function shape\_function = lagrange\_interpolation(xe, index, x)

%

% lagrange interpolation.

%

% @since 2.0.1

% @param {array} [xe] ­n¹ï­þ¨ÇÂI°µ lagrange.

% @param {sym} [x] ÅÜ¼Æ¦WºÙ.

% @return {symfun} [shape\_function] shape function of x.

%

% shape\_function = sym(zeros(size(xe)));

% for index = 1 : length(shape\_function)

xi = xe(index);

% ¦^¶Ç¤£µ¥©ó xi ªº

xj = xe(xe ~= xi);

% ³s­¼

% ¦^¶Ç shape function

shape\_function = prod((x - xj) ./ (xi - xj)); % broadcasting

% end

End

function [] = plot\_grid(nodes, shape\_function\_xi, shape\_function\_eta)

if nargin == 1

shape\_function\_xi = @(x) 0;

shape\_function\_eta = @(x) 0;

plot\_format = 'k--';

else

plot\_format = 'k';

end

natural = -1 : 0.001 : 1;

nodes\_length = length(nodes);

natural\_length = length(natural);

x = zeros(natural\_length, nodes\_length);

y = zeros(natural\_length, nodes\_length);

z = zeros(natural\_length, nodes\_length);

for node = 1 : nodes\_length

for index = 1 : natural\_length

x(index, node) = nodes(node);

y(index, node) = natural(index);

z(index, node) = shape\_function\_xi(x(index, node)) \* shape\_function\_eta(y(index, node));

end

end

plot3(x, y, z, plot\_format);

hold on;

for node = 1 : nodes\_length

for index = 1 : natural\_length

x(index, node) = natural(index);

y(index, node) = nodes(node);

z(index, node) = shape\_function\_xi(x(index, node)) \* shape\_function\_eta(y(index, node));

end

end

plot3(x, y, z, plot\_format);

hold on;

end

function [] = plot\_shape(shape\_function\_xi, shape\_function\_eta)

natural = -1 : 0.001 : 1;

natural\_length = length(natural);

x = zeros(natural\_length, natural\_length);

y = zeros(natural\_length, natural\_length);

z = zeros(natural\_length, natural\_length);

for node = 1 : natural\_length

for index = 1 : natural\_length

x(index, node) = natural(node);

y(index, node) = natural(index);

z(index, node) = shape\_function\_xi(x(index, node)) \* shape\_function\_eta(y(index, node));

end

end

mesh(x, y, z);

hold on;

for node = 1 : natural\_length

for index = 1 : natural\_length

x(index, node) = natural(index);

y(index, node) = natural(node);

z(index, node) = shape\_function\_xi(x(index, node)) \* shape\_function\_eta(y(index, node));

end

end

mesh(x, y, z);

hold on;

end

function f = create\_shape\_function(num\_node\_per\_element, schemes)

%

% shape function and derivatives.

%

% @since 1.1.0

% @param {number} [num\_node\_per\_element] ¤@­Ó element ¦³´X­Ó nodes.

% @param {string} [schemes] corner first or along boundary.

% @return {function} [shape\_function] for Q4 Q8 Q9.

%

% @param {sym} [eta] ¤@­Ó element ¦³´X­Ó nodes.

% @param {sym} [xi] natural coordinates (-1 ... +1).

% @param {array} [shape] Shape functions.

% @return {array} [diff\_shape] derivatives w.r.t. xi and eta.

