

---

# MANUAL FOR THE “Y3D” FEM/DEM COMPUTER PROGRAM

JIANSHENG XIANG, ANTONIO MUNJIZA

## 1 INTRODUCTION

Y3D is a three dimensional combined finite-discrete element method program by Prof Antonio Munjiza & Dr Jiansheng Xiang. This code is part of the Virtual Geoscience Workbench (VGW) developed jointly by Imperial College of Science, Technology and Medicine (ICSTM) & Queen Mary University of London (QMUL) through two related parallel projects at ICSTM and QMUL respectively funded by EPSRC.

The Copyright (C) 2008, to this program (Y3D program) and source code supplied with it belongs to Queen Mary University of London (QMUL) & Imperial College of Science, Technology and Medicine (ICSTM). All rights reserved.

**Source code is provided, however you cannot use it unless you agree with the following:**

---

Copyright (C) 2008,  
Queen Mary University of London (QMUL) & Imperial College of Science, Technology and Medicine (ICSTM).  
All rights reserved.  
Implemented for you by Prof Antonio Munjiza & Dr Jiansheng Xiang.

This code is part of the Virtual Geoscience Workbench (VGW) developed jointly by ICSTM and QMUL through two related parallel projects at ICSTM and QMUL respectively funded by EPSRC.

This code is provided by copyright holders under the GNU Lesser General Public License (LGPL). It is open source code; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License version 3.

This code is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details, <http://www.gnu.org/licenses/lgpl.txt>.

You should have received a copy of the GNU Lesser General Public License along with this code; if not, write to Dr Jiansheng Xiang Prof Antonio Munjiza or Dr John-Paul Latham [j.xiang@imperial.ac.uk](mailto:j.xiang@imperial.ac.uk) [a.munjiza@qmul.ac.uk](mailto:a.munjiza@qmul.ac.uk) or [j.p.latham@imperial.ac.uk](mailto:j.p.latham@imperial.ac.uk)

---

## 2 INPUT FILE

### Data convention:

- 1) Variables names starting with M and N are integer numbers indicating maximum and actual size of arrays used to construct the incore database.
- 2) Variables names starting with I are integer numbers.
- 3) Variables names starting with D are double numbers.
- 4) Variables names starting with D1 and I1 indicate one dimensional array of double and integer numbers respectively.
- 5) Variables names starting with D2 and I2 indicate two dimensional array of double and integer numbers respectively.

*Jiansheng Xiang, Antonio Munjiza*

Copyright (C) 2008, Queen Mary University of London (QMUL) & Imperial College of Science, Technology and Medicine (ICSTM).

### One dimensional arrays:

A one dimensional array I1ANYARRAY[10] comprising of 10 integer numbers is supplied in an input file as follows:

```
/YD/YDE/I1ANYARRAY 10
1 2 3 4 5 6 7 8 9 0
```

where 10 indicates that the name is followed by 10 integer numbers. I1ANYARRAY[0] to I1ANYARRAY[9].

In a similar way, a one dimensional array of double numbers D1ANYARRAY[10] is supplied in an input file as follows:

```
/YD/YDE/D1ANYARRAY 10
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.0
```

where 10 indicates that the name is followed by 10 double numbers. D1ANYARRAY[0]=0.1 to D1ANYARRAY[9]=0.0

### Two dimensional arrays:

A two dimensional array I2ANYARRAY[2][5] comprising of two rows and five columns is supplied in an input file as follows:

```
/YD/YDE/I2ANYARRAY 12 2 5
1 2 3 4 5
21 22 23 24 25
```

where 12 indicates that the arrays is read row by row (i.e. the rightmost index rotates faster than the leftmost index). Thus I2ANYARRAY[0][0]=1, I2ANYARRAY[0][1]=2, I2ANYARRAY[0][2]=3, etc. Also I2ANYARRAY[1][0]=21, I2ANYARRAY[1][1]=22, etc.

Alternatively the same array can be written as follows:

```
/YD/YDE/I2ANYARRAY 21 2 5
1 21
2 22
3 23
4 24
5 25
```

The first number following the name is 21 which indicates that index 1 (the leftmost index) rotates fastest and index 2 (the rightmost index) rotates slowest. Thus I2ANYARRAY[0][0]=1, I2ANYARRAY[0][1]=2, I2ANYARRAY[0][2]=3, etc. Also I2ANYARRAY[1][0]=21, I2ANYARRAY[1][1]=22, etc.

