BEB automation software[1]

Contents

Introduction

The total ionization cross section (TICS) of a molecule is the yield of (positive) ions that result from electron-molecule collisions. The TICS does not distinguish among different ions that may form, but lumps them all together. There are two slightly different definitions of the TICS. The *counting* ionization cross section is appropriate when the number of ions is measured. The *gross* ionization cross section is appropriate when the aggregate charge of the ions is measured. For atoms, the distinction is important because multiply charged ions may be abundant. For molecules, the distinction is less important because multiply charged ions usually fragment into smaller, singly charged ions. The present software predicts the gross TICS by default, but may also be used to estimate the counting TICS.

BEB (Binary-Encounter Bethe) theory is an *ab initio* theory for predicting the TICS of atoms and molecules under electron impact.[2] Its predictions are usually within 15% of reliable experimental measurements.[3,4] This is about as good as the agreement among different measurements. Although the BEB calculation itself can be done in a spreadsheet, the necessary molecular-orbital data must be obtained from electronic-structure calculations. The purpose of the present software is to:

(1) generate input files for those electronic-structure calculations,
(2) extract the necessary data from the resulting output files,
(3) create a molecular-orbital data file, and
(4) apply BEB theory to compute the TICS.

Although the software greatly reduces the labor of performing BEB calculations, the user is responsible for verifying that the automation has been successful in any application. The software is not a reliable "black box."

BEB theory does not specify how the molecular-orbital data should be obtained. Hartree-Fock (HF) theory was used in all the early studies of molecules. Later computations, involving chemists in addition to atomic theorists, tended towards correlated, post-HF calculations to obtain more accurate data. In applications, experimental data have often been used when available, especially for obtaining the important molecular ionization threshold. Experimental and theoretical TICS data for many atoms and molecules are available in an online database.[2] However, this database is static and no longer under development.

When the energy of the incident electron is high, a relativistic version of BEB may be appropriate.[5] This is not implemented in the present software.

BEB theory was developed for neutral targets, but has been adapted for targets that carry a single positive charge.[6,7] The present software is only for neutral targets.

Any quantum chemistry program can produce the necessary molecular orbital data. However, the present software can only handle input and output files for the Gaussian 09 program.[1,8]

Summary of BEB (binary-encounter Bethe) theory

The BEB theory combines two earlier models that worked well only at their respective limits, resulting in a single model that works well for all non-relativistic energies of the incident electron.[6] BEB is the simplest approximation in the family of models developed by Kim and Rudd, and requires the fewest input data. It should be noted that there has been disagreement[9] about the intellectual origin of BEB theory, with some crediting Khare and coworkers[10] and some crediting Kim and coworkers.[6] The procedure implemented here is based upon the publications by Kim and coworkers.

To obtain the TICS of a molecule, first the cross section is computed for each molecular orbital individually, using eq. (1). The parameters that require quantum chemistry calculations are the (vertical) binding energy (B), kinetic energy (U), and number of electrons (N) for the orbital. The other variables in the equation are the Bohr radius (a_0), the Rydberg constant (R), the reduced kinetic energy ($u \equiv U/B$), the energy of the incident electron (T), the reduced energy of the incident electron ($t \equiv T/B$), and a parameter (n) that applies in special situations (see below). Cross sections have units of area.

$$\sigma_{\mathrm{MO}} = \begin{cases} \dfrac{4\pi a_0^2 N}{t+(u+1)/n}\dfrac{R^2}{B^2}\left[\dfrac{\ln t}{2}\left(1-\dfrac{1}{t^2}\right)+1-\dfrac{1}{t}-\dfrac{\ln t}{(t+1)}\right] & \text{for } T > B \\ 0 & \text{for } T \le B \end{cases} \tag{1}$$

The TICS (σ_{T}) for the molecule is the sum of the orbital cross sections, as shown in eq. (2).

$$\sigma_{\mathrm{T}} = \sum_{\mathrm{MOs}} \sigma_{\mathrm{MO}} \tag{2}$$

Although BEB theory is usually called *ab initio*, eq. (1) was not derived strictly from first principles. The denominator in the first factor was obtained, in part, by choosing a reasonable form that performed well. The parameter n, discussed below, has the weakest theoretical basis, but is effective at improving agreement with experiments. Despite having some empiricism in its development, BEB theory has no fitting parameters and is applicable to molecules composed of any chemical elements. This contrasts with semi-empirical methods, which contain many parameters that must be fitted to experimental data.

The parameter n takes positive integer values; usually $n = 1$. Values greater than 1 serve to reduce the effective kinetic energy of the target electron. This may be rationalized classically by considering that the target electron has lower kinetic energy farther from the nuclei, where it may first encounter the incident electron. When the molecular orbital is dominated by an atomic orbital with principal quantum number (usually also labeled n) greater than 2, then this value is used in eq. (1). However, ambiguity arises[11] over the definition of "dominated," how to measure the contribution of atomic orbitals to a molecular orbital, and what to do when different atoms contribute atomic orbitals with large (and possibly different) principal quantum numbers. This ambiguity makes computations somewhat arbitrary. Although the numerical variations are minor, arbitrariness in an undesirable feature in any computational procedure.

