This is a set of notes on using/manipulating Gaussian basis functions in HORTON.

Most of these notes are based on Helgaker et al., Chapter 9.

https://drive.google.com/file/d/1cwLFkfsBBYi2-qgjmQBYjx6XYzcrj9vp/view?usp=sharing

with various contributions from other sources plus the crap I came up with myself. Notation is a weird amalgam of Helgaker and "old HORTON"

https://theochem.github.io/horton/2.0.0/tech_ref_gaussian_basis.html#ref-gaussian-basis-standard-sets

and other things. (Alas, in this section, Helgaker's notation is pretty hideous, probably partly because he seems to have tossed together contents from lots of previous articles he'd written.) I have *tried* to double-check all the formulas by comparing sources (not always deriving them myself). I recommend making sure that I am writing the formula correctly (and maybe double-checking sources also). Key sources are in the google drive directory for gbasis

https://drive.google.com/drive/folders/1d1oLk-7FfxkwpFoUHcr3qYX9zBXnngZc?usp=sharing

and I try to reference them. A good introduction (not code-worthy but OK) is

https://joshuagoings.com/2017/04/28/integrals/

My notation is not harmonized at this stage but I will try to revisit and do that as we go along. I can't decide on the best notation, honestly….it seems many people can't, as a "nice" notation for 2-electron integrals is overcomplicated for simpler tasks.

You should have permission to edit key files and also to add to the directories.

The Gaussian basis functions in HORTON are either Cartesian Gaussians,

$$\chi_{xyz\mathbf{a}}(\mathbf{r}) \equiv b_{xyz}(\mathbf{d},\boldsymbol{\alpha},\mathbf{a},\mathbf{r}-\mathbf{R}_A) \equiv (x-X_A)^{a_x}(y-Y_A)^{a_y}(z-Z_A)^{a_z}\sum_{i=1}^{N_{\text{contractions}}} d_i N_{xyz}(\alpha_i,\mathbf{a})\exp\left(-\alpha_i|\mathbf{r}-\mathbf{R}_A|^2\right) \quad (1)$$

or spherical Gaussians

$$\chi_{\ell m\mathbf{a}}(\mathbf{r}) \equiv b_{\theta\phi}(\mathbf{d},\boldsymbol{\alpha},\ell,m,\mathbf{r}-\mathbf{r}_A) \equiv r^{\ell}S_{\ell}^{m}(\theta,\phi)\sum_{i=1}^{N_{\text{contractions}}} d_i N(\alpha_i,\ell)\exp\left(-\alpha_i|\mathbf{r}-\mathbf{r}_A|^2\right) \quad (2)$$

Here $\mathbf{d}=\left[d_1,d_2,\ldots,d_{N_{\text{contractions}}}\right]$ is a vector of contraction coefficients and $\boldsymbol{\alpha}=\left[\alpha_1,\alpha_2,\ldots,\alpha_{N_{\text{contractions}}}\right]$ is a vector of Gaussian exponents. The *real* spherical harmonics are used (not the complex spherical harmonics). There is some ambiguity about how these should be defined, but we can choose:

$$S_{\ell}^{m}(\theta,\phi) = \begin{cases} \text{Re}\left(Y_{\ell}^{m}(\theta,\phi)\right) & m \geq 0 \\ \text{Im}\left(Y_{\ell}^{m}(\theta,\phi)\right) & m < 0 \end{cases} \quad (3)$$

Note that often we neglect explicit mention of spherical/Cartesian basis functions and just write $\chi_a(\mathbf{r})$ for a normalized, possibly contracted, basis function.

A shell is defined as a set of basis functions with the same ℓ or the same $a_x+a_y+a_z$. There are $\frac{1}{2}(\ell+1)(\ell+2)$ Cartesian Gaussian basis functions in a shell, and $2\ell+1$ spherical Gaussian basis functions in a shell. For Cartesian Gaussians, the "shell label" is $\ell = a_x+a_y+a_z$. It is important to note that a_x, a_y, a_z are always nonnegative integers.

The normalization constants are

$$N_{xyz}(\alpha,\mathbf{a}) = \sqrt{\left(\frac{2\alpha}{\pi}\right)^{3/2}\frac{(4\alpha)^{a_x+a_y+a_z}}{(2a_x-1)!!(2a_y-1)!!(2a_z-1)!!}} \quad (4)$$

$$N_{\theta\phi}(\alpha,\ell) = \sqrt{\left(\frac{2\alpha}{\pi}\right)^{3/2}\frac{(4\alpha)^{\ell}}{(2\ell-1)!!}} \quad (5)$$

In practice, one always includes all basis functions from a given shell, never "partial subshells" of basis functions.

The conversion from Cartesian to Spherical harmonics and back is something that, in "old gbasis" was handled by Sympy. There is, however, an explicit formula for how to do it (given by Helgaker among others). It doesn't matter very much….it is reasonable to store all the "transformation vectors" up to a given order in an .npz

file and use it (or extend it) as needed, but it is also reasonable to compute this file at the start of a calculation. One needs to be careful here, as there is no uniform normalization convention for these things, so taking a result from the literature is nontrivial. I would tend to prefer implementing the explicit formula, as then it is very easy to go to high order as needed. (The same formula is useful in general, as it is used to convert from Cartesian to Spherical multipole moments also.) For the formula, see Helgaker, section 6.4.2 and 6.4.4, with Eq. (6.4.47-50) the key equations.

The basic strategy for Gaussian basis sets is that things are evaluated in Cartesian Gaussians, then converged to Spherical Gaussians if needed (which is usually the case). It is important to note that Cartesian Gaussians (uncontracted Cartesian Gaussians) can always be factored into a product of 1-dimensional Gaussians,

$$b_{xyz}\left(\mathbf{1},\alpha,\mathbf{a},\mathbf{r}-\mathbf{r}_A\right)\equiv b_{1\mathrm{d}}\left(\alpha,a_x,x-X_A\right)b_{1\mathrm{d}}\left(\alpha,a_y,y-Y_A\right)b_{1\mathrm{d}}\left(\alpha,a_z,z-Z_A\right) \tag{6}$$

with

$$b_{1\mathrm{d}}\left(\alpha,a_x,x-X_A\right)=\left(\frac{2\alpha}{\pi}\right)^{1/2}\frac{\left(4\alpha\right)^{a_x}}{\left(2a_x-1\right)!!}\left(x-X_A\right)^{a_x}e^{-\alpha\left(x-X_A\right)^2} \tag{7}$$

(and similar for *y* and *z*). I will define the *unnormalized* 1-dimensional primitive Gaussian centered at the origin

$$g_{a\alpha}\left(x\right)=x^a e^{-\alpha x^2} \tag{8}$$

This is (roughly) the notation in Helgaker. I will also often talk about unnormalized primitive Gaussians, which I'll denote typically as

$$\begin{aligned}
\xi_{\mathbf{a}}\left(\mathbf{r}\right)&\equiv\left(x-X_A\right)^{a_x}\left(y-Y_A\right)^{a_y}\left(z-Z_A\right)^{a_z}\exp\left(-\alpha\left|\mathbf{r}-\mathbf{R}_A\right|^2\right)\\
\xi_{\mathbf{b}}\left(\mathbf{r}\right)&\equiv\left(x-X_B\right)^{b_x}\left(y-Y_B\right)^{b_y}\left(z-Z_B\right)^{b_z}\exp\left(-\beta\left|\mathbf{r}-\mathbf{R}_B\right|^2\right)\\
\xi_{\mathbf{c}}\left(\mathbf{r}\right)&\equiv\left(x-X_C\right)^{c_x}\left(y-Y_C\right)^{c_y}\left(z-Z_C\right)^{c_z}\exp\left(-\gamma\left|\mathbf{r}-\mathbf{R}_C\right|^2\right)\\
\xi_{\mathbf{d}}\left(\mathbf{r}\right)&\equiv\left(x-X_D\right)^{d_x}\left(y-Y_D\right)^{d_y}\left(z-Z_D\right)^{d_z}\exp\left(-\delta\left|\mathbf{r}-\mathbf{R}_D\right|^2\right)
\end{aligned} \tag{9}$$

These just get multiplied by a normalization constant of the form (4) to form an (uncontracted, primitive) Gaussian basis function.

A word about shells…. In general, you can consider *general contractions* where a given set of Gaussian exponent may contribute to several different basis functions. This is typical for ANO basis sets and also for some Pople basis sets (where *s* and *p*-type functions may share exponents). One can exploit this structure in a code, because some of the intermediates are the same. Technically, every Cartesian basis function with the form of Eq. (1) that differs only in terms of the contraction coefficients $\left\{d_i\right\}_{i=1}^{N_{\mathrm{contractions}}}$ and/or the exponents a_x,a_y,a_z (but shares the same Gaussian exponents $\left\{\alpha_i\right\}_{i=1}^{N_{\mathrm{contractions}}}$) is in the same shell. Similarly, every Spherical basis function with the form of Eq. (2) that differs only in terms of the contraction coefficients $\left\{d_i\right\}_{i=1}^{N_{\mathrm{contractions}}}$ and/or the angular momentum quantum numbers ℓ,m (but shares the same Gaussian exponents $\left\{\alpha_i\right\}_{i=1}^{N_{\mathrm{contractions}}}$) is in the same shell. *However*, in this iteration of gbasis, this adds complexity that we prefer not to deal with. Therefore we consider a shell to be defined as a set of basis functions that share the same contraction coefficients and the same exponents, and differ only in a_x,a_y,a_z (Cartesian) or ℓ,m (Spherical).

Given the rarity of using generalized contractions, and the fact we can always support that functionality with the `cgbasis` portion of HORTON, I am not sure it is worth the additional complexity to support generalized contractions. It is certainly not on the immediate agenda.

Evaluating Functions on a Grid.

The first thing we will discuss is evaluating basis functions and related properties (e.g., densities, local kinetic energy, etc.) on a grid. A Gaussian basis function can be specified by its center/position, its contraction coefficients, its angular momentum. That is, the *k*th basis function has the form

$$b_{xyz}\left(\mathbf{d}_k,\boldsymbol{\alpha}_k,\ell_k,a_x,a_y,a_z,\mathbf{R}_A;\mathbf{r}\right) \tag{10}$$

In almost all cases, there is a logical way to structure things.

(a) loop over atoms. These are basis functions that share a center.
(b) loop over shells in atoms. These are basis functions that share the same exponent(s) and contraction coefficient(s).
(c) loop over contractions in each basis function. This is a loop over the primitives that contribute to a given basis function.
(d) calculate primitives.

The idea is that in the innermost step, (d), we evaluate the Cartesian quantities. By looping over contractions at the next level, (c), we quickly reduce the size of our data from the number of primitives to the number of (Cartesian) basis functions. In the next step, (b), we reduce the size of our data further by converting Cartesian basis-set quantities to spherical basis-set quantities, when that is desired (which is 95% of the time).[1] Finally we accumulate everything with an outermost loop over atoms (step (a)). Notice that when you are doing integrals over multiple basis functions, (a) outer loop where you loop over 2 (1-electron integral) or 4 (2-electron integral) choices of atoms (with replacement), (c) middle loop where you loop over 2 (or 4) shells selected from the atoms, (d) inner loop where you loop over 2 (or 4) primitive basis functions selected from the shells in (c).

There are (obviously) ways to improve this, but this is usually close to an "optimal" structure.

