GEBT Manual for Developers[1]

1. Introduction

GEBT can be used as either a callable library or a standalone application, depending on your specific need. A dynamic link library Analysis.dll is provided for it to be used as a callable library. This DLL contains all analysis capabilities of GEBT. For design environment developments, this DLL can act as a plug-n-play black box. What the developers need to provide is the interface so that the DLL can be called and communicated with outside environment.

2. Global Variables needed for GEBT

GlobalDataFun.f90 defines the global variables for GEBT, although they are not passed to/from the DLL. They are necessary for defining the variables passing to/from the DLL. These variables are

- *DBL*: an integer constant to indicate how many digits real numbers should be used. For double precision DBL=8 for single precision DBL=4
- *NDIM*, an integer constant equal to 3
- *NDOF_ND*, an integer constant equal to 12
- *NSTRN*, an integer constant equal to 6
- *MEMB_CONST*, an integer constant equal to 7

3. I/O Variables for Analysis.dll

To take advantage of the analysis modeling capability provided by GEBT, one can call Analysis.dll, which implies the right arguments should pass to and from this DLL. The DLL is invoked as follows (in the format of Fortran 90/95):

```
CALL Analysis(nkp,nelem,ndof_el,nmemb,ncond_pt,nmate, nframe,ndistrfun,ncurv,coord, &
            &  member,pt_condition,material,niter,nstep,sol_pt,sol_mb,error,ncond_mb, &
            &  ntimefun,frame,mb_condition,distr_fun,curvature,omega_a0,omega_a_tf, &
            &  v_root_a0,v_root_a_tf,simu_time,time_function,analysis_flag,init_cond, &
            &  nev,eigen_val,eigen_vec_pt,eigen_vec_mb)
```

with the following variables passed to Analysis.dll

- *nkp* is an integer indicating number of key points.
- *nelem* is an integer number for total number of elements, summation of the divisions for all the members.
- *ndof_el* is an integer number equal to 12 for static analysis, and equal to 18 for all other analyses.
- *nmemb* is an integer number for total number of beam members.
- *ncond_pt* is an integer number for total number of point conditions.

---

[1] You should read GEBT users manual first before you read this manual.

- *nmate* is an integer number for total number of different cross-sections.
- *nframe* is an integer number for total different cross-sectional frames.
- *ndistrfun* is an integer for total number of distribution functions
- *ncurv* is an integer for total number of sets of initial curvature/twists
- *coord* is a 2D real array with dimension as (*nkp,NDIM*). The coordinates of the *i*th point are stored in coord(i,1), coord(i,2), and coord(i,3) respectively.
- *member* is a 2D integer array with dimension as (*nmemb,MEMB_CONST*) holding the seven integers needed for member properties.
- *pt_condition* is a 1D *PrescriInf* type array with dimension *ncond_pt*.
- *material* is a 3D real array with dimensions as *(nmate,ndof_el-NSTRN,NSTRN)*, holding the flexibility matrix at (nmate,1:6,NSTRN), and the mass matrix at (nmate,7:12,NSTRN). For static analysis, only flexibility matrix is needed.
- *niter* is an integer indicating maximum number of iterations. For linear analysis, it is equal to 1.
- *nstep* is an integer indicating number of time steps.
- *ncond_mb* is an integer indicating number of prescribed distribution
- *ntimefun* is an integer indicating the number of time functions.
- *frame* is a 3D real array with dimensions as *(nframe,NDIM,NDIM)*, holding the 3x3 direction cosine matrix for each different cross-sectional frame.
- *mb_condition* is a 1D *PrescriInf* type array with dimension as *ncond_mb*.
- *distr_fun* is a 2D real array with dimensions as *(ndistrfun,NSTRN)*, holding the six parameters for each distribution function.
- *curvature* is a 2D real array with dimensions as *(ncurv,NDIM)*, holding the three parameters (k1, k2, k3) for each set of initial twist and curvature.
- *omega_a0* is a 1D real array with dimension as *NDIM*, holding the angular velocity of the *a* frame.
- *omega_a_tf* is a 1D integer array with dimension as *NDIM*, holding the corresponding time function for angular velocity of the *a* frame.
- *v_root_a0* is a 1D real array with dimension as *NDIM*, holding the linear velocity of the *a* frame.
- *v_root_a_tf* is a 1D integer array with dimension as *NDIM*, holding the corresponding time function for linear velocity of the *a* frame.
- *simu_time* is a 1D real array with two components storing the starting and ending time of the simulation.
- *time_function* is a 1D *TimeFunction* type array with dimension as *ntimefun*.
- *analysis_flag* is an integer, equal to 0 for static analysis, equal to 1 for steady state response, equal to 2 for transient analysis, and equal to 3 for eigenvalue analysis
- *init_cond* is a 2D real array with dimensions as *(nelem,12)*, holding initial displacements/rotations (nelem,1:6) and initial linear and angular velocity (nelem,7:12) for each element. Note this array might be modified by Analysis.dll.
- *nev* is an integer indicating the number of eigenvalues to be calculated. Note this variable might be modified to be *nev*+1 by Analysis.dll.
-

The following variables are passed from Analysis.dll

- *sol_pt* is a 3D real array with dimensions as *(nstep,nkp,NDIM+NDOF_ND)*. For each *i*th step, sol_pt(i,j,1:3) stores the coordinates, sol_pt(i,j,4:9) stores the displacements/ rotations, sol_pt(i,j,10:15) stores the forces/moments, sol_pt(i,j,16:21) stores linear/angular momentum for *j*th key point. Note only sol_pt(i,j,1:15) is available for static analysis.