Fortunately, Huo and Kim found[12] that relativistic effective-core potentials (relativistic ECPs, also called core pseudopotentials) offer an alternative. ECPs replace atomic core orbitals by pseudopotentials, leaving only the valence and near-valence electrons explicit. The remaining orbitals have fewer radial nodes than usual, since fewer are needed for orthogonality. Fewer nodes means lower kinetic energy, and good results are obtained using $n = 1$ in eq. (1). This is the approach implemented in the present software. However, the core orbitals are not ignored. All-electron calculations provide their orbital data, and $n > 1$ may be needed in some cases. Fortunately, the appropriate value of n is rarely ambiguous for such outer-core orbitals.

As stated in the introduction, BEB predictions of TICS are usually within 15% of experimental measurements. However, exceptions have been noted for carbon and nitrogen fluorides.[13,14] BEB theory does not account for autoionization processes, which may be important for atomic targets.[15]

Software installation

This software is written in Python3 and in Perl. It is designed to run under a Linux-type operating system, but might work in other operating systems. To check whether the Python3 and Perl languages are already installed on your computer, try the following commands at a command-line prompt.

```
python3 -V
perl -v
```

If either of these scripting languages is not installed, you must install it. Both languages are available for free from various websites. The python modules `numpy` and `pandas` are also needed.

All program files and documentation are available from https://github.com/kkinist/BEB.

This software does not perform any electronic-structure calculations. It creates input files for the commercial program, Gaussian 09. To obtain the corresponding output files, you must have access to Gaussian 09. Unfortunately, the present software does not accommodate other quantum chemistry packages. Please contact the author if you would like to develop such capability.

Necessary files:

(1) For creating quantum chemistry input files:
   a. `beb_g09build.py`
   b. `beb_subs.py`

(2) For parsing the resulting quantum chemistry output files and creating a molecular-orbital data file:
   a. `beb_g09parse.py`
   b. `beb_subs.py`

(3) For computing the TICS from the orbital data file:
   a. `beb_tbl.pl`

(4) Additional basis sets, described in corresponding `txt` files
   a. `t_plusd.gbs`
   b. `ano_rcc_dz.gbs`
   c. `dzp.gbs`
   d. `dzp_dkh.gbs`
   e. `dkh2.gbs`
   f. `sarc_dkh.gbs`

Optional files:

(1) For running the quantum chemistry tasks in a sensible order:
   a. `run_beb.sh`
   b. `nimag.pl`
   c. `arch.pl`
(2) Examples, for testing your installation: files in sub-directory `tests/`
   a. `*.inp` (input files for the present software; created by user)
   b. `*.gjf` (quantum chemistry input files; created automatically unless otherwise noted in `log` files)
   c. `*_build.log` (commentary during creation of the `*.gjf` files)
   d. `*.out` (quantum chemistry output files; created by running quantum chemistry calculations)
   e. `*_parse.log` (screen output during parsing of the `*.out` files)
   f. `*.bun` (molecular-orbital data files, created automatically)
(3) For plotting output from `beb_tbl.pl`
   a. `beb_plot.py` (requires module `matplotlib`)

How to run a calculation

Create a molecular input file

The first step is to create an input file that specifies the molecule of interest. It's a text file that specifies the molecular charge, the spin multiplicity, and atomic coordinates. The present software can only accommodate a molecular charge of zero (i.e., neutral target molecules).

Here is an example for carbon monoxide. The first line is the charge (0 for neutral) and the spin multiplicity (1 for a spin singlet). The remaining lines are atomic coordinates. The unit of length is Å (1 Å = 10^{-10} m).

File `co.inp`:

```
0 1
C
O 1 r

r 1.
```

In the example input above, internal coordinates are specified (a so-called "Z-matrix"). However, Cartesian coordinates are equally good. Here is equivalent input, using Cartesian coordinates.

File `co.inp` (second version):

```
0 1
C 0. 0. 0.
O 1. 0. 0.
```

You don't have to be careful about the numerical values of the coordinates, because the geometry will be optimized as the first quantum chemistry task.

Generate the quantum chemistry input files

The second step is to use your input file to generate quantum chemistry input files. The program that does this is `beb_g09build.py`. Here is the command to run for the example:

```
beb_g09build.py co
```

You should get several lines of output that look like this:

```
creating file co_opt.gjf
creating file co_bu.gjf
creating file co_ept1.gjf
creating file co_ept2.gjf
creating file co_cc.gjf
creating file co_cc1hi.gjf
creating file co_cc1lo.gjf
creating file co_cc2hi.gjf
creating file co_cc2lo.gjf
Descriptions of the calculations are in the file "co_build.log".
Your calculations will use 1 processor and 1000 Mw of memory.
```

The files with suffix ".gjf" are the quantum chemistry input files, one for each task. They are described in greater detail later. The logfile, `co_build.log`, summarizes important information about the calculations. It is useful when writing a report.

By default, the quantum calculations will run on a single processor and using only 1000 Mw of memory (1 Mw = 8 MiB). You can change this by adding command-line parameters. For example, if you want to use four processors and 16 Mw of memory (all on a single node), use the following command:

```
beb_g09build.py co -n4 -m16
```

If you forget this syntax, just invoke the script without any arguments, like this, to be reminded:

```
beb_09build.py
```

Run the quantum chemistry calculations

The third step is to run the quantum chemistry calculations. An easy way to do this is by using the shell script, `run_beb.sh`. You may run it in batch or in the background, as you typically run quantum chemistry calculations. For example, to run our example interactively in the background, use the following command:

```
run_beb.sh co &
```

If your installation of Gaussian 09 is working properly, all the quantum calculations will run, in sequence. This will generate a quantum chemistry output file (`co_*.out`) for each input file (`co_*.gjf`). It will also leave a checkpoint file (`co.chk`), which is not needed after all the quantum calculations have completed successfully. If the checkpoint file is not desired for other purposes, such as plotting molecular orbitals, it may be deleted.

