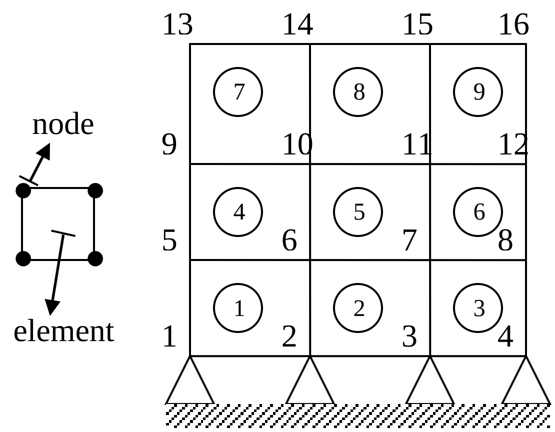
Automatic Assembly of Equations for Finite Element Models

1. Typical input data for building finite element models

A. Control Data
B. Nodal Coordinates
C. Element Connectivity
D. Material Definition
E. Section Definition
F. Boundary Conditions
G. Loading Conditions



2D Example Structure (Q4)

A. Control Data

ndofn=2 % the number of degrees of freedom per node
nnode=4 % the number of nodes per element
npoin=16 % a total number of nodes in the entire structure
nelem=9 % the number of elements
nvfix=4 % the number of nodes where boundary conditions are defined

B. Nodal Coordinates (node number, x, y, z)

1, x_1, y_1, z_1
2, x_2, y_2, z_2
…
16, x_{16}, y_{16}, z_{16}

C. Element Connectivity (element number, n_i, n_j, n_k, n_l) → Stored in '*lnods*'

1, 1, 2, 5, 6
2, 2, 3, 7, 6
…
9, 11, 12, 16, 15

F. Boundary Conditions (node number, index for 1 direction, index for 2 direction) 1: fixed, 0: free

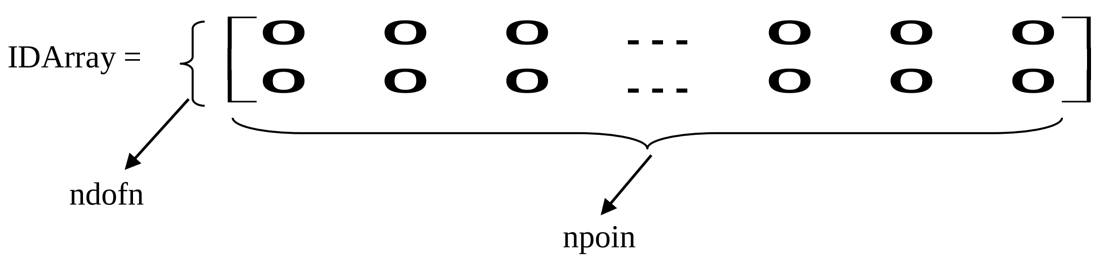
1, 1, 1
2, 1, 1
3, 1, 1
4, 1, 1

MATLAB Code for scanning and storing data for boundary conditions

```
…
% Read boundary conditions
    elseif length(findstr(tline,'*bound'))~=0
        BND_DATA = fscanf(ifid,'%d\n',[3,nvfix]);
        BND_DATA = BND_DATA';
        nofix = BND_DATA(:,1);
        fix_data = BND_DATA(:,2:3);
        % get infomation on boundary condition
        for ivfix=1:nvfix
            for idofn=1:ndofn
                pos = (nofix(ivfix)-1)*ndofn + idofn;
                iffix(pos) = fix_data(ivfix,idofn);
            end
        end
        tline = fgetl(ifid);
% Pass unidentified command
    else
…
```

```
>> iffix = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
>> Size (iffix) = 1    32;
```

2. Constructing Destination Array



MATLAB code for constructing destination array

```
function [IDArray] = GetIDArray()

global npoin ndofn
global iffix nofix
global neq

% initialize of IDArray
for ipoin=1:npoin
    iposi = (ipoin-1)*ndofn;
    for idofn=1:ndofn
        if (iffix(iposi+idofn)==0)
            IDArray(idofn,ipoin) = 0;
        elseif (iffix(iposi+idofn)~=0)
            IDArray(idofn,ipoin) = 1;
        end
    end
end

% Generate a table of equation number
neq = 0;
for ipoin=1:npoin
    for idofn=1:ndofn
       % transfer if DOF is fixed. Otherwise, increment neq
        if (IDArray(idofn,ipoin)==0)
            neq = neq+1;
            IDArray(idofn,ipoin) = neq;
        else
            IDArray(idofn,ipoin) = 0;
        end
    end
end

return
```

$$\Downarrow$$

$$>> \mathbf{IDArray} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 0 & 0 & 0 & 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 & 20 & 22 & 24 \end{bmatrix}$$

Note that the total number of equations (neq) is 24.

3. Automatic Assembly of Equations

MATLAB code for assembling elemental stiffness matrices to construct global stiffness matrix.

```
function [xGK] = GetGlobStiffFull(xGK)


global stiff_files
global nevab nelem npoin neq
global nnode ndofn
global lnods
global IDArray

% Read element stiffness matrices from output file from ABAQUS
nevab = nnode * ndofn;

for ielem=1:nelem

    estif = zeros(nevab, nevab);

    % Read element stiffness matrix
    [estif] = GetElemStiff(estif,ielem);

    % when using IDArray
    for inode=1:nnode
        nod = lnods(ielem,inode);
        for jdofn=1:ndofn
            pos =(inode-1)*ndofn + jdofn;
            KK(pos) = IDArray(jdofn, nod);
        end
    end

    % Assemble element stiffness matrix
    for ievab=1:nevab
        if( KK(ievab) <= 0 )
            continue;
        end
        I=KK(ievab);
        for jevab=1:nevab
            J = KK(jevab);
            if( J < I )
                continue;
            end
            xGK(I,J) = xGK(I,J) + estif(ievab,jevab);
        end
    end

end
…continued
```

You should develop this function to calculate elemental stiffness matrices!!

```
% make it symmetric matrix
for ieq=1:neq
    for jeq=ieq:neq
        xGK(jeq,ieq) = xGK(ieq,jeq);
    end
end

return
```

$$\Downarrow$$

$$>> \mathbf{xGK} = \begin{bmatrix} K_{1,1} & K_{1,2} & . & . & K_{1,24} \\ K_{2,1} & K_{2,2} & . & . & K_{2,24} \\ . & . & . & . & . \\ . & . & . & . & . \\ K_{24,1} & K_{24,2} & . & . & K_{24,24} \end{bmatrix}$$

Note that because restrained degrees of freedom were not considered, **xGK** is invertible as it is and **xGK** is symmetric matrix.