of the most efficient techniques for functions that are polynomials or nearly polynomials. In FEM, the integrals involves polynomials, so Gauss quadrature is a natural choice

- Consider the following integral: $I = \int_a^b f(x)\, dx = ?$ (1)



- Mapping of the 1D domain from the parent domain [-1, 1] to the physical domain [a, b]

$$x = \frac{1}{2}(a+b) + \frac{1}{2}\xi(b-a) \qquad (2)$$

- The above map can also be written directly in terms of the linear shape functions:

$$x = x_1 N_1(\xi) + x_2 N_2(\xi) = a\,\frac{1-\xi}{2} + b\,\frac{1+\xi}{2}$$

(1) $\Rightarrow$ $dx = \frac{1}{2}(b-a)\, d\xi = \frac{\ell}{2} d\xi = \underbrace{J}_{\text{Jacobian}} d\xi$

(1) $\Leftrightarrow$ $I = J\int_{-1}^{1} f(\xi)\, d\xi = J\hat{I}$; $\hat{I} = \int_{-1}^{1} f(\xi)\, d\xi$)

- In the Gauss integration procedure, we approximate the integral by:

points

$$\hat{I} = w_1 f(\xi_1) + w_2 f(\xi_2) + \dots + w_n f(\xi_n)$$

weights

$$\Leftrightarrow \hat{I} = \underbrace{[w_1 \;\; w_2 \; \dots \; w_n]}_{w^T} \cdot \underbrace{\begin{bmatrix} f(\xi_1) \\ f(\xi_2) \\ \vdots \\ f(\xi_n) \end{bmatrix}}_{\mathbb{f}} = w^T \mathbb{f} \qquad (3)$$

- The basic idea of the Gauss integration quadrature is to choose the weights and integration points so that the highest possible polynomial is integrated exactly.
- $f(\xi)$ is approximated by a polynomial as:

$$f(\xi) = \alpha_1 + \alpha_2 \xi + \alpha_3 \xi^2 + \dots + \alpha_m \xi^m = \underbrace{[1 \; \xi \; \xi^2 \dots \xi^m]}_{\mathbb{P}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix}}_{\alpha}$$

- Next, we express the values of the coefficients α_i in terms of the function $f(\xi)$ at the integration points:

$$\begin{aligned} f(\xi_1) &= \alpha_1 + \alpha_2 \xi_1 + \alpha_3 \xi_1^2 + \dots + \alpha_m \xi_1^m \\ f(\xi_2) &= \alpha_1 + \alpha_2 \xi_2 + \alpha_3 \xi_2^2 + \dots + \alpha_m \xi_2^m \quad \text{or} \\ &\vdots \\ f(\xi_n) &= \alpha_1 + \alpha_2 \xi_3 + \alpha_3 \xi_3^2 + \dots + \alpha_m \xi_n^m \end{aligned} \qquad \underbrace{\begin{bmatrix} f(\xi_1) \\ f(\xi_2) \\ \vdots \\ f(\xi_n) \end{bmatrix}}_{f} = \underbrace{\begin{bmatrix} 1 & \xi_1 & \xi_1^2 & \dots & \xi_1^m \\ 1 & \xi_2 & \xi_2^2 & \dots & \xi_2^m \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & \xi_n & \xi_n^2 & \dots & \xi_n^m \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix}}_{\alpha} \quad (4)$$

. m: order or f function

. n: the number of Gauss Point

$$(3)(4) \Rightarrow \quad \hat{I} = W^T M \alpha$$

- Gauss quadrature provides the weights & integration points that yield an exact integral of a polynomial of a given order. To detect what the weights and quadrature points should be, we integrate the polynomial $f(\xi)$

$$\begin{cases} \hat{I} = \int_{-1}^{1} f(\xi)\, d\xi = \int_{-1}^{1} [1 \; \xi \; \xi^2 \dots \xi^m] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix} d\xi = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix} \int_{-1}^{1} [1 \; \xi \; \xi^2 \dots \xi^m]\, d\xi \\ \hat{I} = [w_1 \quad w_2 \dots w_n] \begin{bmatrix} 1 & \xi_1 & \xi_1^2 & \dots & \xi_1^m \\ 1 & \xi_2 & \xi_2^2 & \dots & \xi_2^m \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & \xi_n & \xi_n^2 & \dots & \xi_n^m \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix} \end{cases}$$

