Basis Sets

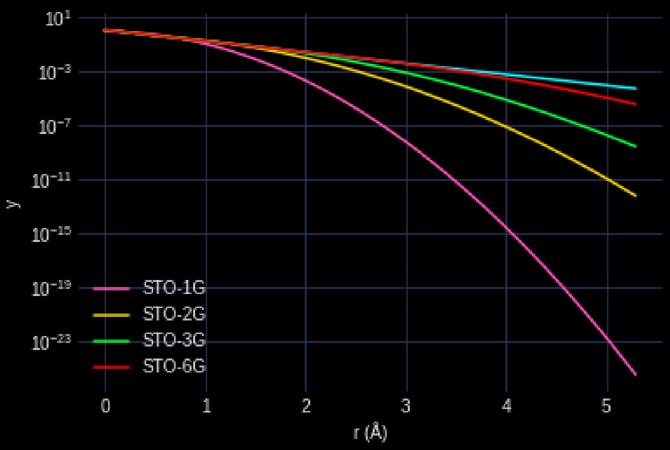
Julio Candanedo
Department of Physics, Arizona State University, Tempe, AZ 85287, USA

1 Gaussian Basis Sets

As we have previously seen with the last few subsections, the characterization of the spatial wavefunction of atoms is typically represented by a eigenfunction of some Hamiltonian. Suppose we chose the hydrogen atom Hamiltonian, the number of bound states is formally countably infinite (classified by their integer quantum numbers), with the number of scattering states being uncountably infinite. Thus even the simple Hydrogen atom already contains infinitely many states. Thus the Hilbert Space is usually treated as a infinite-dimensional vector space $\mathscr{H}$, i.e. the Hamiltonian array would be infinite dimensional, e.e. dim $\mathscr{H} = \infty$! However, if this Hilbert space is approximated to be finite, and this is often a necessity for computational purposes, we will call this approximation the *variational principle.* Using this contracted finite Hilbert Space, $\mathcal{H}$, we may define a N-dimensional basis set. Such that dim $\mathcal{H} = N < \infty$:

$$\begin{aligned} &\mathcal{H} = \mathrm{span}(\{\phi_i\}) \\ &\psi \in \mathcal{H} \\ &\psi = C_1\phi_1 + C_2\phi_2 + \cdots + C_N\phi_N. \end{aligned}$$

If we desire to model M-electrons, the basis set must be larger than $N \geq \frac{M}{2}$. More accurately, the term *minimal basis*, is used to refer to a basis set whose size is the same as the number of occupied orbitals (singly or doubly). These were the first basis sets developed, and were Slater functions (and required numerical integration for 2-integrals), similar to the hydrogenic functions. Later Pople, Boys, et.al. developed Gaussian fits to these Slater functions. The resulting basis sets were named STO-NG, a Slater-function least squares fit with N Gaussian functions (these N Gaussian functions comprising the basis function, are known as *primitive* Gaussian functions in the literature).



Because some electrons are more involved with the chemistry than others, i.e. valence electrons, these are given extra basis functions to capture their changes (this depending on the focus these changes are called a variety of names: multi-zeta basis, diffusive functions, polarization functions, etc...). If core electrons (not valence), from heavy atoms are essentially entirely neglected, then Electron-Core-Potentials (ECP) are often used as well. More modern basis functions have dispensed with the Gaussian-Fitted-Slater-functions notion altogether, instead opting for single Gaussian basis functions with no contraction coefficients, these of are the *correlation-consistent* (cc) basis sets type (e.g. cc-pVTZ).

Nevertheless, an important feature of basis-sets is their normalization. A Gaussian function given their exponential coefficient ζ, and orbital angular momentum parameters $\{n_x, n_y, n_z\}$ (e.g. $(0,0,1)$ for a p_z orbital), has a normalization coefficient given by:

$$N(\zeta; n_x, n_y, n_z) = \left(\frac{2^3\zeta^3}{\pi^3}\right)^{1/4} \left(\frac{(4\zeta)^{n_x+n_y+n_z}}{(2n_x-1)!!(2n_y-1)!!(2n_z-1)!!}\right)^{1/2},$$

$$N(\zeta; \ell) = \left(\left(\frac{2\zeta}{\pi}\right)^{3/2} \frac{(4\zeta)^{\ell}}{(2\ell-1)!!}\right)^{1/2}$$

With primitive contraction coefficients given by:

$$\phi_i = d_{ia} g_a = d_{ia}\left(N(\zeta_a; n_x, n_y, n_z)\exp\left(-\zeta_a \mathbb{r}^2\right)\right).$$

The construction of useful basis sets, is a herculean effort, that typically employs strong *ab initio* (e.g. numerical TISE solutions or Quantum Monte Carlo) methods to solve the atomic structure, which are then fitted using least-squares or modern *machine-learning* techniques. Once calibrated these may be used by SCF techniques to solve more complicated molecular structure. An extensive list of electronic basis function is kept at the Basis Set Exchange (`www.basissetexchange.org`), basis functions from here are used throughout the paper.

