The libint-eigen interface

Laurent Lemmens

Friday 18th August, 2017 14:47

Contents

1 Terminology

In the LibInt2 basis set context, there is some terminology that should be cleared up. A (non-normalized) Cartesian *primitive Gaussian*, centered on $\mathbf{R}(X, Y, Z)$, is a function of the following form:

$$\varphi(\mathbf{r}; \zeta, \mathbf{n}, \mathbf{R}) = (x - X)^{n_x}(y - Y)^{n_y}(z - Z)^{n_z} \exp\left(-\zeta|\mathbf{r} - \mathbf{R}|^2\right) , \tag{1}$$

in which $\mathbf{n}(n_x, n_y, n_z)$ are called the cartesian angular momenta of the primitive. The sum

$$l = n_x + n_y + n_z \tag{2}$$

of the angular momenta, called the *angular momentum* of a primitive, determines its the type: $l = 0$ refers to an s-type, $l = 1$ to a p-type, etc. In general, there are[1]

$$\binom{l + 3 - 1}{3 - 1} = \frac{(l + 1)(l + 2)}{2} \tag{3}$$

primitives corresponding to a given angular momentum l. For example, the following primitives all belong to the case $l = 1$ (i.e. for p-type orbitals):

$$\varphi_{p_x}(\zeta) \qquad \varphi_{p_y}(\zeta) \qquad \varphi_{p_z}(\zeta) , \tag{4}$$

where we have introduced a short-hand notation for primitives: the exponents are given as arguments, the type is given as subscript, and the center is omitted (and to be deduced from context). For clarity:

$$\varphi_{p_x}(\zeta) \equiv \varphi(\mathbf{r}; \zeta, \mathbf{n} = (1, 0, 0), \mathbf{R}) = (x - X) \exp\left(-\zeta|\mathbf{r} - \mathbf{R}|^2\right) . \tag{5}$$

In the following, we will refer to a set of Gaussian primitives with the same angular momentum l that share the same center as a *shell*.

Often, a predetermined/fixed linear combination (also known as a *contraction*) of primitives is taken, leading to a *contracted GTO* (CGTO):

$$d_1\varphi_s(\zeta_1) + d_2\varphi_s(\zeta_2) , \tag{6}$$

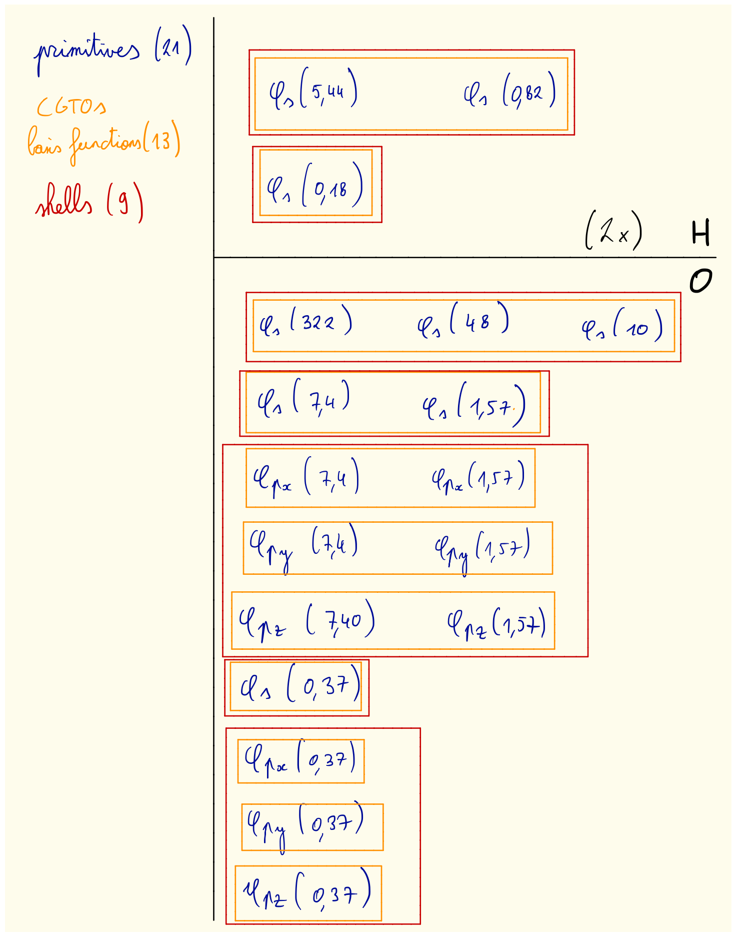
in which the *contraction coefficients* d have to be specified. CGTOs are the functions that are used as *basis functions* (*atomic orbitals*: AOs). Note that a single primitive can also be used as a basis function, in which case the linear combination is just one times that primitive. A

[1] Number of ways to divide l balls in 3 urns: number of combinations with repetition

graphical explanation of the different concepts of primitives, CGTOs, basis functions and shells is given in Figure 1

Molecular orbitals (MOs) are then written as a linear combination of AOs (which are CGTOs), which in turn serve as a one-electron basis to antisymmetrize into the many-electron basis of the *Slater determinants* (SDs). [1]

Internally, LibInt2 stores (normalized) contraction coefficients and exponents in `libint2::Shell` objects.



primitives (21)
CGTOs
basis functions (13)
shells (9)
$\varphi_s(5{,}44)$ $\varphi_s(0{,}82)$
$\varphi_s(0{,}18)$
(2x) H
O
$\varphi_s(322)$ $\varphi_s(48)$ $\varphi_s(10)$
$\varphi_s(7{,}4)$ $\varphi_s(1{,}57)$
$\varphi_{px}(7{,}4)$ $\varphi_{px}(1{,}57)$
$\varphi_{py}(7{,}4)$ $\varphi_{py}(1{,}57)$
$\varphi_{pz}(7{,}40)$ $\varphi_{pz}(1{,}57)$
$\varphi_s(0{,}37)$
$\varphi_{px}(0{,}37)$
$\varphi_{py}(0{,}37)$
$\varphi_{pz}(0{,}37)$

Figure 1: Explanation of the concepts of primitives, CGTOs (basis functions) and shells for water @ 3-21G. φ denotes a Gaussian primitive, its argument being the value of the exponent and its subscript specifying its angular momentum.

2 Are the basis functions (CGTOs) normalized?

When constructing a `libint2::BasisSet` object as in say

```
libint2::BasisSet obs ("STO-3G", atoms);
```

the corresponding file `sto-3g.g94` is read (which is located at your `LIBINT_DATA_PATH` environment variable), in which LibInt2 finds the exponents and contraction coefficients for the given basis for a given element.

In `libint2/basis.h`, we can see the following code (edited for brevity):

```
static ... read_g94_basis_library(...){
    ...
    ref_shells[Z].push_back(
        libint2::Shell{...}
        )
    ...
}
```

which calls a specific constructor of `libint2::Shell`:

```
Shell(...) {
    // embed normalization factors into contraction
  coefficients
    renorm();
}
```

that makes sure that the CGTO is normalized by including the normalization factor in the contraction coefficients. So, **yes**, LibInt2 internally works with normalized basis functions.

References

[1] F. Jensen. *Introduction to Computational Chemistry*. John Wiley & Sons, LTD, second edition, 2007.