$$\int_{-1}^{1} [1 \; \xi \; \xi^2 \dots \xi^m]\, d\xi = [w_1 \; w_2 \dots w_n] \begin{bmatrix} 1 & \xi_1 & \xi_1^2 & \dots & \xi_1^m \\ 1 & \xi_2 & \xi_2^2 & \dots & \xi_2^m \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & \xi_n & \xi_n^2 & \dots & \xi_n^m \end{bmatrix}$$

$\Rightarrow$ Solve this equation for w_i, ξ_i

$\Rightarrow n$ Gauss Point $\longrightarrow$ ($2n$ unknowns, $\underline{(m+1)\text{equations}}$)

m: highest order of f function

. n Gauss Points $\Rightarrow$ can estimate m^{th} order function with 1 condition:

$$2n = m+1$$
$$n = \frac{m+1}{2}$$

Example ① $m=3 \Rightarrow n=2$. This means 2 Gauss Points can estimate exactly a function of order 3^{rd}. Of course, this also mean that 2 Gauss points can estimate a function of order less than 3.

② $m=4 \Rightarrow n=2,5 \Rightarrow 2.5$ Gauss Point can estimate exactly 4^{th} order function. But the number of Gauss Point should be integer $\Rightarrow n=3$!

$$\int_{-1}^{1} [1 \ \xi \ \xi^2 \ldots \ \xi^m] d\xi = [w_1 \ w_2 \ldots \ w_n] \begin{bmatrix} 1 & \xi_1 & \xi_1^2 & \ldots & \xi_1^m \\ 1 & \xi_2 & \xi_2^2 & \ldots & \xi_2^m \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & \xi_n & \xi_n^2 & \ldots & \xi_n^m \end{bmatrix}$$

1^{st} equation:

$$\int_{-1}^{1} 1 \, d\xi = w_1 + w_2 + \ldots + w_n$$

2^{nd} equation:

$$\int_{-1}^{1} \xi \, d\xi = w_1 \xi_1 + w_2 \xi_2 + \ldots + w_n \xi_n$$

$\vdots$

$(m+1)^{th}$ equation

$$\int_{-1}^{1} \xi^m \, d\xi = w_1 \xi_1^m + w_2 \xi_2^m + \ldots + w_n \xi_n^m$$

- 1 Gauss Point : ⇒ 2 equations:

$$\int_{-1}^{1} 1\, d\xi = w_1 \qquad \Rightarrow w_1 = 2$$

$$\int_{-1}^{1} \xi\, d\xi = w_1 \xi_1 \qquad \Rightarrow \xi = 0$$

- 2 Gauss Points ⇒ 4 equations:

$$\int_{-1}^{1} 1\, d\xi = w_1 + w_2 \Rightarrow w_1 + w_2 = 2 \qquad (1)$$

$$\int_{-1}^{1} \xi\, d\xi = w_1 \xi_1 + w_2 \xi_2 \Rightarrow w_1 \xi_1 + w_2 \xi_2 = 0 \qquad (2)$$

$$\int_{-1}^{1} \xi^2\, d\xi = w_1 \xi_1^2 + w_2 \xi_2^2 \Rightarrow w_1 \xi_1^2 + w_2 \xi_2^2 = 2/3 \qquad (3)$$

$$\int_{-1}^{1} \xi^3\, d\xi = w_1 \xi_1^3 + w_2 \xi_2^3 \Rightarrow w_1 \xi_1^3 + w_2 \xi_2^3 = 0 \qquad (4)$$

