APPENDIX B – PROGRAM SOURCE CODE
SDOF ground response model for non-linear soil response using the equivalent linear model. The equation of motion is integrated using a linear acceleration assumption. Damping and shear modulus values for sand and clay are from charts by Seed and Idriss.

GLOBAL strain() AS DOUBLE
GLOBAL zeta() AS DOUBLE
GLOBAL G() AS DOUBLE
GLOBAL vg() AS DOUBLE
GLOBAL ag() AS DOUBLE

SUB NonLin(n AS LONG,Sup AS LONG,SType AS LONG,FName AS STRING,Soil() AS SoilType)

LOCAL pi AS DOUBLE
LOCAL G AS DOUBLE
LOCAL rho AS DOUBLE
LOCAL h AS DOUBLE
LOCAL INtstep AS DOUBLE
LOCAL w AS DOUBLE
LOCAL zeta AS DOUBLE
LOCAL vrel AS DOUBLE
LOCAL velrel AS DOUBLE
LOCAL arel AS DOUBLE
LOCAL a AS DOUBLE
LOCAL i AS LONG
LOCAL j AS DOUBLE
LOCAL strain AS DOUBLE
LOCAL omega2 AS DOUBLE
LOCAL v AS DOUBLE
LOCAL deltav AS DOUBLE
LOCAL deltavel AS DOUBLE
LOCAL Gmax AS DOUBLE
LOCAL SLC AS STRING
LOCAL flag AS LONG
LOCAL eacolx AS DOUBLE
LOCAL ceacx AS DOUBLE
LOCAL j AS LONG
LOCAL nstep AS LONG
LOCAL IGTstep AS DOUBLE
LOCAL dvg AS DOUBLE
LOCAL MaxStrain AS DOUBLE
LOCAL Tol AS DOUBLE
LOCAL TolFlag AS DOUBLE
LOCAL strain0 AS DOUBLE
LOCAL SR AS DOUBLE

REDIM strain(50) AS DOUBLE
REDIM zeta(50) AS DOUBLE
REDIM G(50) AS DOUBLE
REDIM a(1:n) AS DOUBLE

FOR i = 1 TO n
    a(i) = vg(Sup,i) 'copy acceleration history
NEXT

REDIM vg(Sup,n)
pi = 4 * ATN(1)

G = VAL(Soil(SType).G)
Gmax = G
rho = VAL(Soil(SType).UW)
h = VAL(Soil(SType).depth)
INtstep = VAL(cntl.INtstep)
IGTstep = cntl.IGTstep
SLC = Soil(SType).SType
flag = 1 'MAKE FLAG=0 FOR LINEAR RESPONSE
strain0 = 0.01  # assume average strain
strain = strain0
Tol = 0.01   # Tolerance for convergance
SR = 0.65

TolFlag = 0
DO UNTIL TolFlag = 1

IF flag = 1 THEN
   CALL parameters(SLC)
   CALL update(strain, strain(), zeta(), zeta)  # update the damping ratio
   CALL update(strain, strain(), G(), G)      # update the shear modulus
END IF

G = Gmax*G

zeta = zeta / 100
w = (π/2/ h) * SQR(G / rho)
velrel = 0
arel = -a(1)
ceacx = a(1)

j = 1
j1 = 2
nstep = 1
DO UNTIL j > n

   eacolx = ceacx
   ceacx = (a(j) + (a(j1) - a(j))*(nstep * IGtstep) / INtstep)

   dvg = ceacx - eacolx
   omega2 = w^2 + 6 * (1 + zeta * w * IGTstep) / (IGTstep^2)
   v = -dvg + 6 * velrel / IGTstep + 3 * arel +
       zeta * w * (6 * velrel + IGTstep * arel)

   deltax = v / omega2  # incremental displacement
   deltavel = 3 * deltax / IGTstep - 3 * velrel - IGTstep * arel / 2
   velrel = velrel + deltavel  # total displacement @ end of time step
   arel = -(ceacx + w^2 * vrel + 2 * zeta * w * velrel)  # total acceleration @ end of time step

   strain = 100 * ABS(vrel) / h
   IF strain > MaxStrain THEN MaxStrain = strain
   IF nstep*Igtstep >= Instep THEN
      vg(Sup,j1) = vrel  # displacement history
      ag(Sup,j1) = arel+ceacx  # acceleration history
      nstep = 0
      INCR j
      INCR j1
      END IF

