DROG3D

User's Manual for 3-Dimensional Drogue Tracking on a Finite Element Grid with Linear Finite Elements.

written by

Brian O. Blanton

Ocean Processes Numerical Methods Laboratory
Curriculum in Marine Science
University of North Carolina at Chapel Hill
15 - 1 Venable Hall
CB #3300
Chapel Hill, NC 27599-3300

Release 95.1
Spring 1995

email: blanton@marine.unc.edu

Contents

1 Introduction

This document describes the implementation of *DROG3D*, Release 95.1, a 3-dimensional particle tracking algorithm written in FORTRAN 77. It builds on previous documents and code versions and is not intended to be "backward compatible" with the earlier releases. *DROG3D* tracks passive drogues with given harmonic velocity field(s) in a 3-D finite element mesh. The (x,y) elements are linear triangular elements; the interpolation in the vertical is also linear.

Earlier releases of *DROG3D* allowed the inclusion of multiple frequencies per simulation. Release 95.1 is enhanced to allow the simulation of multiple legs, each with multiple frequencies, enabling for example the computation of particle trajectories within evolving seasonal-mean or monthly-mean flow fields in one run. Thus, if flow fields corresponding to monthly variations in wind forcing are available, Release 95.1 of *DROG3D* will read in those files and compute particle trajectories using the sequence of flow fields specified by the user in a single *.din* file. Each set of velocity components and time parameters, called a "leg", is run sequentially. The code internally updates velocity fields at the conclusion of the current leg. This obviates the need for the user to externally create a new *.din* file, with the final drogue positions from the previous leg as starting positions for the next leg, and then restart the code using the new input parameters. The multiple-leg specification of input information occurs in the *.din* file and is described on pages 5-8 of this document.

2 Source codes, input and output files

The files provided in this distribution are:

connect2d.f
DROG3D.f
INPUT.f
SUBROUTINES.f
OUTPUT.f
ZPOSITION.f
COMMON_BLOCK.h
Makefile

The user must provide:

.nod	- node-list file
.ele	- element-list file
.gr2	- domain information file
.din	- tracking run parameters file
.vel	- velocity components file(s)

Output files are:

.pth	- drogue path file
.diag	- diagnostic report file

2.1 Steps to run *DROG3D*

To run *DROG3D* (ensure *COMMON_BLOCK.h* is in the same directory as codes):

1. generate *.gr2* file using *connect2d.f*
2. compile *DROG3D* with *Makefile*; type "`make DROG3D`"
3. specify run parameters in *.din* file
4. run *DROG3D*; the executable is named *DROG3D*

2.2 File Naming Convention

In the text that follows, the convention used when referring to, for example, the file *X.gr2* is that it has a filename *X* and a filetype *.gr2*.

2.3 UNITS

DROG3D makes no assumption about the unit system used to compute the velocity fields or to track drogues. Scaling factors for node coordinates, drogue positions, and velocities are provided in the *.din* file in case of unit mismatch between any variables. Thus, if all variables are MKS, the scaling factors are all 1.

3 *DROG3D* Input Files Structure

There is one `include` file needed by *DROG3D* at compile-time:

- *COMMON_BLOCK.h* - array dimension include file

There are three run-time input files to *DROG3D*, each having a unique suffix.

1. *.gr2* - File describing finite-element domain; generated by *connect2d.f*;
2. *.din* - File containing run parameters and initial drogue positions;
3. *.vel* - File containing velocity components for entire domain at each node.

Upon execution of *DROG3D*, the user is prompted **ONLY** for the filename of the *.din* file; i.e., if the input file is *case1.din*, type only `case1`. The *.din* file contains the name of the *.vel* files to be used in the run, all scaling factors for coordinates and velocities, and time parameters describing the particular tracking run. The name of the *.din* file may be any alphanumeric sequence shorter than 72 characters. The first record of each *.vel* file is expected to contain the name of the finite element grid on which the velocities were computed and thus *DROG3D* opens the appropriate *.gr2* file.