$(1)(4) \Rightarrow \xi_1^2 = \xi_2^2 \Rightarrow \xi_1 = -\xi_2$, plug into 2

$\Rightarrow w_1 - w_2 = 0$

$\Rightarrow w_1 = w_2 = 1$

$(3) \Rightarrow \xi_1^2 = 1/3 \Rightarrow \xi_1 = -1/\sqrt{3}\ ;\ \xi_2 = 1/\sqrt{3}$

- 3 Gauss points:

$\xi_1 \qquad \xi_2 \qquad \xi_3$

$w_1 \qquad 0 \qquad w_2$

Symmetric properties:
$\xi_2 = 0\ ;\ \xi_1 = -\xi_3$

$$\begin{cases} w_1 + w_2 + w_3 = 2 \\ w_1 \xi_1 + \cancel{w_2 \xi_2}^{\to 0} + w_3 \xi_3 = 0 \\ w_1 \xi_1^2 + \cancel{w_2 \xi_2^2}^{\to 0} + w_3 \xi_3^2 = 2/3 \\ w_1 \xi_1^4 + \cancel{w_2 \xi_2^4}^{\to 0} + w_3 \xi_3^4 = 2/5 \end{cases}$$

$\xrightarrow{\xi_1 = -\xi_3} w_1 = w_3$

$\xrightarrow[w_1 = w_3]{\xi_1 = -\xi_3} w_1 \xi_1^2 = 1/3$

$\longrightarrow w_1 \xi_1^4 = 1/5$

$\Rightarrow \xi_1^2 = \frac{3}{5}$

$\Rightarrow \xi_1 = -\sqrt{\frac{3}{5}}\ ;\ \xi_3 = \sqrt{\frac{3}{5}}$

$\Rightarrow w_1 \cdot \frac{3}{5} = \frac{1}{3} \Rightarrow w_1 = \frac{5}{9}$

$\Rightarrow w_3 = \frac{5}{9}$

$\Rightarrow w_2 = 2 - 2 \cdot \frac{5}{9} = \frac{8}{9}$

Example: Evaluate $I = \int_2^5 (x^3 + x^2)\,dx$

- $2n_{gp} - 1 = 3 \Rightarrow n_{gp} = 2 \Rightarrow \begin{cases} w_1 = w_2 = 1 \\ \xi_1 = -\frac{1}{\sqrt{3}} ;\ \xi_2 = \frac{1}{\sqrt{3}} \end{cases}$

- $x = \frac{1}{2}(a+b) + \frac{1}{2}\xi(b-a) = 3.5 + 1.5\xi.$

- $f(\xi) = (3.5 + 1.5\xi)^3 + (3.5 + 1.5\xi)^2$

- $\hat{I} = J\hat{I}\ \frac{\ell}{2}\int_{-1}^{1}\left[(3.5+1.5\xi)^3 + (3.5+1.5\xi)^2\right]d\xi$

$$= \frac{\ell}{2}\left[w_1\left[(3.5+1.5\xi_1)^3 + (3.5+1.5\xi_1)^2\right] + w_2\left[(3.5+1.5\xi_1)^3 + (3.5+1.5\xi_1)^2\right]\right]$$

$$= 191.25$$

- In this case, as Gauss integration is exact, we can check the result by performing analytical integration

$$\int_2^5 (x^3 + x^2)\,dx = \left(\frac{x^4}{4} + \frac{x^3}{3}\right)\Bigg|_2^5 = 191,25$$

Topic: Notebook

2D integration on the domain $[-1, 1] \times [-1, 1]$

$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi, \eta)\, d\xi\, d\eta = \int_{-1}^{1}\left(\int_{-1}^{1} f(\xi, \eta)\, d\xi\right) d\eta$$