   INCR nstep

END LOOP

strain = MaxStrain * SR
IF ABS(1 - strain/strain0) < Tol THEN TolFlag = 1

strain0 = strain

LOOP
   PRINT #20, strain, zeta, G
END LOOP

END SUB
SDOF non-linear soil response routine using the Newton-Raphson 2nd method for solving the equations of motion at the end of the time step. The equation of motion is integrated using a linear acceleration assumption. Damping and shear modulus values for sand and clay are from charts by Seed and Idriss.

DECLARE FUNCTION fx(v AS DOUBLE) AS DOUBLE
DECLARE FUNCTION fxp(dv AS DOUBLE) AS DOUBLE
DECLARE FUNCTION fxpp(dv AS DOUBLE) AS DOUBLE

GLOBAL G AS DOUBLE
GLOBAL Gmax AS DOUBLE
GLOBAL zeta AS DOUBLE
GLOBAL vrel1 AS DOUBLE
GLOBAL velrel1 AS DOUBLE
GLOBAL ceacx AS DOUBLE
GLOBAL IGtstep AS DOUBLE
GLOBAL strain() AS DOUBLE
GLOBAL zeta() AS DOUBLE
GLOBAL G() AS DOUBLE
GLOBAL pi AS DOUBLE
GLOBAL rho AS DOUBLE
GLOBAL dvg AS DOUBLE
GLOBAL h AS DOUBLE
GLOBAL vg() AS DOUBLE
GLOBAL ag() AS DOUBLE

SUB NonLin1(n AS LONG,Sup AS LONG,SType AS LONG,FName AS STRING,Soil() AS SoilType)

LOCAL w AS DOUBLE
LOCAL vrel AS DOUBLE
LOCAL velrel AS DOUBLE
LOCAL arel AS DOUBLE
LOCAL a AS DOUBLE
LOCAL i AS LONG
LOCAL j AS DOUBLE
LOCAL i1 AS LONG
LOCAL nstep AS LONG
LOCAL X1 AS DOUBLE
LOCAL Tol AS DOUBLE
LOCAL TolFlag AS LONG
LOCAL Fsp AS DOUBLE
DIM strain(50) AS DOUBLE
DIM zeta(50) AS DOUBLE
DIM G(50) AS DOUBLE
DIM a(1:n) AS DOUBLE

FOR i = 1 TO n
    a(i) = vg(Sup,i) 'copy acceleration history
NEXT
vg(Sup,1) = 0
pi = 4 * ATN(1)
Tol = 0.005# 'Convergence Tolerance for G(x) and zeta(x)

