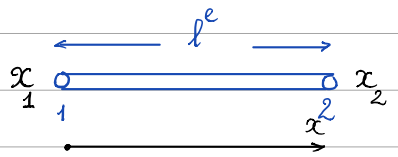
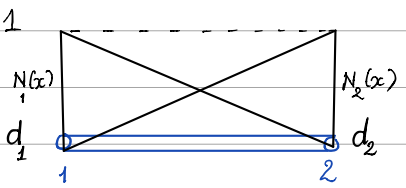
1 Two node linear element:





- To achieve continuity, we express the approximation in the element in terms of nodal values.
- To satisfy completenuss condition, we need to choose at lease a linear polynominal

$$u^e(x) = a_0^e + a_1^e(x)$$

$$\Leftrightarrow \quad u^e(x) = \underbrace{[1 \;\; x]}_{p(x)} \underbrace{\begin{bmatrix} a_0^e \\ a_1^e \end{bmatrix}}_{a^e} = p(x)\,a^e \qquad (1)$$

- Now we express the coefficients of a_0^e, a_1^e in term of values of the approximation at nodes 1 and 2:

$$\begin{cases} u^e(x_1) \equiv u_1^e = a_0^e + a_1^e x_1 \\ u^e(x_2) \equiv u_2^e = a_0^e + a_1^e x_2 \end{cases}$$

$$\Leftrightarrow \underbrace{\begin{bmatrix} u_1^e \\ u_2^e \end{bmatrix}}_{d^e} = \underbrace{\begin{bmatrix} 1 & x_1^e \\ 1 & x_2^e \end{bmatrix}}_{M^e} \underbrace{\begin{bmatrix} a_0^e \\ a_2^e \end{bmatrix}}_{a^e} \Rightarrow a^e = (M^e)^{-1} d^e \qquad (2)$$

(2) → (1)

$$u^e(x) = \underbrace{p(x)(M^e)^{-1}}\, d^e$$

$$g^e = [N_1^e(x) \;\; N_2^e(x)]$$: Element shape function matrix

$$\begin{cases} u_1^e(x_1) = N_1^e(x_1)\,d_1 + N_2^e(x_1)\,d_2 \\ u_2^e(x_2) = N_2^e(x_1)\,d_1 + N_2^e(x_2)\,d_2 \end{cases}$$

$$(M^e)^{-1} = \frac{1}{x_2^e - x_1^e}\begin{bmatrix} x_2^e & -x_1^e \\ -1 & 1 \end{bmatrix} = \frac{1}{l^e}\begin{bmatrix} x_2^e & -x_1^e \\ -1 & 1 \end{bmatrix}$$

$$\cdot\; N^e = [N_1^e \;\; N_2^e] = p(x)(M^e)^{-1} = [1 \;\; x]\begin{bmatrix} x_2^e & -x_1^e \\ -1 & 1 \end{bmatrix}$$

$$= \frac{1}{l^e}\begin{bmatrix} x_2^e - x & -x_1^e + x \end{bmatrix} = \frac{1}{l^e}\begin{bmatrix} l^e - x & x \end{bmatrix}$$

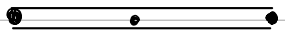
$$\Rightarrow N_1^e = (l^e - x)/l^e \quad ; \quad N_2^e = x/l^e$$

NOTE:

① $N_I(x_J^e) = \delta_{IJ} = \begin{cases} 1 & \text{if } I = J \\ 0 & \text{if } I \neq J \end{cases}$

② At $x = x_K \Rightarrow u^e(x_K^e) = N_1^e(x_K^e) + N_2^e(x_K^e) = 1$

2 Quadratic 1D element



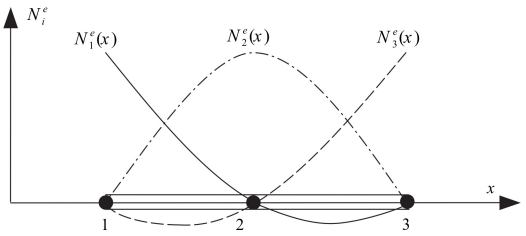


Figure 4.5 The quadratic shape functions for a three-node element.

