Moments for Molecules

We need to evaluate moments for molecules. Cartesian moments are defined as:

$$m_{n_x,n_y,n_z} = \int (x - X_c)^{n_x} (y - Y_c)^{n_y} (z - Z_c)^{n_z} p(\mathbf{r})\, d\mathbf{r}$$

The user should be allowed to specify the center but the usual center would be the center-of-mass of the molecule,

$$\mathbf{R_c} = \frac{\sum_{A=1}^{N_{atoms}} M_A \mathbf{R}_A}{\sum_{A=1}^{N_{atoms}} M_A}$$

where M_A and $\mathbf{R}_A$ denote the atomic masses and positions. It would be good to let the user specify a list of atomic masses (one for each atom; this might be in the calling sequence of the function for computing the center-of-mass only, and not the function for the moments) but the usual tradition is to use the mass of the most abundant isotope for each element. `AtomDB` should have a list (which I'd copy rather than explicitly using `AtomDB`, unless we already have an `AtomDB` dependence.)

When you evaluate the Cartesian moments, you evaluate all the moments up to some order. This means you evaluate all m_{n_x,n_y,n_z} for which $n_x + n_y + n_z \leq \texttt{order}$. The (obvious) implementation exploits the fact that the property density $p(\mathbf{r})$ is (re)used in all the moment evaluations, and since it is usually (by far) the most expensive part of the integrand to evaluate on the grid points, it can be reused. This was the idea of the old `dot_multi` method.

Spherical moments,

$$m_{lm} = \int |\mathbf{r} - \mathbf{R}_c|^l S_l^m(\theta, \phi) p(\mathbf{r})\, d\mathbf{r}$$

can be constructed from the Cartesian moments (the reverse is not true) but they can be constructed directly more efficiently. (Even if you needed both sets of moments, which is rare, computing the spherical moments as integrals increases the cost by less than a factor of two.) There is code for the (real) spherical harmonics in `grid` and `denspart`
I would use the real/imaginary part of the (complex) spherical harmonics. Be careful with normalization which should be $\frac{\color{red}{\backslash} 4\pi}{2l+1}$ in this case (Racah normalization).

Radial moments,

$$m_n = \int |\mathbf{r} - \mathbf{R}_c|^n p(\mathbf{r})\, d\mathbf{r}$$

should also be implemented; they are not (always) computable from Cartesian moments. You could "overload" the solid moments with:

$$m_{nlm} = \int |\mathbf{r} - \mathbf{R}_c|^{n+l} S_l^m(\theta, \phi) p(\mathbf{r})\, d\mathbf{r}$$

I'd use `HORTON` ordering for Cartesian and Spherical moments to ensure easy compatability.

It might be best to code the moments in a very user-friendly way, and have a utility routine to which you pass a function ($p(\mathbf{r})$ in this case) and a list of multiplying factor functions $f_n(\mathbf{r})$, and then return the list of evaluated integrals

$$\int f_n(\mathbf{r}) p(\mathbf{r})\, d\mathbf{r} \qquad\qquad n = 0, 1, \ldots$$

Such a function might be broadly useful. This actually seems easy with the current `integrate` but would require passing a two-dimensional array of values on the grid, where you multiply together values in the first dimension, but then in the second dimension you evaluate separately (returning an array of values).

Implementation Note: One could build molecular moments from atomic moments. This method works fine, and has the advantage of allowing you to integrate functions like $(x - X_A)^{n_x} (y - Y_A)^{n_y} (z - Z_A)^{n_z} p_A(\mathbf{r})$ that are nearly spherical. But one has to engage in rather tedious math to shift the center of the moment expansion to the desired center. (This is especially true for the spherical moments.) To me it seems better to just integrate the moments directly. It would be "fun" to implement the shifting of moment centers approach.