G = VAL(Soil(SType).G)
Gmax = G
\[
\begin{align*}
\rho &= \text{VAL}(\text{Soil(SType).UW}) \\
h &= \text{VAL}(\text{Soil(SType).depth}) \\
\text{INtstep} &= \text{VAL}(\text{cntl.INtstep}) \\
\text{IGTstep} &= \text{cntl.IGTstep} \\
\text{SLC} &= \text{Soil(SType).SType} \\
\text{flag} &= 1 \\
\text{IF} \ \text{Soil(SType).RFlag} = 3 \ \text{THEN} \ \text{flag} = 0 \ \text{FLAG} = 0 \ \text{FOR LINEAR RESPONSE} \\
\text{strain} &= 0# \\
\text{CALL} \ \text{parameters(SLC)} \\
\text{CALL} \ \text{update(strain, strain(), zeta(), zeta)} \ '\text{BE CAREFUL!}' \\
zeta &= \text{zeta} / 100 \\
vrel &= 0 \\
vrel &= 0 \\
arel &= -a(1) \\
ceacx &= -a(1) \\
\text{j} &= 1 \\
\text{j1} &= 2 \\
nstep &= 1 \\
w &= (\pi / 2 / h) * \text{SQR}(G / \rho) \ '\text{natural frequency}' \\
w &= \text{SQR}(2 / h) * \text{SQR}(G / \rho) \ '\text{natural frequency, lumped mass assumption}' \\
\text{DO UNTIL} \ \text{j} > \text{n} \\
&\quad '\text{interpolate input accelerogram ordinate}' \\
ceacolx &= \text{ceacx} \\
ceacx &= (a(j) + (a(j1) - a(j)) * (nstep * \text{IGTstep}) / \text{INtstep}) \\
dvg &= \text{ceacx} - \text{ceacolx} \ '\text{incremental ground acceleration}' \\
vrel1 &= vrel \\
vrel1 &= vrel \\
\text{DO UNTIL} \ \text{TolFlag} = 1 \\
\omega2 &= w^2 + 6 * (1 + zeta * w * \text{IGTstep}) / (\text{IGTstep}^2) \\
v &= -dvg + 6 * vrel1 / \text{IGTstep} + 3 * arel + \\
&\quad zeta * w * (6 * vrel1 + \text{IGTstep} * arel) \\
deltav &= v / \omega2 \ '\text{incremental displacement}' \\
deltavel &= 3 * \text{deltav} / \text{IGTstep} - 3 * vrel1 - \text{IGTstep} * arel / 2 \\
vrel1 &= vrel + \text{deltav} \ '\text{total displacement @ end of time step}' \\
vrel1 &= vrel + \text{deltavel} \ '\text{total velocity @ end of time step}' \\
arel &= -(ceacx + w^2 * vrel1 + 2 * zeta * w * vrel1) \ '\text{total acceleration @ end of time step}' \\
\text{IF} \ \text{flag} = 1 \ \text{THEN} \ '\text{Non-linear Soil behavior}' \\
\text{X1} &= \text{deltav} + 1 / (\text{fxpp}(\text{deltav}) / 2 / \text{fzp}(\text{deltav}) - \text{fxp}(\text{deltav}) / \text{fx}(\text{deltav})) \\
\text{IF} \ \text{ABS}(\text{X1} - \text{deltav}) < \text{Tol} \ \text{THEN} \ \text{TolFlag} = 1 \\
\text{strain} &= 100 * \text{ABS}(vrel + X1) / h \ '\text{average strain in the soil (percent)}' \\
\text{CALL} \ \text{update(strain, strain(), zeta(), zeta)} \ '\text{update the damping ratio}' \\
\text{CALL} \ \text{update(strain, strain(), G(), G)} \ '\text{update the shear modulus}' \\
zeta &= \text{zeta} / 100 \\
G &= G * Gmax \\
w &= (\pi / 2 / h) * \text{SQR}(G / \rho) \ '\text{update the natural frequency}'
\end{align*}
\]
deltav = X1
ELSEIF flag = 0 THEN
EXIT DO
END IF

LOOP
vrel = vrel1
velrel = velrel1
TolFlag = 0
IF nstep\*IGTstep >= InTstep THEN
vg(Sup,j1) = vrel 'displacement history
ag(Sup,j1) = arel + ceacx 'absolute acceleration
Fsp = vrel/H * G 'shear force in the unit area soil column
PRINT #20,j*INTstep,vrel,Fsp

nstep = 0
INCR j
INCR j1
END IF

INCR nstep
LOOP
END SUB

FUNCTION fx(deltav AS DOUBLE) AS DOUBLE
LOCAL strain AS DOUBLE
LOCAL deltavel AS DOUBLE
LOCAL arel AS DOUBLE
LOCAL w AS DOUBLE
LOCAL omega2 AS DOUBLE
LOCAL v AS DOUBLE
strain = 100 * ABS(deltav) / h 'average strain in the soil (percent)
CALL update(strain, strain(), zeta(), zeta) 'update the damping ratio
CALL update(strain, strain(), G(), G) 'update the shear modulus
G = G * Gmax
zeta = zeta / 100
w = (pi /2/ h) * SQR(G / rho) 'update the natural frequency
deltavel = 3 * deltav / IGTstep - 3 * velrel1 - IGTstep * arel / 2 'inc. vel
arel = -(ceacx + w^2 + vrel1 + 2 * zeta * w * velrel1) 'total acceleration @ END of time STEP
omega2 = w^2 + 6 * (1 + zeta * w * IGTstep) / (IGTstep^2)
v = -dvg + 6 * velrel1 / IGTstep + 3 * arel +
zeta * w * (6 * velrel1 + IGTstep * arel)
fx = deltav*omega2 - v
END FUNCTION