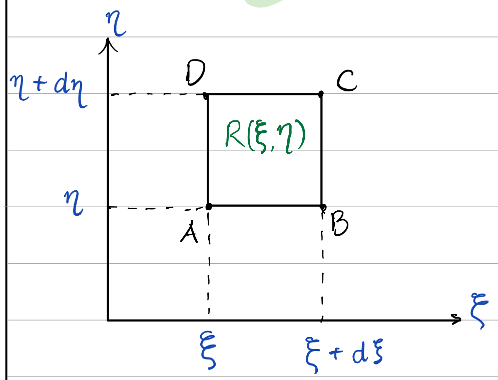
$$= \int_{-1}^{1}\left[\sum_{i=1}^{n_\xi^G} w_i f(\xi_i, \eta)\right] d\eta$$

$$= \sum_{K=1}^{n_\eta^G} \sum_{i=1}^{n_\xi^G} \underbrace{w_i w_K}_{\text{gauss-weight}} f(\underbrace{\xi_i, \eta_K}_{\text{gauss-point}})$$

A Jacobian is required for integrals in more than one variables. Suppose that:

$$x = f(\xi,\eta) \quad ; \quad y = g(\xi,\eta)$$

Let 's see what happens to a small infinisimal box in the uv plane:



Since the side-lengths are infinisimal each side of the box in the uv plane is transformed into a straight line in the xy plane. The result is that the box in the uv plane is transformed into a parallelogram into the xy plane

Suppose:

1. The point (ξ,η) is transformed into the point
$$(x = f(\xi,\eta) \, , \; y = g(\xi,\eta))$$
2. The point $B(\xi + d\xi, \eta)$ is transformed into the point:

Taylor series:

$$\begin{cases} f(\xi + d\xi, \eta) = \overbrace{f(\xi,\eta)}^{x} + f_\xi(\xi,\eta)\, d\xi \\ g(\xi + d\xi, \eta) = \underbrace{g(\xi,\eta)}_{y} + g_\xi(\xi,\eta)\, d\xi \end{cases}$$

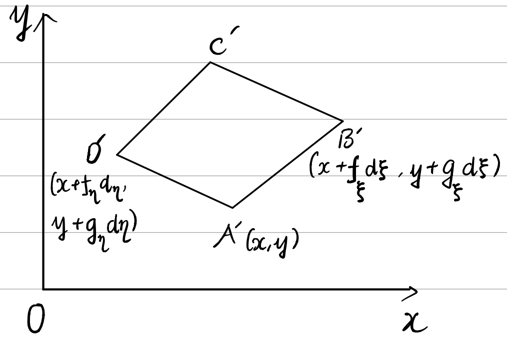
3. The point $D(\xi, \eta + d\eta)$ is transformed into:

$$\begin{cases} f(\xi, \eta + d\eta) = x + \overbrace{f_\eta}^{\frac{\partial f}{\partial \eta}}(\xi,\eta)\, d\eta \\ g(\xi, \eta + d\eta) = y + g_\eta(\xi,\eta)\, d\eta \end{cases}$$

$$\overrightarrow{A'B'} = (f_\xi\, d\xi, \; g_\xi\, d\xi) \quad ; \quad \overrightarrow{A'D'} = (f_\eta\, d\eta, \; g_\eta\, d\eta)$$

- The area of R in the x,y plane is $\overrightarrow{A'B'} \times \overrightarrow{A'D'}$

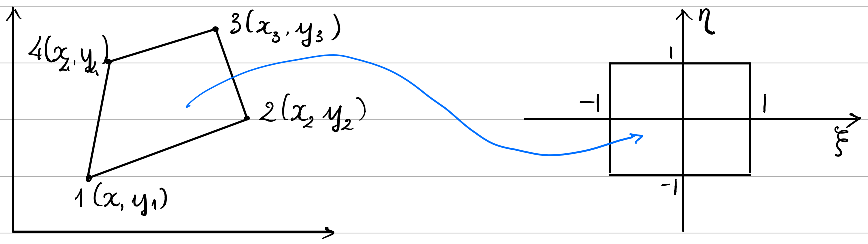
$$\text{Area of } R(x,y) = |\underbrace{f_\xi g_\eta - f_\eta g_\xi}_{\text{Jacobian}}|\, d\xi\, d\eta$$



