Simplified Finite Element Analysis

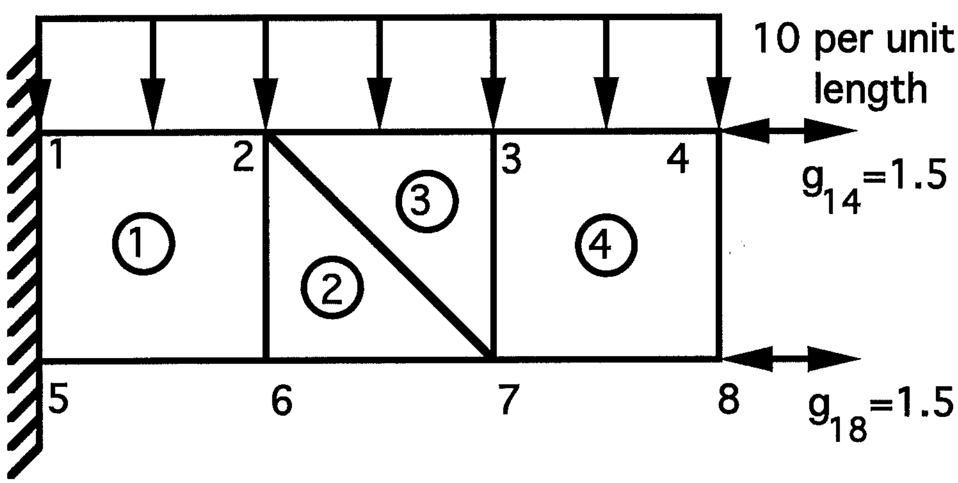
The goal of this write-up is to introduce a structure to demonstrate the basic concepts associated with the calculation of element matrices, assembling them into the global matrices and recovering elemental quantities. The method used to do this uses some simple pseudo code and data structures (sort of). Please note the following assumptions

- all potential dof are associated with node points
- the # of dof per node is the same for all nodes and is equal to n_{sd}

and also consider one alteration form the text

- data structures similar to those in the text will be used with some modifications introduced to make them simpler, slightly more general, and to make it clearer how to handle the prescribed nodal displacements, g_{iA}

example problem to be considered



```
procedure FEA (NNP; NEL; NSD)
   {simplistic view of a finite element analysis process}
   var
      NNP = {n_np, number of nodes mesh}
      NEL = {n_el, number of elements mesh}
      NSD = {n_sd, number of space dimensions}
      NDOF = {n_dof, number of dof in global matrices (please note this is a
                 change from the text in the use of terms)}
      NDOG = {n_dog, number dof associated with non-zero g_iA's}
      ID = {"destination structure" holds "global equations numbers"
              explained in SETUP_EQ}
      K = {K, global stiffness matrix (n_dof,n_dof)}
      F = {F, global force vector  (n_dof)}
      G = {global prescribed displacement vector (n_dog)}
      NEN = {n_en, number of nodes per element}
      LM = {element "location matrix" explained in SETUP_EQ}
      KE = {k^e ,element stiffness matrix (n_ee,n_ee), (n_ee = n_en n_sd)}
      FE = {F^e ,element force vector (n_ee)}
      DISP = {d, global displacement vector (n_dof)}
    begin
      call  SETUP_EQ (NNP,NEL,NSD,ID,NDOF,NDOG,K,F,G)
            {label global equation #'s and initialize global matrices}
      for  I_EL = 1 to NEL {loop over # of elements to calculate
                                      element matrices}
         call ELE_STIFF (NSD,ID,LM,NEN,KE,FE) {calc. element
           stiffness matrix and element location vector}
         call ASSEMBLE (NSD,LM,NEN,KE,FE,G,K,F)
          {assemble the element matrices into the global matrices}
      end {for}
      call SOLVE (NDOF,K,F,DISP) {solve global system (d = K^-1 F)}
      call OUTPUT_DISP. (NNP,NSD,ID,DISP,G)
            {print out the displacements at the nodes}
      for  I_EL = 1 to NEL {calculate and print out stresses or fluxes)
         call OUT_FLUX (NSD,ID,DISP,G)
      end {for}
    end {FEA}
```

procedure SETUP_EQ (NNP;NEL;NSD;ID;NDOF;NDOG;K;F;G)