3.1 *COMMON_BLOCK.h*

The file *COMMON_BLOCK.h* is an include file that contains array initialization parameters for *DROG3D*. It must be located in the same directory in which *DROG3D* is compiled. The following parameters are set in this include file (distribution values are in ()):

ND - max number of horizontal nodes (4000)
NNE - max number of horizontal elements (7000)
NLVL - max number of vertical levels (21)
NLD - max number of land boundaries; not used by code at present (1)
NFR - max number of velocity components per leg (2)
NDROG - max number of drogues in tracking run (1500)
NLEG - max number of legs to be run (5)

3.2 The *.gr2* file and *connect2d.f*

connect2d.f is a pre-processing program that generates the *.gr2* file, which contains all domain (*x,y*) node coordinates and element connectivity information. *connect2d.f* prompts the user for a filename containing horizontal node coordinates and for a filename containing the nodes that comprise each element (an element list). It then prompts the user for the name of the finite element grid on which the velocity fields were computed. This filename of the *.gr2* file **MUST** match the domain name specified in the first record of the velocity files.

connect2d.f must be run before *DROG3D*, but need only be run once for each domain used. All vertical node information is contained in the velocity files, including the number of vertical nodes, and is thus not part of the *.gr2* file (see the *.vel* filetype description).

The FORTRAN statements that *connect2d.f* uses to input the node-coordinate file are as follows:

```
DO I=1,NMND
  READ(8,*,END=24) N,XND(N),YND(N)
END DO
```

where N is the node number, XND and YND are arrays containing *x*- and *y*- coordinates of node N, and NMND is the number of nodes in the node file. The FORTRAN statements that *connect2d.f* uses to input the element-list file are as follows:

```
DO I=1,NMEL
  READ(9,*,END=75) IE,ELEMS(IE,1),ELEMS(IE,2),ELEMS(IE,3)
END DO
```

where IE is the element number, the matrix ELEMS is a list of the three nodes which comprise each element, and NMEL is the number of elements in the element file. The node numbering for each element is expected to be in counterclockwise order. Both formats above conform to Numerical Methods Laboratory: Memorandum on Data File Standards for the Gulf of Maine Project, March 29, 1993, by Christopher Naimie.

3.3 The *.din* file

The *.din* file is created by the user and contains all velocity component information and tracking parameters for all legs, scaling factors (for velocities, nodes and drogues), error criteria, and initial drogue coordinates. The specific format of each line in the *.din* file is given on pages 7-8. Most records in the *.din* file are preceded by one comment line describing the information to follow. Although the specific words in the comment line are unimportant, the code DOES expect the comment to be there. The first record of the *.din* file is a comment line, followed by the number of legs (*nleg*) in the tracking run. Then, FOR EACH LEG, the following structure is expected:

- A comment line denoting the leg number for the following information;
- A line giving leg number (*lgn*), run length (*tpath*) in hours, start time (*tstart*) in hours, the number of time steps in the leg (*ntint*), and the number of velocity components in this leg (*ncomp*). See **NOTE** below;
- *ncomp* lines naming the velocity components for this leg;
- *ncomp* lines giving the "on/off" index *indcomp* and six scaling factors for each component specified. If $indcomp = 0$, the component will not be used in the computation.

The names of the velocity components can contain a valid path, absolute from / (root) or relative to the current working directory, to the file. Once the list of velocity files is read in and before execution of the first leg begins, a check is made to find the specified files. If any files are not found, the code stops and reports the missing files to the *.diag* file. This prevents the premature termination of a run due to a non-existent velocity file, the cause usually being incorrect spelling. This check is for the existence of the files only, not for the correctness of the content of the files.

Each remaining line in the *.din* file specifies information pertaining to all legs. The variable *iprint* denotes how often to report drogue position information to the *.pth* file. If *iprint* is set to *10* then every tenth timestep is output. Scaling factors for the horizontal and vertical node locations and the (*x,y,z*) starting positions for all drogues should be set to 1.0 unless there is unit mismatch between grid coordinates and drogue positions. The horizontal and vertical errors should be in grid coordinate units, and the minimum timestep *dtmin* must be in hours.

