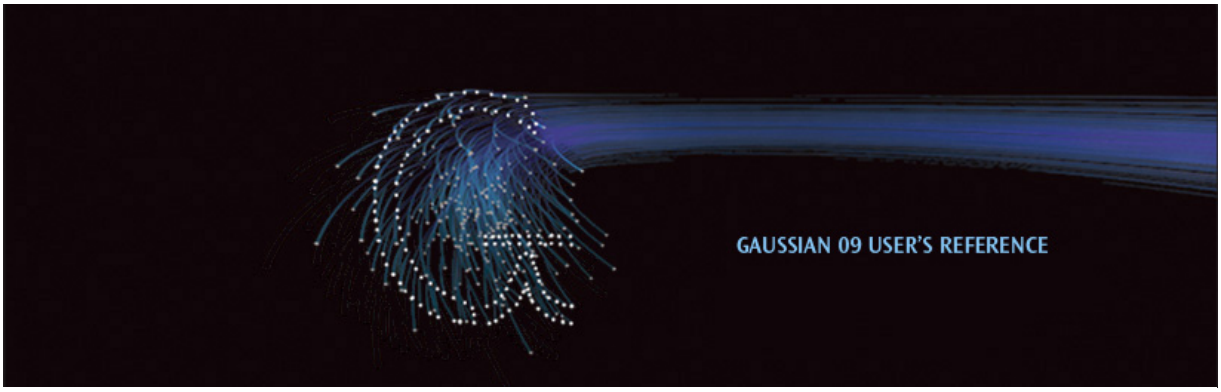

cubegen

Gaussian includes a standalone utility for generating cubes from the data in a formatted checkpoint file (equivalent to the previous Cube keyword). The utility is named **cubegen**, and it has the following syntax:

cubegen *nprocs kind fchkfile cubefile npts format cubefile2*

The parameters, which are not case-sensitive, have the following meanings:

nprocs
Number of shared memory processors used for electrostatic potential calculations. A value of 0 is equivalent to 1 (it is the default). Note that this parameter must be included if other parameters are specified. Previously this parameter was used to specify the amount of memory to allocate. The GAUSS_MEMDEF environment variable should be used instead.

kind
A keyword specifying the type of cube to generate:

MO=*n*	Molecular orbital *n*. The keywords **Homo**, **Lumo**, **All**, **OccA** (all alpha occupied), **OccB** (all beta occupied), **Valence** (all valence orbitals) and **Virtuals** (all virtual orbitals) may also be used in place of a specific orbital number. There is no default for *n*, and an error will occur if it is omitted. **AMO** and **BMO** can be similarly used to select only alpha or beta orbitals (respectively). For open shell systems, **Homo** selects both alpha and beta orbitals.
Density=*type*	Total density of the specified type. The *type* keyword is one of the single density selection options that are valid with the **Density** keyword: **SCF**, **MP2**, **CI**, **QCI,** and so on (note that Current is not supported). The **FDensity=** variation requests the use of the full instead of the frozen core density. The default *type* keyword is **SCF**.
Spin=*type*	Spin density (difference between α and β densities) of the specified type.
Alpha=*type*	Alpha spin density of the specified type. **FAlpha=** requests the use of the full instead of the frozen core density.
Beta=*type*	Beta spin density of the specified type. **FBeta=** requests the use of the full instead of the frozen core density.
Potential=*type*	Electrostatic potential using the density of the specified type.
Gradient	Compute the density and gradient.
Laplacian	Compute the Laplacian of the density ($\nabla^2\rho$).
NormGradient	Compute the norm of the density gradient at each point.
CurrentDensity=*I*	Magnitude of the magnetically-induced (GIAO) current density, where *I* is the applied magnetic field direction (X, Y or Z).
ShieldingDensity=*IJN*	Magnetic shielding densit}. *I* is the direction of the applied magnetic field, *J* is the direction of the induced field (X, Y or Z), and *N* is the number of the nucleus for which the shielding density (GIAO) is to be calculated.

fchkfile
Name of the formatted checkpoint file. **cubegen** will prompt for this filename if it is not specified.

cubefile
Name of the output cube file; test.cube is the default if it is not explicitly specified (i.e., specifying the name of the checkpoint file does not change the default cube filename).

npts
Number of points per side in the cube. A value of **0** selects the default value of 80^3 points distributed evenly over a rectangular

grid generated automatically by the program (not necessarily a cube). Positive values of *npts* similarly specify the number of points per "side"; e.g., 100 specified a grid of 1,000,000 (100^3) points.

The values **-2**, **-3** and **-4** correspond to the keywords **Coarse**, **Medium** and **Fine** and to values of 3 points/Bohr, 6 points/Bohr and 12 points/Bohr (respectively). Negative values of *npts* $\leq$-5 specify spacing of *npts*$*10^{-3}$ Angstroms between points in the grid.