{This procedure is responsible for specifying the equation numbers at each of the possible nodal dof and identifying the essential BC}

var

ID = {Destination matrix which, for each possible dof, gives the equation number (if it is an equation) or the location of the prescribed essential BC in G (if the essential BC is specified). Consider ID as an array where each entity is given a type, t.

$$ID(i, A{:}\ t)\ i = 1(1)n_{sd},\ A = 1(1)n_{np},\ t = \begin{cases} dof & \text{if } i, A \in \eta - \eta_{g_i} \\ dog & \text{if } i, A \in \eta_{g_i} \text{ and } g_{iA} \neq 0 \\ null & \text{if } i, A \in \eta_{g_i} \text{ and } g_{iA} = 0 \end{cases}$$

NTYP = {vector indicating the type of dof for each component at the node.

$$\boxed{NTYP(i) = \begin{cases} dof & \text{if } NTYP(i) \in \eta - \eta_{g_i} \\ dog & \text{if } NTYP(i) \in \eta_{g_i} \text{ and } g_{iA} \neq 0 \\ null & \text{if } NTYP(i) \in \eta_{g_i} \text{ and } g_{iA} = 0 \end{cases}}$$

FG = {vector of either nodal point forces or nodal prescribed essential BC. $FG(i) = \begin{cases} load & \text{if } NTYP(i) = dof \\ g_{iA} & \text{if } NTYP(i) = dog \end{cases}$ }

```
begin
    {initialize global counters}
    NDOF = 0
    NDOG = 0
    for A = 1 to NNP do  {loop over the nodes setting the equation or
        prescribed essential BC number and load the global
        nodal loads or prescribed essential BC}
        read ((NTYP(i), i= 1, NSD) (FG(i),i=1,NSD)
        for i = 1 to NSD do
                if NTYP(i) = dof  then {the possible dof is a dof}
                        NDOF = NDOF+1
                        t = dof
                        ID(i,A:t) = NDOF {give it an equation number}
                        F(NDOF) = FG(i) {if there are nodal loads}
                else {the possible dof is a prescribed essential BC}
                        if NTYP(i) = dog  then {nonzero prescribed
                                                essential BC}
                        NDOG = NDOG+1
                        t = dog
                        ID(i,A:t) = NDOG {pointer to location of value}
                        G(NDOG) = FG(i) (storing value of essential BC}
                        else {zero prescribed essential BC}
                        t = null
                        ID(i,A:t) = 0 {allow us to skip zero multiplies}
                        end {if}
                end {if}
        end {loop over possible dof at node}
    end {loop over nodes}
    {zero out the global stiffness matrix)
    for A = 1 to NDOF do
        for B = 1 to NDOF do
                K(A,B) = 0.0
        end {loop over possible dof at node}
    end {loop over nodes}
end {SETUP_EQ}
```

```
procedure ELE_STIFF (NSD,ID,LM,NEN,KE,FE)
    {main routine for calculating the element matrices}
    var
        NEN = {n_en number of nodes for the element}
        IEN = {vector ofglobal node numbers for the nodes defining the
                element (n_en)}
        LM = {element location vector identifying the equation
               numbers and location of prescribed essential BC for
               the element(n_sd,n_en:t)}
        N = {element shape functions (n_en)}
        B = {Derivatives of shape functions as defined by the
             strain-displacement relations, for n_sd = 2 it is (3,2n_en)}
        S  = {Stress displacement matrix, for n_sd = 2 it is (3,2n_en)}
    begin
        read (NEN, (IEN(a), a=1,NEN)) {read in the elements nodes}
        {set-up the location vector which will keep track of the
         mapping between the local dof numbers and the global
         equation or specified essential BC numbers}
        for a = 1 to NEN do
            A = IEN(a)
            for i = 1 to NSD do
                LM(i,a:t) = ID(i,A:t)
            end {do}
        end {do}
        {read element material properties and obtain nodal coordinate
         info. that we assume were read in before. Common to employ
         numerical integration in this process (will study later). As
         part of this process we calculate the matrices needed in stress
         recovery. We will save those matrices for later. (On today's
         computers it is faster to recalculate.) }
        NINT = {number of integration points to be used}
        NEE = NEN * NSD {size of element stiffness}
        write (NINT, NEN, LM) {need this info. to control stress
                                          recovery process}
        for p = 1 to NEE do {zero out stiffness and load vector (for later)}
            FE(p) = 0.0
            for q = 1 to NEE do
                KE(p,q) = 0.0
            end {do}
        end {do}
```

for INT = 1 **to** NINT **do** {loop over the integration points and add into the appropriate contributions}