Evaluating Derivatives on a Grid:

Note that derivative of a Gaussian is a Hermite polynomial times a Gaussian,

$$\begin{aligned}
\frac{d^k e^{-\alpha(x-X_A)^2}}{dX_A^k} &= \frac{d^k e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2}}{d\left(\sqrt{\alpha}\left(x-X_A\right)^2\right)^k}\left(\frac{d\sqrt{\alpha}\left(x-X_A\right)}{dX_A}\right)^k \\
&= \left(\alpha^{k/2}\right)H_k\left(\sqrt{\alpha}\left(x-X_A\right)\right)e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2} \\
\frac{d^k e^{-\alpha(x-x_A)^2}}{dx^n} &= \frac{d^k e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2}}{d\left(\sqrt{\alpha}\left(x-X_A\right)^2\right)^k}\left(\frac{d\sqrt{\alpha}\left(x-X_A\right)}{dx}\right)^k \\
&= (-1)^k\left(\alpha^{k/2}\right)H_k\left(\sqrt{\alpha}\left(x-X_A\right)\right)e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2} \\
&= \left(-\sqrt{\alpha}\right)^k H_k\left(\sqrt{\alpha}\left(x-X_A\right)\right)e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2}
\end{aligned} \tag{11}$$

This means that to evaluate the derivative of a Cartesian Gaussian, one uses

[1] In an earlier version of my notes I had grouped by shell-types also, but this doesn't seem relevant to me right now…I think it is extra complexity without good reason. If it is helpful, it can be dealt with trivially by "group shells of the same type on an atom.

$$
\begin{aligned}
\frac{d^k \left(x-X_A\right)^n e^{-\alpha\left(x-x_A\right)^2}}{dx^k} &= \sum_{j=0}^{\min(k,n)} \binom{k}{j} \frac{d^{k-j} e^{-\alpha\left(x-x_A\right)^2}}{dx^{k-j}} \frac{d^j \left(x-X_A\right)^n}{dx^j} \\
&= \sum_{j=0}^{\min(k,n)} \binom{k}{j} \left(-\sqrt{\alpha}\right)^{k-j} H_{k-j}\left(\sqrt{\alpha}\left(x-X_A\right)\right) e^{-\left(\sqrt{\alpha}\left(x-X_A\right)\right)^2} \frac{n!\left(x-X_A\right)^{n-j}}{(n-j)!} \\
&= e^{-\left(\sqrt{\alpha}\left(x-X_A\right)\right)^2} \sum_{j=0}^{\min(k,n)} \binom{k}{j} \frac{n!}{(n-j)!} \left(-\sqrt{\alpha}\right)^{k-j} \left(x-X_A\right)^{n-j} H_{k-j}\left(\sqrt{\alpha}\left(x-X_A\right)\right) \\
&= \sum_{j=0}^{k} \binom{k}{j} \frac{d^{k-j}\left(x-X_A\right)^n}{dx^{k-j}} \frac{d^j e^{-\alpha\left(x-x_A\right)^2}}{dx^j} \\
&= \sum_{j=\max(0,k-n)}^{k} \binom{k}{j} \frac{d^{k-j}\left(x-X_A\right)^n}{dx^{k-j}} \frac{d^j e^{-\alpha\left(x-x_A\right)^2}}{dx^j} \\
&= \sum_{j=\max(0,k-n)}^{k} \binom{k}{j} \frac{n!\left(x-X_A\right)^{n-k+j}}{(n-k+j)!} \left(-\sqrt{\alpha}\right)^j H_j\left(\sqrt{\alpha}\left(x-X_A\right)\right) e^{-\alpha\left(x-X_A\right)^2} \\
&= e^{-\alpha\left(x-X_A\right)^2} \sum_{j=\max(0,k-n)}^{k} \left[\binom{k}{j} \frac{n!}{(n-k+j)!} \left(-\sqrt{\alpha}\right)^j\right] \left(x-X_A\right)^{n-k+j} H_j\left(\sqrt{\alpha}\left(x-X_A\right)\right)
\end{aligned}
\tag{12}
$$

The last formula here is the most convenient. Notice that the coefficient can be computed using scipy, e.g.,

$$
\binom{k}{j} \frac{n!}{(n-k+j)!} = \left(\text{scipy.special.comb}\left(k, j\right)\right)\left(\text{scipy.special.perm}\left(n, k-j\right)\right) \tag{13}
$$

There is a little logistics to be considered here too. Evaluating the Hermite polynomial is expensive especially at high order, but there is a recursion. One could code the recursion explicitly, but it is probably better to use numpy. Note that by using

`numpy.polynomial.hermite.hermval`

you can compute several derivatives at once, and this is favorable since one gets to "reuse" a lot of work. (You could also maybe compute several points at once, but since the "coefficients" depend on the points (through $\left(x-X_A\right)^{n-k+j}$), that doesn't seem to be supported.)

The "right" way to structure the code here is to:

(a) read in from the user a list of derivatives that are needed, $\frac{\partial^{k_x+k_y+k_z}}{\partial x^{k_x} \partial y^{k_y} \partial z^{k_z}}$.[2] This can be stored as a matrix or a list of lists.

(b) For every point you need a derivative at
   a. for every shell of contracted basis function in the shell, b_{A,ℓ_A}
   b. for every Gaussian exponent in the basis function,
   c. for every *unique* choice of $k_* = k_x, k_y, k_z$
   d. compute the Hermite polynomial coefficients in the last line of Eq. (12) for every derivative of interest (all unique values of k_*) and $n_* = 1, 2, \ldots \ell$ and evaluate the Hermite expansion. (The $n_* = 0$ and $k_* = 0$ cases are trivial.)

[2] Note that the $k_x = k_y = k_z = 0$ is a (trivial) special case of this method. The "evaluate each basis function at a point" case should probably be coded separately because the consistency of results provides a good test.

e. use the derivatives from the previous step to construct $\frac{\partial^{k_x+k_y+k_z}}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}}$ for each unique Gaussian exponent in the (contracted) basis function
f. Add up the contraction coefficients.
g. Combine Cartesian Gaussians into spherical Gaussians for each shell

(c) This gives all the relevant derivatives of all the basis functions at a point of interest. One is then ready to use these derivatives to evaluate key quantities.

The low-order derivatives (1st and 2nd) should be coded explicitly, because the general formula has excessive overhead. The formulas are:

$$\frac{d^k \left(x-X_A\right)^n e^{-\alpha\left(x-x_A\right)^2}}{dx^k}$$

$$= \left(x-X_A\right)^n e^{-\alpha\left(x-X_A\right)^2} \qquad k=0$$

$$= e^{-\alpha\left(x-X_A\right)^2} \begin{cases} -2\alpha\left(x-X_A\right) & k=1; n=0 \\ n\left(x-X_A\right)^{n-1} - 2\alpha\left(x-X_A\right)^{n+1} & k=1; n\geq 1 \end{cases}$$

$$= e^{-\alpha\left(x-X_A\right)^2} \begin{cases} -2\alpha\left(x-X_A\right) & k=1; n=0 \\ \left(x-X_A\right)^{n-1}\left(n-2\alpha\left(x-X_A\right)^2\right) & k=1; n\geq 1 \end{cases}$$

$$= e^{-\alpha\left(x-X_A\right)^2} \begin{cases} 4\alpha^2\left(x-X_A\right)^2 - 2\alpha & k=2; n=0 \\ 4\alpha^2\left(x-X_A\right)^3 - 6\alpha\left(x-X_A\right) & k=2; n=1 \\ \left(x-X_A\right)^{n-2}\left[4\alpha^2\left(x-X_A\right)^4 - \alpha\left(4n+2\right)\left(x-X_A\right)^2 + n\left(n-1\right)\right] & k=2; n\geq 2 \end{cases}$$

Key properties to evaluate at points:

The input would be the basis set and a set of points. Molecular orbital coefficients, density matrix coefficients, and (more general) quantities expanded in the basis (e.g., 2-electron density matrix) can also be considered. I think basically there should be a fundamental level, which evaluates MOs and their derivatives,

$$\frac{\partial^{k_x+k_y+k_z}\psi_\mu\left(\mathbf{r}_p\right)}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}} = \sum_{a=1}^{N_{\text{basis}}} c_{\mu a} \frac{\partial^{k_x+k_y+k_z}\chi_a\left(\mathbf{r}_p\right)}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}} \tag{14}$$

Here $\chi_a\left(\mathbf{r}\right)$ is an atomic basis function, $\psi_\mu\left(\mathbf{r}\right)$ is a molecular orbital, and $\left\{\mathbf{r}_p\right\}_{p=1}^{N_{\text{points}}}$ is a set of 3-dimensional points at which the derivative of the molecular orbital is expanded. Note that while this is intended as a molecular orbital evaluation/differentiation, it is applicable to any function expanded in the atomic basis,

$$\frac{\partial^{k_x+k_y+k_z}\phi\left(\mathbf{r}_p\right)}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}} = \sum_{\mu=1}^{N_{\text{basis}}} f_a \frac{\partial^{k_x+k_y+k_z}\chi_a\left(\mathbf{r}_p\right)}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}} \tag{15}$$

Various derivatives of the 1-electron reduced density matrix,

$$\gamma\left(\mathbf{r},\mathbf{r}'\right) = \sum_{a=1}^{N_{\text{basis}}}\sum_{b=1}^{N_{\text{basis}}} \gamma_{ab}\chi_a\left(\mathbf{r}\right)\chi_b\left(\mathbf{r}'\right) \tag{16}$$

can be defined as

$$\left[\frac{\partial^{k_x+k_y+k_z}\partial^{l_x+l_y+l_z}\gamma\left(\mathbf{r},\mathbf{r}'\right)}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}\partial x'^{l_x}\partial y'^{l_y}\partial z'^{l_z}}\right]_{\mathbf{r}=\mathbf{r}'=\mathbf{r}_p} = \sum_{a=1}^{N_{\text{basis}}}\sum_{b=1}^{N_{\text{basis}}} \gamma_{ab}\left(\frac{\partial^{k_x+k_y+k_z}\chi_a\left(\mathbf{r}_p\right)}{\partial x^{k_x}\partial y^{k_y}\partial z^{k_z}}\right)\left(\frac{\partial^{l_x+l_y+l_z}\chi_b\left(\mathbf{r}_p\right)}{\partial x^{l_x}\partial y^{l_y}\partial z^{l_z}}\right) \tag{17}$$

I have assumed real basis functions in Eqs. (16) and (17). Note (again) that while this is intended as a way to evaluate arbitrary-order derivatives of the density matrix, in practice it also works for any function expanded in the form of Eq. (16), in the case where $\mathbf{r} = \mathbf{r}'$,

$$f\left(\mathbf{r}_p, \mathbf{r}_p\right) = \sum_{a=1}^{N_{\text{basis}}} \sum_{b=1}^{N_{\text{basis}}} \phi_{ab} \chi_a\left(\mathbf{r}_p\right) \chi_b\left(\mathbf{r}_p\right) \tag{18}$$

Given a set of 6-dimensional points, $\left\{\mathbf{r}_p, \mathbf{r}'_p\right\}_{p=1}^{N_{\text{points}}}$, one can evaluate the 1-electron reduced density matrix (or anything with the form of Eq. (16)) for those points,

$$f\left(\mathbf{r}_p, \mathbf{r}'_p\right) = \sum_{a=1}^{N_{\text{basis}}} \sum_{b=1}^{N_{\text{basis}}} \phi_{ab} \chi_a\left(\mathbf{r}_p\right) \chi_b\left(\mathbf{r}'_p\right) \tag{19}$$

and similarly for the derivatives of that function,

$$\frac{\partial^{k_x+k_y+k_z} \partial^{l_x+l_y+l_z} f\left(\mathbf{r}_p, \mathbf{r}'_p\right)}{\partial x^{k_x} \partial y^{k_y} \partial z^{k_z} \partial x'^{l_x} \partial y'^{l_y} \partial z'^{l_z}} = \sum_{a=1}^{N_{\text{basis}}} \sum_{b=1}^{N_{\text{basis}}} \gamma_{ab} \left(\frac{\partial^{k_x+k_y+k_z} \chi_a\left(\mathbf{r}_p\right)}{\partial x^{k_x} \partial y^{k_y} \partial z^{k_z}} \right) \left(\frac{\partial^{l_x+l_y+l_z} \chi_b\left(\mathbf{r}'_p\right)}{\partial x^{l_x} \partial y^{l_y} \partial z^{l_z}} \right) \tag{20}$$

For the expansions with the form (16)-(20), it may be useful to define one method for the usual symmetric case (where $\phi_{ab} = \phi_{ba}$) and another method for the asymmetric case. It would be OK to make the symmetric case the default, or even to test for this on the fly, as the evaluations are very expensive compared to a symmetry test. numpy can (I think; if it can't, then I'm sure scipy can) exploit symmetry and even sparsity. Whether or not we use symmetry/sparsity here is optional, obviously, and not essential for the first version of gbasis.

It is possible to imagine supporting *3d*-dimensional input points, $\left\{\mathbf{r}_p^{(1)}, \mathbf{r}_p^{(2)}, \ldots, \mathbf{r}_p^{(d)}\right\}_{p=1}^{N_{\text{pts}}}$, and then define the value and derivatives of quantities like:

$$f\left(\mathbf{r}_p^{(1)}, \mathbf{r}_p^{(2)}, \ldots, \mathbf{r}_p^{(d)}\right) = \sum_{a_1=1}^{N_{\text{basis}}} \sum_{a_2=1}^{N_{\text{basis}}} \cdots \sum_{a_d=1}^{N_{\text{basis}}} \phi_{a_1 a_2 \ldots a_d} \chi_{a_1}\left(\mathbf{r}_p^{(1)}\right) \cdots \chi_{a_d}\left(\mathbf{r}_p^{(d)}\right) \tag{21}$$

$$\begin{aligned} &f\left(\mathbf{r}_p^{(1)}, \mathbf{r}_p^{(2)}, \ldots \mathbf{r}_p^{(d)}; \mathbf{r}_p'^{(1)}, \mathbf{r}_p'^{(2)}, \ldots \mathbf{r}_p'^{(d)}\right) \\ &\quad = \sum_{a_1=1}^{N_{\text{basis}}} \sum_{a_2=1}^{N_{\text{basis}}} \cdots \sum_{a_d=1}^{N_{\text{basis}}} \sum_{b_1=1}^{N_{\text{basis}}} \sum_{b_2=1}^{N_{\text{basis}}} \cdots \sum_{b_d=1}^{N_{\text{basis}}} \phi_{a_1 a_2 \ldots a_d; b_1 b_2 \ldots b_d} \chi_{b_1}\left(\mathbf{r}_p^{(1)}\right) \cdots \chi_{b_d}\left(\mathbf{r}_p^{(d)}\right) \chi_{b_1}\left(\mathbf{r}_p'^{(1)}\right) \cdots \chi_{b_d}\left(\mathbf{r}_p'^{(d)}\right) \end{aligned} \tag{22}$$

For Eq. (22), both the cases where $\left\{\mathbf{r}_p'^{(k)} = \mathbf{r}_p^{(k)}\right\}_{k=1}^{d}$ (diagonal element) and the off-diagonal case should be supported, in analogy to Eqs. (17)-(18) and Eqs. (19)-(20), respectively. Obvious when the $\mathbf{r}_p'^{(k)} \neq \mathbf{r}_p^{(k)}$, then *d* has to be even. If it isn't easy/possible to support the *d*-point case, then the *2*-point (and analogous 4-point off-diagonal) case should be supported.

