

VAMUCH Manual for Developers¹

1. Introduction

VAMUCH can be used as either a callable library or a standalone application, depending on your specific need. The program flow of both applications is the same and is illustrated in Figure 1.

The yellow boxes are subroutines handling I/O needed for VAMUCH. First VAMUCH needs to read the finite element mesh of the microstructure which is generated by a preprocessor (the first yellow box). The inputs should include problem control parameters, mesh control parameters, nodal coordinates, elemental connectivity, and material properties. If it is for recovery, VAMUCH should also read the macroscopic information calculated by the structural analysis and the fluctuating functions. The outputs of the constitutive modeling include effective properties and corresponding fluctuating functions. The outputs of the recovery are distribution of microfields within the microstructure.

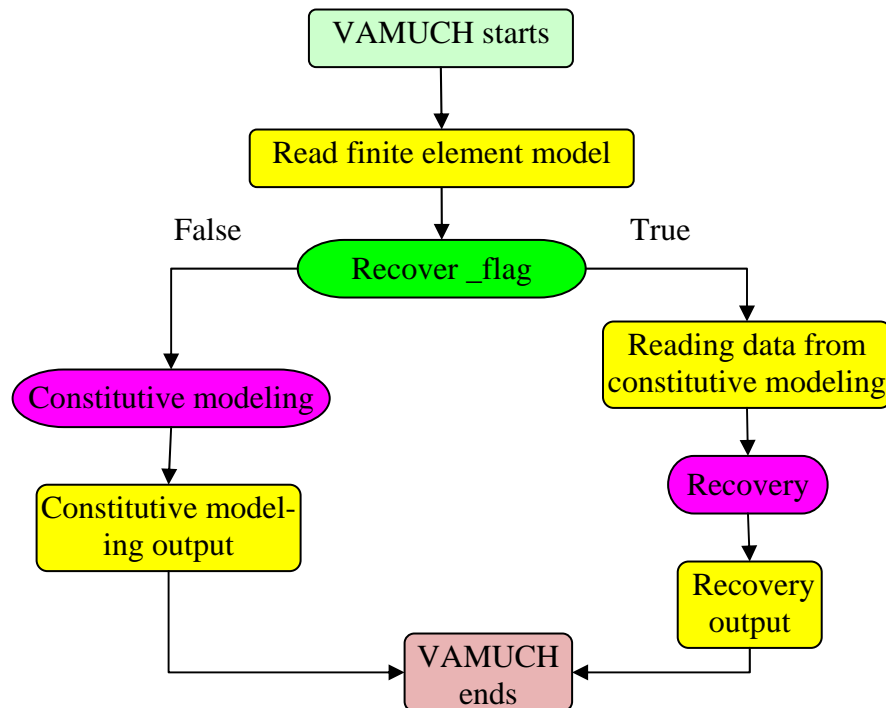


Figure 1. VAMUCH code structure

¹ You should read VAMUCH users manual first before you read this manual.

The red boxes including the constitutive modeling and the recovery are handled by two separated Dynamic Link Libraries (DLLs) called Constitutive.dll and Recovery.dll respectively. These two DLLs contain all analysis capabilities of VAMUCH. For design environment developments, these two DLLs like two plug-n-play black boxes. What the developers need to provide are Interfaces so that these two DLLs can be called and communicated with outside environment.

2. Global Variables needed for VAMUCH

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

- *allo_stat*: an integer variable to indicate status of allocating memory
- *in_stat*: an integer variable to indicate status of I/O process
- *DBL*: an integer constant to indicate how many digits real numbers should be used. For double precision DBL=8 for single precision DBL=4
- *TOLERANCE*: a real constant to simulate a small number
- *PI*: a real constant for π
- *DEG_2_RAD*: a real constant for $\pi/180$
- *FMT_REAL*: the standard output format for a real number: 'ES15.7'
- *FMT_INT*: the standard output format for an integer number: 'I8'

