

## Short Description of M4D Commands by Purpose

with sample input.

See m4d.commands for alphabetical list and fuller descriptions.

c: **command** sample input      **Description**

### Control

c: **infile** pairs 50      **Read (repeatedly) command input from a different file**  
c: **if** C > 0 find ENDINIT continue      **Conditionally (y/n) goto string in input file**  
c: **end**      **Terminate the program**  
c: **printcontrol** off      **Turn off (or 'on' for back on) normal printing**

### Grid setup and related information

c: **gridfrommefp** ellipse1.5  
40 27 2 1  
-6 -5.5 -5. -4.5 -4 -3.5 -3 -2.5 -2 -1.5 -1.25 -1 -0.8 -0.6 -0.4 -0.2 -0.05 0 0.1 0.2  
0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.5 2 2.5 3 4 5 6 7 8.25  
0 0.0817 0.185 0.278 0.395 0.509 0.619 0.724 0.824 0.912  
1 1.05 1.2 1.4 1.6 1.8 2 2.25 2.5 3 4 5 6 7.1 8.2 9.3 10.4  
0 1  
0.      **Make grid starting with a MEFP format grid**  
c: **gridcorner** ab 0 -.5 0 -.5 0 1 end      **Fix grid at corners to make O-grid**  
c: **gridvarmod** "" a omit 0 -1 -.5 1 0 1 a omit -.5 1 -.5 1 0 1  
b omit -.5 1 0 -1 0 1 b omit -.5 1 -.5 1 0 1  
end      **Move grid and modify variables using *abcd***  
c: **gridmatch**      **Determine grid points that have matching locations**  
c: **set\_wherefw** -1 s      **Determine independent equation points**  
c: **set\_wherep**      **Determine independent p-equation points**  
c: **set\_contar**      **Aspect ratios of continuity control volumes**  
c: **walldist**      **Distance to the nearest wall from mid-points**  
c: **wallnorm**      **Wall normal vectors at wall points**

### Control volumes and pressure grid

c: **cvdcinit** .001 3 fi .999 fw .5 wf .5      **Init. cont. c.v. break-up arrays**  
c: **cvdcparm** .001 3 fi .999 fw .5 wf .5      **Reread cont. c.v. break-up parms.**  
c: **cvdreset** vlam dt 10.      **Recalculate cont. c.v. breakup-up arrays**  
c: **set\_cpsleep**      **Set 2-point coefficients to interp. for sleeping pressure points**  
c: **set\_xyzdouble**      **Determine the double size grid and p-grid**  
c: **set\_volcont**      **Volume of each continuity control volume**  
c: **set\_volmom**      **Volume of each momentum control volume**

### Array initialization and modification

c: **arrayhowto** inn.arrayhowto      **Read file name with instructions for making arrays**  
c: **constant** csym c 12 n n n n n n R R s s s s      **Set constants or arrays of constants**  
c: **varinit** U1 0 5.2 0 0      **Initialize on the points variables: as constants**

c: **varinit** test 1 3. 3 2 1 1 -10. 1. 10. 0 15. 0 0 1. 1. 2. 1. 2. 1. 1. Product profile  
 c: **varinit** saveme 3 From dumped arrays  
 c: **varmatch** U1 U2 "" Set matching points of on-the-points variables  
 c: **arrayread** 1 saveme Read in arrays written using arraydump  
 c: **edit** clt c 4 i 0 j 0 k 0 t 0  
     set i 1 1 1 100 1 100 1 100 end Edit an array  
  
 c: **alias** bij 6 b11 b22 b33 b12 b13 b23 Alias names for parts of an array  
 c: **algebra** sum "" add sum .5 0. 1 Perform algebra on arrays  
 c: **copy** xyzpc xyzpcold Copy an array  
 c: **copypart** xyz 5 i 0 1 200 j 0 1 200 k 0 1 200 t 0 1 1 L 3 2 2  
     y 3 i 0 1 200 j 0 1 200 k 0 1 200 Copy sections of an array  
 c: **editabcd** cc d 1 set 10. 1. 1.2 0. 1. 0. 1. 0. 1. end Edit array using *abcd*  
 c: **function** v0 exp v1 Take functions of an array, trig, min, max, etc.  
 c: **mirror** A m c 1 U1 1 U2 1 U3 -1 "" Mirror variables across an a, b, or c boundary  
 c: **arraydelete** 3 ptt sdum rtemp Delete arrays

## Interpolation

c: **copy\_mtop** pm pp 1 Copy between-points array to p-points array  
 c: **copy\_ptom** pp pm Copy p-points array to between-points array  
 c: **interp\_gtom** 1 vlamg vlam Interp. variables, g-points to between the points  
 c: **interp\_gtop** 1 pg pp Interpolate variables, g-points to p-points  
 c: **interp\_ptod** 1 pp pd Interp. variables p-points to double grid  
 c: **interp\_ptog** 2 pp pg ppadd pgadd Interp. variables p-points to g-points  
 c: **ppreset** Reset pressure for new p-point locations  
 c: **eqppts2ppts** rhsc rhscp 0 Set a p-points array from an Eq. p-points array  
 c: **eqpts2gpts** rhsU1 rhsU1g 1 Set a g-points array from an equation points array  
 c: **gpts2eqpts** U1 u1eq Make an equation point array from a grid point array  
 c: **ppts2eqppts** xtraflow rhscadd Make p-Eq. array from p-point array  
 c: **valueat** pgat1.5.0 pg 1 .5 0 Value of property at specified a,b,c