B{of INT} = {calc. strain disp. relationship on shape functions N}

S{of INT} = D{of INT} B{of INT} $\sigma = DBd^e$

write (S{of INT})

KE = KE + B{of INT}^{T} * S{of INT} * WEIGHT{of INT}

end {do element stiffness}

{Calculate the contribution to the element loads due to body forces and tractions. The contribution due to non-zero essential boundary conditions is handled in ASSEMBLE. Assuming a 2-D case, the contributions can include body forces, f_i, edge loads, h_i, and prescribed edge essential BC, g_i. To simplify the presentation we will assume i) all quantities are uniform in value, ii) there are body forces in both directions, iii) each element edge has two node points, and iv) one edge from node b to c has an edge load in the 2 direction.}

{body force contribution}

for a = 1 **to** NEN **do**

for i = 1 **to** NSD **do** (w, f)

p = NSD *(a-1) + i

FE(p) = FE(p) + $\int_{\Omega^e} N_a f_i d\Omega$

end {do}

end {do}

{edge load contribution - For the specific case given there will be contributions to only two terms in the element load vector}

p = NSD*(b-1) + 2

FE(p) = FE(p) + $\int_{\Gamma^e_{b-c}} N_b(b-c)h_2 d\Gamma$ $(w, h)_\Gamma$

p = NSD*(c-1) + 2

FE(p) = FE(p) + $\int_{\Gamma^e_{b-c}} N_c(b-c)h_2 d\Gamma$

end {ELE_STIFF}

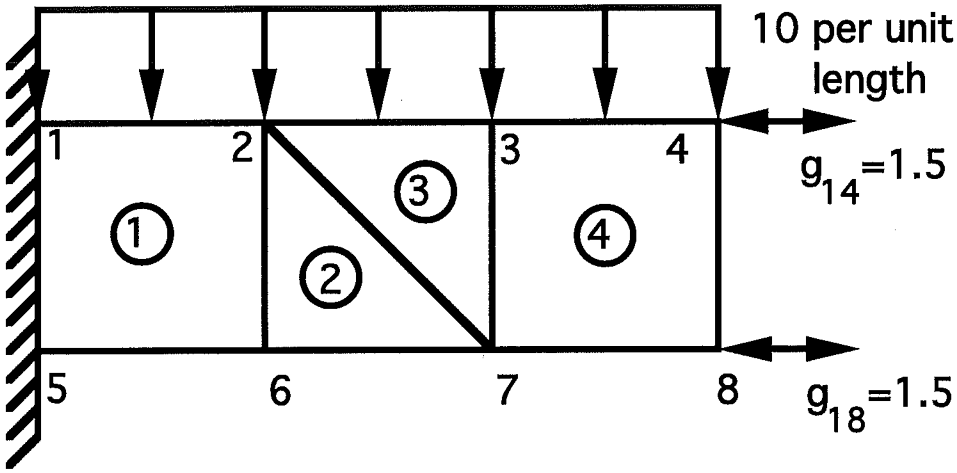
```
procedure ASSEMBLE (NSD,LM,NEN,KE,FE,G,K,F)
    {procedure to assemble the element contributions into the global
     matrices. Note in a real program the sparseness of the global
     matrix would be taken in account and only the terms from the
     main diagonal up would be considered. Here we use full matrices
     and add everything.}
  var
  begin
     p = 0
     {the outer loop set is going over rows of stiffness matrix and load vector)
     for  a = 1 to NEN do {loop over number of nodes in the element}
          for i = 1 to NSD do
               p = p + 1   <- row counter
               if t of LM(i,a:t) = dof  then {row is associated with a dof}
               {process terms in the row since it a dof}
               {if row not associated with a dof – skip row ans go to next row}
                    P = LM(i,a:t)
                    F(P) = F(P) + FE(p)
                    q = 0
                    {loop set to go over columns of stiffness matrix}
                    for b = 1 to NEN do {loop over number of nodes in
                              the element}
                         for j = 1 to NSD do
                              q = q + 1   <- column counter
                              if t of LM(j,b:t) = dof  then
                              {column associated with dof}
                                   Q = LM(j,b:t)
                                   K(P,Q) = K(P,Q) + KE(p,q)
                              end {if}
                              if t of LM(j,b:t) = dog  then
                              {column associated with dog}
                                   Q = LM(j,b:t)
                                   F(P) = F(P) - G(Q)* KE(p,q)
                              end {if}
                         end {do over NSD}
                    end {do over NEN}
               end {if}
          end {do over NSD}
     end {loop over number of nodes in element}
  end {ASSEMBLE}
```