%

switch num\_node\_per\_element

case 4

f = @shape\_function\_Q4;

case 8

if nargin == 1 || strcmp(schemes, 'corner first')

f = @shape\_function\_Q8;

else

f = @shape\_function\_Q8\_along\_boundary;

end

case 9

if nargin == 1 || strcmp(schemes, 'corner first')

f = @shape\_function\_Q9;

else

f = @shape\_function\_Q9\_along\_boundary;

end

end

function [shape, diff\_shape] = shape\_function\_Q4(xi, eta)

shape = 1 / 4 \* [ (1 - xi) \* (1 - eta), (1 + xi) \* (1 - eta), (1 + xi) \* (1 + eta), (1 - xi) \* (1 + eta)];

diff\_shape = 1 / 4 \* [

- (1 - eta), 1 - eta, 1 + eta, - (1 + eta);

- (1 - xi), - (1 + xi), 1 + xi, 1 - xi;

];

end

function [shape, diff\_shape] = shape\_function\_Q8(xi, eta)

shape = 1 / 4 \* [

(1-xi)\*(1-eta)\*(-xi-eta-1), ...

(1+xi)\*(1-eta)\*(xi-eta-1), ...

(1+xi)\*(1+eta)\*(xi+eta-1), ...

(1-xi)\*(1+eta)\*(-xi+eta-1), ...

2\*(1-eta)\*(1-xi^2), ...

2\*(1+xi)\*(1-eta^2), ...

2\*(1+eta)\*(1-xi^2), ...

2\*(1-xi)\*(1-eta^2)

];

diff\_shape = [

- ((eta - 1)\*(eta + xi + 1))/4 - ((eta - 1)\*(xi - 1))/4, ...

((eta - 1)\*(eta - xi + 1))/4 - ((eta - 1)\*(xi + 1))/4, ...

((eta + 1)\*(eta + xi - 1))/4 + ((eta + 1)\*(xi + 1))/4, ...

((eta + 1)\*(xi - eta + 1))/4 + ((eta + 1)\*(xi - 1))/4, ....

(xi\*(2\*eta - 2))/2, ...

1/2 - eta^2/2, ...

-(xi\*(2\*eta + 2))/2, ...

eta^2/2 - 1/2;

- ((xi - 1)\*(eta + xi + 1))/4 - ((eta - 1)\*(xi - 1))/4, ...

((xi + 1)\*(eta - xi + 1))/4 + ((eta - 1)\*(xi + 1))/4, ...

((xi + 1)\*(eta + xi - 1))/4 + ((eta + 1)\*(xi + 1))/4, ...

((xi - 1)\*(xi - eta + 1))/4 - ((eta + 1)\*(xi - 1))/4, ...

xi^2/2 - 1/2, ....

-(eta\*(2\*xi + 2))/2, ...

1/2 - xi^2/2, ...

(eta\*(2\*xi - 2))/2

];

end

function [shape, diff\_shape] = shape\_function\_Q9(xi, eta)

shape = [

(eta\*xi\*(eta - 1)\*(xi - 1))/4, ...

(eta\*xi\*(xi/2 + 1/2)\*(eta - 1))/2, ...

(eta\*xi\*(xi/2 + 1/2)\*(eta + 1))/2, ...

(eta\*xi\*(eta + 1)\*(xi - 1))/4, ...

-(eta\*(eta - 1)\*(xi - 1)\*(xi + 1))/2, ...

-xi\*(xi/2 + 1/2)\*(eta - 1)\*(eta + 1), ...

-(eta\*(eta + 1)\*(xi - 1)\*(xi + 1))/2, ...

-(xi\*(eta - 1)\*(eta + 1)\*(xi - 1))/2, ...

(eta - 1)\*(eta + 1)\*(xi - 1)\*(xi + 1)

];

diff\_shape = [

(eta\*xi\*(eta - 1))/4 + (eta\*(eta - 1)\*(xi - 1))/4,...

(eta\*xi\*(eta - 1))/4 + (eta\*(xi/2 + 1/2)\*(eta - 1))/2,...

(eta\*xi\*(eta + 1))/4 + (eta\*(xi/2 + 1/2)\*(eta + 1))/2,...

(eta\*xi\*(eta + 1))/4 + (eta\*(eta + 1)\*(xi - 1))/4,...

-(eta\*(eta - 1)\*(xi - 1))/2 - (eta\*(eta - 1)\*(xi + 1))/2,...

-(xi/2 + 1/2)\*(eta - 1)\*(eta + 1) - (xi\*(eta - 1)\*(eta + 1))/2,...