These helper functions provide the essential capacity to analyze density matrices (down the road) and the electron density, kinetic energy density, and related quantities in the near term. The key helper function is:

$$d^{k_x,k_y,k_z,l_x,l_y,l_z} \gamma\left(\mathbf{r}_p\right) \equiv \left[\frac{\partial^{k_x+k_y+k_z} \partial^{l_x+l_y+l_z} \gamma\left(\mathbf{r}, \mathbf{r}'\right)}{\partial x^{k_x} \partial y^{k_y} \partial z^{k_z} \partial x'^{l_x} \partial y'^{l_y} \partial z'^{l_z}} \right]_{\mathbf{r}=\mathbf{r}'=\mathbf{r}_p} = \sum_{a=1}^{N_{\text{basis}}} \sum_{b=1}^{N_{\text{basis}}} \gamma_{ab} \left(\frac{\partial^{k_x+k_y+k_z} \chi_a\left(\mathbf{r}_p\right)}{\partial x^{k_x} \partial y^{k_y} \partial z^{k_z}} \right) \left(\frac{\partial^{l_x+l_y+l_z} \chi_b\left(\mathbf{r}_p\right)}{\partial x^{l_x} \partial y^{l_y} \partial z^{l_z}} \right) \tag{23}$$

I think we should support (at least, easy to add to this list) the following. I'm assuming (pretty much throughout this whole section) that we are using real orbitals. It costs a factor of ~2 to allow for complex orbitals, but we should probably do that I guess.

evaluating the density:

$$\rho\left(\mathbf{r}_p\right) = \sum_{a=1}^{N_{\text{basis}}} \sum_{b=1}^{N_{\text{basis}}} \gamma_{ab} \chi_a\left(\mathbf{r}_p\right) \chi_b\left(\mathbf{r}_p\right) = d^{000000} \gamma\left(\mathbf{r}_p\right) \tag{24}$$

evaluating the gradient of the density:

$$\nabla\rho(\mathbf{r}_p) = 2\begin{bmatrix} \sum_{a=1}^{N_{\text{basis}}}\sum_{b=1}^{N_{\text{basis}}}\gamma_{ab}\left(\chi_a(\mathbf{r}_p)\frac{\partial\chi_b(\mathbf{r}_p)}{\partial x}\right) \\ \sum_{a=1}^{N_{\text{basis}}}\sum_{b=1}^{N_{\text{basis}}}\gamma_{ab}\left(\chi_a(\mathbf{r}_p)\frac{\partial\chi_b(\mathbf{r}_p)}{\partial y}\right) \\ \sum_{a=1}^{N_{\text{basis}}}\sum_{b=1}^{N_{\text{basis}}}\gamma_{ab}\left(\chi_a(\mathbf{r}_p)\frac{\partial\chi_b(\mathbf{r}_p)}{\partial z}\right) \end{bmatrix} = 2\begin{bmatrix} d^{000100}\gamma(\mathbf{r}_p) \\ d^{000010}\gamma(\mathbf{r}_p) \\ d^{000001}\gamma(\mathbf{r}_p) \end{bmatrix} \tag{25}$$

evaluating the Laplacian of the density:

$$\nabla^2\rho(\mathbf{r}_p) = 2\left(d^{100100}\gamma(\mathbf{r}_p) + d^{200100}\gamma(\mathbf{r}_p) + d^{010010}\gamma(\mathbf{r}_p) + d^{020000}\gamma(\mathbf{r}_p) + d^{001001}\gamma(\mathbf{r}_p) + d^{002000}\gamma(\mathbf{r}_p)\right) \tag{26}$$

evaluating the second-derivative tensor of the density:

It is helpful to define $\mathbf{e}_i$ as a vector that is "1" in position *i* and "0" everywhere else. So

$$\frac{\partial^2\rho(\mathbf{r}_p)}{\partial r_i\partial r_j} = 2\left(d^{\mathbf{e}_i+\mathbf{e}_j}\gamma + d^{\mathbf{e}_i+\mathbf{e}_{3+j}}\gamma\right) \qquad [r_0, r_1, r_2] = [x, y, z] \tag{27}$$

evaluate arbitrary-order derivatives of the density:

$$\begin{aligned} \frac{\partial^{L_x+L_y+L_z}\rho(\mathbf{r}_p)}{\partial x^{L_x}\partial y^{L_y}\partial z^{L_z}} &= \sum_{l_x=0}^{L_x}\sum_{l_y=0}^{L_y}\sum_{l_z=0}^{L_z}\binom{L_x}{l_x}\binom{L_y}{l_y}\binom{L_z}{l_z}\sum_{a=1}^{N_{\text{basis}}}\sum_{b=1}^{N_{\text{basis}}}\gamma_{ab}\left(\frac{\partial^{l_x+l_y+l_z}\chi_a(\mathbf{r}_p)}{\partial x^{l_x}\partial y^{l_y}\partial z^{l_z}}\right)\left(\frac{\partial^{L_x+L_y+L_z-l_x-l_y-l_z}\chi_b(\mathbf{r}_p)}{\partial x^{L_x-l_x}\partial y^{L_y-l_y}\partial z^{L_z-l_z}}\right) \\ &= \sum_{l_x=0}^{L_x}\sum_{l_y=0}^{L_y}\sum_{l_z=0}^{L_z}\binom{L_x}{l_x}\binom{L_y}{l_y}\binom{L_z}{l_z}\gamma_{\mu\nu}\left(d^{l_x,l_y,l_z;L_x-l_x,L_y-l_y,L_z-l_z}\gamma(\mathbf{r}_p)\right) \\ &= 2\,\mathrm{Re}\left[\sum_{l_x=0}^{\mathrm{floor}\left(\frac{1}{2}L_x\right)}\sum_{l_y=0}^{L_y}\sum_{l_z=0}^{L_z}\binom{L_x}{l_x}\binom{L_y}{l_y}\binom{L_z}{l_z}\left(d^{l_x,l_y,l_z;L_x-l_x,L_y-l_y,L_z-l_z}\gamma(\mathbf{r}_p)\right)\right] \end{aligned} \tag{28}$$

I'm not 100% sure about the last formula but the idea is that one can use the symmetry to reduce the cost by a factor of 2.

evaluate the stress tensor (various forms)

$$\boldsymbol{\sigma}_{\alpha,\beta}(\mathbf{r})_{i,j} \equiv -\frac{1}{2}\begin{bmatrix} \alpha\left(\frac{\partial^2\gamma(\mathbf{r},\mathbf{r}')}{\partial r_i\partial r_j'} + \frac{\partial^2\gamma(\mathbf{r},\mathbf{r}')}{\partial r_i'\partial r_j}\right) \\ -(1-\alpha)\left(\frac{\partial^2\gamma(\mathbf{r},\mathbf{r}')}{\partial r_i\partial r_j} + \frac{\partial^2\gamma(\mathbf{r},\mathbf{r}')}{\partial r_i'\partial r_j'}\right) \\ +\delta_{ij}\beta\nabla^2\rho(\mathbf{r}) \end{bmatrix}_{\mathbf{r}=\mathbf{r}'} \qquad i, j = \{x, y, z\}$$

$$\equiv -\frac{1}{2}\sum_{ab}\gamma_{ab}\begin{bmatrix} \alpha\left(\frac{\partial\chi_a(\mathbf{r})}{\partial r_i}\frac{\partial\chi_b(\mathbf{r})}{\partial r_j} + \frac{\partial\chi_a(\mathbf{r})}{\partial r_j}\frac{\partial\chi_b(\mathbf{r})}{\partial r_i}\right) \\ -(1-\alpha)\left(\chi_b(\mathbf{r})\frac{\partial^2\chi_a(\mathbf{r})}{\partial r_i\partial r_j} + \chi_a(\mathbf{r})\frac{\partial^2\chi_b(\mathbf{r})}{\partial r_i\partial r_j}\right) \end{bmatrix} - \tfrac{1}{2}\beta\delta_{ij}\nabla^2\rho(\mathbf{r}) \qquad i, j = \{x, y, z\}$$

$$\equiv \tfrac{1}{2}2\,\mathrm{Re}\left[\left(\alpha d^{\mathbf{e}_i+\mathbf{e}_{3+j}}\gamma(\mathbf{r}) - (1-\alpha)d^{\mathbf{e}_i+\mathbf{e}_j}\gamma(\mathbf{r})\right)\right] - \tfrac{1}{2}\beta\delta_{ij}\nabla^2\rho(\mathbf{r})$$

evaluating the Ehrenfest force (divergence of the stress tensor)

$$F_j(\mathbf{r}) = -\frac{1}{2}\sum_{a,b}\gamma_{ab}\left[\sum_{i=x,y,z}\left(\begin{array}{l}\dfrac{\partial^2\chi_a(\mathbf{r})}{\partial r_i^2}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_j}+\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_j}\dfrac{\partial^2\chi_b(\mathbf{r})}{\partial r_i^2}\\ \dfrac{\partial\chi_a(\mathbf{r})}{\partial r_i}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_i\partial r_j}+\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_i\partial r_j}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_i}\end{array}\right)\right] \qquad i,j=\{x,y,z\} \tag{29}$$

$$= -\mathrm{Re}\left[d^{2\mathbf{e}_i+\mathbf{e}_{3+j}}\gamma(\mathbf{r}_p)+d^{\mathbf{e}_i+\mathbf{e}_{3+i}+\mathbf{e}_{3+j}}\gamma(\mathbf{r}_p)\right]$$

evaluating the Ehrenfest Hessian (symmetrized, if the user prefers that)

$$H_{jk}(\mathbf{r}) = \frac{\partial F_j(\mathbf{r})}{\partial r_k} = \sum_{i=x,y,z}\frac{\partial}{\partial r_i\partial r_k}\sigma_{ij}(\mathbf{r}) \qquad j,k=x,y,z$$

$$H_{jk}(\mathbf{r}) = -\frac{1}{2}\sum_{a,b}\gamma_{ab}\left[\sum_{i=x,y,z}\left(\begin{array}{l}\dfrac{\partial^2\chi_a(\mathbf{r})}{\partial r_i^2\partial r_k}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_j}+\dfrac{\partial^2\chi_a(\mathbf{r})}{\partial r_i^2}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_j\partial r_k}\\ +\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_j\partial r_k}\dfrac{\partial^2\chi_b(\mathbf{r})}{\partial r_i^2}+\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_j}\dfrac{\partial^2\chi_b(\mathbf{r})}{\partial r_i^2\partial r_k}\\ +\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_i\partial r_k}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_i\partial r_j}+\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_i\partial r_j\partial r_k}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_i}\\ +\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_i}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_i\partial r_j\partial r_k}+\dfrac{\partial\chi_a(\mathbf{r})}{\partial r_i\partial r_j}\dfrac{\partial\chi_b(\mathbf{r})}{\partial r_i\partial r_k}\end{array}\right)\right] \qquad j,k=\{x,y,z\} \tag{30}$$

$$= -\mathrm{Re}\left[d^{2\mathbf{e}_i+\mathbf{e}_k+\mathbf{e}_{3+j}}\gamma(\mathbf{r}_p)+d^{2\mathbf{e}_i+\mathbf{e}_{3+j}+\mathbf{e}_{3+k}}\gamma(\mathbf{r}_p)+d^{\mathbf{e}_i+\mathbf{e}_k+\mathbf{e}_{3+i}+\mathbf{e}_{3+j}}\gamma(\mathbf{r}_p)+d^{\mathbf{e}_i+\mathbf{e}_j+\mathbf{e}_k+\mathbf{e}_{3+i}}\gamma(\mathbf{r}_p)\right]$$

This matrix is not symmetric, so it is often convenient to add its transpose and divide by two.

evaluate the positive-definite kinetic energy density

$$t_+(\mathbf{r}) \equiv \left[\tfrac{1}{2}\nabla_{\mathbf{r}}\cdot\nabla_{\mathbf{r}'}\gamma(\mathbf{r},\mathbf{r}')\right]_{\mathbf{r}=\mathbf{r}'=\mathbf{r}_p} = \tfrac{1}{2}\left(d^{100100}\gamma(\mathbf{r}_p)+d^{010010}\gamma(\mathbf{r}_p)+d^{001001}\gamma(\mathbf{r}_p)\right) \tag{31}$$

evaluate general forms of the kinetic energy density

$$t_\alpha(\mathbf{r}) \equiv t_+(\mathbf{r})+\alpha\nabla^2\rho(\mathbf{r}) \tag{32}$$

Xiao has done work on this and knows key formulas, so I didn't bother to list everything….

Integrals of Gaussians:

A separate set of notes on overlap integrals for arbitrary numbers of Gaussians (using a different algorithm to this one) is attached. Farnaz has implemented that.