1.0.1 Downloading the Basis Sets

⚠ Show code downloading basis functions!!

Gaussian and PSI4 codes use .gbs files for characterizing their basis set. The work descibed here requires extensive use of customized basis sets. For ease of obtaining this, we present a python code to read these files from the Basis-Set-Exchange website.

1.0.2 Examples of Basis Sets

Lets illustrate an example for the simple STO-3G Hydrogen:

```
****
H    0
S    3   1.00
      0.3425250914E+01     0.1543289673E+00
      0.6239137298E+00     0.5353281423E+00
      0.1688554040E+00     0.4446345422E+00
****
```

In the form of equations this is (upto 3 decimal places for clarity), using N the normalization function described above:

$$\begin{aligned}\phi(r) &= 0.154N(3.425,0)\exp\left(-3.425\, r^2\right)\\ &+ 0.535N(0.624,0)\exp\left(-0.624\, r^2\right)\\ &+ 0.445N(0.169,0)\exp\left(-0.169\, r^2\right).\end{aligned}$$

Now lets show another more complicated example, Huzinaga MIDI basis for Neon.

```
****
Ne    0
S    3   1.00
    456.953          0.067
     68.365          0.389
     14.630          0.671
S    2   1.00
     19.327         -0.080
      1.442          0.595
S    1   1.00
      0.444          1.000
P    2   1.00
     13.353          0.129
      2.779          0.480
P    1   1.00
      0.601          1.000
****
```

Neon in this basis has 9 functions: 1 for the core 1s orbital, a Double-Zeta valance 2sp orbitals yielding 8 orbitals. In order of the Basis functions:

$$\begin{aligned}
\phi_1(r) &= 0.067N(456.953, 0)\exp\left(-456.953r^2\right) + 0.389N(68.365, 0)\exp\left(-68.365r^2\right)\\
&\quad + 0.671N(14.630, 0)\exp\left(-14.630r^2\right)\\
\phi_2(r) &= -0.080N(19.327, 0)\exp\left(-19.327r^2\right) + 0.595N(1.442, 0)\exp\left(-1.442r^2\right)\\
\phi_3(r) &= N(0.444, 0)\exp\left(-0.444r^2\right)\\
\phi_4(r) &= 0.129N(13.353, 1)\exp\left(-13.353r^2\right) + 0.480N(2.779, 1)\exp\left(-2.779r^2\right)\\
\phi_5(r) &= 0.129N(13.353, 1)\exp\left(-13.353r^2\right) + 0.480N(2.779, 1)\exp\left(-2.779r^2\right)\\
\phi_6(r) &= 0.129N(13.353, 1)\exp\left(-13.353r^2\right) + 0.480N(2.779, 1)\exp\left(-2.779r^2\right)\\
\phi_7(r) &= N(0.601, 1)\exp\left(-0.601r^2\right)\\
\phi_8(r) &= N(0.601, 1)\exp\left(-0.601r^2\right)\\
\phi_9(r) &= N(0.601, 1)\exp\left(-0.601r^2\right)
\end{aligned}$$

These basis functions can be complied into a list indexed by μ, $\phi_\mu(r)$, and the objective of the SCF procedure would be to determine the multiplicative coefficients of these orbitals.

2 Basis Set Exchange

The Basis Set Exchange is online database of Gaussian Basis Sets used in electronic structure calculations. It is maintained by the Molecular Sciences Software Institute (MolSSI) and the Pacific Northwest National Lab/Environmental Molecular Sciences Laboratory (PNNL/EMSL). It may be addressed by the following url: `https://www.basissetexchange.org/`.

⚠ Note that basis sets from different electronic structure packages and this website might be slightly different. Leading to ~ 0.1 Hartree in SCF energy differences.

3 Code to Download

The output of this code are two lists, containing `BasisFunctions` and `ECPbasis` objects. The `BasisFunctions` 3-dimensional list has the following hierarchy: Atom | Shell | Basis-function (*Shell* denotes a group of $2\ell - 1$ basis functions, for a *shell* with orbital angular momentum ℓ.). While, the `ECPbasis` 2-dimensional list contains Atom | basis-function. These objects are defined for convenience for storing the entire molecular (many atom, many shell, many primitive) basis-sets.