Names of all the variables must be present in the input file (data need not be supplied for variables printed in italic) as shown in the following table:

VARIABLE	DESCRIPTION
<i>/* comment */</i>	Anything between <i>/*</i> and <i>*/</i> is ignored, i.e. is considered as a comment (do not forget to put space before and after the <i>*/</i> )
<i>/YD/YDC/MCSTEP</i>	Maximum number of time steps
<i>/YD/YDC/NCSTEP</i>	Actual number of time steps (cannot be greater than <i>/YD/YDC/MCSTEP</i> )
<i>/YD/YDC/DCGRAX</i>	Acceleration of gravity (in x direction)
<i>/YD/YDC/DCGRAY</i>	Acceleration of gravity (in y direction)
<i>/YD/YDC/DCGRAZ</i>	Acceleration of gravity (in z direction)
<i>/YD/YDC/DCSIZEC</i>	Maximum size of coordinate in any direction (size of physical space – corresponding outputs are normalized using this value)
<i>/YD/YDC/DCSIZF</i>	Maximum size of force in any direction (corresponding outputs are normalized

	using this value)
/YD/YDC/DCSIZV	Maximum size of velocity in any direction (corresponding outputs are normalized using this value)
/YD/YDC/DCSTEC	Size of the time step (see in the book how to calculate critical-maximum time step)
/YD/YDC/DCTIME	Current time. i.e. time at start of this run.
/YD/YDC/DCURELX	The under relaxation factor for mass matrix
/YD/YDC/INITER	the number of iteration for multi-pass algorithm
/YD/YDC/ICOUTF	Output frequency – every so many time steps complete state of the system is recorded in a file with extension .ym which can be visualized using M program, which is FEM/DEM Visualizer accompanying Y program.
/YD/YDC/ICOUTI	Serial number of first output associated with this run.
/YD/YDC/ICOUTP	Number of characters for each number in output file (for example: three characters is equivalent to six significant digits)
/YD/YDC/IWFAST	0. slow mode, 1. fast mode
/YD/YDE/MELEM	Maximum number of finite elements (with fracture the actual number of finite elements increases during the run, however, it should not exceed this number)
/YD/YDE/NELEM	Actual number of finite elements at the beginning of this run.
/YD/YDE/MELST	Maximum number of state variables per finite element.
/YD/YDE/NELST	Actual number of state variables per finite element.
/YD/YDE/MELNO	Maximum number of nodes per finite element.
/YD/YDE/NELNO	Actual number of nodes per finite element.
/YD/YDE/D2ELST	[MELST][MELEM] array containing state variables for all finite elements.
/YD/YDE/I1ELCF	Head of a list of contacting couples for each finite element.
/YD/YDE/I1ELPR	Set of properties associated with each element.
/YD/YDE/I2ELTO	[MELNO][MELEM] topology array containing nodes for each finite element.
/YD/YDE/D3TCS	[ndime][ndime][melem] -Cauchy stress for each finite element.
/YD/YDE/D1EMCT	[melem] total elemental mass for each finite element.
/YD/YDE/I1ELBE	[melem] array containing a flag indicating that a element is on the boundary (usually set to non-zero for all elements indicating the boundary. Any element with the flag set to zero is considered not to be on the boundary)
/YD/YDI/MICOUP	Maximum number of contacting couples of finite elements.
/YD/YDI/NICOUP	Actual number of contacting couples of finite elements (always set to zero)
/YD/YDI/IIECFE	Internal variable used for contact (always set to -2)
/YD/YDI/DIEDI	Internal variable (travel since last detection) used to trigger contact detection (always set to a large number in order to trigger contact detection immediately at the start of this run)
/YD/YDI/DIEZON	Size of the buffer around each finite element for contact detection purposes (usually 1/5 of the size of the smallest finite element)
/YD/YDI/D1IESL	[MELEM] array used to store sliding distance between couples in contact (usually set to zero)
/YD/YDI/I1IECN	[MICOUP] array used to store next contacting couple in the list.
/YD/YDI/I1IECT	[MICOUP] array used to store target finite element for each contacting couple.

