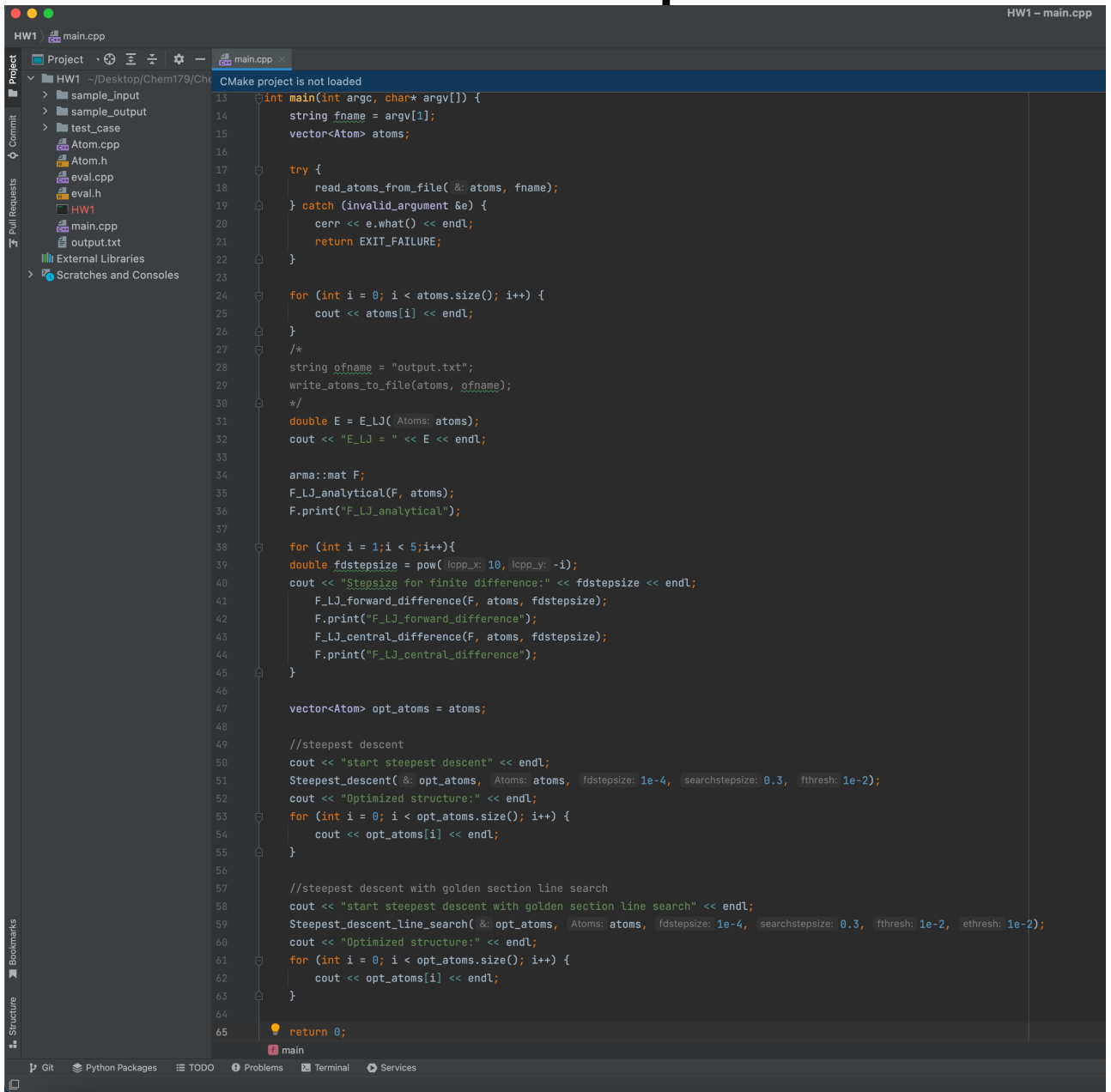
Chem 179 - Lab 2

Xiao Liu, Feb. 8th 2024
xiao_liu@berkeley.edu

Biweekly office hours (starting next week):
Monday 3-5pm, at Gilman basement common area

There are 2 large tables and 1 medium-sized table, might be at one of them depending on availability

HW submission: please include **ALL** parts of your code



Your GitHub should look like this

ref_hw1	add reference code for HW1
sample_input	change in sample input
sample_output	codes for HW1
test_case	add hw1 test and hw2 code
Atom.cpp	codes for HW1
Atom.h	codes for HW1
eval.cpp	codes for HW1
eval.h	codes for HW1
main.cpp	codes for HW1

I won't be able to grade your HW if you only have an executable but don't have any source code in your GitHub repo

Tonight is really the deadline

Linking Armadillo on Datahub (Linux) or Mac

```
jovyan@jupyter-xiao-5fliu:~/Chem179-Spring2024$ cd /usr/include
jovyan@jupyter-xiao-5fliu:/usr/include$ ls
aio.h           charls       drvrsmem.h        fmtmsg.h    gmpxx.h         langinfo.h       linux              Mongoose.hpp          netdb.h        paths.h              pwd.h         spatialite           sudo_plugin.h  ttyent.h       wctype.h
aliases.h       CharLS       elf.h             fnmatch.h   gnumake.h       lastlog.h        locale.h           mqueue.h              neteconet      pcre2.h              rdma          spatialite.h         suitesparse    uchar.h        webp
alloca.h        clang        endian.h          form.h      gnu-versions.h  libaec.h         longnam.h          mtd                   netinet        pcre2posix.h         readline      spatialite_private.h superlu        ucontext.h     wordexp.h
aom             complex.h    envz.h            freexl.h    grp.h           libde265         ltdl.h             mysql                 netipx         pngconf.h            re_comp.h     spawn.h              syscall.h      udunits2.h     X11
argp.h          converter.h  err.h             fstab.h     gshadow.h       libdeflate.h     lz4frame.h         nc_tparm.h            netiucv        png.h                regex.h       sqlext.h             sysexits.h     udunits.h      x265_config.h
argz.h          cpio.h       errno.h           fts.h       hdf             libgen.h         lz4frame_static.h  ncurses_dll.h         netpacket      pnglibconf.h         regexp.h      sql.h                syslog.h       ulimit.h       x265.h
ar.h            crypt.h      error.h           ftw.h       hdf5            libheif          lz4.h              ncurses.h             netrom         poll.h               resolv.h      sqlite3ext.h         szlib.h        unctrl.h       x86_64-linux-gnu
armadillo       ctype.h      eti.h             fyba        iconv.h         libintl.h        lz4hc.h            ncursesw              netrose        poppler              rpc           sqlite3.h            tar.h          unicode        xcb
armadillo_bits  cursesapp.h  etip.h            gconv.h     ifaddrs.h       libltdl          lzma               net                   nfs            postgresql           rpcsvc        sqlspi.h             termcap.h      unistd.h       xen
arpa            cursesf.h    execinfo.h        gdal        inttypes.h      libpng           lzma.h             netash                nl_types.h     printf.h             sched.h       sqltypes.h           term_entry.h   uodbc_extras.h xercesc
arpack          curses.h     expat_external.h  geodesic.h  jbig85.h        libpng16         malloc.h           netatalk              nss.h          proc_service.h       scsi          sqlucode.h           term.h         uodbc_stats.h  zconf.h
asm-generic     cursesm.h    expat.h           geos        jbig_ar.h       libqhull         math.h             netax25               obstack.h      proj                 search.h      stab.h               termio.h       uriparser      zdict.h
assert.h        cursesp.h    fcntl.h           geos_c.h    jbig.h          libqhullcpp      mcheck.h           netcdf_aux.h          odbcinstext.h  proj_constants.h     semaphore.h   stdc-predef.h        termios.h      utime.h        zlib.h
blosc-export.h  cursesw.h    features.h        geotiff     jerror.h        libqhull_r       memory.h           netcdf_dispatch.h     odbcinst.h     proj_experimental.h  setjmp.h      stdint.h             tgmath.h       utmp.h         zstd_errors.h
blosc.h         cursslk.h    features-time64.h getopt.h    jmorecfg.h      librttopo_geom.h menu.h            netcdf_filter_build.h ogdi           proj.h               sgtty.h       stdio_ext.h          thread_db.h    utmpx.h        zstd.h
boost           dav1d        fenv.h            gif_lib.h   jpegint.h       librttopo.h      minizip            netcdf_filter.h       openjpeg-2.1   proj_symbol_rename.h shadow.h      stdio.h              threads.h      values.h
byteswap.h      dirent.h     finclude          GL          jpeglib.h       libxml2          misc               netcdf.h              openjpeg-2.4   protocols            signal.h      stdlib.h             tic.h          video
bzlib.h         dlfcn.h      fitsio2.h         glob.h      json-c          limits.h         mntent.h           netcdf_mem.h          openssl        pthread.h            sound         string.h             time.h         wait.h
c++             drm          fitsio.h          glpk.h      kml             link.h           monetary.h         netcdf_meta.h         panel.h        pty.h                spatialindex  strings.h            tirpc          wchar.h
jovyan@jupyter-xiao-5fliu:/usr/include$ cd -
/home/jovyan/Chem179-Spring2024
jovyan@jupyter-xiao-5fliu:~/Chem179-Spring2024$ cd /usr/lib
jovyan@jupyter-xiao-5fliu:/usr/lib$ ls
apt          cpp            git-core  kernel                libdfalt.a         libgdal.a          libmfhdfalt.a         libogdi.so      libvpf.so.4    mime            os-release            python3         ssl         terminfo    X11
bfd-plugins  dbus-1.0       gnupg     ld-linux.so.2         libdfalt.la        libgdal.so         libmfhdfalt.la        libogdi.so.4    libvpf.so.4.1  modprobe.d      pam.d                 python3.10      sysctl.d    tmpfiles.d  x86_64-linux-gnu
binfmt.d     dpkg           gnupg2    libarmadillo.so       libdfalt.so        libgdal.so.30      libmfhdfalt.so        libogdi.so.4.1  llvm-14        modules-load.d  pkgconfig             R               systemd     udev
clang        environment.d  gold-ld   libarmadillo.so.10    libdfalt.so.0      libgdal.so.30.0.1  libmfhdfalt.so.0      libR.so         locale         ogdi            pkg-config.multiarch  rstudio-server  sysusers.d  usrmerge
compat-ld    gcc            init      libarmadillo.so.10.8.2 libdfalt.so.0.0.0  libhdf4.settings   libmfhdfalt.so.0.0.0  libvpf.so       lsb            openssh         python2.7             sasl2           tcltk       valgrind
```

On Datahub (Linux):
jovyan@jupyter-xiao-5fliu:~/Chem179-Spring2024/HW1$ **g++ -o HW1 *.cpp -I/usr/include -L/usr/lib -larmadillo**

On Mac (install Armadillo through “brew install armadillo” in the command line first):
g++ -o HW1 *.cpp -I/usr/local/opt/armadillo/include -L/usr/local/opt/armadillo/lib –larmadillo

Windows users please consider using Datahub or install WSL and do everything in the Linux subsystem

How to use my code (**compile first**!):
jovyan@jupyter-xiao-5fliu:~/Chem179-Spring2024/HW1$ **./HW1 sample_input/SD_with_line_search/1.txt > test_1.out**

How to use Armadillo: https://arma.sourceforge.net/docs.html; read my code for practical examples

- Recap of the Lecture
  - The Variational Principle
  - Molecular Orbital Theory
  - Linear Combination of Atomic Orbitals (LCAO)
- Commonly Used Atomic Orbitals Basis
  - Slater-type Orbital (STO) and Gaussian-type Orbital (GTO)
  - Shell of Primitive Gaussian-type Orbitals
- Q1: Numerical Integration of the 1-D Overlap Integral
  - Rectangular Rule
  - Trapezoidal Rule
- Q2: Analytical Integration of the Overlap Integral Between (GTO) Shells
  - Gaussian Product Theorem and Binomial Theorem
  - Useful Integrals

Variational Principle

Time-independent Schrödinger equation (SE) with Born-Oppenheimer approximation

$$\hat{H}(\boldsymbol{\tau}_i;\mathbf{R}_a)\Psi(\boldsymbol{\tau}_i;\mathbf{R}_a) = E(\mathbf{R}_a)\Psi(\boldsymbol{\tau}_i;\mathbf{R}_a)$$

$\boldsymbol{\tau}$ – Electron coordinates + spin $\{x, y, z, \sigma\}$

$\mathbf{R}$ – Nuclear coordinates

$$\hat{H} = \hat{T} + \hat{V}_{en} + \hat{V}_{ee} = -\frac{1}{2}\sum_i^n \nabla_i^2 - \sum_i^n \sum_a^N \frac{Z_a}{R_{ia}} + \sum_i^n \sum_{j>i}^n \frac{1}{r_{ij}}$$

Electron-electron repulsion

Kinetic energy of electrons

Electron-nucleus attraction

Solving SE analytically is impossible for systems beyond 2 electrons due to the pair-wise many-body Electron-electron repulsion

Need approximated methods

Variational principle $\quad E_G = \langle \Psi_G | \hat{H} | \Psi_G \rangle \geq E$

Ψ_G is a normalized trial function – a guess

We can approach the true ground state energy E by minimizing E_G (but under some constraints)

Molecular Orbital Theory

We can approximate the many-body wavefunction with a Single Slater determinant

$$\Psi(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_n) = \frac{1}{\sqrt{n!}} \begin{vmatrix} \psi_1(\boldsymbol{\tau}_1) & \psi_1(\boldsymbol{\tau}_2) & \cdots & \psi_1(\boldsymbol{\tau}_n) \\ \psi_2(\boldsymbol{\tau}_1) & \psi_2(\boldsymbol{\tau}_2) & \cdots & \psi_2(\boldsymbol{\tau}_n) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_n(\boldsymbol{\tau}_1) & \psi_n(\boldsymbol{\tau}_2) & \cdots & \psi_n(\boldsymbol{\tau}_n) \end{vmatrix}$$

Why?

Antisymmetry and Pauli principle:
Electrons are fermions (half-integer spin), their wavefunctions are antisymmetric with respect to exchanging electron labels **(row / column swap)**
2 electrons can't share the same set of 4 quantum numbers **(determinant will be 0)**

$$\Psi(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_n) = -\Psi(\boldsymbol{\tau}_2, \boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_n)$$

where ψ_i are molecular orbitals, and satisfy

$$\langle \psi_i | \psi_j \rangle = \int \psi_i^*(\boldsymbol{\tau}) \psi_j(\boldsymbol{\tau})\, d\boldsymbol{\tau} = \delta_{ij}$$

This wavefunction is also normalized

$$\langle \Psi | \Psi \rangle = \int \Psi^*(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_n) \Psi(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_n)\, d\boldsymbol{\tau}_1 d\boldsymbol{\tau}_2 \cdots d\boldsymbol{\tau}_n = 1$$

So it can work as a trial wavefunction!

Linear Combination of Atomic Orbitals (LCAO)

Mean field theory **approximates** the effect of all the other particles on any given particle as an averaged effect, thus reducing a many-body problem to a single-body problem **(Treating 1 electron at a time!)**

$$\hat{H} = \hat{T} + \hat{V}_{en} + \hat{V}_{ee} = -\frac{1}{2}\sum_i^n \nabla_i^2 - \sum_i^n \sum_a^N \frac{Z_a}{R_{ia}} + \sum_i^n \sum_{j>i}^n \frac{1}{r_{ij}} \quad \Rightarrow \quad \hat{H} = \sum_i \hat{h}_{\text{eff}}(i) = \sum_i -\frac{1}{2}\nabla_i^2 + v_{\text{eff}}(\mathbf{r}_i)$$

1-body operator

It can be proved that for 1-body operators, if we use a Slater determinant as our trial function, we will get

$$E = \langle \Psi | \hat{H} | \Psi \rangle = \sum_i \langle \psi_i | \hat{h}_{\text{eff}} | \psi_i \rangle$$

But we still don't know the form of the molecular orbitals!

A lookback from VB theory: express molecular orbitals as the linear combination of (pre-defined) atomic orbitals (LCAO)

$$\psi_i(\boldsymbol{\tau}) = \sum_\mu^N \omega_\mu(\boldsymbol{\tau}) C_{\mu i} \qquad N >> n$$

MO AO unknown coefficients

$$E = \sum_i \langle \psi_i | \hat{h}_{\text{eff}} | \psi_i \rangle = \sum_i \sum_\mu \sum_\nu C_{\mu i}^* \langle \omega_\mu | \hat{h}_{\text{eff}} | \omega_\nu \rangle C_{\nu i} = \sum_i \sum_\mu \sum_\nu C_{\mu i}^* h_{\mu\nu} C_{\nu i}$$

Linear Combination of Atomic Orbitals (LCAO)

$$\psi_i(\boldsymbol{\tau}) = \sum_{\mu}^{N} \omega_\mu(\boldsymbol{\tau}) C_{\mu i}$$

MO AO unknown coefficients

Usually, we can (iteratively) minimize the energy to identify these MO coefficients (as shown in Lecture 8)

(In real quantum chemistry calculations, MO coefficients are calculated on-the-fly!)

$$E = \sum_i \langle \psi_i | \hat{h}_{\text{eff}} | \psi_i \rangle = \sum_i \sum_\mu \sum_\nu C^*_{\mu i} \langle \omega_\mu | \hat{h}_{\text{eff}} | \omega_\nu \rangle C_{\nu i} = \sum_i \sum_\mu \sum_\nu C^*_{\mu i} h_{\mu\nu} C_{\nu i}$$

In the meantime, we may need to calculate integrals in the following form:

$$\langle \omega_\mu | \hat{O}_1 | \omega_\nu \rangle = \int \omega^*_\mu(\boldsymbol{\tau}) \hat{O}_1(\boldsymbol{r}) \omega_\nu(\boldsymbol{\tau})\, d\boldsymbol{r}\, d\sigma = \int \omega^*_\mu(\boldsymbol{r}) \hat{O}_1(\boldsymbol{r}) \omega_\nu(\boldsymbol{r})\, d\boldsymbol{r}$$

$\hat{O}_1 = \hat{1}$ Overlap matrix $S_{\mu\nu} = \int \omega^*_\mu(\boldsymbol{r}) \omega_\nu(\boldsymbol{r})\, d\boldsymbol{r}$ We will compute this integral in HW2

Will be used in extended Hückel theory in solving $HC = SC\epsilon$ in HW3

Slater-type Orbital (STO) and Gaussian-type Orbital (GTO)

We need a set of atomic orbitals to span the molecular orbitals

$$\psi_i(\boldsymbol{\tau}) = \sum_{\mu}^{N} \omega_\mu(\boldsymbol{\tau}) C_{\mu i}$$

A variety of choices on $\omega_\mu(r)$, but two type of functions are historically used

$|r - R| = 0$ $\quad$ $|r - R| \to \infty$

- Slater-type Orbital (STO)

$$\omega_{1s}^{\mathrm{STO}}(\zeta, \boldsymbol{r} - \boldsymbol{R}) = \sqrt{\frac{\zeta^3}{\pi}} e^{-\zeta|\boldsymbol{r}-\boldsymbol{R}|}$$

ζ – Slater orbital exponent r – electron coordinate R – Atomic center

$$\left(\frac{de^{-\zeta r}}{dr}\right)_{r=0} \neq 0$$

Finite slope

Decay slower

Exact form of the 1s orbital of the hydrogen atom, correctly describe the qualitative features of the molecular orbitals

- Gaussian-type Orbital (GTO)

$$\omega_{1s}^{\mathrm{GTO}}(\alpha, \boldsymbol{r} - \boldsymbol{R}) = \left(\frac{2\alpha}{\pi}\right)^{\frac{3}{4}} e^{-\alpha|\boldsymbol{r}-\boldsymbol{R}|^2}$$

α – Gaussian orbital exponent r – electron coordinate R – Atomic center

$$\left(\frac{de^{-\alpha r^2}}{dr}\right)_{r=0} = 0$$

Zero slope

Decay more rapidly

GTO is preferred due to computational reason

GTOs

Why is GTO preferred?

- Variables can be easily separated

$$e^{-\alpha(\boldsymbol{r}-\boldsymbol{R})^2} = e^{-\alpha(x-R_x)^2} e^{-\alpha(y-R_y)^2} e^{-\alpha(z-R_z)^2}$$

3-D integrals → 1-D integrals

- Gaussian Product Theorem

$$e^{-\alpha(x-X_A)^2} e^{-\beta(x-X_B)^2} = P \cdot e^{-\gamma(x-X_P)^2}$$

The product of two Gaussians is another Gaussian orbital

$$\gamma = \alpha + \beta \qquad X_P = \frac{\alpha X_A + \beta X_B}{\alpha + \beta} \qquad P = exp\left[\frac{-\alpha\beta(X_A - X_B)^2}{\alpha + \beta}\right]$$

Useful in two electron integrals

$$\int\int \phi_\mu^*(\boldsymbol{r_1})\phi_\nu(\boldsymbol{r_1})\frac{1}{|\boldsymbol{r_1}-\boldsymbol{r_2}|}\phi_\lambda^*(\boldsymbol{r_2})\phi_\sigma(\boldsymbol{r_2})\, d\boldsymbol{r_1}\, d\boldsymbol{r_2}$$

GTOs with higher angular momentum quantum numbers

The general form of a primitive gaussian function

$$\omega(\alpha, \boldsymbol{r}-\boldsymbol{R}) = N(x-X)^l(y-Y)^m(z-Z)^n e^{-\alpha(\boldsymbol{r}-\boldsymbol{R})^2}$$

We will ignore the normalizer in HW2

$L = l + m + n$ specifies the angular momentum numbers

	Shell Type	Number of Primitive Gaussian Functions	(l, m, n)
$L = 0$	s shell	1	(0,0,0)
$L = 1$	p shell	3	x (1,0,0) y (0,1,0) z (0,0,1)
$L = 2$	d shell	6	x^2 (2,0,0) y^2 (0,2,0) z^2 (0,0,2) xy (1,1,0) yz (0,1,1) xz (1,0,1)
...			
$L = n$		$\frac{(n+1)(n+2)}{2}$	

In HW2.2, you are going to evaluate the overlap integrals between two shells of normalized cartesian gaussian functions

Input Format in Problem set 2

Overlap integral between two primitive gaussian functions $$S^{AB} = \int_x \int_y \int_z \omega_A(\alpha, \boldsymbol{r} - \boldsymbol{R_A}) \omega_B(\beta, \boldsymbol{r} - \boldsymbol{R_B})\, dx dy dz = S_x^{AB} S_y^{AB} S_z^{AB}$$

For 1-D integral $$S_x^{AB} = \int_x N_A N_B (x - X_A)^{l_A} (x - X_B)^{l_B} e^{-\alpha(x - X_A)^2} e^{-\beta(x - X_B)^2}\, dx$$ **We will ignore the normalizer in HW2**

2.1 1-D numerical overlap integral between two primitive gaussian functions

example2.1.txt

D: > example2.1.txt

```
0.0 1.0 0
1.0 1.0 1
```

Input File format:

[X-coordinate of the centroid of ω_1] [exponent of ω_1] [X-axis component of angular momentum of ω_1]
[X-coordinate of the centroid of ω_2] [exponent of ω_2] [X-axis component of angular momentum of ω_2]

2.2 Analytical 3-D overlap integral between two shells of normalized primitive gaussian functions

example2.2.txt

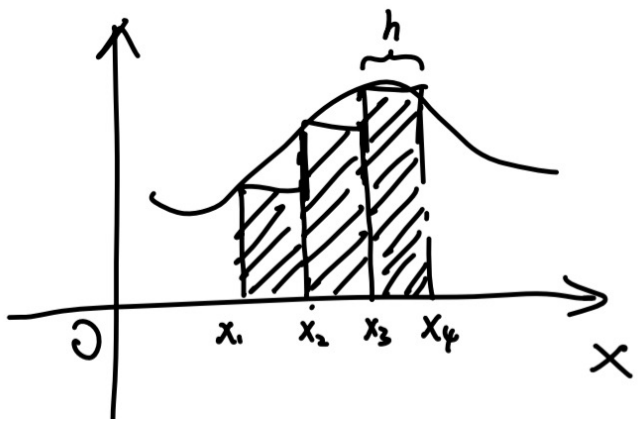
D: > example2.2.txt

```
1.0 0.0 0.0 1.0 0
0.0 1.0 0.0 2.0 1
```

Input File format:

[X] [Y] [Z] [Exponent] [Angular momentum] (for shell 1 in the bra)
[X] [Y] [Z] [Exponent] [Angular momentum] (for shell 2 in the ket)

Q1: 1-D Numerical Integration



Calculate the integral of a function over a specified interval using numerical methods rather than analytical solutions

- Rectangular Rule $$\int_{x_i}^{x_i+h} f(x)\,dx = h f(x_i)$$

The truncation error is $O(h^2)$

- Error Analysis

Primitive function

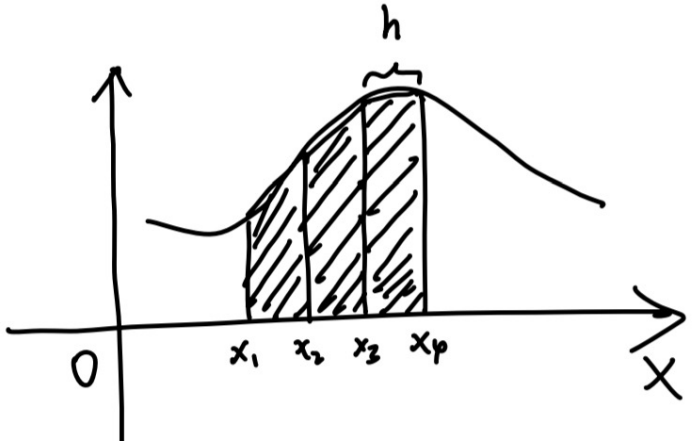
$$\int_{x_i}^{x_i+h} f(x) = F(x_i+h) - F(x_i) = h f(x_i) + \frac{h^2}{2} f'(x_i) + O(h^3)$$

Taylor expansion $$F(x_i+h) = F(x_i) + h f(x_i) + \frac{h^2}{2} f'(x_i) + O(h^3)$$

- Extended Rectangular Rule $$\int_{x_0}^{x_n} f(x)\,dx = \sum_{i=0}^{n-1} h f(x_i) \qquad h = \frac{x_n - x_0}{n}$$

The error is $$O\left(n \cdot \left(\frac{x_n - x_0}{n}\right)^2\right) = O\left(\frac{(x_n - x_0)^2}{n}\right)$$

Q1: 1-D Numerical Integration



Trapezoidal Rule

$$\int_{x_i}^{x_i+h} f(x)\,dx = \frac{h}{2}\left[f(x_i) + f(x_i+h)\right]$$

Error Analysis

$$\int_{x_i}^{x_i+h} f(x) = F(x_i+h) - F(x_i) = hf(x_i) + \frac{h^2}{2}f'(x_i) + O(h^3)$$

Taylor Expansion

$$f'(x_i) = \frac{f(x_i+h) - f(x_i)}{h} + O(h)$$

Forward Difference

$$\int_{x_i}^{x_i+h} f(x) = hf(x_i) + \frac{h^2}{2}\left[\frac{f(x_i+h) - f(x_i)}{h} + O(h)\right] + O(h^3) = \frac{h}{2}\left[f(x_i) + f(x_i+h)\right] + O(h^3)$$

The truncation error is $O(h^3)$

Extended Trapezoidal Rule

$$\int_{x_0}^{x_n} f(x)\,dx = \sum_{i=0}^{n-1}\frac{h}{2}\left[f(x_i) + f(x_i+h)\right] = \frac{h}{2}\left[f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1} f(x_i)\right]$$

$$h = \frac{x_n - x_0}{n}$$

The error is

$$O\left(n\cdot\left(\frac{x_n - x_0}{n}\right)^3\right) = O\left(\frac{(x_n - x_0)^3}{n^2}\right)$$

Q1: 1-D Numerical Integration

$$S_x^{AB} = \int_x N_A N_B (x - X_A)^{l_A} (x - X_B)^{l_B} e^{-\alpha(x-X_A)^2} e^{-\beta(x-X_B)^2} dx$$

We will ignore the normalizer in HW2

The integral is evaluated over the range from $-\infty$ to ∞

How to set a proper range and the number of points for numerical integration?

For Gaussian-type orbitals, it may be helpful to use the extended trapezoidal rule **Fast decay**

For functions that exhibit a slow decay rate towards the endpoints, quadratures can be performed through variable transformation **(Widely used in DFT code!)**

$$I = \int_c^d f[x(t)] \frac{dx}{dt} dt$$

Numerical Recipes, chapter 4.5

Check Numerical Recipes, chapter 4 for a comprehensive overview of more advanced numerical integration algorithms

Q2: Analytical Integration

Evaluate overlap integral between two primitive gaussian functions

$$S^{AB} = \int_x \int_y \int_z \omega_A(\alpha, \boldsymbol{r} - \boldsymbol{R_A}) \omega_B(\beta, \boldsymbol{r} - \boldsymbol{R_B})\, dx dy dz = S_x^{AB} S_y^{AB} S_z^{AB}$$

$$S_x^{AB} = \int_x N_A N_B (x - X_A)^{l_A} (x - X_B)^{l_B} e^{-\alpha(x-X_A)^2} e^{-\beta(x-X_B)^2}\, dx$$

We will ignore the normalizer in HW2

How to evaluate this integral analytically?

Remember Gaussian Product theorem!

$$S_x^{AB} = N_A N_B P \int_x (x - X_A)^{l_A} (x - X_B)^{l_B} e^{-(\alpha+\beta)(x-X_P)^2}\, dx \qquad X_P = \frac{\alpha X_A + \beta X_B}{\alpha + \beta} \qquad P = exp\left[\frac{-\alpha\beta(X_A - X_B)^2}{\alpha + \beta}\right]$$

What about the polynomial terms?

Binomial Theorem for explicit expansion of x

$$(x - X_A)^{l_A} = [(x - X_P) + (X_P - X_A)]^{l_A} = \sum_{i=0}^{l_A} \binom{l_A}{i} (x - X_P)^i (X_P - X_A)^{l_A - i} \qquad \binom{l_A}{i} = \frac{l_A!}{i!(l_A - i)!}$$

Then, do a variable substitution in the integral by replacing x with $x - X_P$

Q2: Some useful integrals

$$\int_{-\infty}^{\infty} e^{-\alpha x^2}\, dx = \sqrt{\frac{\pi}{\alpha}}$$

Proof

$$\left(\int_{-\infty}^{\infty} e^{-\alpha x^2}\, dx\right)^2 = \left(\int_{-\infty}^{\infty} e^{-\alpha x^2}\, dx\right)\left(\int_{-\infty}^{\infty} e^{-\alpha y^2}\, dy\right)$$

$$= \int_x \int_y e^{-\alpha(x^2+y^2)}\, dx dy$$

$$= \int_0^{2\pi} d\theta \int_0^{\infty} r e^{-\alpha r^2}\, dr$$

Transform to polar coordinates

$$= \frac{\pi}{\alpha}$$

n is a non-negative integer

$$\int_{-\infty}^{\infty} x^{2n+1} e^{-\alpha x^2}\, dx = 0$$

Parity of the integrand

There is zero overlap between an s shell and a p shell when they are on the same center!

$$\int_{-\infty}^{\infty} x^{2n} e^{-\alpha x^2}\, dx = \frac{(2n-1)!!}{(2\alpha)^n}\sqrt{\frac{\pi}{\alpha}} = \frac{\Gamma(n+\frac{1}{2})}{\alpha^{n+\frac{1}{2}}}$$

Integration by parts or leverage the property of gamma function

Double factorial: (2n-1)!! = (2n-1)*(2n-3)*(2n-5) ……* 1

The product of all the integers from 1 to n with the same parity as n

Q2: Some useful integrals

Gamma function $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$ Recurrence relationship $\Gamma(z+1) = z\Gamma(z)$

The integral we want to compute $I(n) = \int_{-\infty}^{\infty} x^{2n} e^{-\alpha x^2}\, dx$

Transform the integral into the form of Gamma function (do variable substitution) $u = \alpha x^2 \quad x^2 = \frac{u}{\alpha} \quad dx = \frac{du}{2\sqrt{\alpha u}}$

$$I(n) = 2\int_0^\infty x^{2n} e^{-\alpha x^2}\, dx$$

$$= 2\int_0^\infty \left(\frac{u}{\alpha}\right)^n e^{-u} \frac{1}{2\sqrt{\alpha u}}\, du$$

$$= \alpha^{-n-\frac{1}{2}} \Gamma(n + \frac{1}{2})$$

We know $\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$

Use recurrence relationship of Gamma function

$$I(n) = \alpha^{-n-\frac{1}{2}} \prod_{i=0}^{n-1} \left(i + \frac{1}{2}\right) \Gamma\left(\frac{1}{2}\right) = \frac{1}{2^n \alpha^{n+\frac{1}{2}}} \prod_{i=0}^{n-1} (2i+1)\sqrt{\pi} = \frac{(2n-1)!!}{2^n \alpha^{n+\frac{1}{2}}} \sqrt{\pi}$$

Combine them together, we can derive the analytical integral of S_x^{AB} **Try to derive it yourself**

$$S_x^{AB} = exp\left[\frac{-\alpha\beta(X_A - X_B)^2}{\alpha+\beta}\right] \sqrt{\frac{\pi}{\alpha+\beta}} \sum_{i=0}^{l_A} \sum_{j=0}^{l_B} \binom{l_A}{i} \binom{l_B}{j} \frac{(i+j-1)!!(X_P - X_A)^{l_A - i}(X_P - X_B)^{l_B - j}}{[2(\alpha+\beta)]^{\frac{i+j}{2}}}$$

This is the correct form

$$S_x^{AB} = \exp\left[-\frac{\alpha\beta(X_A - X_B)^2}{\alpha+\beta}\right] \sqrt{\frac{\pi}{\alpha+\beta}} \sum_{i=0}^{l_A} \sum_{j=0}^{l_B} \binom{l_A}{i} \binom{l_B}{j} \frac{(i+j+1)!!(X_P - X_A)^{l_A - i}(X_P - X_B)^{l_B - j}}{[2(\alpha+\beta)]^{(i+j)/2}} \quad (2.9)$$

There is a typo in the original HW2 document equation 2.9

HW2 Suggestions:

Keep in mind the principles of good software design. Try to create modular and extensible code – you may want to reuse your HW2 code in the upcoming HW3.

It's not necessary to use Armadillo yet (but probably will be in HW3), but feel free to use it if you believe it can help you better
[You can always use vector from std, and even construct vector of vectors if you want to build your own "matrix" data structure]

Please try not to "translate" from Python to C++