- *sol_mb* is a 3D real array with dimensions as *(nstep,nelem,NDIM+ndof_el)*. For each *i*th step, sol_mb(i,j,1:3) stores the mid-point coordinates, sol_mb(i,j,4:9) stores the displacements/ rotations, sol_mb(i,j,10:15) stores the forces/moments, sol_mb(i,j,16:21) stores linear/angular momentum for *j*th element. Note only sol_mb(i,j,1:15) is available for static analysis.
- *error* is a character variable with length 300 to store the error message of the program.
- *eigen_val* is a 2D real array with dimensions as *(2,nev+1)*, where (1,i) and (2,i) stores the real and imaginary parts of the *i*th eigenvalue.
- *eigen_vec_pt* is a 3D real array with dimensions as *(nev+1,nkp,NDIM+NDOF_ND)*, storing the corresponding component of the eigenvectors for each key point, the storage is the same as sol_pt.
- *eigen_vec_mb* is a 3D real array with dimensions as *(nev+1,elem,,NDIM+NDOF_ND)*, storing the corresponding component of the eigenvectors for each element, the storage is the same as sol_mb.

4. Definition of two types

There are two types defined for GEBT: *PrescriInf* and *TimeFunction*. *PrescriInf* is defined in Fortran 90/95 as

```
TYPE PrescriInf
    PRIVATE
      INTEGER   ::id
      INTEGER   ::dof(NSTRN)
      REAL(DBL) ::value(NSTRN)
      INTEGER   ::time_fun_no(NSTRN)
      INTEGER   ::follower(NSTRN)
      REAL(DBL) ::value_current(NSTRN)
END TYPE PrescriInf
```

- *id* is an integer indicates where it is applied, could be a node # or member #.
- *dof* is an integer array of dimension NSTRN, storing the prescribed degrees of freedom. It is also used to store the # of distribution function for each degree of freedom.
- *value* is a real array of dimension NSTRN, storing the magnitude of the prescribed values.
- *time_fun_no* is an integer array of dimension NSTRN, storing the # of time function for each prescribed degrees of freedom.
- *follower* is an integer array of dimension NSTRN, indicating whether the prescribed condition is a follower condition or not.
- *value_current* is a real array of dimension NSTRN, storing the time updated values of each prescribed condition.

```
TYPE TimeFunction
    PRIVATE
         INTEGER   ::fun_type
         REAL(DBL) ::ts, te
         INTEGER   ::entries
         REAL(DBL),POINTER::time_val(:)
         REAL(DBL),POINTER::fun_val(:)
```

```
          REAL(DBL),POINTER::phase_val(:)
END TYPE TimeFunction
```

- *fun_type* is an integer indicates the time function type, could be 0 for user defined time function or 1 for harmonic time function.
- *ts* is a real number indicating the starting time and *te* is a real number indicating the ending time for the time function definition.
- *entries* is an integer number indicated how many entries are used for defining the time function.
- *time_val* is pointer for a real array of dimension *entries*. For user defined time function, it stores the time in increasing order. For harmonic time function, it stores the amplitude.
- *fun_val* is pointer for a real array of dimension *entries*. For user defined time function, it stores the functional value. For harmonic time function, it stores the period.
- *phase_val* is pointer for a real array of dimension *entries*. It is only needed for harmonic time function storing the phase of the harmonic function.

5. Standalone Application

The standard release includes the standalone application including Constitutive.dll and Recovery.dll and the files needed to compile the standalone application including CPUtime.f90, main.f90, IO.f90, GlobalDataFun.f90, PrescribedCondition.f90, and TimeFunction.f90. Interface are provided in main.f90 so that the DLL can be called properly. IO.f90 defines/inputs/outputs all the arguments needed to pass to/from the DLL. GlobalDataFun.f90 defines some global constants and functions needed for IO.f90. If you are familiar with Fortran language, these files might be able to facilitate your development. The developers are also free to modify the source codes to add more capabilities which are design to take advantage of GEBT analysis incarnated in Analysis.dll.

If the variables as explained previously are defined correctly, the remaining task for the developer to integrate GEBT is to provide the interface necessary for calling the DLL. For example, for Fortran 90/95, the needed interface is provided in main.f90.