/YD/YDN/MNODIM	Maximum number of degrees of freedom per node (usually 2 in 2D)
/YD/YDN/NNODIM	Actual number of degrees of freedom per node (usually 2 in 2D)
/YD/YDN/MNOPO	Maximum number of nodes.
/YD/YDN/NNOPO	Actual number of nodes.
/YD/YDN/D2NCC	[MNODIM][MNOPO] array containing the current coordinates of the nodes.
/YD/YDN/D2NCI	[MNODIM][MNOPO] array containing the initial coordinates of the nodes (usually corresponding to undeformed system)
/YD/YDN/D2NFC	[MNODIM][MNOPO] array containing the current nodal forces due to contact.
/YD/YDN/D2NFT	[MNODIM][MNOPO] array containing the current total nodal forces.
/YD/YDN/D1NMCT	[MNOPO] array containing the current mass for each node.
/YD/YDN/D2NVC	[MNODIM][MNOPO] array containing the current velocities for each node.
/YD/YDN/I1NOBF	[MNOPO] array containing a flag indicating that a node is on the boundary (usually set to 1 for all nodes indicating the boundary. Any node with the flag set to zero is considered not to be on the boundary)
/YD/YDN/I1NOPR	[MNOPO] array containing an ID of a property set associated with each node.
/YD/YDP/MPROP	Maximum number of properties sets.
/YD/YDP/NPROP	Actual number of properties sets.
/YD/YDP/D1PEFR	[MPROP] array containing sliding friction coefficient
/YD/YDP/D1PEKS	[MPROP] array containing viscous damping of material for finite elements for each of the properties (for D1PEKS equal $2h\sqrt{E\rho}$ , finite element smaller than $h$ is critically damped)
/YD/YDP/D1PELA	[MPROP] array containing Lamé elastic constant. (l)
/YD/YDP/D1PEMU	[MPROP] array containing Lamé elastic constant. (m)
/YD/YDP/D1PEPE	[MPROP] array containing penalty term for contact (usually 2 to 100 times greater than lambda)
/YD/YDP/D1PERO	[MPROP] array containing density.
/YD/YDP/I1PTYP	[MPROP] array containing the type of each property. Indicates type of object to which this property is associated.
/YD/YDB/MBCON	Maximum number of boundary conditions sets.
/YD/YDB/NBCON	Actual number of boundary conditions sets.
/YD/YDB/D1BNAX	[MPROP] array containing amplitude of acceleration in local x direction for each node.
/YD/YDB/D1BNAY	[MPROP] array containing amplitude of acceleration in local y direction for each node.
/YD/YDB/D1BNAZ	[MPROP] array containing amplitude of acceleration in local z direction for each node.
/YD/YDB/D1BNFX	[MPROP] array containing amplitude of force in local x direction for each node.
/YD/YDB/D1BNFY	[MPROP] array containing amplitude of force in local y direction for each node.
/YD/YDB/D1BNFZ	[MPROP] array containing amplitude of force in local z direction for each node.
/YD/YDB/D1BNVX	[MPROP] array containing amplitude of velocity in local x direction for each node.
/YD/YDB/D1BNVY	[MPROP] array containing amplitude of velocity in local y direction for each node.
/YD/YDB/D1BNVZ	[MPROP] array containing amplitude of velocity in local z direction for each

*Jiansheng Xiang, Antonio Munjiza*

Copyright (C) 2008, Queen Mary University of London (QMUL) & Imperial College of Science, Technology and Medicine (ICSTM).

	node.
/YD/YDB/I1BNVX	[MPROP] array (if >0) indicates that a velocity is applied to a node in local x direction.
/YD/YDB/I1BNVY	[MPROP] array (if >0) indicates that a velocity is applied to a node in local y direction.
/YD/YDB/I1BNVZ	[MPROP] array (if >0) indicates that a velocity is applied to a node in local z direction.
\$YDOIT	It is a command to execute the program.
\$YSTOP	It is a command to stop the program.

### 3 REMARKS

Y3D computer program is a research code. As such it does not make any checks of the input data and will usually crash if the input file is not compiled properly. It is especially sensitive to variables starting with letters M and N, which control dynamic-array- sizes. However, should any of the variables necessary for the processing of the job be missing, memory allocation for that variable may not be performed and a usual result is a pointer-type crash. We advise that file generation be taken in stages from the file which runs smoothly. Should the program run for a few time steps, it usually finishes the whole job. Thus, it is a good idea to run a job for say 10 time steps after every small modification of input file in order to run a short “test” that it is correct. Should variable names be wrong in the input file, all variable names before the wrong one are recorded in the file Ytmp. By looking at this file, one can see what variables Y has accepted.