- We use a complete second-order polynominal approximation

$$u^e(x) = a_0^e + a_1^e x + a_2^e x^2 = \underbrace{[1 \; x \; x^2]}_{P(x)} \underbrace{\begin{bmatrix} a_0^e \\ a_1^e \\ a_2^e \end{bmatrix}}_{a^e} \quad (1)$$

- 2 nodes are placed at the end of the element so that the global approximation will be continuous. The 3rd node is placed at center → symmetrically pleasing.
- We express (a_0^e, a_1^e, a_2^e) in terms of nodal values of:

$$\begin{aligned} u_1^e &= a_0^e + a_1^e x_1^e + a_2^e x_1^e \\ u_2^e &= a_0^e + a_1^e x_2^e + a_2^e x_2^e \\ u_3^e &= a_0^e + a_1^e x_3^e + a_2^e x_3^e \end{aligned} \Rightarrow \underbrace{\begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix}}_{d^e} = \underbrace{\begin{bmatrix} 1 & x_1^e & x_1^{e^2} \\ 1 & x_2^e & x_2^{e^2} \\ 1 & x_3^e & x_3^{e^2} \end{bmatrix}}_{M^e} \underbrace{\begin{bmatrix} a_0^e \\ a_e^1 \\ a_e^2 \end{bmatrix}}_{a^e}$$

$$\Rightarrow a^e = (M^e)^{-1} d^e \quad (2)$$

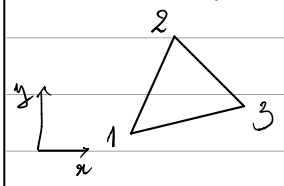
- (1) (2): $u^e(x) = \underbrace{P(x)(M^e)^{-1}}_{g^e = [N_1^e(x) \; N_2^e(x) \; N_3^e(x)]} d^e$

$$N_1 = \frac{2}{l_e^2}(x - x_2^e)(x - x_3^e)$$

$$N_2 = -\frac{2}{l_e^2}(x - x_1^e)(x - x_3^e)$$

$$N_3 = \frac{2}{l_e^2}(x - x_1^e)(x - x_2^e)$$

3) Triangle elements :



$$u^e(x,y) = a_0^e + a_1^e x + a_2^e y$$

$$= \underbrace{[1 \;\; x \;\; y]}_{P(x,y)} \underbrace{\begin{bmatrix} a_0^e \\ a_1^e \\ a_2^e \end{bmatrix}}_{a^e} \qquad (1)$$

$$\begin{aligned} u_1^e &= a_0^e + a_1^e x_1^e + a_2^e y_1^e \\ u_1^e &= a_0^e + a_1^e x_1^e + a_2^e y_1^e \\ u_1^e &= a_0^e + a_1^e x_1^e + a_2^e y_1^e \end{aligned} \Rightarrow \underbrace{\begin{bmatrix} u_1^e \\ u_2^e \\ u_3^e \end{bmatrix}}_{d^e} = \underbrace{\begin{bmatrix} 1 & x_1^e & y_1^e \\ 2 & x_2^e & y_2^e \\ 1 & x_3^e & y_3^e \end{bmatrix}}_{M^e} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}}_{a^e}$$

$$\Rightarrow a^e = (M^e)^{-1} d^e \qquad (2)$$

(2)(1): $u^e = \underbrace{P (M^e)^{-1}}_{g^e} d^e$

↳ $[N_1^e(x,y) \;\; N_2^e(x,y) \;\; N_3^e(x,y)]$

$$(M^e)^{-1} = \frac{1}{2A^e} \begin{bmatrix} y_2^e - y_3^e & y_3^e - y_1^e & y_1^e - y_2^e \\ x_3^e - x_2^e & x_1^2 - x_3^e & x_2^e - x_1^e \\ x_2^e y_3^e - x_3^e y_2^e & x_3^e y_1^e - x_1^e y_3^e & x_1^e y_2^e - x_2^e y_1^e \end{bmatrix}$$

A^e: area of the element

$$2A^e = \det(M^e) = (x_2^e y_3^e - x_3^e y_2^e) - (x_1^e y_3^e - x_3^e y_1^e) + (x_1^e y_2^e - x_2^e y_1^e)$$

$$\Rightarrow N_1^e = \frac{1}{2A^e} \left\{ x_2^e y_3^e - x_3^e y_2^e + (y_2^e - y_3^e) x + (x_3^e - x_2^e) y \right\}$$

$$N_2^e = \frac{1}{2A^e} \left\{ x_3^e y_1^e - x_1^e y_3^e + (y_3^e - y_1^e) x + (x_1^e - x_3^e) y \right\}$$

$$N_3 = \frac{1}{2A^e} \left\{ x_1^e y_2^e - x_2^e y_1^e + (y_1^e - y_2^e) x + (x_2^e - x_1^e) y \right\}$$