3.1 Classes

We have the following two classes:

```
class BasisFunctions(object):
    ''' A class that contains all our basis function data
       Attributes:
       origin: array/list containing the coordinates of the Gaussian origin
       shell: orbital angular momentum for the orbital
       exps:  list of primitive Gaussian exponents
       coefs: list of primitive Gaussian coefficients
       norm:  normalization
    '''
    def __init__(self,origin=[0.0,0.0,0.0],shell=None,exps=None,coefs=None, atom=None, norm=1.):
       self.origin = np.asarray(origin)
       self.shell = shell
       self.exps = exps
       self.coefs = coefs
       self.norm = norm
       self.atom = atom
       self.oam = None
       self.expn = None

    def getNormCo(self):
       return
           ((2**3)*(self.exps**3)/(np.pi**3))**(1/4)*((4*self.exps)**(self.oam)/factorial2(2*int(self.oam)
           - 1, exact=True))**(1/2)

class ECPbasis(object):
    ''' A class that contains all our basis function data
       Attributes:
       origin: array/list containing the coordinates of the Gaussian origin
       shell: orbital angular momentum for the orbital
       exps:  list of primitive Gaussian exponents
       coefs: list of primitive Gaussian coefficients
       norm:  normalization
    '''
    def __init__(self,origin=[0.0,0.0,0.0],r=None,exps=None,coefs=None, atom=None, norm=1.,
        name=None):
       self.origin = np.asarray(origin)
       self.r = r
       self.exps = exps
       self.coefs = coefs
       self.norm = norm
       self.atom = atom
       self.name = None
```

3.2 Main Function

```
def get_GBSfile(input_basis, Zset, alt_basis=None, file=None, file_dir='', empty=None, version=0,
    output_gbs="newbasis"):
    Zset = np.unique(Zset)

    ### reexamine Zset for empty
    if empty is not None:
        for i, ele in enumerate(Zset):
            if ele == empty:
                Zset.pop(i)

    ### get gbs, ecp objects for a set of atoms
    atom_gbs = []
    atom_ecp = []
    for element in Zset:
        gbs_temp, ecp_temp = getBSEBasis(input_basis, element, version, alt_basis=alt_basis)
        atom_gbs.append(gbs_temp)
        atom_ecp.append(ecp_temp)

    ### Get from File, if file is None:
    if file is not None:
        gbs_base, element_name_list = get_file_basis(file, path=file_dir) ## !!! only does gbs
        for index, element in enumerate(element_name_list):
            if len(np.where(Z_dictonary[Zset] == element)[0]) > 0:
                location_BSE = np.where(Z_dictonary[Zset] == element)[0][0]
                temp = gbs_base[0] + atom_gbs[location_BSE]
                atom_gbs[location_BSE] = temp

            else:
                atom_gbs.append(gbs_base[index])
                atom_ecp.append([])

    ### print out gbs objects to .gbs file
    output = "****\n"
    for i, atom in enumerate(atom_gbs):
        output += write_atom_gbsstring(atom)

    ### print empty atom
    if empty is not None:
        output += Z_dictonary[empty] + "  " + str(0) + "\n" + "****"

    ### print ecp to gbs
    out_ecp = ""
    for i, atom in enumerate(atom_ecp): ## run thru all atoms! some might be empty!!
        if atom: ## check if it is empty
            out_ecp += write_atom_ecpstring(atom) + "\n\n"

    output += "\n\n" + out_ecp

    ## write file with 'output' string
    file = open(output_gbs + '.gbs',"w")
    file.write( output )
    file.close()

    return atom_gbs, atom_ecp ## atom gbs: ATOM, SHELL, 2n+1 basis functions
```

4 Examples

4.1 ECP Basis

Lets analyze how to obtain a CRENBL basis using this code for Hydrogen (element 1) and Oxygen (element 8).

```
atom_gbs, atom_ecp = get_GBSfile("CRENBL", [1,8])
```

Using command: `atom_ecp` , we obtain:

```
[[],
 [<__main__.ECPbasis at 0x7efe8b4506d0>,
  <__main__.ECPbasis at 0x7efe8b450190>]]
```

demonstrating, the first atom has no ECP basis, while the second atom (element 8) has two. Command: `atom_ecp[1]` yields the 1st atom (listed in `get_GBSfile` command) ECP basis, for example:

```
[<__main__.ECPbasis at 0x7efe8b4506d0>, <__main__.ECPbasis at 0x7efe8b450190>].
```

Now, attributes for each object may be obtained, for instance: `atom_ecp[1][0].exps`, yields.

```
array([ 10.02859998, 34.19799995, 100.00389957])
```

4.2 GBS

Similarly, the GBS basis for the same basis-set may be obtained (in this case stored in `atom_gbs`). For instance, for atom 0 (hydrogen), `atom_gbs[0]`, we get for example: (the list of basis functions).

```
[[<__main__.BasisFunctions at 0x7efe8b451750>],
 [<__main__.BasisFunctions at 0x7efe8b4510d0>],
 [<__main__.BasisFunctions at 0x7efe8b451890>],
 [<__main__.BasisFunctions at 0x7efe8b451b90>]]
```

Each of these are s functions, hence only have one basis-function per shell. Attributes for each basis-function may be obtained by selecting the basis-function from the list, e.g. `atom_gbs[0][3][0].coefs`.

```
array([1.])
```