FUNCTION fxp(dv AS DOUBLE) AS DOUBLE
LOCAL h1 AS DOUBLE
LOCAL i AS LONG
LOCAL fxp AS DOUBLE
REDIM f1(1:6) AS DOUBLE
\[ h_1 = 0.01 \]

\[ f_1(1) = -fx(dv - 3h_1) \]
\[ f_1(2) = 9*fx(dv - 2h_1) \]
\[ f_1(3) = -45*fx(dv - h_1) \]
\[ f_1(4) = 45*fx(dv + h_1) \]
\[ f_1(5) = -9*fx(dv + 2h_1) \]
\[ f_1(6) = fx(dv + 3h_1) \]

FOR i = 1 TO 6
  \[ fxp1 = fxp1 + f1(i) \]
NEXT

\[ fxp = fxp1 /60 / h_1 \]

END FUNCTION

FUNCTION fxpp(dv AS DOUBLE) AS DOUBLE

LOCAL h1 AS DOUBLE
LOCAL i AS LONG
LOCAL fxp1 AS DOUBLE
DIM f1(1:7) AS DOUBLE

\[ h_1 = 0.01 \]

\[ f_1(1) = 2*fx(dv - 3h_1) \]
\[ f_1(2) = -27*fx(dv - 2h_1) \]
\[ f_1(3) = 270*fx(dv - h_1) \]
\[ f_1(4) = -490*fx(dv) \]
\[ f_1(5) = 270*fx(dv + h_1) \]
\[ f_1(6) = -27*fx(dv + 2h_1) \]
\[ f_1(7) = 2*fx(dv + 3h_1) \]

FOR i = 1 TO 7
  \[ fxp1 = fxp1 + f1(i) \]
NEXT

\[ fxpp = fxp1 /180 / h_1 / h_1 \]

END FUNCTION

SUB interpolate (valstart AS DOUBLE, valend AS DOUBLE, strainst AS DOUBLE, strainend AS DOUBLE, strain AS DOUBLE, value AS DOUBLE)

LOCAL slope AS DOUBLE

slope = (valstart - valend) / (strainst - strainend)
value = slope * (strain - strainst) + valstart
IF value = 0 THEN value = valstart
END SUB

SUB parameters (SoilType AS STRING)

LOCAL i AS LONG
LOCAL j AS LONG
LOCAL a AS DOUBLE
LOCAL NumPts AS LONG

\[ i = 1 \]
DO
  a = VAL(READ$(i))
  IF a = -1 THEN EXIT LOOP
  strain(i) = a
INCR i
LOOP
NumPts = i
IF UCASE$(SoilType) = "S" THEN

  i = 1
  DO
    INCR NumPts
    a = VAL(READ$(NumPts))
    IF a = -1 THEN EXIT LOOP
    G(i) = a
    IF i = 1 THEN G(0) = G(1)
    INCR i
  LOOP

  i = 1
  DO
    INCR NumPts
    a = VAL(READ$(NumPts))
    IF a = -1 THEN EXIT LOOP
    zeta(i) = a
    IF i = 1 THEN zeta(0) = zeta(1)
    INCR i
  LOOP

ELSE

  i = NumPts
  j = 1
  DO
    a = VAL(READ$(i))
    IF a = -1 THEN INCR j
    IF j = 2 THEN EXIT LOOP
    END IF
    INCR i
  LOOP

  NumPts = i

  i = 1
  DO
    a = VAL(READ$(NumPts))
    IF a = -1 THEN EXIT LOOP
    G(i) = a
    IF i = 1 THEN G(0) = G(1)
    INCR i
  LOOP

  i = 1
  DO
    a = VAL(READ$(NumPts))
    IF a = -1 THEN EXIT LOOP
    zeta(i) = a
    IF i = 1 THEN zeta(0) = zeta(1)
    INCR i
  LOOP

END IF

'strain data points
DATA 0,0.0001,0.0002,0.0005,0.001,0.002,0.005,0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,-1

'shear modulus data points for sand
DATA 1,1.0998,0.98,0.949,0.917,0.832,0.729,0.6,0.421,0.291,0.188,0.098,0.060,0.036,0.016,-1