In all cases, there are ways to *screen* integrals. There is more on that at the end of the notes, but the general idea is that one can screen at the atomic loop level (loop (a), above) and also at the shell level (loop c, above).

Overlap and Multipole Moment Integrals:

The fundamental multipole moment integral has the form

$$S_{ab}^{efg} \equiv \int\xi_a(\mathbf{r})(x-X_C)^e(y-Y_C)^f(z-Z_C)^g\,\xi_b(\mathbf{r})\,d\mathbf{r} = \int\left\{\begin{array}{l}\left[(x-X_A)^{a_x}(y-Y_A)^{a_y}(z-Z_A)^{a_z}\,e^{-\alpha|\mathbf{r}-\mathbf{R}_A|^2}\right]\\ \times(x-X_C)^e(y-Y_C)^f(z-Z_C)^g\\ \times\left[(x-X_B)^{b_x}(y-Y_B)^{b_y}(z-Z_B)^{b_z}\,e^{-\beta|\mathbf{r}-\mathbf{R}_B|^2}\right]\end{array}\right\}d\mathbf{r} \tag{33}$$

We define the helper function,

$$s_{ij}^{e} \equiv \int_{-\infty}^{\infty} \left(x - X_A\right)^i \left(x - X_B\right)^j \left(x - X_C\right)^e \exp\left(-\alpha\left(x - X_A\right)^2\right) \exp\left(-\beta\left(x - X_B\right)^2\right) dx \tag{34}$$

and so

$$S_{ab}^{efg} = s_{a_x b_x}^{e} s_{a_y b_y}^{f} s_{a_z b_z}^{g} \tag{35}$$

Multiplying the raw integral in Eq. (35) by the normalization coefficients for the primitive Gaussians, contracting the normalized primitives to form contracted Cartesians, and (optionally) converting the Cartesians to sphericals gives the key AO integral,

$$\int \chi_a(\mathbf{r}) \chi_b(\mathbf{r}) \left(x - X_C\right)^e \left(y - Y_C\right)^f \left(z - Z_C\right)^g d\mathbf{r} \tag{36}$$

Notice that the $e = f = g = 0$ case is the normal overlap integral.

We can evaluate s_{ij}^{e} using a standard recursion due to Obara and Saika,

$$\begin{aligned} s_{i+1,j}^{e} &= X_{PA} s_{ij}^{e} + \tfrac{1}{2p}\left(i s_{i-1,j}^{e} + j s_{i,j-1}^{e} + e s_{ij}^{e-1}\right) \\ s_{i,j+1}^{e} &= X_{PB} s_{ij}^{e} + \tfrac{1}{2p}\left(i s_{i-1,j}^{e} + j s_{i,j-1}^{e} + e s_{ij}^{e-1}\right) \\ s_{ij}^{e+1} &= X_{PC} s_{ij}^{e} + \tfrac{1}{2p}\left(i s_{i-1,j}^{e} + j s_{i,j-1}^{e} + e s_{ij}^{e-1}\right) \end{aligned} \tag{37}$$

The last term in this expression is zero, and the last equations in this expression is unnecessary, when one is dealing with overlaps (not moments).

In Eq. (37), p is the sum of the exponents of the Gaussians,

$$p = \alpha + \beta \tag{38}$$

X_P is the weighted average-center of the Gaussians,

$$X_P = \frac{\alpha X_A + \beta X_B}{p} \tag{39}$$

the (signed) distance to the moment center is:

$$\begin{aligned} X_{PA} &= X_P - X_A \\ X_{PB} &= X_P - X_B \\ X_{PC} &= X_P - X_C \end{aligned} \tag{40}$$

Finally, μ is ½ the harmonic mean of the Gaussian exponents,

$$\mu = \frac{\alpha\beta}{p} \tag{41}$$

The recursion in Eq. (37) is initialized using the overlap of the *s*-type Gaussians,

$$s_{00}^{0} = \sqrt{\frac{\pi}{p}} \exp\left(-\mu\left(X_A - X_B\right)^2\right). \tag{42}$$

and we take the convention that $s_{-1,0}^{0} = s_{i,-1}^{0} = s_{i,j}^{-1} = 1$, just so you don't risk an undefined quantity in the recursion.

Now there are *many* integrals to compute. In general, if the shells of interest have angular momentum $\ell(i)$ and $\ell(j)$, one needs to consider $0 \le i \le \ell(i)$ and $0 \le j \le \ell(j)$. Let us assume, without loss of generality, that $\ell(i) \le \ell(j)$. (This can always been done since $s_{ij}^{e} = s_{ji}^{e}$.) Then we first do the first recursion

`for i=0,1,…` $\ell(i)$`-1`

$$s_{i+1,0}^{0} = X_{PA} s_{i0}^{0} + \left(\tfrac{1}{2p}\right)\left(i s_{i-1,0}^{0}\right)$$

Next we do the second recursion; we can do this in a "vectorized" way since we know s_{i0}^{0} for $0 \le i \le \ell(i)$. So

`for j=0,1,…` $\ell(j)$`-1`

$$s_{i,j+1}^{0} = X_{PB} s_{ij}^{0} + \tfrac{1}{2p}\left(i s_{i-1,j}^{0} + j s_{i,j-1}^{0}\right) \qquad 0 \le i \le \ell(i)$$

This gives us the overlap integral. Now, finally, we generate the moment with the final recursion, again doing it in a "vectorized" way. Ordinarily we would want all moments up to some order, and in this way we generate all moments up to some order automatically,

for d=0,1,…e-1

$$s_{ij}^{d+1} = X_{PC} s_{ij}^{d} + \tfrac{1}{2p}\left(i s_{i-1,j}^{d} + j s_{i,j-1}^{d} + e s_{ij}^{d-1} \right) \qquad 0 \le i \le \ell(i); 0 \le j \le \ell(j) \tag{43}$$

Once the key quantities are computed, they are multiplied by contraction coefficients/normalizations and then transformed to spherical basis functions (if desired).

To convert Cartesian 2^q-pole moments (that is, a set of moments with $q = e + f + g$, where q = 0 is the monopole/charge, q = 1 is the dipole, q = 2 is the quadrupole, q = 3 is the octupole, etc.) to Spherical 2^q-pole moments one uses exactly the same transformation as the one to convert Cartesian basis functions to spherical basis functions.

Kinetic Energy and Related Quantities.

Looking at Eqs. 9.3.35-9.3.37 and Eqs. 9.3.30 and 9.3.31 in Helgaker, the momentum, angular momentum, and kinetic energy can be evaluated directly from the overlap integrals, the only requirement is that you evaluate to 2 higher orders. I.e., you need to evaluate all the way to $s^0_{\ell(i)+2,\ell(j)}$. It is handy to use the symmetry here, also. Specifically, for the expectation value of the momentum,

$$\left\langle \hat{\mathbf{p}} \right\rangle = \int \xi_a(\mathbf{r}) - i\nabla \xi_b(\mathbf{r})\, d\mathbf{r} = -i \begin{bmatrix} d^1_{a_x b_x} s^0_{a_y b_y} s^0_{a_z b_z} \\ s^0_{a_x b_x} d^1_{a_y b_y} s^0_{a_z b_z} \\ s^0_{a_x b_x} s^0_{a_y b_y} d^1_{a_z b_z} \end{bmatrix} \tag{44}$$

for the expectation value of the angular momentum

$$\left\langle \hat{\mathbf{L}} \right\rangle = \int \xi_a(\mathbf{r}) \left(-i\mathbf{r} \times \nabla \right) \xi_b(\mathbf{r})\, d\mathbf{r} = -i \begin{bmatrix} s^0_{a_x b_x} s^1_{a_y b_y} d^1_{a_z b_z} - s^0_{a_x b_x} d^1_{a_y b_y} s^1_{a_z b_z} \\ d^1_{a_x b_x} s^0_{a_y b_y} s^1_{a_z b_z} - s^1_{a_x b_x} s^0_{a_y b_y} d^1_{a_z b_z} \\ s^1_{a_x b_x} d^1_{a_y b_y} s^0_{a_z b_z} - d^1_{a_x b_x} s^1_{a_y b_y} s^0_{a_z b_z} \end{bmatrix} \tag{45}$$

and the kinetic energy (a scalar)

$$\left\langle \hat{T} \right\rangle = \int \xi_a(\mathbf{r}) \left(\tfrac{-1}{2} \nabla^2 \right) \xi_b(\mathbf{r})\, d\mathbf{r} = \tfrac{-1}{2} \left(d^2_{a_x b_x} s^0_{a_y b_y} s^0_{a_z b_z} + s^0_{a_x b_x} d^2_{a_y b_y} s^0_{a_z b_z} + s^0_{a_x b_x} s^0_{a_y b_y} d^2_{a_z b_z} \right) \tag{46}$$

where

$$\begin{aligned} d^1_{ij} &= 2\alpha s^0_{i+1,j} - i s^0_{i-1,j} \\ d^2_{ij} &= 4\alpha^2 s^0_{i+2,j} - 2\alpha(2i+1) s^0_{ij} + i(i-1) s^0_{i-2,j} \end{aligned} \tag{47}$$

Expectation Values of Derivatives:

For quantities like

$$\begin{aligned} D^{efg}_{ab} &\equiv \int \xi_a(\mathbf{r}) \left[\frac{\partial^{e+f+g}}{\partial x^e \partial y^f \partial z^g} \right] \xi_b(\mathbf{r})\, d\mathbf{r} \\ &= \int \left\{ \begin{aligned} &\left[(x - X_A)^{a_x} (y - Y_A)^{a_y} (z - Z_A)^{a_z} e^{-\alpha |\mathbf{r} - \mathbf{R}_A|^2} \right] \\ &\times \frac{\partial^{e+f+g}}{\partial x^e \partial y^f \partial z^g} \\ &\times \left[(x - X_B)^{b_x} (y - Y_B)^{b_y} (z - Z_B)^{b_z} e^{-\beta |\mathbf{r} - \mathbf{R}_B|^2} \right] \end{aligned} \right\} d\mathbf{r} \end{aligned} \tag{48}$$

one can use a similar trick to what was done for the moments. One defines We define the helper function,

$$d_{ij}^{e} \equiv \int_{-\infty}^{\infty} \left(x - X_A\right)^i \exp\left(-\alpha\left(x - X_A\right)^2\right) \frac{d^e}{dx^e} \left(x - X_B\right)^j \exp\left(-\beta\left(x - X_B\right)^2\right) dx \tag{49}$$

and so

$$D_{ab}^{efg} = d_{a_x b_x}^{e} d_{a_y b_y}^{f} d_{a_z b_z}^{g} \tag{50}$$

Multiplying the raw integral in Eq. (35) by the normalization coefficient, contracting the integrals to form contracted Cartesians, and converting the Cartesians to sphericals gives the key AO integral,

$$\int \chi_a(\mathbf{r}) \frac{\partial^{e+f+g}}{\partial x^e \partial y^f \partial z^g} \chi_b(\mathbf{r}) d\mathbf{r} \tag{51}$$

Notice that the $e = f = g = 0$ case is the normal overlap integral.

We can evaluate d_{ij}^{e} using a standard recursion due to Obara and Saika,

$$\begin{aligned} d_{i+1,j}^{e} &= X_{PA} d_{ij}^{e} + \tfrac{1}{2p}\left(i d_{i-1,j}^{e} + j d_{i,j-1}^{e} - 2\beta e \cdot d_{ij}^{e-1}\right) \\ d_{i,j+1}^{e} &= X_{PB} d_{ij}^{e} + \tfrac{1}{2p}\left(i d_{i-1,j}^{e} + j d_{i,j-1}^{e} + 2\alpha e \cdot d_{ij}^{e-1}\right) \\ d_{ij}^{e+1} &= 2\alpha d_{i+1,j}^{e} - i d_{i-1,j}^{e} \end{aligned} \tag{52}$$

Here α and β are the Gaussian exponents for the basis functions *i* and *j*. The other notation is the same as in Eqs. (38)-(42). The same basic strategy can be used as was used for the overlap, but if one already knows the overlaps, then one can use just the final relationship in Eq. (52) because $d_{ij}^{0} = s_{ij}^{0}$. It is safe to assume that the overlaps are known, as I can't imagine a case where you need the derivatives and not the overlaps. So initialize $d_{-1,j}^{0} = d_{\ell(i),-1}^{0} = 1; d_{i,j}^{0} = s_{i,j}^{0}$ for $0 \le i \le \ell(i); 0 \le j \le \ell(j)$. Now I need to compute $d_{\ell+1,j}^{0}, d_{\ell(i)+2,j}^{0}, \ldots, d_{\ell(i)+e,j}^{0} = s_{\ell+1,j}^{0}, s_{\ell(i)+2,j}^{0}, \ldots, s_{\ell(i)+e,j}^{0}$ using $d_{\ell(i)+\varepsilon+1,j}^{0} = X_{PA} d_{\ell(i)+\varepsilon,j}^{0} + \tfrac{1}{2p}\left(\ell(i) d_{\ell(i)+\varepsilon-1,j}^{0} + j d_{\ell(i)+\varepsilon,j-1}^{0}\right)$ for all $0 \le j \le \ell(j)$ and $\varepsilon = 0,1,\ldots e-1$. Note that this recursion can be done on the whole array of *j* values at once, though the ε's have to be treated sequentially. Finally, I compute the elements I need using the last Eq. (52). I.e., for all $\varepsilon = 0,1,\ldots e-1$, $d_{ij}^{\varepsilon+1} = 2\alpha_i d_{i+1,j}^{\varepsilon} - i d_{i-1,j}^{\varepsilon}$. All *j* can be treated at the same time, as can all $i = \ell(i), \ell(i)+1, \ldots, \ell(i)+e-\varepsilon$