-(eta\*(eta + 1)\*(xi - 1))/2 - (eta\*(eta + 1)\*(xi + 1))/2,...

-(xi\*(eta - 1)\*(eta + 1))/2 - ((eta - 1)\*(eta + 1)\*(xi - 1))/2,...

(eta - 1)\*(eta + 1)\*(xi - 1) + (eta - 1)\*(eta + 1)\*(xi + 1);

(eta\*xi\*(xi - 1))/4 + (xi\*(eta - 1)\*(xi - 1))/4, ...

(eta\*xi\*(xi/2 + 1/2))/2 + (xi\*(xi/2 + 1/2)\*(eta - 1))/2, ...

(eta\*xi\*(xi/2 + 1/2))/2 + (xi\*(xi/2 + 1/2)\*(eta + 1))/2, ...

(eta\*xi\*(xi - 1))/4 + (xi\*(eta + 1)\*(xi - 1))/4, ...

-(eta\*(xi - 1)\*(xi + 1))/2 - ((eta - 1)\*(xi - 1)\*(xi + 1))/2, ...

-xi\*(xi/2 + 1/2)\*(eta - 1) - xi\*(xi/2 + 1/2)\*(eta + 1), ...

-(eta\*(xi - 1)\*(xi + 1))/2 - ((eta + 1)\*(xi - 1)\*(xi + 1))/2, ...

-(xi\*(eta - 1)\*(xi - 1))/2 - (xi\*(eta + 1)\*(xi - 1))/2, ...

(eta - 1)\*(xi - 1)\*(xi + 1) + (eta + 1)\*(xi - 1)\*(xi + 1)

];

end

function [shape, diff\_shape] = shape\_function\_Q8\_along\_boundary(xi, eta)

shape = 1 / 4 \* [

(1-xi)\*(1-eta)\*(-xi-eta-1), ...

2\*(1-eta)\*(1-xi^2), ...

(1+xi)\*(1-eta)\*(xi-eta-1), ...

2\*(1+xi)\*(1-eta^2), ...

(1+xi)\*(1+eta)\*(xi+eta-1), ...

2\*(1+eta)\*(1-xi^2), ...

(1-xi)\*(1+eta)\*(-xi+eta-1), ...

2\*(1-xi)\*(1-eta^2)

];

diff\_shape = [

- ((eta - 1)\*(eta + xi + 1))/4 - ((eta - 1)\*(xi - 1))/4, ...

(xi\*(2\*eta - 2))/2, ...

((eta - 1)\*(eta - xi + 1))/4 - ((eta - 1)\*(xi + 1))/4, ...

1/2 - eta^2/2, ...

((eta + 1)\*(eta + xi - 1))/4 + ((eta + 1)\*(xi + 1))/4, ...

-(xi\*(2\*eta + 2))/2, ...

((eta + 1)\*(xi - eta + 1))/4 + ((eta + 1)\*(xi - 1))/4, ....

eta^2/2 - 1/2;

- ((xi - 1)\*(eta + xi + 1))/4 - ((eta - 1)\*(xi - 1))/4, ...

xi^2/2 - 1/2, ....

((xi + 1)\*(eta - xi + 1))/4 + ((eta - 1)\*(xi + 1))/4, ...

-(eta\*(2\*xi + 2))/2, ...

((xi + 1)\*(eta + xi - 1))/4 + ((eta + 1)\*(xi + 1))/4, ...

1/2 - xi^2/2, ...

((xi - 1)\*(xi - eta + 1))/4 - ((eta + 1)\*(xi - 1))/4, ...

(eta\*(2\*xi - 2))/2

];

end

function [shape, diff\_shape] = shape\_function\_Q9\_along\_boundary(xi, eta)

shape = [

(eta\*xi\*(eta - 1)\*(xi - 1))/4, ...

-(eta\*(eta - 1)\*(xi - 1)\*(xi + 1))/2, ...