## Output to file

c: **arraydump** saveme U1 U2 U3 pp "" Dump arrays to a named file  
 c: **arraydumpmore** saveme U1 U2 U3 pp "" Append dump to a named file  
 c: **gridtomefp** gridfile 0 Dump 3d grid, MEFP format  
 c: **lineoutput** linenearwall angle U1 U2 pg ""  
     b -1.01 0 0 0 1 a -1 1.01 0 0 1  
     b 1.01 1 0 0 0 e Lineplot file - results along a, b, or c grid lines  
 c: **lineoutputijk** linediag y qturb omturb ""  
     3 i 0 j 0 k 0 2 2 1 23 1 23 0 Lineplot file - output, any dimensioned  
 arrays

## Print output and analysis

c: **areafLOWint** a 100. -10. 10. 0. 1. U1 qturb "" Area, flow integrals on grid planes

c: <b>arraylist</b>	List information for all arrays
c: <b>aveijk</b> U1 U1avi s n n ""	Take i, j, and/or k averages
c: <b>comment</b> This is my comment!	Read and print a one line comment
c: <b>geom8print</b> 0 10 0 10 0 1 0 1 0 1	Print results from geom8 routines
c: <b>geomcprint</b> 0 10 0 10 0 1 0 1 0 1	Print results from geomc routines
c: <b>print</b> U1 0	Print arrays, g-point, p-point, mid-point or other
c: <b>prints</b> pp 4 i 1 1 5 j 1 1 5 k 1 1 5 l t 0 1 1	Print parts of an array
c: <b>printscoef</b> coef_n coef_i coef_c 0 0 3 1 5 1 5 1 1 1 1	Print parts of coef. array
c: <b>rmsminmax</b> U1 U2 U2 dU1 dU2 dU3 ""	R.m.s, min and max of arrays

### Boundary conditions

c: <b>inletinit</b> 1 1 1 1 200 1 200 1 200 0. .9397 .3420 0	Momentum inlet bndry
**not recommended, marginally stable	
c: <b>inletreset</b>	Reset the velocity on the inlet boundary
c: <b>exitinit</b> 1 1 -1 200 1 1 1 200 200 200 1 200 2 4 1 .7	Exit p-bndry parms
c: <b>bijwallmarv</b>	Set <i>bij</i> on the walls, MARV model
c: <b>bijwallsplat</b>	Set <i>bij</i> on the walls using "splat" wall reflection
c: <b>omwall</b> .089 1 0	Set <i>omturb</i> at walls based on input parameters
c: <b>omwallcoakley</b>	Set <i>omturb</i> at wall for modified Coakley model
c: <b>omwallmarv</b>	Set <i>omturb</i> at walls for MARV model
c: <b>omwallmarvs</b>	Set <i>omturb</i> at walls for MARVS model

### Coefficients for momentum Eqs., etc.

c: <b>coefinit</b> 4	Create the main coefficient arrays
c: <b>coefconv</b> 1	Set convection coefficients, set 1 (linear)
c: <b>coefconvstep</b> 1	Set convection coefficients, set 1 (stepwise approx.)
c: <b>coefvisc</b> vlam 2	Set viscous coefficients, set 2 (linear)
c: <b>coefviscstep</b> vlam 2	Set viscous coefficients, set 2 (stepwise approx.)
c: <b>coefdt</b> dt 3 n	Set coefficients for the time term, set 3
c: <b>coefzero</b> cpflop_n cpflop_i cpflop_c 3 2 0 1 i	Zero a coef. set for point types
c: <b>coefadd</b> 0 2 1 1. 2 1.	Form a sum of coefficient sets
c: <b>coefficient</b> 2 0 0 .4 -1.5 2 i w	Analyze and fix a set of coefficients
c: <b>coefcplus</b> 0 cplusa	Determine the sum of the positive coefficients

### Right hand side for momentum Eqs., etc.

c: <b>coefrhs</b> U1up rhsU1 old 3 -1.	Update the r.h.s of Eq. using the coefficient array
c: <b>bijrhsmarv</b>	Evaluate right hand side <i>bij</i> Eqs., MARV model
c: <b>bijrhsmarvs</b>	Evaluate right hand side <i>bij</i> Eqs., MARVS model
c: <b>bijrhsmarvex</b> MA c5 c7 pkdktot	Eval. r.h.s <i>bij</i> Eqs. with model variations
c: <b>momrhrs</b> new 0 pp bij zrotation ""	Contributions to r.h.s. momentum equations
c: <b>omrhSCOakley</b>	R.h.s <i>omturb</i> equations, Coakley model
c: <b>omrhsmarv</b>	R.h.s <i>omturb</i> equations, MARV (and MARVS) model
c: <b>qturbrhs</b>	Right hand side of the <i>qturb</i> equations
c: <b>vint2eqpts</b> hksm hkse n	Integrate array for source term addition to r.h.s. of Eq.

c: **wallflux** rhstt fluxda      Add specified wall flux/area to the r.h.s. of Eq.