The quantity dudv is the area of the box $R(\xi,\eta)$

$\Rightarrow$ Area of $R(x,y) = J \, .$ Area of $R(\xi,\eta)$

Topic: *Gauss Integration - 2D - Arbitrary domain* **Notebook**



The relationship between convex quadrilateral in the physical coordinate Oxy and the standard domain $[-1,1]\times[-1,1]$ in the natural coordinate $(O\xi\eta)$ is given by:

$$x = N_1(\xi,\eta)x_1 + N_2(\xi,\eta)x_2 + N_3(\xi,\eta)x_3 + N_4(\xi,\eta)x_4$$
$$y = N_1(\xi,\eta)y_1 + N_2(\xi,\eta)y_2 + N_3(\xi,\eta)y_3 + N_4(\xi,\eta)y_4$$

in which (x_i, y_i), $i = 1,2,3,4$ are the coordinates of 4 nodes in physical coordinate system Oxy.

- N_i, $i = 1, 2, 3, 4$ are shape functions of the quadrilateral in the physical coordinate Oxy:

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta) \quad ; \quad N_3 = \frac{1}{4}(1+\xi)(1+\eta)$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta) \quad ; \quad N_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

- Now: $I = \iint_{\Omega_{xy}} f(x,y)\,dx\,dy = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)\det J\,d\xi\,d\eta$

- In which $\det J$ is the determinant of Jacobian matrix, which relate the convex quadrilateral Ω_{xy} in the physical coordinate system Oxy with the standard domain $[-1,1]\times[-1,1]$ in the natural coordinate system $O\xi\eta$:

$$J = \left[\frac{\partial(x,y)}{\partial(\xi,\eta)}\right] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}$$

Topic:

Notebook

Notice that, the code here is based on Julia 1.0.0

In [1]:

```
using Pkg;
Pkg.add("SymPy");
using SymPy;
using LinearAlgebra;
```

```
  Updating registry at `/home/jrun/.julia/registries/JuliaPro`
  Updating git-repo `https://pkg.juliacomputing.com/registry/JuliaPro`
 Resolving package versions...
  Updating `~/.julia/Project.toml`
 [no changes]
  Updating `~/.julia/Manifest.toml`
 [no changes]
```

In [2]:

```
#-------------------------------------------------------------------------------
function gauss_integration(nGauss, dim)
#-------------------------------------------------------------------------------
# PURPOSE:
#     Determine Gauss point's coordinate and the corresponding Gauss weight
# SYNTAX:
#     gauss_integration(nGauss, rGauss, dim)
# INPUT:
#     nGauss: the number of Gauss point
#     dim   : dimension of the problem (dim = 1 or dim = 2 or dim = 3)
# OUPUT:
#      gausspoint_coordinate: The Gauss point's coordinate
#      gausspoint_weight: The Gauss point's weight
#-------------------------------------------------------------------------------

# Initiate gausspoint_coordinate and gausspoint_weight
   gausspoint_coordinate = zeros(nGauss^dim, dim)
   gausspoint_weight = Float64[];

#******** the integration domain is [-1 1] for all of direction:***************

    #------------- Limit the number of Gauss point up to 5 --------------------
    if (nGauss > 5)
        println("The number of Gauss point shouldn't be more than 5")
    end

    #------------- The number of Gauss point in one direction -----------------
    if nGauss == 1
        point = 0.0
        weight = 2.0
        j
    elseif nGauss == 2
        point = [-0.577350269189626
                  0.577350269189626]
        weight = [1.0 1.0]

    elseif nGauss == 3
        point = [ 0
                 -0.774596669241483
                  0.774596669241483];

        weight = [8/9
                  5/9
                  5/9];

    elseif nGauss == 4
        point = [-0.3399810435848563
                  0.3399810435848563
                 -0.8611363115940526
                  0.8611363115940526];

        weight = [0.6521451548625461
                 0.6521451548625461
                 0.3478548451374538
                 0.3478548451374538];
    elseif nGauss == 5
        point = [ 0
```

```
                -0.5384693101056831
                 0.5384693101056831
                -0.9061798459386640
                 0.9061798459386640];

        weight = [0.5688888888888889
                 0.4786286704993665
                 0.4786286704993665
                 0.2369268850561891
                 0.2369268850561891];

    end
    #-----------------------------DIMENSION -----------------------------------
    # One dimension problem
    if dim == 1
        for i = 1:nGauss
            gausspoint_coordinate[i,:] = [point[i]]
            push!(gausspoint_weight, weight[i])
        end
        return gausspoint_coordinate, gausspoint_weight

    # Two dimension problem
    elseif dim == 2
        n = 0
        for i = 1:nGauss
            for j = 1:nGauss
                n = n + 1
                gausspoint_coordinate[n,:] = [point[i] point[j]]
                push!(gausspoint_weight, weight[i] * weight[j])
            end
        end
        return gausspoint_coordinate, gausspoint_weight

    # Three dimension problem
    elseif dim == 3
        n = 0
        for i = 1:nGauss
            for j =  1:nGauss
                for k = 1:nGauss
                    n = n + 1
                    gausspoint_coordinate[n,:] = [point[i] point[j] point[k]]
                    push!(gausspoint_weight, weight[i] * weight[j] * weight[k])
                end
            end
        end
        return gausspoint_coordinate, gausspoint_weight
    end
end
```