Interaction with a point charge (electrostatic potential, electron-nuclear attraction, and similar):

Consider a Gaussian integral with the form

$$V^{(0)}(\mathbf{a};\mathbf{b}) \equiv \int \xi_a(\mathbf{r}) \xi_b(\mathbf{r}) g\left(\left|\mathbf{r} - \mathbf{R}_C\right|\right) d\mathbf{r} \tag{53}$$

The most important case is the case where $g\left(\left|\mathbf{r} - \mathbf{R}_C\right|\right) = -\dfrac{Z_C}{\left|\mathbf{r} - \mathbf{R}_C\right|}$ (electron-nuclear attraction potential, but also the electrostatic potential (change $Z_A \to 1$) and interaction with an external point charge (change $Z_A \to q_A$). Recall that the unnormalized Cartesian Gaussian primitive basis functions have the form

$$\begin{aligned} \xi_{\mathbf{a}}(\mathbf{r}) &\equiv \left(x - X_A\right)^{a_x} \left(y - Y_A\right)^{a_y} \left(z - Z_A\right)^{a_z} \exp\left(-\alpha\left|\mathbf{r} - \mathbf{R}_A\right|^2\right) \\ \xi_{\mathbf{b}}(\mathbf{r}) &\equiv \left(x - X_B\right)^{b_x} \left(y - Y_B\right)^{b_y} \left(z - Z_B\right)^{b_z} \exp\left(-\beta\left|\mathbf{r} - \mathbf{R}_B\right|^2\right) \end{aligned} \tag{54}$$

(This notation is basically from Ahlrichs, because I like his notation better than others here.) The integral in Eq. (53) I write in shorthand,

$$V^{(0)}(\mathbf{a};\mathbf{b}) = V^{(0)}\left(a_x a_y a_z; b_x b_y b_z\right) \tag{55}$$

and I will introduce the auxiliary quantities, $V^{(N)}(\mathbf{a};\mathbf{b})$. Since $V^{(k)}(\mathbf{a};\mathbf{b}) = V^{(k)}(\mathbf{b};\mathbf{a})$, and <u>it turns out to be most efficient to choose the *most contracted* function in the first position (basis function **a**)</u>. Second, because $V^{0}(\mathbf{a}|\mathbf{b}) = V^{0}(\mathbf{b}|\mathbf{a})$ it is favorable to compute the case where $\ell_a \ge \ell_b$. So if the two functions have the same level of contraction, putting the highest angular momentum in the first position is an advantage.

One should first do overlap screening (see below and attached notes).

Even if the shell is predicted to be important, one should still check the overlap integral, $S_{ab} = \int \chi_a(\mathbf{r})\chi_b(\mathbf{r})d\mathbf{r}$ and only perform the following rather tedious procedure if that integral is larger than some threshold. Using the notation,

$$\begin{aligned} p &\equiv \alpha + \beta \\ \mathbf{R}_P &= \frac{\alpha\mathbf{R}_A + \beta\mathbf{R}_B}{p} \\ \mu &= \frac{\alpha\beta}{p} \\ \mathbf{R}_{PA} &= \mathbf{R}_P - \mathbf{R}_A \\ \mathbf{R}_{PB} &= \mathbf{R}_P - \mathbf{R}_B \\ \mathbf{R}_{AB} &= \mathbf{R}_A - \mathbf{R}_B \\ \mathbf{R}_{PC} &= \mathbf{R}_P - \mathbf{R}_C \end{aligned} \tag{56}$$

the recursion starts with:

$$V^{(N)}(\mathbf{0};\mathbf{0}) = \exp\left(-\mu|\mathbf{R}_A - \mathbf{R}_B|^2\right) G_N\left(p, p|\mathbf{R}_P - \mathbf{R}_C|^2\right) \tag{57}$$

The evaluation of the Boys function will be considered later, but for the Coulomb case one is using Eq. (39) in Ahlrich's paper, which can be rewritten as a confluent hypergeometric function (cf. eq. 9.8.39 of Helgaker) or as an incomplete gamma function (eq. 9.8.20 of Helgaker). There are now *two* recursions we need. We basically have

$$\begin{aligned} V^{(m)}(a_x+1,a_y,a_z;\mathbf{0}) &= (X_P - X_A)V^{(m)}(a_x,a_y,a_z;\mathbf{0}) - (X_P - X_C)V^{(m+1)}(a_x,a_y,a_z;\mathbf{0}) \\ &\quad + \frac{a_x}{2p}\left[V^{(m)}(a_x-1,a_y,a_z;\mathbf{0}) - V^{(m+1)}(a_x-1,a_y,a_z;\mathbf{0})\right] \\ &= X_{PA}V^{(m)}(a_x,a_y,a_z;\mathbf{0}) - X_{PC}V^{(m+1)}(a_x,a_y,a_z;\mathbf{0}) \\ &\quad + \frac{a_x}{2p}\left[V^{(m)}(a_x-1,a_y,a_z;\mathbf{0}) - V^{(m+1)}(a_x-1,a_y,a_z;\mathbf{0})\right] \end{aligned} \tag{58}$$

and then we transfer angular momentum to the other function using

$$V^{(m)}(\mathbf{a};b_x+1,b_y,b_z) = V^{(m)}(a_x+1,a_y,a_z;b_x,b_y,b_z) + X_{AB}V^{(m)}(a_x,a_y,a_z;b_x,b_y,b_z) \tag{59}$$

Obviously there are similar relations for *y* and *z*.

Here is a possible implementation. I assume that the angular momentum of orbital *a* is ℓ_a and for orbital *b* is ℓ_b. As mentioned, it is most efficient to choose $\ell_a \geq \ell_b$. The idea is to perform a vertical recursion, then contract, then perform a horizontal recursion.

Vertical Recursion

1. Initialize $V^{(m)}(0,0,0;0,0,0)$ for $m = \ell_a - 1, \ell_a + 1, 2, \ldots \ell_a + \ell_b$ using Eq. (57). This can be done using a recursion relation (sequentially) or explicitly in a vectorized fashion (computing all terms at the same time). The vectorized implementation is easier to code, and may not be much less efficient, and the recursion relation is only known for certain special choices of $g(|\mathbf{r}-\mathbf{r}'|)$ anyway. All other values of $V^{(m)}(a_x,a_y,a_z;\mathbf{0}) = 0$ can be initialized to zero.
2. The basic strategy we use is to first form all relevant choices for one nonzero index, then all relevant choices for two nonzero indices, then all relevant choices for three nonzero indices.

3. For one nonzero index, we need all values of the form $V^{(m)}(a,0,0;\mathbf{0})$ where $\ell_a \le a \le \ell_a + \ell_b$ and $0 \le m + a \le \ell_a + \ell_b$ (plus permutations of *x*,*y*,*z*).

for $a = 1,\ldots \ell_a + \ell_b$

for all $m = 0,1,\ldots,\ell_a + \ell_b - a$

!!Note that all *m* can be treated at once (vectorized) and that the next three lines can be treated simultaneously/vectorized also.

$$V^{(m)}(a,0,0;\mathbf{0}) = X_{PA}V^{(m)}(a-1,0,0;\mathbf{0}) - X_{PC}V^{(m+1)}(a-1,0,0;\mathbf{0}) + \frac{a-1}{2p}\left[V^{(m)}(a-2,0,0;\mathbf{0}) - V^{(m+1)}(a-2,0,0;\mathbf{0})\right.$$

$$V^{(m)}(0,a,0;\mathbf{0}) = Y_{PA}V^{(m)}(0,a-1,0;\mathbf{0}) - Y_{PC}V^{(m+1)}(0,a-1,0;\mathbf{0}) + \frac{a-1}{2p}\left[V^{(m)}(0,a-2,0;\mathbf{0}) - V^{(m+1)}(0,a-2,0;\mathbf{0})\right]$$

$$V^{(m)}(0,0,a;\mathbf{0}) = Z_{PA}V^{(m)}(0,0,a-1;\mathbf{0}) - Z_{PC}V^{(m+1)}(0,0,a-1;\mathbf{0}) + \frac{a-1}{2p}\left[V^{(m)}(0,0,a-2;\mathbf{0}) - V^{(m+1)}(0,0,a-2;\mathbf{0})\right.$$

4. Two nonzero indices. We make the recursions as short as we can. The key is that we need all possible choices of indices where $\ell_a \le a_1 + a_2 \le \ell_a + \ell_b$, and we choose $a_1 \ge a_2$ for efficiency (to shorten the length of the recursions). The outermost loops (and all highlighted loops, in general) can be vectorized because what happens inside it is independent of the other loops. The goal is to make all values of $V^{(m)}(a_1, a_2 \le a_1, 0)$ for which $\ell_a \le a_1 + a_2 \le \ell_a + \ell_b$ and $0 \le m + a_1 + a_2 \le \ell_a + \ell_b$ (plus permutations of *x*,*y*,*z*).

for $q = \ell_a, \ell_a + 1,\ldots,\ell_a + \ell_b$

for $a_1 = \left[\text{floor}\left(\tfrac{1}{2}q\right) + \text{mod}(q,2)\right],\ldots,q-1$

for $a_2 = 1,\ldots,q - a_1$

for all $m = 0,1,\ldots \ell_a + \ell_b - a_1 - a_2$ (this loop can be vectorized)

$$V^{(m)}(a_2,a_1,0;\mathbf{0}) = X_{PA}V^{(m)}(a_2-1,a_1,0;\mathbf{0}) - X_{PC}V^{(m+1)}(a_2-1,a_1,0;\mathbf{0}) + \frac{a_2-1}{2p}\left[V^{(m)}(a_2-2,a_1,0;\mathbf{0}) - V^{(m+1)}(a_2-2,a_1,0;\mathbf{0})\right]$$

$$V^{(m)}(a_2,0,a_1;\mathbf{0}) = X_{PA}V^{(m)}(a_2-1,0,a_1;\mathbf{0}) - X_{PC}V^{(m+1)}(a_2-1,0,a_1;\mathbf{0}) + \frac{a_2-1}{2p}\left[V^{(m)}(a_2-2,0,a_1;\mathbf{0}) - V^{(m+1)}(a_2-2,0,a_1;\mathbf{0})\right]$$

$$V^{(m)}(a_1,a_2,0;\mathbf{0}) = Y_{PA}V^{(m)}(a_1,a_2-1,0;\mathbf{0}) - Y_{PC}V^{(m+1)}(a_1,a_2-1,0;\mathbf{0}) + \frac{a_2-1}{2p}\left[V^{(m)}(a_1,a_2-2,0;\mathbf{0}) - V^{(m+1)}(a_1,a_2-2,0;\mathbf{0})\right]$$

$$V^{(m)}(0,a_2,a_1;\mathbf{0}) = Y_{PA}V^{(m)}(0,a_2-1,a_1;\mathbf{0}) - Y_{PC}V^{(m+1)}(0,a_2-1,a_1;\mathbf{0}) + \frac{a_2-1}{2p}\left[V^{(m)}(0,a_2-2,a_1;\mathbf{0}) - V^{(m+1)}(0,a_2-2,a_1;\mathbf{0})\right]$$

$$V^{(m)}(a_1,0,a_2;\mathbf{0}) = Z_{PA}V^{(m)}(a_1,0,a_2-1;\mathbf{0}) - Z_{PC}V^{(m+1)}(a_1,0,a_2-1;\mathbf{0}) + \frac{a_2-1}{2p}\left[V^{(m)}(a_1,0,a_2-2;\mathbf{0}) - V^{(m+1)}(a_1,0,a_2-2;\mathbf{0})\right]$$

$$V^{(m)}(0,a_1,a_2;\mathbf{0}) = Z_{PA}V^{(m)}(0,a_1,a_2-1;\mathbf{0}) - Z_{PC}V^{(m+1)}(0,a_1,a_2-1;\mathbf{0}) + \frac{a_2-1}{2p}\left[V^{(m)}(0,a_1,a_2-2;\mathbf{0}) - V^{(m+1)}(0,a_1,a_2-2;\mathbf{0})\right]$$