going over rows of k^e and f^e

```
procedure OUTPUT_DISP.(NNP,NSD,ID,DISP,G)
    {This procedure is responsible for printing the displacements at nodes}
    var
      DISP = {d, global displacement vector       (n_dof)}
      NNP = {n_np, number of nodes mesh}
      NSD = {n_sd, number of space dimensions}
      ID = {"destination structure" holds "global equations numbers"
              explained in SETUP_EQ}
      G = {global prescribed displacement vector       (n_dog)}
      DISP_N = {displacements at one node (n_sd)}
    begin
        for A = 1 to NNP do  {loop over the nodes setting the equation or
            prescribed essential BC number and load the global
            nodal loads or prescribed essential BC}
            for i = 1 to NSD do
                    P = ID(i,A:t)
                    if t of ID(i,A:t) = dof  then { dof is a dof}
                            DISP_N(i) = DISP(P)
                    else {the possible dof is a prescribed essential BC}
                            if  t of ID(i,A:t) = dog  then {nonzero prescribed
                                                            essential BC}
                            DISP_N(i) = G(P)
                            else {zero prescribed essential BC}
                            DISP_N(i) = 0.0
                            end {if}
                    end {if}
            end {loop over dof at node}
            write (A,DISP_N(i),i=1,NSD) {write the node's displacements}
        end {loop over nodes}
    end {OUTPUT_DISP.}
```

```
procedure OUT_FLUX(NSD,ID,DISP,G)
    {This procedure is responsible for calculating and printing the
     stresses or fluxes}
    var
      DISP = {d, global displacement vector        (n_dof)}
      NSD = {n_sd, number of space dimensions}
      ID = {"destination structure" holds "global equations numbers"
              explained in SETUP_EQ}
      G = {G, global prescribed displacement vector        (n_dog)}
      DE = {d^e,element displacement vector (n_ee = n_sd n_en)}
      S = {stress displacement matrix}
      SIG = {stress vector - size defined by S matrix)
    begin
        read (NINT, NEN, LM) {read info. to control stress
                                            recovery process}
        {need to get the displacements for the element}
        p=0
        for a = 1 to NEN do  {loop over the nodes in the element}
            for i = 1 to NSD do
                    p=p+1
                    if t of LM(i,a:t) = dof  then { dof is a dof}
                    P = LM(i,a:t)
                          DE(p) = DISP(P)
                    else {the possible dof is a prescribed essential BC}
                          if  t of LM(i,a:t) = dog  then {nonzero prescribed
                                                     essential BC}
                          P = LM(i,a:t)
                          DE(p) = G(P)
                          else {zero prescribed essential BC}
                          DE(p) = 0.0
                          end {if}
                    end {if}
            end {loop over dof at node}
        end {loop over number of nodes in the element}
        for INT = 1 to NINT do {loop over the integration points and add
                                      into the appropriate contributions}
            read (S{of INT})
            SIG = S * DE
            write (SIG)
        end {for}
    end {OUTPUT_FLUX}
```