'damping ratio's for sand
DATA 0.5, 0.5, 0.8, 1.3, 1.9, 2.5, 3.7, 5.3, 7.7, 12, 15.3, 18.7, 22.6, 24.4, 25.9, 27.3, -1

'shear modulus data points for clay
DATA 2500, 2500, 2450, 2420, 2400, 2320, 2300, 2200, 2120
DATA 2050, 2000, 1930, 1800, 1700, 1600, 1500, 1400, 1220, 1150
DATA 1000, 952, 904, 851, 793, 745, 698, 634, 580, 507
DATA 459, 380, 300, 275, 250, 230, 205, 175, 150, 130
DATA 100, 92,-1

damping ratio's for clay
DATA 2.63, 2.63, 2.63, 2.63, 2.63, 2.63, 2.63, 2.63, 2.63, 2.63, 2.63, 2.63
DATA 2.63, 2.63, 2.63, 2.89, 3.16, 3.42, 3.55, 3.81, 3.97
DATA 4.23, 4.42, 4.74, 5, 5.39, 5.66, 6.05, 6.45, 6.84, 7.37
DATA 7.89, 8.42, 9.21, 9.74, 10.79, 11.32, 12.5, 13.68, 14.74, 16.05
DATA 17.37, 18.42,-1

END SUB

SUB update (strain AS DOUBLE, strain() AS DOUBLE, value() AS DOUBLE, value AS DOUBLE)
LOCAL i0 AS LONG
LOCAL i AS LONG
i0 = 0
i = 1
DO
IF strain < strain(i) THEN
CALL interpolate(value(i0), value(i), strain(i0), strain(i), strain, value)
EXIT LOOP
END IF
INCR i
INCR i0
LOOP
END SUB

*********************************************************************************
Calculate response history using superposition of psuedo-static displacements and dynamic response
*********************************************************************************

#INCLUDE "Response.bas" 'Modal Response Module
#INCLUDE "Nonlin.bas" 'Equivalent Stiffness Module (Ground Motion Response Model)
#INCLUDE "Nonlin1.bas" 'Nonlinear and Linear Module (Ground Motion Response Model)

DECLARE SUB Response(N AS LONG, j AS LONG, i AS DOUBLE, c AS DOUBLE, w AS DOUBLE, z AS DOUBLE)
DECLARE SUB Nonlin(n AS LONG, Sup AS LONG, SType AS LONG, FName AS STRING, Soil() AS SoilType)

SUB RespHist(Lcase AS LONG, cntl AS cntlType, Mode() AS ModeType, FN() AS FuncType, LC() AS LTCType)
LOCAL i AS LONG
LOCAL n AS LONG
LOCAL i1 AS LONG
LOCAL i2 AS LONG
LOCAL TotModes AS LONG
LOCAL MDOF AS LONG
LOCAL TotPoints AS LONG
LOCAL TotPoints1 AS LONG
LOCAL TotPnts AS LONG
LOCAL OutTstep AS DOUBLE
LOCAL IGtstep AS DOUBLE
LOCAL INtstep AS DOUBLE
LOCAL Sup AS LONG
LOCAL n AS LONG
LOCAL w AS DOUBLE
LOCAL zeta AS DOUBLE
LOCAL nstep AS LONG
LOCAL nstep1 AS LONG
LOCAL j AS LONG
LOCAL j1 AS LONG
LOCAL Nsup AS LONG
LOCAL eacolx AS DOUBLE
LOCAL ceacx AS DOUBLE
LOCAL dvg AS DOUBLE
LOCAL NDOF AS LONG
LOCAL n1 AS DOUBLE
LOCAL time AS DOUBLE
LOCAL t AS STRING
LOCAL a AS STRING
LOCAL d AS STRING
LOCAL v AS STRING
LOCAL a1 AS STRING
LOCAL i3 AS LONG
LOCAL RFlag AS LONG
LOCAL rd AS STRING
LOCAL rd1 AS STRING

TotModes = 2&  'Defaults for this 2 DOF application.
MDOF = 2&    'Could be changing for future application.
Sup = 2&

cntl.MaxRD1 = 0#  'initialize maximum relative displacement
cntl.MaxRD2 = 0#  'initialize maximum relative displacement

i1 = LC(Lcase).Ftn(1)