Out[2]:

```
gauss_integration (generic function with 1 method)
```

In [3]:

```
#=
Example1: Calculate the integration of
f(x) =  0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5
here x = [-1, 1]
=#


nGauss =  3
dim = 1

gauss_point, gauss_weight = gauss_integration(nGauss, dim)


f(x) =  0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5

I = 0

# loop over all of gauss points
for i = 1:length(gauss_point)
    I = I + f(gauss_point[i]) * gauss_weight[i]
end

@show I


# Analytical solution
x = Sym("x")
I_analytical = integrate( 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5, (x, -1, 1))
@show I_analytical
```

```
I = -492.93333333333237
I_analytical = -492.933333333333
```

Out[3]:

-492.933333333333

In [4]:

```
#=
Example 2: Calculate the integration of
f(x) =  0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5
where x = [0, 2]
=#

nGauss =  3
dim = 1

# function f
f1(x) =  0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5

a = 0 # lower bound of the limit
b = 2 # upper bound of the limit
J = (b-a)/2 # Jacobian value

I = 0
gauss_point, gauss_weight = gauss_integration(nGauss, dim)

for i = 1:length(gauss_point) # loop over Gauss points
    x = (a+b)/2 + (b-a)/2* gauss_point[i]
    I = I + J * f1(x) * gauss_weight[i]
end
@show I


# ANALYTICAL SOLUTION
x = Sym("x")
I_analytical = integrate(0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5,(x,0,2))
@show I_analytical
```

```
I = 723.7333333333321
I_analytical = 723.733333333334
```

Out[4]:

723.733333333334

In [9]:

```
#=
Example 2: Calculate the integration of
f(x) =  0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5
where x = [0, 2]
=#


# function f
f1(x) =  0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5

a = 0 # lower bound of the limit
b = 2 # upper bound of the limit
J = (b-a)/2 # Jacobian value

println("The number of Gauss Points is equal to the necessary one")
# CHECK WITH THE LOWER NUMBER OF GAUSS POINT
I = 0
nGauss =  3
dim = 1
gauss_point, gauss_weight = gauss_integration(nGauss, dim)

for i = 1:length(gauss_point) # loop over Gauss points
    x = (a+b)/2 + (b-a)/2* gauss_point[i]
    I = I + J * f1(x) * gauss_weight[i]
end
@show I

println("The number of Gauss Points is higher than the necessary one")
# CHECK WITH THE HIGHER NUMBER OF GAUSS POINT
I = 0
nGauss =  4
dim = 1
gauss_point, gauss_weight = gauss_integration(nGauss, dim)

for i = 1:length(gauss_point) # loop over Gauss points
    x = (a+b)/2 + (b-a)/2* gauss_point[i]
    I = I + J * f1(x) * gauss_weight[i]
end
@show I

println("The number of Gauss Points is less than the necessary one")
# CHECK WITH THE LOWER NUMBER OF GAUSS POINT
I = 0
nGauss =  2
dim = 1
gauss_point, gauss_weight = gauss_integration(nGauss, dim)

for i = 1:length(gauss_point) # loop over Gauss points
    x = (a+b)/2 + (b-a)/2* gauss_point[i]
    I = I + J * f1(x) * gauss_weight[i]
end
@show I
```

```
The number of Gauss Points is equal to the necessary one
I = 723.7333333333321
The number of Gauss Points is higher than the necessary one
```

```
I = 723.7333333333331
The number of Gauss Points is less than the necessary one
I = 528.1777777777781
```