Application of data structure



$n_{np} = 8,\ n_{el} = 4,\ n_{sd} = 2$

node information

node number	1	2	3	4	5	6	7	8
NTYP	$\begin{Bmatrix} null \\ null \end{Bmatrix}$	$\begin{Bmatrix} dof \\ dof \end{Bmatrix}$	$\begin{Bmatrix} dof \\ dof \end{Bmatrix}$	$\begin{Bmatrix} dog \\ dof \end{Bmatrix}$	$\begin{Bmatrix} null \\ null \end{Bmatrix}$	$\begin{Bmatrix} dof \\ dof \end{Bmatrix}$	$\begin{Bmatrix} dof \\ dof \end{Bmatrix}$	$\begin{Bmatrix} dog \\ dof \end{Bmatrix}$
FG	$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 1.5 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$	$\begin{Bmatrix} 1.5 \\ 0 \end{Bmatrix}$

ID - Destination Matrix

$$\begin{matrix} \text{node} \rightarrow & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \text{i} = 1 & 0{:}\,null & 1{:}\,dof & 3{:}\,dof & 1{:}\,dog & 0{:}\,null & 6{:}\,dof & 8{:}\,dof & 2{:}\,dog \\ \text{i} = 2 & 0{:}\,null & 2{:}\,dof & 4{:}\,dof & 5{:}\,dof & 0{:}\,null & 7{:}\,dof & 9{:}\,dof & 10{:}\,dof \end{matrix}$$

$n_{dof} = 10,\ n_{dog} = 2$

element information

element number	1	2	3	4
n_{en}	4	3	3	4
IEN	$\begin{Bmatrix} 5 \\ 6 \\ 2 \\ 1 \end{Bmatrix}$	$\begin{Bmatrix} 6 \\ 7 \\ 2 \end{Bmatrix}$	$\begin{Bmatrix} 7 \\ 3 \\ 2 \end{Bmatrix}$	$\begin{Bmatrix} 7 \\ 8 \\ 4 \\ 3 \end{Bmatrix}$

LM - location vectors for elements

element 1

$$\begin{bmatrix} 0: null & 6: dof & 1: dof & 0: null \\ 0: null & 7: dof & 2: dof & 0: null \end{bmatrix}$$

element 2

$$\begin{bmatrix} 6: dof & 8: dof & 1: dof \\ 7: dof & 9: dof & 2: dof \end{bmatrix}$$

element 3

$$\begin{bmatrix} 8: dof & 3: dof & 1: dof \\ 9: dof & 4: dof & 2: dof \end{bmatrix}$$

element 4

$$\begin{bmatrix} 8: dof & 2: dog & 1: dog & 3: dof \\ 9: dof & 10: dof & 5: dof & 4: dof \end{bmatrix}$$

Based on LM consider where terms in element stiffness matrices will be used

element 1

$$\mathbf{k}^1 = \begin{bmatrix} - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & K_{6,6} & K_{6,7} & K_{6,1} & K_{6,2} & - & - \\ - & - & K_{7,6} & K_{7,7} & K_{7,1} & K_{7,2} & - & - \\ - & - & K_{1,6} & K_{1,7} & K_{1,1} & K_{1,2} & - & - \\ - & - & K_{2,6} & K_{2,7} & K_{2,1} & K_{2,2} & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{bmatrix}$$

element 4

$$\mathbf{k}^4 = \begin{bmatrix} K_{8,8} & K_{8,9} & F_8 & K_{8,10} & F_8 & K_{8,5} & K_{8,3} & K_{8,4} \\ K_{9,8} & K_{9,9} & F_9 & K_{9,10} & F_9 & K_{9,5} & K_{9,3} & K_{9,4} \\ - & - & - & - & - & - & - & - \\ K_{10,8} & K_{10,9} & F_{10} & K_{10,10} & F_{10} & K_{10,5} & K_{10,3} & K_{10,4} \\ - & - & - & - & - & - & - & - \\ K_{5,8} & K_{5,9} & F_5 & K_{5,10} & F_5 & K_{5,5} & K_{5,3} & K_{5,4} \\ K_{3,8} & K_{3,9} & F_3 & K_{3,10} & F_3 & K_{3,5} & K_{3,3} & K_{3,4} \\ K_{4,8} & K_{4,9} & F_4 & K_{4,10} & F_4 & K_{4,5} & K_{4,3} & K_{4,4} \end{bmatrix}$$