ShapeFunction

In [1]:
```
using JFEM: ShapeFunction, Quad4

shape = ShapeFunction(Quad4)

# define local coordinate
ξ = [0.0,0.0]

@show N = shape(ξ)
@show ∂N∂ξ = shape(Val{:grad}, ξ)

# define geometry x of element
x = [[0.0,0.0], [1.0,0.0], [1.0,1.0], [0.0,1.0]]

@show ∂N∂x = shape(Val{:grad}, ξ, x)
@show J = shape(Val{:jacobian}, ξ, x) # Jacobian matrix ∂x/∂ξ
@show detJ = shape(Val{:detj}, ξ, x)  # determinant of Jacobian matrix
```

```
N = shape(ξ) = [0.25 0.25 0.25 0.25]
∂N∂ξ = shape(Val{:grad},ξ) = [-0.25 0.25 0.25 -0.25; -0.25 -0.25 0.25 0.25]
∂N∂x = shape(Val{:grad},ξ,x) = [-0.5 0.5 0.5 -0.5; -0.5 -0.5 0.5 0.5]
J = shape(Val{:jacobian},ξ,x) = [0.5 0.0; 0.0 0.5]
detJ = shape(Val{:detj},ξ,x) = 0.25
```

Out[1]: 0.25

```
typealias Quad4 Element{2,4}


function ShapeFunction(::Type{Quad4})
    function N(point)
        A = Matrix{Float64}(1,4)
        ξ, η = point
        A[1] = 0.25 * (1-ξ) * (1-η)
        A[2] = 0.25 * (1+ξ) * (1-η)
        A[3] = 0.25 * (1+ξ) * (1+η)
        A[4] = 0.25 * (1-ξ) * (1+η)
        return A
    end
    function ∇N(point)
        A = Matrix{Float64}(2,4)
        ξ, η = point
        A[1] = 0.25 * (-1+η)
        A[2] = 0.25 * (-1+ξ)
        A[3] = 0.25 * ( 1-η)
        A[4] = 0.25 * (-1-ξ)
        A[5] = 0.25 * ( 1+η)
        A[6] = 0.25 * ( 1+ξ)
        A[7] = 0.25 * (-1-η)
        A[8] = 0.25 * ( 1-ξ)
        return A
    end
    return ShapeFunction(N,∇N)
end


function create_gauss_points(::Type{Quad4})
    gauss_line = GaussLine(2,2)
    gauss_points = [GaussPoint(i...) for i in gauss_line]
    return gauss_points
end
```

```julia
# Created by K.Nakamura on 2015-11-21.
# e-mail: nakamura-keita-kn@ynu.jp


immutable ShapeFunction
    N::Function
    ∇N::Function
end

function ShapeFunction(local_coord::Vector, P::Function, ∇P::Function)
    coef = vcat([P(x...) for x=local_coord]...) |> inv
    N(point) = P(point...) * coef
    ∇N(point) = ∇P(point...) * coef
    return ShapeFunction(N, ∇N)
end

function (shape::ShapeFunction)(point)
    return shape.N(point)
end

function (shape::ShapeFunction)(point, dim::Int64)
    base = shape(point)
    N = zeros(dim, dim*length(base))
    for i = 1:dim
        N[i,i:dim:end] = base
    end
    return N
end

function (shape::ShapeFunction)(::Type{Val{:grad}}, point)
    return shape.∇N(point)
end

function (shape::ShapeFunction)(::Type{Val{:jacobian}}, point, geometry::Vector)
    ∇N = shape(Val{:grad}, point)
    return ∇N * hcat(geometry...)'
end

function (shape::ShapeFunction)(::Type{Val{:detj}}, point, geometry::Vector)
    J = shape(Val{:jacobian}, point, geometry)
    n, m = size(J)
    if n == m  # volume element
        return det(J)
    end
    # else: refer to Nakamura-san code
end

function (shape::ShapeFunction)(::Type{Val{:grad}}, point, geometry::Vector)
    ∇N = shape(Val{:grad}, point)
    J = shape(Val{:jacobian}, point, geometry)
    return inv(J) * ∇N
end
```