IGtstep = cntl.IGtstep
INtstep = VAL(FN(i1).INtstep)
OutTstep = cntl.tstep

i = LC(Lcase).Ftn(1)
i1 = LC(Lcase).Ftn(2)

TotPoints = FN(i).NumRec
TotPoints1 = FN(i1).NumRec  'total integration points

IF TotPoints1 > TotPoints THEN TotPoints = TotPoints1

TotPnts = TotPoints + 1&

REDIM ag(1:Sup,1:TotPnts) AS DOUBLE
REDIM vg(1:Sup,1:TotPnts) AS DOUBLE
DIM d(1:MDOF,1:TotPnts) AS DOUBLE
DIM v(1:MDOF,1:TotPnts) AS DOUBLE
DIM a(1:MDOF,1:TotPnts) AS DOUBLE

DIM gd(1:Sup,1:TotPnts) AS DOUBLE

REDIM f(1:3) AS LONG

f(1) = FREEFILE
f(2) = f(1) + 1
f(3) = f(2) + 1

OPEN RTRIM$(cntl.FileLoc)+RTRIM$(cntl.FileNme) + ".L" + LTRIM$(STR$(Lcase)) FOR OUTPUT AS #f(3)

'input time history functions

FOR n = 1 TO Sup
  i1 = LC(Lcase).Ftn(n)
OPEN RTRIM$(cntl.EQFileLoc) + RTRIM$(FN(i1).FName) + LTRIM$(FN(i1).XTN) FOR INPUT AS #f(n)

IF cntl.IGtstep > VAL(FN(i1).INtstep) THEN
  MSGBOX "Integration Time Step Cannot Be Greater Than Input Time Step",_%MB_ICONWARNING,"Error - Integration Time Step For Load Case " + STR$(Lcase)
  EXIT SUB
END IF

i2 = LC(Lcase).ST(n)
IF Soil(i2).SType = "R" THEN RFlag = -1 ELSE RFlag = 1

i = 1
DO UNTIL i = TotPoints
  INPUT #f(n), vg(n,i)
  vg(n,i) = vg(n,i) * LC(Lcase).Scale(n)
  IF RFlag = -1 THEN
    ag(n,i) = vg(n,i)
    vg(n,i) = 0
  END IF
  INCR i
LOOP
CLOSE f(n)

IF FN(i1).HType = "A" AND RFlag = 1 THEN 'propagate acc. time history thru soil column
  i3 = FREEFILE
  cntl.MaxTime = 2 * TotPoints * (1 + INtstep/OutTstep)
  OPEN RTRIM$(cntl.FileLoc) + cntl.FileNme + "L" + STR$(n) + "S" + STR$(n) + ".S" + LTRIM$(STR$(i2)) FOR OUTPUT AS #i3
  file = FN(i1).FName
  cntl.INtstep = FN(i1).INtstep
  OPEN "Param.out" FOR APPEND AS #20
  SELECT CASE Soil(i2).RFlag
    CASE 1,3
      CALL NonLin1(TotPoints,n,i2,file,Soil()) 'nonlinear or linear response
    CASE 2
      CALL NonLin(TotPoints,n,i2,file,Soil()) 'equivalent stiffness method
    CASE ELSE
      MSGBOX "Invalid Ground Motion Response Model",_,"Error"
      EXIT SUB
  END SELECT
  time = 0
  FOR i = 1 TO TotPoints
    PRINT #i3,time,vg(n,i),ag(n,i)
    time = time + VAL(cntl.INtstep)
  NEXT
  CLOSE #i3
END IF
NEXT

FOR NSup = 1 TO Sup
  FOR i = 1 TO TotModes
    w = Mode(i).w
    zeta = Mode(i).zeta

    nstep = 1
    nstep1 = 1
    j = 1 'counter for number of input load points
    j1 = 2 'counter for number of output time steps
    za = 0 : zv = 0 : zd = 0 'initialize the modal response totals

DO UNTIL j > TotPoints
   eacolx = ceax
   ceax = ag(NSup,j) + 
   (ag(NSup,j+1) - ag(NSup,j))]*(nstep * IGtstep) / INtstep
   dvg = ceax - eacolx

   'Modal response due to support input
   CALL response(NSup,j,IGtstep,ceax,w,zeta,dvg)