### Solve equations and update variables

c: **eqnsolvebij** cab new 2 i w tc 10 end 0      Solve *bij* equations for *dbij*  
c: **eqnsolves** rhsc "" dcc new 2 i I tc 20 end 0      Solve on-the-grid-points equations  
c: **eqnupdate** m 1 2 i w      Update the velocity from the momentum equations  
c: **varupdate** qturb dq log      Update a variable with its change, safely  
c: **varupdate** bij dbij b      Update *bij* with a realizabilty check

### Continuity = pressure correction equations

c: **blocksetabc** p 6 6 6  
0 1.281 2.501 3.721 4.941 6.039  
-1 -.6 -.2 .2 .6 1.  
-1 -.6 -.2 .2 .6 1.      Set up block for multiblock options for eqnsolvep  
c: **momcam** .5      Center-pt coeffs of  $dU$ , abbreviated momentum equations  
c: **momcamddt**      Alternatives to cam from momcam  
c: **set\_cpda** .0 .05 .5 .15 0 1 i      Set coeffs. of pressure in momentum equations  
c: **set\_cpflop**      Collect p-coefs for each independent momentum equation  
c: **contpcdu**      Set continuity Eq.  $dp$  coefficients based on  $dU$   
c: **contpcfixed** 1. 1. 1. -1. 0 -20      Fix up *cpc* for stable p-correction equations  
c: **contpcexit**      Set  $dp$  coefficients for exit boundary condition  
c: **contpcinlet**      Set  $dp$  coefficients for the inlet boundary  
c: **contrhsu**      Set r.h.s of continuity equation based on mass fluxes  
c: **contrhsp**      Add to the r.h.s of continuity the retained pressure modification  
c: **contrhsexit**      Set r.h.s of continuity for exit boundary condition  
c: **eqnsolvep** ijk 100 c 1 end 0      Solve pressure correction equations for  $dp$   
c: **contdu**      Update the velocities to satisfy continuity from  $dp$

### Turbulent viscosity models

c: **visccoakley**      Turbulent viscosity, Coakley 2-eq. model  
c: **viscles** clles      An LES style mixing length turbulent viscosity  
c: **viscmarv**      Turbulent viscosity for *bij* and *qturb*, MARV model  
c: **viscmarvheat**      Turb. viscosity for heat, MARV compatible  
c: **viscnoise**      Turb. viscosity for noise

### Misc.

c: **ddtall**      Calc.  $1/dt$  for a variety of timescales  
c: **gradprop** 3 U1 ""      Take gradients of properties  
c: **prinstress**      Calc. principle components of  $\overline{u_i u_j}$ ,  $S_{ij}$ ,  $\nabla \times \underline{U}$   
c: **qbijles**      Arrays for a (trial) Reynolds stress based LES  
c: **rayleigh** 0 10 1 3 0 1 2 cyw cyi cypd cely      Arrays and search for a (trial) Rayleigh instability model  
c: **set\_gbij**      Calculate turbulence anisotropy parameter  
c: **set\_pkdk** vcoakley      Calculate  $P/k$

c: **set\_srate** Calculate the magnitude of the strain rate

### Plot package

c: **abcmask** maskc 1. c 0 .05 .1 .3 .5 0 Set points in array at specified a, b, or c

c: **bar** imagebar 250 50 b Create a labeled colorbar image (fill properties)

c: **giflist** prename 1 50 Write to a file, a list of .gif names (for videos)

c: **gridinfo** Set grid info arrays for xyzdouble

c: **image** outgif imagebar qbar Create, combine modify or output plot images

c: **iplaneint** lineave rho qturb pt "" Lineplot file, area and mass-ave on i-planes

c: **keyword** conv conv.line 2

TIME 1 t rmsmm 2 U1rms U1max

Analyze an arraydump file to give a lineplot file

c: **lineplot** lineave imageave "rms and max U1"

100 400 140 450 1 7 time 0 1.5 .5 "" 0 6 1

t 1 0 U1rms 1 0 2 3 'r'

t 1 0 U1max 1 0 26 3 'm'

end

Create a lineplot pixel image

c: **picture** image 500 500 1000 fgv Draw fill contours, grid lines, vectors

c: **xytocgrid** c: xytocgrid cg .3 .5 5 0 0 .5 1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 .5  
.5 100 1. qturb pg "" Create and interp. prop. to c-grid or flat grid

c: **xyzicut** outgif cut 5 1. 0. 0. -.143 -.111 -.079 -.048 -.016

1. 2. 0. 1. U1 U2 U3 cps cpt ""

Cut grid in i-grid direction to specified parallel planes