5. Three nonzero indices. The "for" loops can be vectorized and are just an explicit way to generate all integers for which $\ell_a \le a_1 + a_2 + a_3 \le \ell_a + \ell_b$. At this stage we need to generate all integrals $V^{(m)}(a_1,a_2,a_3)$ with $\ell_a \le a_1 + a_2 + a_3 \le \ell_a + \ell_b$ and $0 \le m + a_1 + a_2 + a_3 \le \ell_a + \ell_b$

for $q = \ell_a, \ell_a + 1,\ldots,\ell_a + \ell_b$

for $a_1 = \left[\text{floor}\left(\tfrac{1}{3}q\right) + \begin{Bmatrix} 1 \text{ if } \text{mod}(q,3) \ne 0 \\ 0 \text{ if } \text{mod}(q,3) = 0 \end{Bmatrix}\right],\ldots,q-2$

for $a_2 = 1,\ldots, q - a_1 - 1$

for $a_3 = 1,\ldots, q - a_1 - a_2$

for all $m = 0,1,\ldots \ell_a + \ell_b - a_1 - a_2 - a_3$ (this loop can be vectorized)

$$V^{(m)}(a_3,a_1,a_2;\mathbf{0}) = X_{PA}V^{(m)}(a_3-1,a_1,a_2;\mathbf{0}) - X_{PC}V^{(m+1)}(a_3-1,a_1,a_2;\mathbf{0}) + \frac{a_3-1}{2p}\left[V^{(m)}(a_3-2,a_1,a_2;\mathbf{0}) - V^{(m+1)}(a_3-2,a_1,a_2\right.$$

$$V^{(m)}(a_3,a_2,a_1;\mathbf{0}) = X_{PA}V^{(m)}(a_3-1,a_2,a_1;\mathbf{0}) - X_{PC}V^{(m+1)}(a_3-1,a_2,a_1;\mathbf{0}) + \frac{a_3-1}{2p}\left[V^{(m)}(a_3-2,a_2,a_1;\mathbf{0}) - V^{(m+1)}(a_3-2,a_2,a_1\right.$$

$$V^{(m)}(a_1,a_3,a_2;\mathbf{0}) = Y_{PA}V^{(m)}(a_1,a_3-1,a_2;\mathbf{0}) - Y_{PC}V^{(m+1)}(a_1,a_3-1,a_2;\mathbf{0}) + \frac{a_3-1}{2p}\left[V^{(m)}(a_1,a_3-2,a_2;\mathbf{0}) - V^{(m+1)}(a_1,a_3-2,a_2;\mathbf{0}\right.$$

$$V^{(m)}(a_2,a_3,a_1;\mathbf{0}) = Y_{PA}V^{(m)}(a_2,a_3-1,a_1;\mathbf{0}) - Y_{PC}V^{(m+1)}(a_2,a_3-1,a_1;\mathbf{0}) + \frac{a_3-1}{2p}\left[V^{(m)}(a_2,a_3-2,a_1;\mathbf{0}) - V^{(m+1)}(a_2,a_3-2,a_1;\mathbf{0}\right.$$

$$V^{(m)}(a_1,a_2,a_3;\mathbf{0}) = Z_{PA}V^{(m)}(a_1,a_2,a_3-1;\mathbf{0}) - Z_{PC}V^{(m+1)}(a_1,a_2,a_3-1;\mathbf{0}) + \frac{a_3-1}{2p}\left[V^{(m)}(a_1,a_2,a_3-2;\mathbf{0}) - V^{(m+1)}(a_1,a_2,a_3-2;\right.$$

$$V^{(m)}(a_2,a_1,a_3;\mathbf{0}) = Z_{PA}V^{(m)}(a_2,a_1,a_3-1;\mathbf{0}) - Z_{PC}V^{(m+1)}(a_2,a_1,a_3-1;\mathbf{0}) + \frac{a_3-1}{2p}\left[V^{(m)}(a_2,a_1,a_3-2;\mathbf{0}) - V^{(m+1)}(a_2,a_1,a_3-2;\right.$$

Contract. I.e., now one moves to contracted basis functions. So in our notation, $\xi_a, \xi_b \to \chi_a, \chi_b$.

=====got to here=====

Horizontal Recursion:

One has all integrals of the form $V^{(0)}(a_x,a_y,a_z;\mathbf{0})$ with $\ell_a \le a_x + a_y + a_z \le \ell_a + \ell_b$ at this stage. We finally generate our integrals using the horizontal recursion, specifically,

$$V^{(0)}(\mathbf{a};b_x+1,b_y,b_z) = V^{(0)}(a_x+1,a_y,a_z;b_x,b_y,b_z) + X_{AB}V^{(0)}(\mathbf{a};b_x,b_y,b_z) \tag{60}$$

The first step is to generate all sets of (a_x,a_y,a_z) with $a_x + a_y + a_z = \ell_a$. This can be done explicitly as follows, and this entire loop (highlighted text) can be vectorized, because "order doesn't matter" for this loop.

for $a_x = 0,1,\ldots,\ell_a$

for $a_y = 0,1,\ldots,\ell_a - a_x$

$a_z = \ell_a - a_x - a_y$

In obvious notation I specify the result from the first loop as $\mathbf{a} = (a_x,a_y,a_z)$.

The second step is to generate all sets of $b_x + b_y + b_z = \ell_b$. It turns out to be a *very* similar trick to the vertical recursion.

1. initialize $V^{(0)}(\mathbf{a};b_x,b_y,b_z) = 0$ when any of $\{b_x,b_y,b_z\} = -1$ for all **a**.
2. The basic strategy we use is to first form all possible choices for one nonzero index, then all possible choices for two nonzero indices, then all possible choices for three nonzero indices.
3. One nonzero index:

$m_{\max} = \ell_b$

$b = -1$

while $m_{\max} \ge 1$

$m_{\max} = m_{\max} - 1$

$b = b + 1$

for all $m = 0,1,\ldots m_{\max}$

!!Note that all m can be treated at once (vectorized) and that the next three lines can be treated simultaneously/vectorized also.

$$V^{(0)}\left(a_x+m,a_y,a_z;b+1,0,0\right)=V^{(0)}\left(a_x+m+1,a_y,a_z;b,0,0\right)+X_{AB}V^{(0)}\left(a_x+m,a_y,a_z;b,0,0\right)$$
$$V^{(0)}\left(a_x,a_y+m,a_z;0,b+1,0\right)=V^{(0)}\left(a_x,a_y+m+1,a_z;0,b,0\right)+Y_{AB}V^{(0)}\left(a_x,a_y+m,a_z;0,b,0\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;0,0,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;0,0,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;0,0,b\right)$$

4. Two nonzero indices. We make the recursions as short as we can. We use the integer division, floor$\left[\frac{1}{2}\ell_b\right]$ to determine how to do this. We first generate pairs of integers with $i > j$. Then we evaluate the integrals we need… The outermost for loop can be vectorized. This is just an explicit way to generate all integers for which $m_{\max} < i$ and $m_{\max}+i=\ell_b$.

```
for i = floor[½ℓ_b]+1,2,…ℓ_b−1
```
$m_{\max}=\ell_b-i$
```
    while m_max ≥ 1
```
$m_{\max}=m_{\max}-1$
```
        b = 0
        for all m = 0,1,2,…,m_max
```
!!Note that all m can be treated at once (vectorized) and that the next six lines can be treated simultaneously/vectorized also.

$$V^{(0)}\left(a_x+m,a_y,a_z;b+1,i,0\right)=V^{(0)}\left(a_x+m+1,a_y,a_z;b,i,0\right)+X_{AB}V^{(0)}\left(a_x+m,a_y,a_z;b,i,0\right)$$
$$V^{(0)}\left(a_x+m,a_y,a_z;b+1,0,i\right)=V^{(0)}\left(a_x+m+1,a_y,a_z;b,0,i\right)+X_{AB}V^{(0)}\left(a_x+m,a_y,a_z;b,0,i\right)$$
$$V^{(0)}\left(a_x,a_y+m,a_z;i,b+1,0\right)=V^{(0)}\left(a_x,a_y+m+1,a_z;i,b,0\right)+Y_{AB}V^{(0)}\left(a_x,a_y+m,a_z;i,b,0\right)$$
$$V^{(0)}\left(a_x,a_y+m,a_z;0,b+1,i\right)=V^{(0)}\left(a_x,a_y+m+1,a_z;0,b,i\right)+Y_{AB}V^{(0)}\left(a_x,a_y+m,a_z;0,b,i\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;i,0,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;i,0,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;i,0,b\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;0,i,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;0,i,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;0,i,b\right)$$
```
            b = b + 1
```

There is a second case if ℓ_b is even, that is if $\mathrm{mod}\left(\ell_b,2\right)=0$. In that case we can have $i=m_{\max}=\frac{1}{2}\ell_b$. We need to do the following:

```
if mod(ℓ_b,2) = 0
```
$i=\frac{1}{2}\ell_b$

$m_{\max}=\ell_b$
```
    while m_max ≥ 1
```
$m_{\max}=m_{\max}-1$
```
        b = 0
        for all m = 0,1,2,…,m_max
```
!!Note that all m can be treated at once (vectorized) and that the next three lines can be treated simultaneously/vectorized also. Which of the three equations above should be used depends on what the "optimal loop order" is in Python, i.e., if the last indices loop faster or the first indices loop faster. I assumed that the "innermost loops" are fastest. Note that this is rather important, as it can be a factor of 100 faster if the loops are ordered correctly.

$$V^{(0)}\left(a_x,a_y+m,a_z;i,b+1,0\right)=V^{(0)}\left(a_x,a_y+m+1,a_z;i,b,0\right)+Y_{AB}V^{(0)}\left(a_x,a_y+m,a_z;i,b,0\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;i,0,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;i,0,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;i,0,b\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;0,i,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;0,i,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;0,i,b\right)$$
b = b + 1

5. Three nonzero indices. The "for" loops can be vectorized and are just an explicit way to generate all nonzero integers for which $m_{\max}\le j\le i$ and $m_{\max}+j+i=\ell_b$.

for i = 1,2, ℓ_b-2

$$j_{\min}=\begin{cases}\frac{1}{2}\left(\ell_b-i\right) & \mod\left(\ell_b-i,2\right)=0\\ \text{ceiling}\left[\frac{1}{2}\left(\ell_b-i\right)\right] & \text{otherwise}\end{cases}$$

for j = $j_{\min}, j_{\min}+1,\ldots \ell_b-i-1$

$m_{\max}=\ell_b-i-j$

while $m_{\max}\ge 1$

$m_{\max}=m_{\max}-1$

b = 0

for all $m=0,1,2,\ldots,m_{\max}$

!!Note that all *m* can be treated at once (vectorized) and that the next six lines can be treated simultaneously/vectorized also.

$$V^{(0)}\left(a_x+m,a_y,a_z;b+1,i,j\right)=V^{(0)}\left(a_x+m+1,a_y,a_z;b,i,j\right)+X_{AB}V^{(0)}\left(a_x+m,a_y,a_z;b,i,j\right)$$
$$V^{(0)}\left(a_x+m,a_y,a_z;b+1,j,i\right)=V^{(0)}\left(a_x+m+1,a_y,a_z;b,j,i\right)+X_{AB}V^{(0)}\left(a_x+m,a_y,a_z;b,j,i\right)$$
$$V^{(0)}\left(a_x,a_y+m,a_z;i,b+1,j\right)=V^{(0)}\left(a_x,a_y+m+1,a_z;i,b,j\right)+Y_{AB}V^{(0)}\left(a_x,a_y+m,a_z;i,b,j\right)$$
$$V^{(0)}\left(a_x,a_y+m,a_z;j,b+1,i\right)=V^{(0)}\left(a_x,a_y+m+1,a_z;j,b,i\right)+Y_{AB}V^{(0)}\left(a_x,a_y+m,a_z;j,b,i\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;i,j,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;i,j,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;i,j,b\right)$$
$$V^{(0)}\left(a_x,a_y,a_z+m;j,i,b+1\right)=V^{(0)}\left(a_x,a_y,a_z+m+1;j,i,b\right)+Z_{AB}V^{(0)}\left(a_x,a_y,a_z+m;j,i,b\right)$$
b = b + 1

2-electron integrals:

The two-electrons are similar. ***The following notes need to be revised in a similar way to how I revised the notes for the 1-electron integrals.***

One has

$$\mathbf{I}^{(0)}\left(\mathbf{a},\mathbf{b}\,|\,\mathbf{c},\mathbf{d}\right)\equiv\iint\xi_a\left(\mathbf{r}_1\right)\xi_b\left(\mathbf{r}_2\right)g\left(\left|\mathbf{r}_1-\mathbf{r}_2\right|\right)\xi_c\left(\mathbf{r}_2\right)\xi_d\left(\mathbf{r}_2\right)d\mathbf{r}_1d\mathbf{r}_2 \tag{61}$$

$$\begin{aligned}\xi_{\mathbf{a}}\left(\mathbf{r}\right)&\equiv\left(x-X_A\right)^{a_x}\left(y-Y_A\right)^{a_y}\left(z-Z_A\right)^{a_z}\exp\left(-\alpha\left|\mathbf{r}-\mathbf{R}_A\right|^2\right)\\ \xi_{\mathbf{b}}\left(\mathbf{r}\right)&\equiv\left(x-X_B\right)^{b_x}\left(y-Y_B\right)^{b_y}\left(z-Z_B\right)^{b_z}\exp\left(-\beta\left|\mathbf{r}-\mathbf{R}_B\right|^2\right)\\ \xi_{\mathbf{c}}\left(\mathbf{r}\right)&\equiv\left(x-X_C\right)^{c_x}\left(y-Y_C\right)^{c_y}\left(z-Z_C\right)^{c_z}\exp\left(-\gamma\left|\mathbf{r}-\mathbf{R}_C\right|^2\right)\\ \xi_{\mathbf{d}}\left(\mathbf{r}\right)&\equiv\left(x-X_D\right)^{d_x}\left(y-Y_D\right)^{d_y}\left(z-Z_D\right)^{d_z}\exp\left(-\delta\left|\mathbf{r}-\mathbf{R}_D\right|^2\right)\end{aligned} \tag{62}$$

and notation

$$
\begin{aligned}
\zeta &= \alpha + \beta \\
\eta &= \gamma + \delta \\
\rho &= \frac{\zeta\eta}{\zeta+\eta} \\
\mathbf{R}_P &= \frac{\alpha\mathbf{R}_A + \beta\mathbf{R}_B}{\zeta} \\
\mathbf{R}_Q &= \frac{\gamma\mathbf{R}_C + \delta\mathbf{R}_D}{\eta} \\
T &= \rho\left|\mathbf{R}_P - \mathbf{R}_Q\right| \\
S_{ab} &= \exp\left(-\frac{\alpha\beta}{\zeta}\left|\mathbf{R}_A - \mathbf{R}_B\right|^2\right) \\
S_{cd} &= \exp\left(\frac{-\gamma\delta}{\eta}\left|\mathbf{R}_C - \mathbf{R}_D\right|^2\right)
\end{aligned}
\tag{63}
$$

One defines

$$
I^{(m)}\left(\mathbf{00}\,|\,\mathbf{00}\right) \equiv \left(\frac{\pi}{\zeta+\eta}\right)^{3/2} S_{ab}S_{cd}G_m\left(\rho,T\right) \tag{64}
$$

Notice that if $\left(\frac{\pi}{\zeta+\eta}\right)^{3/2}\cdot S_{ab}S_{cd} < \varepsilon$, the integral can be safely neglected.