If some output files are already present, they will be left alone. The corresponding calculations will be skipped. This is helpful if a calculation fails for some reason, such as an insufficient number of coupled-cluster iterations. You can edit the corresponding `gjf` file, delete the failed `out` file, and re-run the `run_beb.sh` script without repeating the successful tasks. Some familiarity with the quantum chemistry program will be needed, to know how to fix problems.

Not all the quantum chemistry tasks are essential to obtain data needed for BEB calculations. However, the complete set will provide data of good quality, similar to those tested in ref. 4.

It is recommended to inspect the quantum chemistry output files to verify that the geometry is what you expect and that all the calculations have completed successfully.

Generate the molecular-orbital data file

The fourth step is to generate the molecular-orbital data file. The program `beb_g09parse.py` constructs this file by extracting information from the quantum chemistry output files. For our carbon monoxide example, the following command will create the orbital data file, `co.bun`. The file suffix (`bun`) comes from the three parameters that the file specifies (*B*, *U*, and *N* for each orbital).

```
beb_g09parse.py co
```

Any warning messages may indicate problems with the quantum chemistry calculations. To save the screen output to a file, such as `co_parse.log`, use the following command or an equivalent.

```
beb_g09parse.py co | tee co_parse.log
```

For our example, the file `co.bun` is printed below.

```
# Results from automated beb_g09build.py and beb_g09parse.py
# Molecule is 'co' (singlet CO)
# Double-ionization threshold = 41.45 eV from CCSD(T) (to triplet dication)
#MO      B/eV     U/eV     N        Q        DblIon   Special  Remarks
1        562.31   794.63   2        1        Yes      none     Koopmans
2        309.25   436.40   2        1        Yes      none     Koopmans
3        41.39    78.04    2        1        No       none     Koopmans
4        20.06    71.86    2        1        No       none     P3-3rd
5        16.90    53.96    4        1        No       none     P3-3rd
7        13.93    43.12    2        1        No       none     CCSD(T)
```

Run the BEB calculation

The last step is to use the `bun` file to calculate the molecular TICS using BEB theory. The following command will generate many lines of screen output.

```
beb_tbl.pl co.bun
```

First it will echo the data in the `bun` file. Then it will report the total number of electrons for which it read data. Normally this will equal the number of electrons in your molecule. However, sometimes it may be convenient to exclude deep core electrons because they are energetically inaccessible or because their cross sections are small enough to ignore.

Most of the output is the table of incident electron energies (column labeled "Energy", in units of eV) and corresponding values of the TICS (column labeled "BEB", in units of $Å^2$). The listed energies range from threshold to 5 keV, increasing by intervals that increase as the energy increases. To compute the TICS at a specific value of the incident energy (T), specify that energy as an additional command-line argument, as shown below.

```
beb_tbl.pl co.bun 144
```

This reports the value (2.649 $Å^2$) of the TICS at T = 144 eV. As mentioned in the Introduction, the standard uncertainty of a BEB prediction is about 15%. Thus, the four reported digits are not all significant.

The cross sections of individual molecular orbitals are also available, by appending another command line argument (such as the word "details") after the specified energy.

```
beb_tbl.pl co.bun 144 details
```

In addition to the earlier results, the above command prints the following information:

```
Orbital contributions:
MO       B        xsec
3        41.39    0.120217
4        20.06    0.436636
5        16.90    1.230006
7        13.93    0.862550
```

As before, the values of *B* are in eV and the values of the orbital ionization cross section ("xsec") are in Å^2 (1 Å^2 = 10^{-20} m^2). Most of the printed digits are not significant; they are provided in case they are needed for purposes like plotting or differentiation.

Detailed example: carbon monosulfide

In this section, the example of carbon monosulfide (CS) is examined in detail. Begin by creating an input file, `cs.inp`, specifying the charge (0), spin multiplicity (singlet), and approximate molecular geometry. Here is one way to do it.

```
0 1
C
S 1 1.5

```

Quantum chemistry input files

Generate the quantum chemistry input files with the following command:

```
beb_g09build.py cs -n4
```

Each of the 10 quantum chemistry input files is described in the following sub-sections.