(eta\*xi\*(xi/2 + 1/2)\*(eta - 1))/2, ...

-xi\*(xi/2 + 1/2)\*(eta - 1)\*(eta + 1), ...

(eta\*xi\*(xi/2 + 1/2)\*(eta + 1))/2, ...

-(eta\*(eta + 1)\*(xi - 1)\*(xi + 1))/2, ...

(eta\*xi\*(eta + 1)\*(xi - 1))/4, ...

-(xi\*(eta - 1)\*(eta + 1)\*(xi - 1))/2, ...

(eta - 1)\*(eta + 1)\*(xi - 1)\*(xi + 1)

];

diff\_shape = [

(eta\*xi\*(eta - 1))/4 + (eta\*(eta - 1)\*(xi - 1))/4,...

-(eta\*(eta - 1)\*(xi - 1))/2 - (eta\*(eta - 1)\*(xi + 1))/2,...

(eta\*xi\*(eta - 1))/4 + (eta\*(xi/2 + 1/2)\*(eta - 1))/2,...

-(xi/2 + 1/2)\*(eta - 1)\*(eta + 1) - (xi\*(eta - 1)\*(eta + 1))/2,...

(eta\*xi\*(eta + 1))/4 + (eta\*(xi/2 + 1/2)\*(eta + 1))/2,...

-(eta\*(eta + 1)\*(xi - 1))/2 - (eta\*(eta + 1)\*(xi + 1))/2,...

(eta\*xi\*(eta + 1))/4 + (eta\*(eta + 1)\*(xi - 1))/4,...

-(xi\*(eta - 1)\*(eta + 1))/2 - ((eta - 1)\*(eta + 1)\*(xi - 1))/2,...

(eta - 1)\*(eta + 1)\*(xi - 1) + (eta - 1)\*(eta + 1)\*(xi + 1);

(eta\*xi\*(xi - 1))/4 + (xi\*(eta - 1)\*(xi - 1))/4, ...

-(eta\*(xi - 1)\*(xi + 1))/2 - ((eta - 1)\*(xi - 1)\*(xi + 1))/2, ...

(eta\*xi\*(xi/2 + 1/2))/2 + (xi\*(xi/2 + 1/2)\*(eta - 1))/2, ...

-xi\*(xi/2 + 1/2)\*(eta - 1) - xi\*(xi/2 + 1/2)\*(eta + 1), ...

(eta\*xi\*(xi/2 + 1/2))/2 + (xi\*(xi/2 + 1/2)\*(eta + 1))/2, ...

-(eta\*(xi - 1)\*(xi + 1))/2 - ((eta + 1)\*(xi - 1)\*(xi + 1))/2, ...

(eta\*xi\*(xi - 1))/4 + (xi\*(eta + 1)\*(xi - 1))/4, ...

-(xi\*(eta - 1)\*(xi - 1))/2 - (xi\*(eta + 1)\*(xi - 1))/2, ...

(eta - 1)\*(xi - 1)\*(xi + 1) + (eta + 1)\*(xi - 1)\*(xi + 1)

];

end

end

function [jacobian\_matrix, jacobian\_inv, diff\_shape\_xy] = form\_jacobian(node\_coordinates, diff\_shape)

%

% cal jacobian.

%

% @since 1.0.1

% @param {type} [node\_coordinates] nodal coordinates at element level.

% @param {type} [diff\_shape] derivatives w.r.t. xi and eta.

% @return {type} [jacobian\_matrix] Jacobian matrix.

% @return {type} [jacobian\_inv] inverse of Jacobian Matrix.

% @return {type} [diff\_shape\_xy] derivatives w.r.t. x and y

% @see dependencies

%

jacobian\_matrix = diff\_shape \* node\_coordinates;

jacobian\_inv = inv(jacobian\_matrix);

% TODO: check which one is fast

diff\_shape\_xy = jacobian\_matrix \ diff\_shape;

% diff\_shape\_xy = jacobian\_inv \* diff\_shape;

end