A value of **-1** says to read the cube specification from the input stream, according to the following format:

IFlag, X_0, Y_0, Z_0	*Output unit number and initial point.*
N_1, X_1, Y_1, Z_1	*Number of points and step-size in the X-direction.*
N_2, X_2, Y_2, Z_2	*Number of points and step-size in the Y-direction.*
N_3, X_3, Y_3, Z_3	*Number of points and step-size in the Z-direction.*

IFlag is the output unit number. If *IFlag* is less than 0, then a formatted file will be produced; otherwise, an unformatted file will be written.

If N_1<0 the input cube coordinates are assumed to be in Bohr, otherwise, they are interpreted as Angstroms. $|N_1|$ is used as the number of X-direction points in any case; N_2 and N_3 specify the number of points in the Y and Z directions, respectively. Note that the three axes are used exactly as specified; they are not orthogonalized, so the grid need not be rectangular.

The value **-5** says to read in an arbitrary list of points from standard input. If you enter this input by hand, terminate the input with an end-of-file (i.e., **Ctrl-D** under Unix). Alternatively, you can redirect standard input to a file containing the list of points (do not place a blank line or **Ctrl-D** at the end of the file).

format
Format of formatted output files: **h** means include header (this is the default); **n** means don't include header. This parameter is ignored when unformatted cube files are produced.

cubefile2
If specified, the size for the generated cube is taken from this file. This option is useful when creating cubes for later arithmetic operations such as difference densities where you need the exact same grid to be used for all cubes. In order for the cube dimensions to be taken from the specified cube file, the *npts* parameters must be -1 and the specified file must have been created with a header.

If no parameters are specified, **cubegen** will prompt for *fchkfile* and run the following:

cubegen 1 density=scf test.cube 80 h

This generates a file named test.cube (with header) containing the SCF density in a rectangular grid of 80^3 points.

OUTPUT FILE FORMATS

All values in the cube file are in atomic units, regardless of the input units.

For density and potential grids, unformatted files have one row per record (i.e., N_1*N_2 records each of length N_3). For formatted output, each row is written out in format (6E13.5). In this case, if N_3 is not a multiple of six, then there may be blank space in some lines.

The norm of the density gradient and the Laplacian are also scalar (i.e., one value per point), and are written out in the same manner. Density+Gradient grids are similar, but with two writes for each row (of lengths N_3 and $3*N_3$).
Density+Gradient+Laplacian grids have 3 writes per row (of lengths N_3, $3*N_3$, and N_3).

For example, for a density cube, the output file looks like this:

```
NAtoms, X-Origin, Y-Origin, Z-Origin NVal          NVal is the #points/value
N1, X1, Y1, Z1                 # of increments in the slowest running direction
N2, X2, Y2, Z2
N3, X3, Y3, Z3                 # of increments in the fastest running direction
IA1, Chg1, X1, Y1, Z1          Atomic number, charge, and coordinates of the first atom
…
IAn, Chgn, Xn, Yn, Zn          Atomic number, charge, and coordinates of the last atom
(N1*N2) records, each of length N3      Values of the density at each point in the grid
```

Note that a separate write is used for each record.

For molecular orbital output, *NAtoms* will be less than zero, and an additional record follows the data for the final atom (in format 10I5 if the file is formatted):

```
NMO, (MO(I),I=1,NMO)     Number of MOs and their numbers
```

If N_{MO} orbitals were evaluated, then each record is $N_{MO}*N_3$ long and has the values for all orbitals at each point together.

If one wishes to read the values of the density, Laplacian, or potential back into an array dimensioned X(N_3,N_2,N_1), code like the following Fortran loop may be used:

```
      Do 10 I1 = 1, N1
      Do 10 I2 = 1, N2
         Read(n,'(6E13.5)') (X(I3,I2,I1),I3=1,N3)
      10 Continue
```

where *n* is the unit number corresponding to the cube file.

If the origin is (X_0,Y_0,Z_0), and the increment is (X_1,Y_1,Z_1), then point (I_1,I_2,I_3) has the coordinates:

X-coordinate: $X_0+(I_1-1)*X_1+(I_2-1)*X_2+(I_3-1)*X_3$

Y-coordinate: $Y_0+(I_1-1)*Y_1+(I_2-1)*Y_2+(I_3-1)*Y_3$

Z-coordinate: $Z_0+(I_1-1)*Z_1+(I_2-1)*Z_2+(I_3-1)*Z_3$

The output is similar if the gradient or gradient and Laplacian of the charge density are also requested, except that in these cases there are two or three records, respectively, written for each pair of I_1, I_2 values. Thus, if the density and gradient are to be read into arrays D(N_3,N_2,N_1), G(3,N_3,N_2,N_1), RL(N_3,N_2,N_1), a correct set of Fortran loops would be:

```
      Do 10 I1 = 1, N1
      Do 10 I2 = 1, N2
         Read(n,'(6F13.5)')  (D(I3,I2,I1),I3=1,N3)
         Read(n,'(6F13.5)')  ((G(IXYZ,I3,I2,I1),IXYZ=1,3), I3=1,N3)
 10   Continue
```

where again *n* is the unit number corresponding to the cube file.