cs_opt.gjf

Here are the contents of the file `cs_opt.gjf`, with line numbers added to facilitate description. The number of lines will vary, depending upon the molecule and upon how the coordinates are specified.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # B3LYP/GEN gfinput OPT FREQ scf=xqc 6D
 5
 6 BEB step 1: geom and freqs for cs
 7
 8 0 1
 9 C
10 S 1 1.5
11
12 C 0
13 6-31G(d)
14 ****
15 S 0
16 6-31G(d)
17 ****
18
```

This should look familiar to users of this quantum chemistry software. However, a line-by-line description is given in the table below. The purpose of this computation (step 1) is to generate a reasonable geometry for the molecule. The resulting geometry will be used in all subsequent computations.

Line	*Description*
1	The file `cs.chk` is the communication between one quantum chemistry calculation and the next.
2	Four processors were requested for all the calculations ("`-n4`" argument to the program "`beb_g09build.py`")
3	The default value of 1000 Mw of memory was requested for all the calculations.
4	• The hybrid density functional known as B3LYP[16-18] is used for the geometry optimization (`OPT`). • Vibrational frequencies (`FREQ`) are also computed, merely to verify that they are all real-valued and that the final geometry is a local minimum in energy. • The basis set will be specified later in `GEN` format. • The option `gfinput` will cause the basis set information to be printed in the output file, for future reference if needed. • The option `6D` specifies that polarization d-functions will be applied in sets of six instead of five. This is appropriate for some old basis sets, such as the `6-31G(d)` sets used here. • The option `scf=xqc` makes it more likely that the SCF equations will converge.
5	must be blank
6	Comment line. The recommended order for the quantum chemistry calculations is the step number.
7	must be blank
8	molecular charge and spin multiplicity, taken from the user-created input file
9-10	molecular coordinates, taken from the user-created input file
11	must be blank
12	Basis-set input, in `GEN` format, for all carbon atoms (terminated with zero)
13	Name of basis set to use for carbon
14	Terminate each basis-set input using four asterisks
15-17	Basis-set specification for all sulfur atoms
18	blank line

You can edit many details without breaking the parser for the resulting quantum chemistry output files. For example, you may change the level of theory from B3LYP/6-31G(d) to MP2/cc-pVTZ by changing the appropriate lines as described in the documentation for the quantum chemistry software. However, changes that you make will not be mentioned in the commentary file `*_build.log` unless you edit that file (recommended).

If you would like to make a permanent change to your computational protocol, then it will be more convenient to modify the program `beb_g09build.py`, instead of editing the files that it

generates. Such changes *will* be reflected in the automatically generated log file. For example, to use MP2/cc-pVTZ for all geometry optimizations, change "`B3LYP`" to "`MP2`" in the string-variable `directive` in the section for step 1 (near line 414). To change the basis set, go the function `specify_basis()`. There are three lists that specify the basis sets for light, medium, and heavy atoms [as defined in function `sort_atoms()`]. To change the basis sets for step 1, change the values of `light[1]` and `middle[1]` to `cc-pVTZ`. Whether or not you modify the program, always verify that the computations performed are the computations that you want. This is especially important for molecules that are open-shell or that contain d-block or f-block elements.

cs_bu.gjf

The purpose of this computation (step 2) is to compute the binding energy and kinetic energy for each molecular orbital in the molecule. Many of these binding energies will be superseded in later computational steps. As in step 1, you can edit this file to change the details of the computation. However, Hartree-Fock theory is recommended. Informal tests using Kohn-Sham orbitals have not been encouraging.

The contents of this file are displayed in the text box. The unique contents are described in the following table.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # HF/GEN gfinput geom=check IOP(6/81=3) scf=xqc pop(all,thresh=1)
 5
 6 BEB step 2: Hartree-Fock B and U for cs
 7
 8 0 1
 9
10 C 0
11 6-311G(d,p)
12 ****
13 S 0
14 6-311G(d,p)
15 ****
16
```

Line	*Description*
4	• The geometry is taken from step 1 (`geom=check`). • The command `IOP(6/81=3)` requests that orbital kinetic energies be printed. • The population-analysis command, `pop(all,thresh=1)`, requests a Mulliken population analysis for all orbitals, printing all contributions of at least 1%. • Polarization d-functions will be applied in sets of five (the default for this quantum chemistry program).
11, 14	The basis set `6-311G(d,p)` is used on all carbon and sulfur atoms.

Step 2 is for all orbitals, including the core. If the molecule contains heavy atoms (beyond Kr, $Z > 36$), scalar relativistic effects will be included by using the second-order Douglas-Kroll-Hess Hamiltonian.

cs_bupp.gjf

Step 3 is only needed when the molecule contains atoms heavier than neon ($Z > 10$). Its purpose is to obtain ECP-adjusted values of kinetic energies for the valence orbitals.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # HF/GEN gfinput geom=check IOP(6/81=3) pseudo=read scf=xqc pop(all,thresh=1)
 5
 6 BEB step 3: pseudopotential B and U for cs
 7
 8 0 1
 9
10 C 0
11 6-311G(d,p)
12 ****
13 S 0
14 SDDall
15 ****
16 S 0
17  D   1 1.00       0.000000000000
18       6.5000000000e-01  0.1000000000D+01
19 ****
20
21 S 0
22 SDDall
23
```

Line	*Description*
4	Pseudopotential information will be read near the end of the input (`pseudo=read`).
14	The basis set `SDDall`, which is designed to pair with the pseudopotential, is used on all sulfur atoms.
16-19	An additional set of polarization d-functions is added to all sulfur atoms, for consistency with 6-311G(d,p) on other atoms. The exponent (0.65) is taken from the corresponding 6-311G(d,p) basis set.
20	blank
21-22	The Stuttgart/Köln pseudopotential SDDall is requested on sulfur, to replace the core electrons.

cs_ept1.gjf

The purpose of step 4 is to obtain refined binding energies for the valence orbitals, which contribute most to the TICS. The file is shown below. Unique lines are described in the following table.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # EPT(OVGF+P3,ReadOrbitals)/GEN gfinput geom=check guess=check scf=xqc pop(all,thresh=1)
 5
 6 BEB step 4: EPT B for cs
 7
 8 0 1
 9
10 C 0
11 6-311+G(d,p)
12 ****
13 S 0
14 6-311+G(d,p)
15 ****
16
17 1 5 5 5
18
```

Line	*Description*
4	• Electron-propagator theory[19,20] is used to compute correlated values of orbital ionization energies (`EPT`). • Both `OVGF` and `P3` approximations will be applied. • The orbitals to be calculated will be specified at the end of the file (`ReadOrbitals`). • The initial guess for the wavefunction will be taken from the previous calculation (`guess=check`).
11, 14	The basis set 6-311+G(d,p) is specified for all sulfur and carbon atoms.
17	Orbitals 1 through 5 (not counting the frozen core) among the alpha orbitals, and orbitals 5 through 5 among the beta orbitals will be computed. This corresponds to all the alpha orbitals. This does not actually trigger any separate calculations for beta orbitals (they are identical to the alpha orbitals for a closed-shell molecule such as carbon monsulfide).

cs_ept2.gjf

The purpose of step 5 is to obtain a value for the vertical double ionization energy (VIE2). This is the energy to remove two electrons from the molecule without changing its geometry. The value computed here may be superseded by coupled-cluster calculations in a later step. However, coupled-cluster calculations are very expensive and not always practical.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # EPT(OVGF+P3,ReadOrbitals)/GEN gfinput geom=check guess=check scf=xqc pop(all,thresh=1)
 5
 6 BEB step 5: cation EPT for VIE2 for cs
 7
 8 1 2
 9
10 C 0
11 6-311+G(d,p)
12 ****
13 S 0
14 6-311+G(d,p)
15 ****
16
17 5 5 4 4
18
```

Line	*Description*
8	The molecular charge is specified as +1 and spin multiplicity as a doublet. So this is a calculation of the energy to remove an electron from the CO^+ ion, at the geometry of the neutral molecule.
17	Ionization energies will be computed for alpha orbital 5 (not counting the frozen core orbitals), yielding a singlet dication, and beta orbital 4, yielding a triplet dication. These are the highest occupied molecular orbitals in CO^+.

The spin state here (doublet) is chosen to be between reasonable guesses for the low- and high-spin states of the dication.

One common problem is for a low-spin ion or dication to converge to an excited state. You can fix this by hand in your favorite way. You can also use `beb_g09build.py` to precede all the post-HF calculations with a HF instability-following step, by using with the extra command-line argument, ‘stabilize’, as shown below.

```
beb_g09build.py co -n4 -m16 stabilize
```

cs_cc.gjf

Step 6 is the first of five high-level calculations that will provide refined values for the vertical first and second ionization energies of the molecule. These are expensive calculations and not always feasible.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # CCSD(T,maxcyc=100)/GEN gfinput geom=check scf=xqc
 5
 6 BEB step 6: CCSD(T) for neutral cs
 7
 8 0 1
 9
10 C 0
11 cc-pVTZ
12 ****
13 @t_plusd.gbs
14
```

Line	*Description*
1	• CCSD(T) theory (coupled-cluster theory with single- and double-excitations, and a perturbative estimate of triples)[21] is requested. • Up to 100 iterations (`maxcyc=100`) of the coupled-cluster solver are permitted.
11	The cc-pVTZ basis set, designed for post-HF calculations,[22] is specified for carbon.
13	The basis-set file, `t_plusd.gbs`, is required because it contains the cc-pV(T+d)Z basis set[23] for sulfur.

cs_cc1hi.gjf

Step 7 is the high-level calculation for the molecular ion in its higher-spin state. The result is combined with that from step 6 to get a refined value for the vertical ionization energy (VIE).

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # CCSD(T,maxcyc=100)/GEN gfinput geom=check scf=xqc guess=check
 5
 6 BEB step 7: high-spin CCSD(T) for cation of cs
 7
 8 1 4
 9
10 C 0
11 cc-pVTZ
12 ****
13 @t_plusd.gbs
14
```

Line	*Description*
8	The electric charge is +1 and the spin multiplicity is quartet.

cs_cc1lo.gjf

Step 8 is the high-level calculation for the lower-spin state of the molecular ion. The result is combined with that from step 6 to get a refined value for the VIE. The lower value of the VIE (from step 7 or step 8) is the adopted value for the molecular ionization threshold.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # CCSD(T,maxcyc=100)/GEN gfinput geom=check scf=xqc
 5
 6 BEB step 8: low-spin CCSD(T) for cation of cs
 7
 8 1 2
 9
10 C 0
11 cc-pVTZ
12 ****
13 @t_plusd.gbs
14
```

Line	*Description*
4	The command `guess=check` is not used because it was found in some cases (including CS) not to yield the lowest doublet state.
8	Charge of +1 and spin multiplicity of doublet.

cs_cc2hi.gjf

Step 9 is the high-level calculation for the higher-spin state of the molecular dication. The result is combined with that from step 6 to get a refined value for the vertical double ionization energy

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # CCSD(T,maxcyc=100)/GEN gfinput geom=check scf=xqc guess=check
 5
 6 BEB step 9: high-spin CCSD(T) for dication of cs
 7
 8 2 3
 9
10 C 0
11 cc-pVTZ
12 ****
13 @t_plusd.gbs
14
```

(VIE2). The relevance of the VIE2 is explained later, in the description of the BEB input file (i.e., the molecular-orbital data file).

Line	*Description*
8	Charge = +2 and spin multiplicity = triplet

cs_cc2lo.gjf

Step 10 is the last quantum chemistry calculation. It is the high-level calculation for the lower-spin state of the molecular dication. The result is combined with that from step 6 to get a refined value for VIE2. The lower value (from step 9 or step 10) is the adopted value.

```
 1 %chk=cs.chk
 2 %nprocs=4
 3 %mem=1000mw
 4 # CCSD(T,maxcyc=100)/GEN gfinput geom=check scf=xqc
 5
 6 BEB step 10: low-spin CCSD(T) for dication of cs
 7
 8 2 1
 9
10 C 0
11 cc-pVTZ
12 ****
13 @t_plusd.gbs
14
```

Line	*Description*
8	Charge = +2 and spin multiplicity = singlet

Running the quantum chemistry calculations

The shell script, `run_beb.sh`, is intended to facilitate running the quantum chemistry calculations. It will skip any computation for which an output file already exists. The command `g09` must already be defined in the shell environment.

The most interesting thing that `run_beb.sh` does is to check the results of step 1, using the auxiliary programs `nimag.pl` and `arch.pl`. If the optimized geometry is not a local minimum, the series of calculations is halted. You must correct this problem before proceeding. Usually, this problem results from making the input geometry too symmetrical. After correcting the molecular geometry in your input (`*.inp`) file, re-run `beb_g09build.py`, delete the old `*_opt.out` file, and run `run_beb.sh` again.

Especially for large molecules, the high-level CCSD(T) calculations (steps 6 to 10) may fail because of insufficient time or disk space. This is OK. The results of the electron-propagator theory (EPT) calculations (steps 4 and 5) will be used instead. The results will probably still be good.

For very large molecules, even the EPT calculations may not be feasible. In that case, Hartree-Fock values will be used for all the orbital binding energies. These tend to be somewhat too high, leading to TICS values that may be somewhat too low. In addition, the value used for VIE2 will simply be a guess. The only file that is essential for getting a TICS result is the file `*_bu.out`.

In either of these large-molecule situations, another option is to try the CCSD(T) or EPT calculations with smaller basis sets. That will make the calculations cheaper, but will also reduce their accuracy somewhat. To change the basis sets, simply edit the corresponding `*_cc*.gjf` or `*_ept?.gjf` files. Be sure to use the same basis sets in all five `*_cc*.gjf` files and the same basis sets in both the `*_ept?.gjf` files. The parsing script does not check that.

Screen output from `beb_g09parse.py`

After all the desired quantum chemistry calculations are complete, run the parsing script, `beb_g09parse.py`, to automatically extract information from the output files and to create the orbital data file. Watch for warning messages. For example, if the values of VIE from the EPT and CCSD(T) calculations are quite different, a CCSD(T) calculation may have converged to an excited state. If so, you would need to edit the corresponding input file (as described in the documentation for the quantum chemistry software) to correct the problem.

As each quantum chemistry output file is parsed, the extracted information is printed to the screen. The only file that is required is `*_bu.out`; all others are optional (but recommended). Please examine this output to be sure it looks reasonable. The table below lists some things to check.

Step	*Filename*	*To note*
1	`*_opt.out`	Geometry should be a local minimum (`nimag = 0`)
2	`*_bu.out`	Orbital spectrum (energies) should look reasonable.
3	`*_bupp.out`	The number of electrons replaced by the ECP should be what you expect.
4	`*_ept1.out`	Usually the `P3-3rd` results (best) have acceptable pole strengths and are listed. All the valence orbitals should be listed.
5	`*_ept2.out`	One alpha and one beta orbital should be listed.
6 to 10	`*_cc*.out`	The spin states should be reasonable. If not, edit the corresponding input files (`*.gjf`) and re-run.
summary	(none)	A summary of the selected results is printed. Energy thresholds are used for lumping orbitals together as (effectively) degenerate. You can change these thresholds by editing `beb_g09parse.py` (variable named `degen_thresh`).

For heavy elements, please check that the number of frozen-core orbitals is what you want in the EPT and CCSD(T) calculations. The defaults in the quantum chemistry software do not always provide a good energy separation between frozen and active orbitals.

The orbital data file

The orbital data file is the input file for the actual BEB computation done by `beb_tbl.pl`. Here is the automatically generated file for the carbon monosulfide example.

```
  1 # Results from automated beb_g09build.py and beb_g09parse.py
  2 # Molecule is 'cs' (singlet CS)
  3 # Double-ionization threshold = 32.63 eV from CCSD(T) (to triplet dication)
  4 #MO     B/eV    U/eV    N       Q       DblIon  Special Remarks
  5 1       2503.53 3297.08 2       1       Yes     none    Koopmans
  6 2       309.12  436.68  2       1       Yes     none    B Koopmans; U ECP
  7 3       245.16  509.15  2       1       Yes     none    Koopmans
  8 4       182.03  478.56  6       1       Yes     none    Koopmans
  9 7       30.03   24.02   2       1       No      none    B Koopmans; U ECP
 10 8       17.10   29.33   2       1       No      none    B P3-3rd; U ECP
 11 9       11.34   33.20   2       1       No      none    B CCSD(T); U ECP
 12 10      12.74   20.00   4       1       No      none    B P3-3rd; U ECP
```

Line	*Description*
1	Comment lines begin with '#'.
2	• The molecule's name (`cs`) is taken from the name of the user-created input file (`*.inp`). • The spin state (`singlet`) is also taken from the input file. • The stoichiometry (`CS`) is read from the file `*_bu.out`.
3	The value of VIE2, its source, and the corresponding spin state of the dication are listed in this comment.
4	These are column labels, to improve readability.
5	• This is molecular orbital number 1. The orbital numbering is from Hartree-Fock (HF). • Its binding energy B = 2503.53 eV and its kinetic energy U = 3297.08 eV. • It holds 2 electrons. • The value of Q is always 1 in BEB theory. It has different values in more rigorous variants of BE theory.[6] • Because B > VIE2, double-ionization is possible from this orbital (`Yes` in the column `DblIon`). See below for more about double ionization. • See below for more about `Special` situations. '`none`' indicates no special treatment. • The value of B is from Koopmans' theorem, i.e., from a HF calculation. Values of U are always from HF.
6	The value of U is from HF with a relativistic effective core potential (ECP).
7	`No` is in the `DblIon` column because B < VIE2.
8	This value of B is from a P3 third-order EPT calculation.
9	This value of B is from CCSD(T) calculations. Since this is the highest occupied molecular orbital (HOMO), this is the molecular VIE. For this molecule, the HOMO is not the last orbital listed because HF gives the wrong energy ordering.
10	There are four electrons in this orbital. This is the degenerate pair of valence pi orbitals.

Special situations

The `Special` field is most often used when the molecular orbital is dominated by atomic orbital(s) with principal quantum number greater than 2. When using ECPs (core pseudo-potentials) for heavy atoms, as done here, this field is seldom needed. However, it may be needed for some outer-core orbitals. For example, the special treatment for a core 3s orbital can be invoked by putting `3s` in the `Special` field. The script looks for a numeral followed by one character of [spdf]. The numeral is used as the value of n in eq. (1) The program `beb_g09parse.py` handles this automatically, based upon Mulliken populations.

Another special situation is for the inner core (K and L shells, that is, n = 1 and 2) of heavy atoms. Kim et al. tentatively proposed a modified denominator, shown in eq. (34) of ref. 5. This is triggered here by the entry `heavy_core` in the `Special` field. The script `beb_g09parse.py` does this automatically for atoms heavier than argon (Z > 18). There are few experimental data available to evaluate the accuracy of this modification. If you wish, you can turn it off by replacing `heavy_core` with `none` in the `bun` file. There is usually only a small effect upon the TICS.

Yet another situation, *not* handled by `beb_g09parse.py` or by `beb_g09build.py`, is for target molecules that are ions with charge +1. In this case, n = 2 is used in eq. (1) for all orbitals.[7] This is triggered in `beb_tbl.pl` by the entry `ion` in the `Special` field. Note that this practice is based upon comparison with only a few experiments.

Double ionization

A value of `Yes` in the column `DblIon` indicates that the orbital contribution should be doubled to account for double ionization. This is appropriate for a gross ionization cross section. For a counting cross section, the values may all be set to `No`. However, that makes sense mostly for TICS of atoms. Doubly charged molecular ions usually fragment into two singly-charged ions. If that always happens, then the counting cross section will increase to be equal to the gross cross section, and the values of `Yes` make sense also for the counting cross section.

BEB theory is a one-electron theory and cannot account for processes that involve more than one target electron at a time. Consequently, BEB theory cannot account for direct double (or higher) ionization. Instead, it is assumed that double ionization occurs only through an Auger mechanism (i.e., a deep hole left by single ionization is filled by a valence electron and the resulting energy transferred to a third target electron, which is ejected). In practice, this means doubling the contribution from orbitals that are deep enough.

The same reasoning suggests that even deeper orbitals should have their contributions multiplied by three or more. The script `beb_tbl.pl` does not handle this. The reason is that the cross sections for such deep orbitals are so small that the bigger multiplier would make a negligible change. However, if you wish to include Auger-based triple-ionization events, you can get the same numerical result by tripling the number of electrons in the orbitals whose binding energies exceed the threshold for vertical triple ionization, that is, by tripling their values of N in eq. (1).

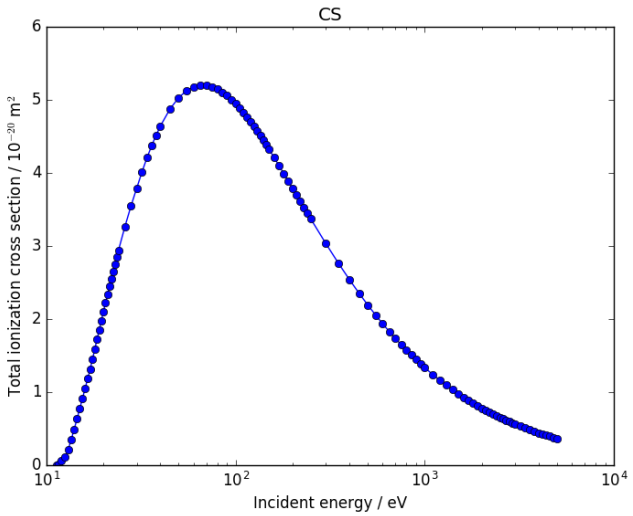
Using experimental binding energies

Sometimes experimental data are available for the vertical binding energies of some molecular orbitals. Usually such data are from photoelectron spectra. To replace the theoretical values of *B* with other values, simply edit the `bun` file. It is recommended to update the `Remarks` field to note the experimental origin of the value of *B*, and to add the literature citation as a comment line.

Plotting with `beb_plot.py`

This is a convenience program for quick plotting. It takes one command-line argument: the name of the BUN file. If the suffix is omitted, it is assumed to be `.bun`.

```
beb_plot.py cs
```



Suggested citation

Please acknowledge using this software by a citation similar to this:

Irikura, K. K. *BEB automation software*; vers. 2017-06-26; National Institute of Standards and Technology: Gaithersburg, MD, 2017; https://github.com/kkinist/BEB.

Contact information

Karl K. Irikura (karl.irikura@nist.gov)
Chemical Sciences Division, National Institute of Standards and Technology,
Gaithersburg MD 20899-8320

Last revised June 26, 2017

References

(1) Certain commercial materials and equipment are identified in this paper in order to specify procedures completely. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the material or equipment identified is necessarily the best available for the purpose.

(2) Kim, Y.-K.; Irikura, K. K.; Rudd, M. E.; Ali, M. A.; Stone, P. M.; Chang, J.; Coursey, J. S.; Dragoset, R. A.; Kishore, A. R.; Olsen, K. J.; Sansonetti, A. M.; Wiersma, G. G.; Zucker, D. S.; Zucker, M. A. *Electron-Impact Cross Sections for Ionization and*

Excitation National Institute of Standards and Technology, Gaithersburg, MD: 2005; http://physics.nist.gov/ionxsec.
(3) Kim, Y.-K.; Irikura, K. K. In *Atomic and Molecular Data and Their Applications*; Berrington, K. A., Bell, K. L., Eds.; American Institute of Physics: College Park, Maryland, 2000; Vol. CP543, p 220.
(4) Scott, G. E.; Irikura, K. K. *Surf. Interf. Anal.* **2005**, *37*, 973.
(5) Kim, Y.-K.; Santos, J. P.; Parente, F. *Phys. Rev. A* **2000**, *62*, 052710.
(6) Kim, Y.-K.; Rudd, M. E. *Phys. Rev. A* **1994**, *50*, 3954.
(7) Kim, Y.-K.; Irikura, K. K.; Ali, M. A. *J. Res. Natl. Inst. Stand. Technol.* **2000**, *105*, 285.
(8) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Mennucci, B.; Petersson, G. A.; Nakatsuji, H.; Caricato, M.; Li, X.; Hratchian, H. P.; Izmaylov, A. F.; Bloino, J.; Zheng, G.; Sonnenberg, J. L.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Montgomery, J. A., Jr.; Peralta, J. E.; Ogliaro, F.; Bearpark, M.; Heyd, J. J.; Brothers, E.; Kudin, K. N.; Staroverov, V. N.; Keith, T.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Rega, N.; Millam, J. M.; Klene, M.; Knox, J. E.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Zakrzewski, V. G.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Dapprich, S.; Daniels, A. D.; Farkas, O.; Foresman, J. B.; Ortiz, J. V.; Cioslowski, J.; Fox, D. J. *Gaussian 09*; vers. D.01; Gaussian, Inc.: Wallingford, CT, 2013.
(9) Kim, Y. K.; Rudd, M. E. *J. Phys. B-At. Mol. Opt. Phys.* **2000**, *33*, 1981.
(10) Jain, D. K.; Khare, S. P. *Journal of Physics B: Atomic and Molecular Physics* **1976**, *9*, 1429.
(11) Scott, G. E.; Irikura, K. K. *J. Chem. Theory Comput.* **2005**, *1*, 1153.
(12) Huo, W. M.; Kim, Y.-K. *Chem. Phys. Lett.* **2000**, *319*, 576.
(13) Huo, W. M.; Tarnovsky, V.; Becker, K. H. *Chem. Phys. Lett.* **2002**, *358*, 328.
(14) Gupta, D.; Choi, H.; Song, M.-Y.; Karwasz, G. P.; Yoon, J.-S. *Eur. Phys. J. D* **2017**, *71*, 88.
(15) Kim, Y.-K.; Stone, P. M. *Phys. Rev. A* **2001**, *64*, 052707.
(16) Lee, C.; Yang, W.; Parr, R. G. *Phys. Rev. B* **1988**, *37*, 785.
(17) Becke, A. D. *J. Chem. Phys.* **1993**, *98*, 5648.
(18) Stephens, P. J.; Devlin, F. J.; Chabalowski, C. F.; Frisch, M. J. *J. Phys. Chem.* **1994**, *98*, 11623.
(19) Ortiz, J. V. *J. Chem. Phys.* **1996**, *104*, 7599.
(20) Zakrzewski, V. G.; Ortiz, J. V.; Nichols, J. A.; Heryadi, D.; Yeager, D. L.; Golab, J. T. *Int. J. Quantum Chem.* **1996**, *60*, 29.
(21) Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. *Chem. Phys. Lett.* **1989**, *157*, 479.
(22) Dunning, T. H., Jr. *J. Chem. Phys.* **1989**, *90*, 1007.
(23) Dunning, T. H., Jr.; Peterson, K. A.; Wilson, A. K. *J. Chem. Phys.* **2001**, *114*, 9244.