Most of the recursions are almost *exactly* the same as before. It is favorable to choose $\ell_a \geq \ell_c; \ell_a \geq \ell_b; \ell_c \geq \ell_d$.

One starts with the vertical recursion, which is the same as listed before, with slightly different initialization, and the fact that the upper limit of $\ell_a + \ell_b$ is replaced by an upper limit of $\ell_a + \ell_b + \ell_c + \ell_d$. The new initialization is:

Vertical Recursion

1. Initialize $I^{(m)}\left(0,0,0;0,0,0\,|\,0,0,0;0,0,0\right)$ for $m = 0,1,2,\ldots \ell_a + \ell_b + \ell_c + \ell_d$ using Eq. (64). This can be done using a recursion relation (sequentially) or explicitly in a vectorized fashion (computing all terms at the same time). The vectorized implementation is easier to code, and may not be much less efficient, and the recursion relation is only known for certain special choices of $g\left(\left|\mathbf{r}_1 - \mathbf{r}_2\right|\right)$. Also initialize $I^{(m)}\left(a_x,a_y,a_z;\mathbf{0}\,|\,\mathbf{0};\mathbf{0}\right) = 0$ when any of $\left\{a_x,a_y,a_z\right\} = -1$ for $m = 0,1,2,\ldots,\ell_a + \ell_b + \ell_c + \ell_d$.
2. The basic strategy we use is to first form all possible choices for one nonzero index, then all possible choices for two nonzero indices, then all possible choices for three nonzero indices. The recursion is

$$
\begin{aligned}
I^{(m)}\left(a_x+1,a_y,a_z;\mathbf{0}\,|\,\mathbf{0};\mathbf{0}\right) &= \left(X_P - X_A\right)I^{(m)}\left(a_x,a_y,a_z;\mathbf{0}\,|\,\mathbf{0};\mathbf{0}\right) + \frac{\rho}{\zeta}\left(X_P - X_Q\right)I^{(m+1)}\left(a_x,a_y,a_z;\mathbf{0}\,|\,\mathbf{0};\mathbf{0}\right) \\
&+ \frac{a_x}{2\zeta}\left(I^{(m)}\left(a_x-1,a_y,a_z;\mathbf{0}\,|\,\mathbf{0};\mathbf{0}\right) - \frac{\rho}{\zeta}I^{(m+1)}\left(a_x-1,a_y,a_z;\mathbf{0}\,|\,\mathbf{0};\mathbf{0}\right)\right)
\end{aligned}
\tag{65}
$$

Notice that this is *very* similar in form to Eq. (58) so clearly the same routines can be used in both cases.

Electron-Transfer Recursion

The "new" recursion is the electron-transfer recursion. It is used to switch $\ell_c + \ell_d$ elements of angular momentum from **a** to **c**. The algorithm is (again) essentially the same as the vertical recursion, so I will just list the key idea, namely that one uses relations like

$$I^{(0)}\left(a_x,a_y,a_z;\mathbf{0}\,|\,c_x+1,c_y,c_z;\mathbf{0}\right)=\left[\left(X_Q-X_C\right)+\frac{\zeta}{\eta}\left(X_P-X_A\right)\right]I^{(0)}\left(a_x,a_y,a_z;\mathbf{0}\,|\,c_x,c_y,c_z;\mathbf{0}\right)$$
$$+\frac{a_x}{2\eta}I^{(0)}\left(a_x-1,a_y,a_z;\mathbf{0}\,|\,c_x,c_y,c_z;\mathbf{0}\right)+\frac{c_i}{2\eta}I^{(0)}\left(a_x,a_y,a_z;\mathbf{0}\,|\,c_x-1,c_y,c_z;\mathbf{0}\right) \quad (66)$$
$$-\frac{\zeta}{\eta}I^{(0)}\left(a_x+1,a_y,a_z;\mathbf{0}\,|\,c_x,c_y,c_z;\mathbf{0}\right)$$

Contract. I.e., now one moves to contracted basis functions. So in our notation, $\xi_a,\xi_b,\xi_c,\xi_d \rightarrow \chi_a,\chi_b,\chi_c,\chi_d$

Horizontal Recursion

This is *exactly* the same as for the 1-electron integrals. One uses relations like:

$$I^{(0)}\left(a_x,a_y,a_z;b_x+1,b_y,b_z\,|\,\mathbf{cd}\right)=I^{(0)}\left(a_x+1,a_y,a_z;b_x,b_y,b_z\,|\,\mathbf{cd}\right)+X_{AB}V^{(0)}\left(a_x,a_y,a_z;b_x,b_y,b_z\,|\,\mathbf{cd}\right)$$
$$I^{(0)}\left(\mathbf{ab}\,||\,c_x,c_y,c_z;d_x+1,d_yd_z\right)=I^{(0)}\left(\mathbf{ab}\,|\,c_x+1,c_y,c_z;d_x,d_yd_z\right)+X_{CD}I^{(0)}\left(\mathbf{ab}\,|\,c_x,c_y,c_z;d_x,d_yd_z\right) \quad (67)$$

Screening:

There are two main types of screening, overlap screening and Coulomb screening. There is no broad consensus for either, so one simply does something that makes sense, and then uses user-defined threshholds to be sure nothing too bad happens.

In overlap screening, screening occurs at the atom level and the shell level. See the set of notes on this. This means that when one is looping over atoms (loop a) and shells (loop c) one may end up setting an integral to zero.

In the innermost loop (over primitives) one performs one last screening: set the integral to zero in the innermost loop if $s_{00}^0 < \varepsilon$. (If that is true, then one can safely neglect the overlap/moment/kinetic-energy/derivatives integral.) Note that the "right" value for ε may be quite small, especially for moments. In general one can use a little bit coarser threshold for overlaps, and one needs potentially tighter threshholds for moments/derivatives (especially high-order moments/derivatives).

My notes on overlap screening are in the same directory as this document.

https://drive.google.com/file/d/1HXor65UzGlcqAwTa0C_h-CTeISSuJNL9/view?usp=sharing

Note that the basic strategy is that one first identifies a distance threshold beyond which atomic overlaps are negligible or shell-overlaps are negligible. After storing these thresholds, one can avoid computing certain integrals, which are set to zero. There are many ways to implement it, but one way would be to initialize the matrix of integrals one wishes to evaluate to "empty" or some placeholder, then fill zero's where one does not need to compute the integrals, and (finally) compute only integrals that had not (already) been assigned to zero.

In Coulomb screening for the 2-electron integrals, we'll use the scheme proposed by Ochsenfeld, which seems a reasonable compromise between cost and ease-of-programming.

https://drive.google.com/file/d/1u7eObmZpeDu38-qSeDeU0In3Im6_962U/view?usp=sharing

The basic idea is that one forms an estimate for $I^{(0)}\left(\mathbf{a};\mathbf{b}\,|\,\mathbf{c};\mathbf{d}\right)$ (the 2-electron integral) and then one neglects computing integrals where this estimate is sufficiently low. To do this, one first forms all the integrals where only two shells enter, namely,

$$Q_{\mathbf{ab}}=\sqrt{I^{(0)}\left(\mathbf{a};\mathbf{b}\,|\,\mathbf{a};\mathbf{b}\right)} \quad (68)$$

and

$$M_{\mathbf{ac}}\equiv\sqrt{I^{(0)}\left(\mathbf{a};\mathbf{a}\,|\,\mathbf{c};\mathbf{c}\right)} \quad (69)$$

Then the magnitude of the 2-electron integral can be estimated as (cf. Eqs. 11-13 in Ochsenfeld's paper)

$$\left|I^{(0)}\left(\mathbf{a};\mathbf{b}\,|\,\mathbf{c};\mathbf{d}\right)\right| \approx \frac{Q_{\mathbf{ab}}Q_{\mathbf{cd}}}{\sqrt{Q_{\mathbf{aa}}Q_{\mathbf{bb}}Q_{\mathbf{cc}}Q_{\mathbf{dd}}}}\max\left(M_{\mathbf{ac}}M_{\mathbf{bd}}, M_{\mathbf{ad}}M_{\mathbf{bc}}\right) \tag{70}$$

If the predicted magnitude is small enough, then one can neglect the integral. Typically one choose a threshold of about 10^{-10} (which is typically good enough for microHartree accuracy) but sometimes smaller (or larger) thresholds are used.

Boys functions:

We should implement the Boys functions in Ahlrich's paper. This is especially important for the error function (Eq. (52)) and the $\left|\mathbf{r}_1-\mathbf{r}_2\right|^{\alpha}$ interactions (Eq. 49)), but we can implement all of his forms, and also a helper function for the "general" form in Eqs. (46)-(48), perhaps by (explicitly) supporting any interaction that is a polynomial (using Eq. (48)) and implicitly supporting *any* function using numerical integration (using Eq. (46)).

There is a bit of work to figure out for those general cases, however.

https://drive.google.com/file/d/19fQvkuVGA5PVOfbZ3AhwmRFW34gnQ5Fw/view?usp=sharing

We should consider using mpmath here, so that the Boys function is computed to very high accuracy. For the 2-electron integrals there can be numerically ill-conditioning in the recursion, though the recursion is usually stable except for very high angular momentum on sharp (nondiffuse) basis functions. Being able to compute to high precision is worth it, then, but we should not compromise a lot of performance to do it (maybe have it as an option, though I'm not sure how that would work). That is, ideally one could use `scipy.special` or `mpmath`, with a user switch to mpmath. I'm not sure what the performance hit is, though….but that isn't a *huge* deal here.

For the Coulomb case one uses Eq. (39) in Ahlrich's paper, which can be rewritten as a confluent hypergeometric function (cf. eq. 9.8.39 of Helgaker) or as an incomplete gamma function (eq. 9.8.20 of Helgaker). It is best, however, to use the recursion (unless there is a very efficient implementation), i.e.,

$$F_{n+1}\left(x\right) = \frac{\left(2n+1\right)F_n\left(x\right)-\exp\left(-x\right)}{2x} \tag{71}$$

as one always needs all Boys functions up to a set order. Cf. Eq. 9.8.13 in Helgaker. Ideally one could derive recursions for the other Boys functions too, though in some cases the Boys function is easy to evaluate directly, and "performance" is scarcely the issue here. Note that the conversion between the Boys function considered by Ahlrichs and this Boys function is simply

$$G_0\left(\rho,T\right) \equiv \frac{2\pi}{\rho}F_0\left(T\right) \tag{72}$$

ZORA.

We can include relativistic effects, at the scalar relativistic level, using the zeroth order regular approximation. This involves a *numerical* integration, in a similar way to the exchange-correlation integral. The key ZORA integral is:

$$t_{ab} = \int\left(\nabla\chi_a\left(\mathbf{r}\right)\cdot\nabla\chi_b\left(\mathbf{r}\right)\right)\cdot\frac{\tilde{V}\left(\mathbf{r}\right)}{4\left(137.035999139\right)^2-2\tilde{V}\left(\mathbf{r}\right)}d\mathbf{r} \tag{73}$$

(cf. Eq. (14) in the following reference)

https://drive.google.com/open?id=1EbNFPl7rO-nE0gmavP-ah_etmLIAv2ez

It is probably most sensible to pass $\nabla\chi_a\left(\mathbf{r}\right)\cdot\nabla\chi_b\left(\mathbf{r}\right)$ as a function to the `grid` module for this evaluation, which has to be done numerically, but which can often be pruned based on the location/width of the Gaussians under consideration. It is helpful to *precompute* $\tilde{V}\left(\mathbf{r}\right)$ on the entire grid, so this is a case where dot-multi in `grid` is useful. This is automatic if one does a Harris-function guess, wherein one

1. Forms the promolecular density.

2. Evaluates the Kohn-Sham potential for that density. $\tilde{V}(\mathbf{r})$ is the Kohn-Sham potential of the promolecular density.