Out[9]:

```
528.1777777777781
```

In [6]:

```
#=
Example 3: Calculate the integration of
f(x, y) =  0.2 + 25x - 200y^2 + 675x^3 - 900y^4 + 400x^5
where x = [-1, 1], y =[-1, 1]
=#

nGauss = 5
dim = 2

# function f
f4(x,y) =  0.2 + 25x - 200y^2 + 657x^3 - 900y^4 + 400x^5

I = 0
gauss_point, gauss_weight = gauss_integration(nGauss, dim)

for i = 1:length(gauss_weight) # loop over Gauss points
    I = I +  f4(gauss_point[i,1], gauss_point[i,2]) * gauss_weight[i]
end

@show I

#Analytical

x = Sym("x")
y = Sym("y")
I_analytical = integrate(0.2 + 25x - 200y^2 + 657x^3 - 900y^4 + 400x^5, (x,-1,1), (y
@show I_analytical
```

```
I = -985.8666666666667
I_analytical = -985.866666666667
```

Out[6]:

-985.866666666667

In [7]:

```
#=
Example 4: Reference: https://ctec.tvu.edu.vn/ttkhai/TCC/21_Tich_phan_hai_lop.htm
Calculate integration of F = ∬dxdy, with the domain is limited by:
   x + y = 1
   x + y = 2
   2x - y = 1
   2x - y = 3
=#

ξ = Sym("xi")
η = Sym("eta")

N1 = 1/4*(1-ξ)*(1-η)
N2 = 1/4*(1+ξ)*(1-η)
N3 = 1/4*(1+ξ)*(1+η)
N4 = 1/4*(1-ξ)*(1+η)

X =[4/3, 5/3, 1, 2/3]
Y =[-1/3, 1/3, 1, 1/3]

x = [N1 N2 N3 N4]*X
y = [N1 N2 N3 N4]*Y

J =[diff(x,ξ) diff(y,ξ);diff(x,η) diff(y,η)]

detJ = det(J)

gauss_point, gauss_weight = gauss_integration(3,2)


I = 0

for i = 1:length(gauss_weight)
    I += 1 * detJ(gauss_point[i,1], gauss_point[i,2]) * gauss_weight[i]
end
@show I

# ANALYTICAL SOLUTION
@show I_analytical = integrate(detJ,(ξ,-1,1),(η,-1,1))
```