NOTE: The specification of *tpath* and *ntint* allows *DROG3D* to compute the timestep as *tpath/ntint*. In a multiple-leg run, the timestep for the entire run is calculated from *tpath* and *ntint* as specified for the first leg. The remaining legs MUST have timesteps equal, to the fourth decimal place, to the timestep calculated for the first leg. If this is NOT the case, the code stops immediately and reports the leg number and problem to the *.diag* file. This check has been implemented to ensure that the time interval between outputs of the drogue tracks, as specified by *iprint*, remains constant between legs.

A 2-leg example of the *.din* file follows. In the first leg, there are two velocity components (two *.vel* files), `comp1` and `comp2`. Both files reside in the current working directory. Leg 2 has one component, called `comp3`, in the directory `/users/john_doe/velocities`. The comments in () below should **NOT** appear in a real *.din* file.

```
Specify number of legs in run                                  (comment line)
2                                                              (nleg)
LEG 1 PARAMETERS                                               (leg 1 comment line)
1       1.0    0.0    10     2                                 (lgn, tpath, tstart, ntint, ncomp)
comp1                                                          (velocity component list for leg 1)
comp2
1       1.0    1.0    1.0    1.0    1.0    1.0                 (index and scaling factors for comp1)
1       1.0    1.0    1.0    1.0    1.0    1.0                 (index and scaling factors for comp2)
LEG 2 PARAMETERS                                               (leg 2 comment line)
2       1.0    1.0    10     1                                 (lgn, tpath, tstart, ntint, ncomp)
/users/john_doe/velocities/comp3                               (velocity component list for leg 2)
1       1.0    1.0    1.0    1.0    1.0    1.0                 (index and scaling factors for comp3)
END OF LEG INFORMATION                                         (end of leg comment line)
Specify iprint, the number of timesteps between outputs
1
Scaling factors for grid in x,y,z directions
1.00    1.00   1.00
Specify horizontal error, vertical error, and minimum time step in hours
0.10    0.01   0.01
Specify scaling factors for drogue coordinates in x,y,z directions
1.00    1.00   1.00
Specify number of starting drogues, ndr
1
Specify ndr starting positions (x,y,z)
25000. -99000. -5.
```

The following is a complete outline for the sample *.din* file and the FORTRAN READ statements from *INPUT.f*. Refer to the previous example *.din* file for comparison:

comment - comment for number of legs; not used in computations
READ (11,'a') COMMENT

nleg - number of legs in tracking run
READ (11,) NLEG*

comment - leg number 1 comment; not used in computations
READ (11,'a') COMMENT

lgn,tpath,tstart,ntint,ncomp - leg number,
tracking length of leg 1 (hours),
time at start of leg 1 (hours),
number of time steps in leg 1,
number of velocity components in leg 1
READ (11,) LGN,TPATH,TSTART,NTINT,NCOMP*

vellist - velocity file list for leg 1
READ (11,'a') VELLIST(LGN,J) J=1 TO NCOMP

indcomp,scampu,scphau,scampv,scphav,scampw,scphaw
- velocity scaling factors
READ (11,) INDCOMP,SCAMPU,SCPHAU,SCAMPV,SCPHAV,*
SCAMPW,SCPHAW

comment - leg number 2 comment; not used in computations
READ (11,'a') COMMENT

lgn,tpath,tstart,ntint,ncomp - leg number,
tracking length of leg 2 (hours),
time at start of leg 2 (hours),
number of time steps in leg 2,
number of velocity components in leg 2
READ (11,) LGN,TPATH,TSTART,NTINT,NCOMP*

vellist - velocity file list for leg 2
READ (11,'a') VELLIST(LGN,J) J=1 TO NCOMP

indcomp,scampu,scphau,scampv,scphav,scampw,scphaw
- velocity scaling factors
READ (11,) INDCOMP,SCAMPU,SCPHAU,SCAMPV,SCPHAV,*
SCAMPW,SCPHAW

comment - END OF LEG INFORMATION comment; not used in computations
READ (11,'a') COMMENT

comment - descriptor for iprint; not used in computations
READ (11,'a') COMMENT

iprint - output interval
READ (11,) IPRINT*

comment - descriptor for domain scaling; not used in computations
READ (11,'a') COMMENT

scndx,scndy,scndz - node coordinate scaling factors
READ (11,) SCNDX,SCNDY,SCNDZ*

comment - descriptor for run parameters; not used in computations
READ (11,'a') COMMENT

epshor,epsvert,dtmin - error values, minimum timestep increment
READ (11,) EPSHOR,EPSVERT,DTMIN*

comment - descriptor for drogue scaling factors; not used in computations
READ (11,'a') COMMENT

scdrx,scdry,scdrz - drogue units scaling factors
READ (11,) SCDRX,SCDRY,SCDRZ*

comment - descriptor for number of drogues; not used in computations
READ (11,'a') COMMENT

ndr - # of drogues at beginning of track
READ (11,) NDR*

comment - descriptor for drogue coordinates; not used in computations
READ (11,'a') COMMENT

xdr(i),ydr(i),zdr(i) - initial x,y,z coordinates of drogue i
READ (11,) XDR(I),YDR(I),ZDR(I) I=1 TO NDR*

3.4 The *.vel* file

The file suffixed *.vel* contains flow field information for each domain node for one velocity component. The general velocity components (*u,v,w*) are obtained from

$$\begin{aligned} u &= Re[A_u e^{i(\omega t - \phi_u)}] \\ v &= Re[A_v e^{i(\omega t - \phi_v)}] \\ w &= Re[A_w e^{i(\omega t - \phi_w)}] \end{aligned}$$

where A_u, A_v and A_w are amplitudes, ω is a frequency, and ϕ_u, ϕ_v and ϕ_w are phases.

The first record of the *.vel* file states the *.gr2* filename of the finite element grid on which the velocities were calculated. Next is an alphanumeric description of the velocity component which follows. This is read as a comment line. The number of vertical nodes, *nnv*, is on the third line of the file. The number of components in this velocity file is next. This number is always 1 and although it is read in by *DROG3D*, it is not used by the tracking code. The fifth record is the frequency of the velocity component, in radians/sec. (If the period of a component is ∞, then the frequency should be entered as 1.e-10 on this line.) Each remaining line of the *.vel* file contains a node number, the depth at that node, and the amplitudes and phases for the (*u,v,w*) components at that node, as follows:

nn	$Z_{(nn,nnv)}$	$A_{u(nn,nnv)}$	$\phi_{u(nn,nnv)}$	$A_{v(nn,nnv)}$	$\phi_{v(nn,nnv)}$	$A_{w(nn,nnv)}$	$\phi_{w(nn,nnv)}$
nn	$Z_{(nn,nnv-1)}$	$A_{u(nn,nnv-1)}$	$\phi_{u(nn,nnv-1)}$	$A_{v(nn,nnv-1)}$	$\phi_{v(nn,nnv-1)}$	$A_{w(nn,nnv-1)}$	$\phi_{w(nn,nnv-1)}$
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
nn	$Z_{(nn,2)}$	$A_{u(nn,2)}$	$\phi_{u(nn,2)}$	$A_{v(nn,2)}$	$\phi_{v(nn,2)}$	$A_{w(nn,2)}$	$\phi_{w(nn,2)}$
nn	$Z_{(nn,1)}$	$A_{u(nn,1)}$	$\phi_{u(nn,1)}$	$A_{v(nn,1)}$	$\phi_{v(nn,1)}$	$A_{w(nn,1)}$	$\phi_{w(nn,1)}$

where *nn* is the horizontal node number and *nnv* is the number of vertical nodes at each horizontal node location. See the READ statements on page 10 for the expected format.

DROG3D assumes that there are the same number of vertical nodes under each surface (*x,y*) node, regardless of the total depth. It is not necessary that vertical nodes be equally spaced under each surface node. The number of nodes in the vertical, *nnv*, and *u,v,w* components of each frequency are required for all nodes. *DROG3D* expects to read the amplitudes and phases for all vertical nodes under each surface node and the depth at that particular node. NOTE: The code also expects that the velocity components of the bottom-most level will be read first, and progress upward toward the surface under each surface node (*z* is positive upward with *z*=-*h* at the bottom).

The following FORTRAN statements show the assumed *.vel* file structure for velocity input:

```
      READ(10,'a')GRIDNAME
      READ(10,'a')HEADER
      READ(10,*)NNV
      READ(10,*)IFREQ
      READ(10,*)FREQ
      DO 66 I = 1,NMND                - loop over horizontal node
         DO 67 K = 1,NNV              - loop through vertical nodes
            READ (10,*) NNO,DEP,      - read depth, velocity components
     +                  AX,PX,
     +                  AY,PY,
     +                  AZ,PZ
            DEPTH(I,K)=DEP*SCNDZ      - scale according to factors
            AMPX(I,K,NFQT)=AX*SCAMPU
            AMPY(I,K,NFQT)=AY*SCAMPV
            AMPZ(I,K,NFQT)=AZ*SCAMPW
            PHIX(I,K,NFQT)=PX*SCPHAU
            PHIY(I,K,NFQT)=PY*SCPHAV
            PHIZ(I,K,NFQT)=PZ*SCPHAW
   67    CONTINUE
   66 CONTINUE
```

where *nmnd* is the number of (x,y) nodes, *nnv* is the number of vertical nodes, and *dep* is the depth at node i level k. The outer loop scans over the horizontal node (i counter), the inner loop reads the velocity information for all nodes under i, where k=*1* (one) indicates the bottom level, and *nnv* is the surface level.

4 Output files *.pth* and *.diag*

The output file suffixed .pth has the same filename as the *.din* file specified by the user. The first record is the finite element grid name on which the velocity fields were computed. Then, a complete echo of the *.din* file follows. A flag, 'XXXX', is then written to delimit the drogue tracks from the *.din* file echo. The next record contains the number of output timesteps, the total tracking length in seconds, and the number of drogues initially located within the domain. Output to the *.pth* file after this record occurs only in the subroutine *OUTPUT.f*. The user may want to restructure the present format in this subroutine.

Currently, a 4-column matrix is output. The first three columns represent the (*x,y,z*) coordinates of each drogue at each output timestep as specified by *iprint*. The fourth column is the bottom depth at the drogue's (*x,y*) position. For example, if 10 drogues are initially located within the domain, the first 10 rows of the matrix are initial positions of the 10 drogues. The next 10 rows are the positions after *iprint* timesteps, until the final output.

The output file suffixed *.diag* also has the same name as the input *.din* name. Its first record is the name of the *.din* file used for the current run. The parameter values specified in the *COMMON_BLOCK.h* include file are then reported. Tracking parameters for the entire run are written to the *.diag* file, as well as leg-specific parameters and run-time diagnostic information of the *DROG3D* run, including messages regarding drogue encounters with the bottom and drogue elimination through a horizontal boundary. The last record of the *.diag* file, upon successful completion of the tracking run, reports the total number of drogues eliminated during all legs of the run.

Drogues are never allowed to exit through the bottom of the domain. If a drogue penetrates the bottom, the time remaining in the current timestep is determined, and, based on the bottom velocities of the current element *j*, the drogue's position at the end of the step is projected. The drogue is placed at the bottom normal to its projected position. This relocation of the drogue continues until it no longer hits the bottom.

DROG3D computes the vertical position of each drogue at each timestep in the subroutine *ZPOSITION*. By default, this subroutine returns the new vertical position based on the vertical velocities at the current position of the drogue. This is considered completely passive tracking. *ZPOSITION* also includes code to demonstrate a simple vertical behavior. The behavioral function is commented out upon distribution of *DROG3D* and is documented in comment lines in the subroutine *ZPOSITION*.

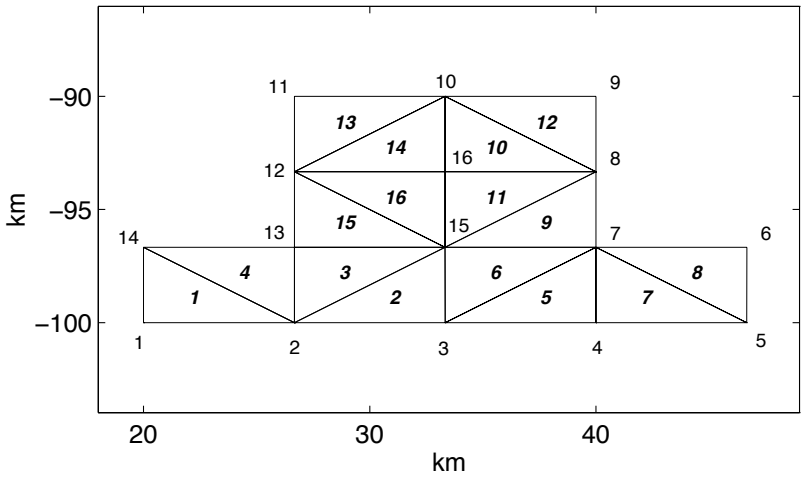
5 Acknowledgements

Support for the preparation of this document and 3-D coding was provided by National Science Foundation Grant numbers OCE-9013887 and OCE-9018388.

A Appendix - Test Case

A.1 Domain and Setup

A simple test case is provided in the *DROG3D* distribution. The test mesh is described by a 16-node *.nod* file and a 16-element *.ele* file. The figure below shows a plot of the test mesh. Element numbers appear in *italics* in the center of each element. Node numbers appear in normal text near the node.



Two test case velocities are provided. The first component is called `test0f.vel` and is a 0-frequency, 1. *m/s* amplitude wind to the northeast. The second component is called `testm2.vel` and is an **M2**-period, unit amplitude, tide. The phases are:

$$\phi_u=0. \qquad \phi_v=\frac{\pi}{2} \qquad \phi_w=0.$$

The velocity scaling factors in the `test.din` input file scale the unit amplitudes to magnitudes appropriate for the given domain.

The `test.din` file is a sample input file for *DROG3D*. It provides information for a tracking run with the following parameters:

- 2 legs
- 2 drogues
- output every timestep

The first leg is 124.2 *hrs* (10 **M2** periods) and the timestep is .2484 *hrs* (12.42/50). It includes both velocity components. The 0-frequency component is scaled to 1. *cm/sec* in both u and v and to 0. in w. The **M2** component is scaled to 10. *cm/sec* in u and v and to 0. in w.

The second leg uses only the northeast wind, 0-frequency component, with scaling in u to 1. *cm/sec* and in v to -1. *cm/sec*. This scaling generates a wind in the southeast direction, although the velocity file contains a wind to the northeast. The duration of leg 2 is the same as leg 1, but the starting time is 124.2 *hrs*.

To run the test case, the user must first compile *connect2d.f* by typing `make conn2d` at the **UNIX** prompt, and then run `conn2d` to generate the *.gr2* file. Next, the user must compile *DROG3D*, using the *Makefile* provided. Type `make DROG3D` at the **UNIX** prompt. Make sure that the `include` file *COMMON_BLOCK.h* resides in the same directory as the *Makefile* and *DROG3D* source codes. This should be the case immediately after distribution. The dimensions set in *COMMON_BLOCK.h* are sufficient for running the test case.

Finally, the user should execute the tracking algorithm by typing `DROG3D` at the **UNIX** prompt. The code queries the user for the name of the *.din* file to run. The user should enter `test`.

A.2 Results

The resulting files `test.pth` and `test.diag` will be output to the current working directory. The trajectories of the two drogues should be as follows:

Drogue 1: Leg 1 conditions will move drogue 1 the to the northeast in a spiraling manner, due to the influence of the m2 component, for 10 m2 periods. Leg 2 will then move the drogue to the southeast for another 10 *M2* periods but without the *M2* component.

Drogue 2: Drogue 2 is placed near enough to a boundary that leg 1 conditions will move the drogue through the northern boundary at time 4.3404 days. Drogue 2 follows the same motion as Drogue 1, displaced to the northeast, until it exits the domain near node 10 during leg 1. The drogue does not re-enter the tracking at the start of leg 2.

The following figure shows a plot of the drogue trajectories within the domain boundary.