However, that breaks modularity in a fairly bad way. It's more sensible to add to the database of atomic densities (splines) a database of atomic electrostatic potentials (also splines). Finally, we assume that the local density approximation is good enough, and use the fact that the most important values in Eq. (73) occur close to the nuclei, so it is important to choose a correlation functional that is accurate near the high-density limit. This means that we choose:

$$
\begin{aligned}
E_x^{\text{Dirac}}[\rho] &= \frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}\int \rho^{4/3}(\mathbf{r})\,d\mathbf{r} \\
v_x^{\text{Dirac}}[\rho] &= \left(\frac{3}{\pi}\right)^{1/3}\rho^{1/3}(\mathbf{r}) \approx \left(\frac{3}{\pi}\right)^{1/3}\left(\sum_A \rho_A^0(\mathbf{r})\right)^{1/3} \\
E_c^{\text{Chachiyo}}[\rho] &= \int \rho(\mathbf{r})\left(a\ln\left(1+\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{1/3}+\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{2/3}\right)\right) \\
a &= \tfrac{1}{2\pi^2}(\ln 2 - 1) \qquad b = 20.4562557 \\
v_c^{\text{Chachiyo}}[\rho] &= a\ln\left(1+\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{1/3}+\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{2/3}\right)+\left(\frac{a\rho(\mathbf{r})\left[\tfrac{1}{3}\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{-2/3}+\tfrac{2}{3}\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{-1/3}\right]}{1+\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{1/3}+\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{2/3}}\right) \\
&= a\left[\ln\left(1+\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{1/3}+\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{2/3}\right)+\left(\frac{\tfrac{1}{3}\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{1/3}+\tfrac{2}{3}\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{2/3}}{1+\left(\tfrac{4\pi}{3}\right)^{1/3} b\rho^{1/3}+\left(\tfrac{4\pi}{3}\right)^{2/3} b\rho^{2/3}}\right)\right]
\end{aligned}
\tag{74}
$$

Approximating the density in these density functionals with the promolecular density, we then have:

$$
\tilde{V}(\mathbf{r}) = \sum_{A=1}^{N_{\text{atoms}}} -\frac{Z_A}{|\mathbf{r}-\mathbf{R}_A|} + \underbrace{\Phi_A^0(|\mathbf{r}-\mathbf{R}_A|)}_{\substack{\text{fit to electrostatic} \\ \text{potential of } A\text{'s} \\ \text{pro-atomic density}}} + v_{xc}^{\text{Dirac-Chachiyo}}\left[\sum_{A=1}^{N_{\text{atoms}}} \rho_A^0(|\mathbf{r}-\mathbf{R}_A|)\right] \tag{75}
$$

The Chaychiyo functional is *J. Chem. Phys.* **145**, 021101 (2016).

It is helpful to use a *relativisitic* atomic density and electrostatic potential here. It might be good to read in a density from a ZORA calculation from ADF or Gaussian to form the reference densities/electrostatic potentials here.

Nuclear Gradients:

When evaluating the gradient with respect to the atomic nucleus, one needs to evaluate the derivative of molecular integrals. This is *relatively* simple: using Eqs. (11) and (12),

$$
\begin{aligned}
\frac{d^k e^{-\alpha(x-X_A)^2}}{dX_A^k} &= \frac{d^k e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2}}{d\left(\sqrt{\alpha}(x-X_A)^2\right)^k}\left(\frac{d\sqrt{\alpha}(x-X_A)}{dX_A}\right)^k \\
&= \left(\alpha^{k/2}\right) H_k\left(\sqrt{\alpha}(x-X_A)\right) e^{-\left(\sqrt{\alpha}(x-X_A)\right)^2}
\end{aligned}
\tag{76}
$$

This means that to evaluate the derivative of a Cartesian Gaussian, one uses

$$\begin{aligned}\frac{d^k\left(x-X_A\right)^n e^{-\alpha\left(x-X_A\right)^2}}{dX_A^k} &= \sum_{j=\max(0,k-n)}^{k}\binom{k}{j}\frac{d^{k-j}\left(x-X_A\right)^n}{dX_A^{k-j}}\frac{d^j e^{-\alpha\left(x-X_A\right)^2}}{dX_A^j} \\ &= \sum_{j=\max(0,k-n)}^{k}\binom{k}{j}\frac{(-1)^{k-j}\,n!\left(x-X_A\right)^{n-k+j}}{(n-k+j)!}\left(\sqrt{\alpha}\right)^j H_j\left(\sqrt{\alpha}\left(x-X_A\right)\right)e^{-\alpha\left(x-X_A\right)^2} \\ &= e^{-\alpha\left(x-X_A\right)^2}\sum_{j=\max(0,k-n)}^{k}\left[\binom{k}{j}\frac{n!(-1)^{k-j}}{(n-k+j)!}\left(\sqrt{\alpha}\right)^j\right]\left(x-X_A\right)^{n-k+j} H_j\left(\sqrt{\alpha}\left(x-X_A\right)\right)\end{aligned} \tag{77}$$

This is just another Gaussian basis function, and one can expand out the Hermite series into powers of $\left(x-X_A\right)$ explicitly using

`numpy.polynomial.hermite.herm2poly`

With this understanding, one can evaluate the derivatives of Gaussian integrals to arbitrary order. (It gets a bit tedious, though, which is why it seems people who do this sort of thing always use Hermite basis functions as intermediate quantities.) It would be a unique feature, so if we can do it easily, it might be worth doing, but it is rather complicated because the product rule for higher derivatives leads to a proliferation of terms. For first derivatives (the most important case, by far),

$$\begin{aligned}\frac{d\left(x-X_A\right)^n e^{-\alpha\left(x-X_A\right)^2}}{dX_A} &= -n\left(x-X_A\right)^{n-1} e^{-\alpha\left(x-X_A\right)^2} - 2\alpha\left(x-X_A\right)^{n+1} e^{-\alpha\left(x-X_A\right)^2} = -\frac{d\left(x-X_A\right)^n e^{-\alpha\left(x-X_A\right)^2}}{dx} \\ \frac{d\left(x-X_A\right)^n e^{-\alpha\left(x-X_A\right)^2}}{dX_B} &= \delta_{AB}\left(-n\left(x-X_A\right)^{n-1} e^{-\alpha\left(x-X_A\right)^2} - 2\alpha\left(x-X_A\right)^{n+1} e^{-\alpha\left(x-X_A\right)^2}\right)\end{aligned} \tag{78}$$

In general, then, we can evaluate first derivatives of overlap integrals (and also kinetic energy integrals, electron-nuclear repulsion integrals, and electron-electron repulsion integrals) by just evaluating them for one-higher-angular momentum, then using Eq. (78) to distribute the terms.

For the electron-nuclear repulsion term, however, there is an extra term:

$$\begin{aligned}\int\frac{\partial}{\partial X_D}\frac{\xi_a(\mathbf{r})\xi_b(\mathbf{r})}{\left|\mathbf{r}-\mathbf{R}_C\right|}d\mathbf{r} &= \int\frac{\partial\xi_a(\mathbf{r})}{\partial X_D}\frac{\xi_b(\mathbf{r})}{\left|\mathbf{r}-\mathbf{R}_C\right|}d\mathbf{r} + \int\frac{\partial\xi_b(\mathbf{r})}{\partial X_D}\frac{\xi_a(\mathbf{r})}{\left|\mathbf{r}-\mathbf{R}_C\right|}d\mathbf{r} + \int\xi_a(\mathbf{r})\xi_b(\mathbf{r})\frac{\partial}{\partial X_D}\frac{1}{\left|\mathbf{r}-\mathbf{R}_C\right|}d\mathbf{r} \\ &= \int\frac{\partial\xi_a(\mathbf{r})}{\partial X_D}\frac{\xi_b(\mathbf{r})}{\left|\mathbf{r}-\mathbf{R}_C\right|}d\mathbf{r} + \int\frac{\partial\xi_b(\mathbf{r})}{\partial X_D}\frac{\xi_a(\mathbf{r})}{\left|\mathbf{r}-\mathbf{R}_C\right|}d\mathbf{r} + \int\xi_a(\mathbf{r})\xi_b(\mathbf{r})\frac{\delta_{CD}\left(x-X_C\right)}{\left|\mathbf{r}-\mathbf{R}_C\right|^3}d\mathbf{r}\end{aligned} \tag{79}$$

The first two terms are easy to deal with through Eq. (78), but the last term requires special treatment. The key is to note that the one-electron integrals depend on the nuclear center, $\mathbf{R}_C$, only through the Boys function, which enters into the expressions linearly. Cf. Eq. (53). One then evaluates

$$\frac{\partial}{\partial X_C}V^{(N)}(\mathbf{0};\mathbf{0}) = \exp\left(-\mu\left|\mathbf{R}_A-\mathbf{R}_B\right|^2\right)\frac{\partial G_N\left(p\left|\mathbf{R}_P-\mathbf{R}_C\right|^2\right)}{\partial X_C} \tag{80}$$

but (see, for example Ahlrichs Eq. 12),

$$\begin{aligned}G_{N+1}(p,T) &= -\frac{\partial}{\partial T}G_N(p,T) \\ \frac{\partial G_N(p,T)}{\partial R_C} &= \frac{\partial G_N(p,T)}{\partial T}\frac{\partial T}{\partial R_C} = -G_{N+1}(p,T)\left(\frac{\partial p\left|\mathbf{R}_P-\mathbf{R}_C\right|^2}{\partial R_C}\right) \\ &= p\left(X_P-X_C\right)G_{N+1}(p,T)\end{aligned} \tag{81}$$

Here I have used $T = p\left|\mathbf{R}_P-\mathbf{R}_C\right|^2$. Using Eq. (81), it's clear that we can initialize the 1-electron integrals with

$$V^{(N+1)}(\mathbf{0};\mathbf{0}) = \exp\left(-\mu\left|\mathbf{R}_A-\mathbf{R}_B\right|^2\right)G_{N+1}\left(p\left|\mathbf{R}_P-\mathbf{r}'\right|^2\right) \tag{82}$$

and move down the recursion in the same way, but now $N=1$ is the physical integral,

$$\int \xi_a(\mathbf{r})\xi_b(\mathbf{r})\frac{\partial g\left(|\mathbf{r}-R_C|\right)}{\partial R_C}d\mathbf{r} \equiv (\alpha+\beta)(X_P-X_C)V^{(1)}(\mathbf{a};\mathbf{b}) \\ =(\alpha+\beta)(X_P-X_C)V^{(1)}\left(a_x a_y a_z; b_x b_y b_z\right) \tag{83}$$

The gradient of the nuclear-nuclear repulsion energy is

$$\frac{\partial}{\partial X_A}\sum_{A>B}\frac{Z_A Z_B}{|\mathbf{R}_A-\mathbf{R}_B|}=Z_A\sum_{B\neq A}\frac{Z_B\left(X_B-X_A\right)}{|\mathbf{R}_A-\mathbf{R}_B|^3} \tag{84}$$

There is an easy way to check Eq. (83): use a very steep Gaussian (which approximates a Gaussian) located at $\mathbf{r}=\mathbf{r}'$ in the two-electron integral and differentiate that expression with respect to x'. Note that the whole set of gradient integrals can be generated simultaneously using Eq. (82). The nuclear gradients can be calculated simultaneously with the original 1-electron integrals if one initializes $V^{(m)}(0,0,0;0,0,0)$ for one extra order, $m=0,1,2,\ldots,\ell_a+\ell_b+1$. This is not surprising since the "extra order" is exactly the same as what one needs when evaluating the overlap, kinetic-energy, and 2-electron integrals because of the "degree-raising" feature of Eq. (78) .

Some notes for the future:

It would be good to look at periodic boundary conditions
https://drive.google.com/drive/folders/1qX7ayEWnMaWZV54p5Exs_SmfhhACUiEy?usp=sharing
and, more urgently and importantly, density-fitting
https://drive.google.com/file/d/1KDk2B29kmR5FzHGkgnGDULQ4xHKPIpuI/view?usp=sharing
https://drive.google.com/open?id=1MPcWeA6fTrLRWb7aD0c4-XCkAtCwWQst
We will not bother to preserve the Cholesky part of the code, because density-fitting is, I think, a better option. We still will need to figure out how to use density-fitting, effectively, in a seniority-zero theory. (So there is a "science objective" here too.)

The periodic boundary conditions integrals seem tricky, though with a pseudopotential (yet another piece of implementation…) they should be doable. For now I would like to see if there is some nice trick to avoid them.

For density-fitting, it will help significantly for small systems, but screening is more important for larger systems. That is, density-fitting helps even for small systems (it is best for products of basis functions that are close together) and screening is best for large systems (it is best for products of basis functions that are far apart).

Paul W. Ayers
March 4-13, 2019