3. I/O Variables for Constitutive.dll

To take advantage of the constitutive modeling, one needs to call Constitutive.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 ConstitutiveModeling (ndim,analysis,max_node_elem,ndof_node,nstr_3D,nnode,nelem,&
& nmate,coord,mat_type,element,orth,material,ref_temp,nsiz,&
& V0,VT,Deff,Deff_F,Beff,Cveff,error)
```

with the following variables passed to Constitutive.dll

- *ndim* is an integer indicating the dimension of the microstructure.
- *analysis* is an integer indicating the analysis type.
- *max_node_elem* is an integer indicating the max # of nodes allowed in the element: 2 for 1D UCs; 9 for 2D UCs; 20 for 3D UCs.
- *ndof_node* is an integer indicating the # of dofs per node: 3 for thermoelastic analysis, 1 for conduction analysis
- *nstr_3D* is an integer indicating the # of strain variables (or temperature gradient): 6 for thermoelastic analysis, 3 for conduction analysis
- *nnode* is an integer number for total number of nodes.
- *nelem* is an integer number for total number of element.
- *nmate* is an integer number for total number of material types.
- *coord* is a 2D real array with dimension as (*nnode*, *ndim*). Note these values are

changed after execution due to optimizing the input mesh.

- *mat_type* is a 1D integer array with dimension as *nelem* holding the material type for each element.
- *element* is a 2D integer array with dimension as (*nelem*, *max_node_elem*) holding the nodes needed for elemental connectivity. Note these values are changed after execution due to optimizing the input mesh.
- *orth* is a 1D integer array with dimension as *nmate* holding an integer to indicate the anisotropy of the material.
- *material* is a 2D real array with dimension as (*nmate*,28) holding up to 28 real numbers for the thermoelastic constants of each material.
- *ref_temp* is a real number for the reference temperature when the material is stress free.
- *nsize* is an integer number for the total dofs of the problems: 3*nnode

The following variables are passed from Constitutive.dll

- *V0* is a 2D real array with dimension as (*nsize*,*nstr_3D*) storing the fluctuating function.
- *VT* is a 1D real array with dimension as *nsize* storing the fluctuating function due to temperature.
- *Deff*, *Deff_F* are two real arrays with dimension as (*nstr_3D*, *nstr_3D*) storing the effective stiffness (conductivity matrix and the corresponding compliance matrix.
- *Beff*, *is* one real array with dimension as (*nstr_3D*) storing the effective thermal expansion coefficients.
- *Cveff*, *is* a real number storing the effective specific heat.
- *error* is a character variable with length 300 to store the error message of the program.

4. I/O Variables for Recovery.dll

To use the recovery capability of VAMUCH, one needs to call Recovery.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 Recovery(ndim,analysis,max_node_elem,ndof_node,nstr_3D,nnode,nelem,nmate,coord,&  
  & mat_type,element,orth,material,nsize,max_gp,V0,VT,macro_u,macro_der,macro_T,&  
  & k_F,nd_F,disp_3D_F,ss_F,ss_nd_F,ss_elem,error)
```

where *ndim*, *analysis*, *max_node_elem*, *ndof_node*, *nstr_3D*, *nnode*, *nelem*, *nmate*, *coord*, *mat_type*, *element*, *orth*, *material*, *nsize*, *max_gp*, *V0*, *VT*, *macro_u*, *macro_der*, *macro_T* are inputs to this DLL. In addition to those variables common to Constitutive.dll, the additional variables are defined as follows:

- *macro_u* is a real array of dimension (*ndof_node*) storing the macroscopic displacements/temperature
- *macro_der* is a real array of dimension (*ndof_node*,3) storing the gradient of the macroscopic displacements/temperature
- *Macro_T* is the macroscopic temperature

The following variables are passed from Recover.dll

- k_F is an integer number used to indicate the total number of Gaussian points we have recovered the 3D stresses and strains
- nd_F is an integer number used to indicate the total number of nodes we have recovered the 3D stresses and strains
- $disp_{3D_F}$ is a 2D real array with dimension as $(nnode, ndim+ndof_node)$ storing the recovered 3D displacements.
- ss_F is a 2D real array with dimension as $(nelem*max_gp, ndim+2*nstr_{3D})$ storing the recovered 3D stresses and strains at Gaussian points, where $ss_F(:,1:ndim)$ denotes the position, $ss_F(:,ndim+1:ndim+nstr_{3D})$ denotes the strains, $ss_F(:,ndim+nstr_{3D}+1:ndim+2*nstr_{3D})$ denotes the stresses. Only the first k_F rows are meaningful.
- ss_{nd_F} is a 2D real array with dimension as $(nelem*max_node_elem, ndim+2*nstr_{3D})$ storing the recovered 3D stresses and strains at nodal points, where $ss_{nd_F}(:,1:ndim)$ denotes the position, $ss_{nd_F}(:,ndim+1:ndim+nstr_{3D})$ denotes the strains, $ss_{nd_F}(:,ndim+nstr_{3D}+1:ndim+2*nstr_{3D})$ denotes the stresses. Only the first nd_F rows are meaningful.
- ss_{elem} is a 2D real array with dimension as $(nelem, 2*nstr_{3D})$ storing the average of recovered 3D stresses and strains of all the Gaussian points within one element, where $ss_{elem}(:,1:nstr_{3D})$ denotes the strains, $ss_{elem}(:,nstr_{3D}+1:2*nstr_{3D})$ denotes the stresses. This array can be used to facilitate contour plot for visualization.

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, and GlobalDataFun.f90, in addition to the two DLLs. Interfaces are provided in main.f90 so that the two DLLs can be called properly. IO.f90 defines/inputs/outputs all the arguments needed to pass to/from the two DLLs. 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 VAMUCH incarnated in Constitutive.dll and Recovery.dll.

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