   'Calculate total DOF response
   IF nstep*IGtstep >= INTstep THEN 'goto next load increment
   IF nstep1*IGtstep >= OutTstep THEN 'calc and save response values
      FOR NDOF = 1 TO MDOF
         d(NDOF,j1) = d(NDOF,j1) + 
         (zd + vg(NSup,j1*OutTstep/INTstep))*Mode(i).MPF(NSup)*Mode(i).Phi(NDOF) 'Total Displacement Response
         v(NDOF,j1) = v(NDOF,j1) + 
         zv*Mode(i).MPF(NSup)*Mode(i).Phi(NDOF)
         a(NDOF,j1) = a(NDOF,j1) + 
         za*Mode(i).MPF(NSup)*Mode(i).Phi(NDOF)
      NEXT
      INCR j1
      nstep1 = 0
   END IF
   INCR j
   nstep = 0
   END IF
   INCR nstep
   INCR nstep1
END LOOP

PRINT #f(3),"File Name = "+cntl.FileNme
PRINT #f(3),"Time","Rel Disp (DOF 1)","Disp (DOF 1)","Vel (DOF 1)","Acc (DOF 1)","Rel Disp (DOF 2)","Disp (DOF 2)","Vel (DOF 2)","Acc (DOF 2)"

   time = 0
   TotPoints = TotPoints * INtstep / OutTstep
   FOR j = 1 TO TotPoints
      t = FORMAT$(time,"##.###")
      rd = FORMAT$(d(1,j)-vg(1,j*OutTstep/INTstep),"####.#######") 'Relative Displacement - DOF 1
      d = FORMAT$(d(1,j),"####.#######") 'Displacement - DOF 1
      v = FORMAT$(v(1,j),"###.#####") 'Velocity - DOF 1
      a = FORMAT$(a(1,j),"####.#####") 'Acceleration - DOF 1
      rd1 = FORMAT$(d(2,j)-vg(2,j*OutTstep/INTstep),"####.#######") 'Relative Displacement - DOF 2
      d1 = FORMAT$(d(2,j),"####.#######") 'Displacement - DOF 2
      v1 = FORMAT$(v(2,j),"###.#####") 'Velocity - DOF 2
      a1 = FORMAT$(a(2,j),"####.#####") 'Acceleration - DOF 2
      IF ABS(VAL(rd)) > ABS(cntl.MaxRD1) THEN cntl.MaxRD1 = ABS(VAL(rd))
      IF ABS(VAL(rd1)) > ABS(cntl.MaxRD2) THEN cntl.MaxRD2 = ABS(VAL(rd1))
      PRINT #f(3),t,rd,d,v,a,rd1,d1,v1,a1
      time = time + OutTstep
   NEXT
   i = FREEFILE
   OPEN RTRIM$(cntl.FileLoc)+RTRIM$(cntl.FileNme) + ".MAX" FOR APPEND AS i
   PRINT #i,Lcase,cntl.MaxRD1,cntl.MaxRD2
   CLOSE
END SUB
Calculate the modal dynamic response histories

GLOBAL zd AS DOUBLE
GLOBAL zv AS DOUBLE
GLOBAL za AS DOUBLE

SUB Response(sup AS LONG,j0 AS LONG,IGtstep AS DOUBLE,ceacx AS DOUBLE,_
w AS DOUBLE,zeta AS DOUBLE,dvg AS DOUBLE)

LOCAL omega2 AS DOUBLE
LOCAL drel AS DOUBLE
LOCAL deltad AS DOUBLE
LOCAL deltav AS DOUBLE

omega2 = w ^ 2 + 6*(1 + zeta * w * IGtstep) / (IGtstep ^ 2)  'w^2*
drel = -dvg + 6*zv/IGtstep + 3*za + zeta*w*(6*zv + IGtstep*za) 'a*

deltad = drel / omega2  'incremental displacement
deltav = 3 * deltad/IGtstep - 3 * zv - IGtstep * za / 2  'incremental velocity

zd = zd + deltad  'modal displacement @ end of integration time step
zv = zv + deltav  'modal velocity @ end of integration time step

'modal acceleration @ end of time step
za =  - ceacx - (w ^ 2 * zd + 2 * zeta * w * zv)

END SUB