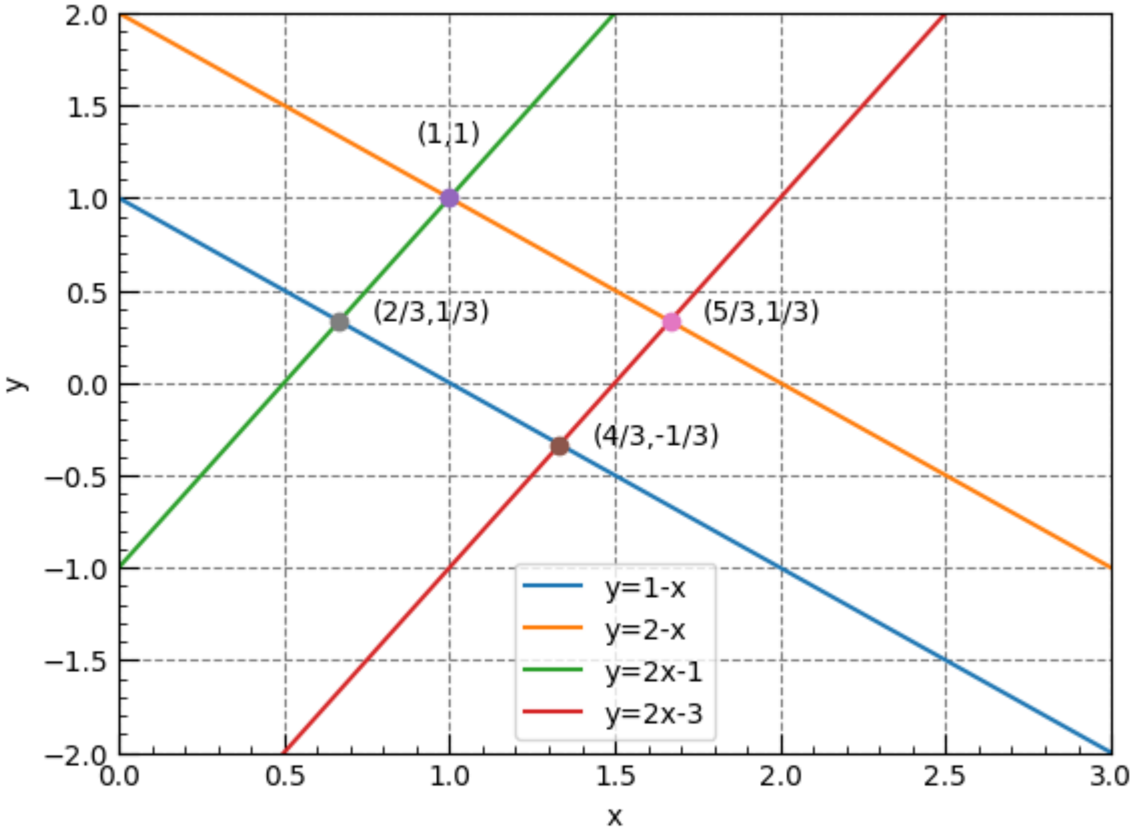
```
I = 0.6666666666666667
I_analytical = integrate(detJ, (ξ, -1, 1), (η, -1, 1)) = 0.666666666666
667
```

Out[7]:

0.6666666666666667

In [8]:

```
# Plot the domain of integartion in exercise 4
using PyPlot
@show x = collect(range(0, stop = 3, length = 4))
@show y1 = [1-i for i in x]
y2 = [2-i for i in x]
y3 = [2i - 1 for i in x]
y4 = [2i - 3 for i in x]
#using PyPlot
plot(x,y1 , label = "y=1-x")
plot(x,y2 , label = "y=2-x")
plot(x,y3 , label = "y=2x-1")
plot(x,y4 , label = "y=2x-3")

plot([1],[1],"o")
text(0.9,1.3, "(1,1)")

plot([4/3],[-1/3],"o")
text(4/3+0.1,-1/3,"(4/3,-1/3)")

plot([5/3],[1/3],"o")
text(5/3+0.1,1/3,"(5/3,1/3)")

plot([2/3],[1/3],"o")
text(2/3+0.1,1/3,"(2/3,1/3)")
xlabel("x")
ylabel("y")


tick_params(which = "both", direction = "in", color = "black")
tick_params(which="major", length=7)
tick_params(which="minor", length=3)
grid(linestyle = "--", linewidth = 0.8, color = "grey")
grid("on")
minorticks_on()
xlim(0,3)
ylim(-2,2)
legend()
```




```
x = collect(range(0, stop=3, length=4)) = [0.0, 1.0, 2.0, 3.0]
y1 = [1 - i for i = x] = [1.0, 0.0, -1.0, -2.0]
```

```
/usr/local/lib/python2.7/dist-packages/matplotlib/cbook/deprecation.p
y:107: MatplotlibDeprecationWarning: Passing one of 'on', 'true', 'of
f', 'false' as a boolean is deprecated; use an actual boolean (True/Fa
lse) instead.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

Out[8]:

```
PyObject <matplotlib.legend.Legend object at 0x7f282b4d1490>
```

In [ ]: