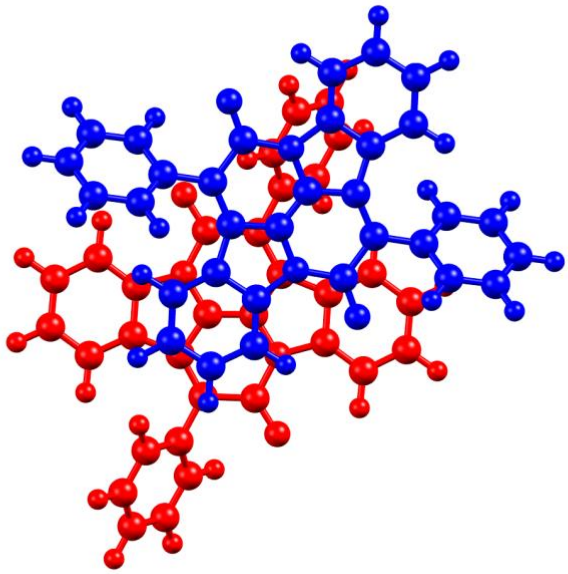
Program Simple Manual



Version 3.0

Alexandr Zaykov, Eric Anthony Buchanan, Milena Jovanović, Zdeněk Havlas, and Josef Michl

Table of Contents

1. Introduction

Program Simple suite (*Simple*), is a package of programs allowing its user to qualitatively compute coupling elements of a phenomenon called "singlet fission" (SF) within the boundaries of Simple theory as derived in the referenced articles.[1,2] The core of the theory though lies in Dirac's (Fermi's) golden rule.[3] It further expands on this by the use of semi-classical Marcus theory[4,5] to assess the SF rate.

The aim of this package is not to provide exact or quantitative data for one structure, for that is the purpose of higher order methods. *Simple* should give the user the ability to search through the whole 6D space of rotations and translations of a monomer to arrive to an optimal dimer structure, or to compare the known crystal structure to it. During its run, *Simple* also prints out other useful information such as the magnitude of Davydov splitting.

The manual itself is split into three main parts. First of them is dedicated to the input file structure for *Simple.* It covers the ways of constructing it manually for small molecules from the output files generated by Weinhold's NBO analysis (often embedded into *Gaussian* or *ORCA* QM programs) and explains its structure on real world example.[6–8] Because this process could end up being long and tiresome for larger molecules, the suite provides an interface that converts the output of the above mentioned analysis into the input file of *Simple*. The use of this tool is also covered in this section.

The second part then describes the run of *Simple* after the successful making of the input file. It describes what parts of the program you should call to obtain the desired result, how to call them, and how to read and/or interpret the results.

The last part describes the way of obtaining and installing the program and also includes contacts section should the user need help or have a bug report to file in.

2. Input File

2.1 Structure of the Input File

The structure of the input file is going to be shown on a model molecule of ethylene (Fig. 1).

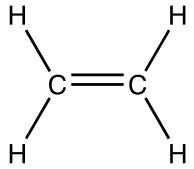


Figure 1. Structure of the model molecule, ethylene.

The whole input file is going to look in the following way (Fig. 2).

```
Title :
C2H4 dimer - 6-311G
Processors : 44
Method :
sf-lj 1.0
GeomA :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
GeomB :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
Const :
T Z  2.0
T Y  1.0
T X  0.0
R Z 90.0
R Y  0.0
R X  0.0
Scan :
NAO :
  C 3
P 3 0.0 0.0 0.2929d0
20.96420000     0.04024870
 4.80331000     0.23759400
 1.45933000     0.81585400
P 1 0.0 0.0 0.5275d0
 0.48345600     1.00000000
P 1 0.0 0.0 0.4343d0
 0.14558500     1.00000000
MOs :
 hA
 1  0.5
 2  0.5
 lA
 1  0.5
 2 -0.5
 hB
 1  0.5
 2  0.5
 lB
 1  0.5
 2 -0.5
dE(CT) : 1.0
lambda : 2.0
```

Figure 2. The whole input file structure for a dimer of ethylenes.

```
Title :
C2H4 dimer - 6-311G
#Le comment
Processors : 44
Method :
sf-lj 1.0
#rate
#LJ-rep
#omit
GeomA :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
GeomB :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
```

Figure 3. Title, method, parallelization, and geometry part of the input.

If we start the dissection of the input file (Fig. 3), it begins with **`Title :`** declaration, *under** which you can specify the title of your calculation. Under the title "**`C2H4 dimer - 6-311G`**", a line starts with **`#`** that is used to denote a commented line. Line starting with this symbol is not read by *Simple*.

Under this section you may find the **`Processors :`** with a number *next* to this keyword. *Simple* has been parallelized using OpenMP directives and as tested, it scales well with up-to 48 threads.

Under the next line that says **`Method :`**, one specifies the goal of the program is supposed to achieve. There are three to choose from:

a. **`sf-lj 1.0`**

   This method tells the program to calculate coupling elements of singlet fission (T^2) according to the theory by Havlas and Michl. If the program searches through or optimizes in predefined 6D space, it maximizes T^2. To do so, it employs a search function defined in Eq. 1, in which a is the multiplicative factor defined in the input file. This equation by itself does not bear any physical meaning. *Default and recommended value is* **`1.0`**.

*The way *Simple* handles input files is somewhat rigid. Do pay attention to the prepositions in *italics*.

$$F = aE_{\mathrm{REP}} - T^2 \tag{1}$$

The repulsion potential E_{REP} is a smooth version of hard-sphere potential defined in Eq. 2, in which $\Delta \boldsymbol{x}$ is the distance between the atoms of the two examined molecules and $\sum r_{\mathrm{vdW}}$ is the sum of the van der Waals radii of both the atoms. This repulsion effectively prevents the molecule entwining.

$$E_{\mathrm{REP}} = 10^{106} \cdot \exp\left[-244.1 \cdot \left(\frac{\triangle \boldsymbol{x}}{\sum r_{\mathrm{vdW}}}\right)^2\right] \tag{2}$$

It includes calculation of Davydov splitting, biexciton binding energy and addition to the endoergicity of the process.

b. **`rate`**

This method tells the program to calculate rate of singlet fission (k) according to the theory by Marcus. If the program searches through predefined 6D space, it maximizes k_{SF}. To do so, it employs a search function defined in Eq. 3, which, as its predecessor in Eq. 2, bears no physical meaning.

$$F = E_{\mathrm{REP}} - \hbar k_{\mathrm{SF}} \tag{3}$$

c. **`LJ-rep`**

This method tells the program to calculate only the repulsion potential E_{REP} as defined in Eq. 2. If the program searches through predefined 6D space, it maximizes its magnitude.

d. **`omit`**

This is an optional keyword. It tells the program to omit structures with too large repulsion. This will speed up scanning considerably.

Next section with xyz coordinates of the monomers is marked by **GeomA :** and **GeomB :**. In this example, both geometries are identical and are at the very same place. This is the recommended start point of the calculation, yet one can easily choose to predefine a structure of a dimer. In the latter case, it is needed to define the orbitals for each monomer separately.

```
GeomB :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
Scan :
T Z  2.0
T Y  1.0
T X  0.0
R Z 90.0
R Y  0.0
R X  0.0
```

Figure 4. Translations and rotations in *Simple*, *one dimer*.

```
GeomB :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
Scan :
T Z -4.0  16  0.5
T Y -4.0  16  0.5
T X -4.0  16  0.5
R Z  0.0  18 10.0
R Y  0.0  18 10.0
R X  0.0  18 10.0
```

Figure 5. Translations and rotations in *Simple*, *user-defined scan grid*.

```
GeomB :
 C    0.6630315    0.0000000    0.0000000
 C   -0.6630315    0.0000000    0.0000000
 H    1.2338880    0.9217687    0.0000000
 H   -1.2338880   -0.9217687    0.0000000
 H   -1.2338880    0.9217687    0.0000000
 H    1.2338880   -0.9217687    0.0000000
Scan :
coarse
#medium
#fine
```

Figure 6. Translations and rotations in *Simple*, *pre-defined scan grid*.

Next section (Figs. 4, 5, and 6), defines the way *Simple* handles molecular geometries. Fig. 4 shows a possibility how to define a single translation (**T**) and

rotation (**R**) vector to form a dimer *under* **Scan** : keyword. Rotations are defined as counterclockwise, except for the x-axis rotation, which clockwise. The outcome of this is depicted in Fig. 7, where the blue and red structure is the monomer A and B, respectively.

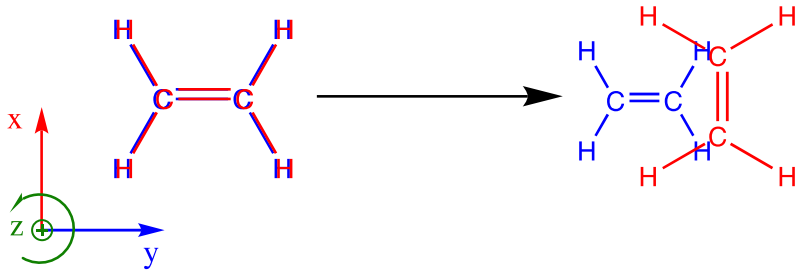


Figure 7. The effects of translations and rotations as defined in Fig. 4, blue is the monomer A and red is the monomer B (translated and rotated one). All rotations are defined as *counterclockwise*.

Fig. 5 shows the possibility of the user to define his/her own grid where the program searches for the maximum value of the sought quantity defined in the **Method** : section. *Under* **Scan** : line, one defines translations and rotations in all 6 dimensions – the first two strings are similar to the previous case section, the *next* three numbers stand for: the starting point (**-4.0**), the number of steps (**16**), the step size in Ångströms (**0.5**). *Beware of the following features.*

a. The program searches for the maximum value by comparing the adjacent points.
b. The scan in the example will take *some* time – in the example: $16^3 \times 18^3 \doteq 24$M points. On the other hand, one structure is completed usually within milliseconds of CPU time or even less.

Fig. 6 shows the easiest and recommended way of searching the 6D space – predefined grids for scanning. There are three grids to choose from: **fine**, **medium**, and **coarse**. The last grid should be used for larger molecules.

The test on a molecule with roughly 40 atoms (108 NAOs) showed **coarse** grid (420M points) to complete in a little less than a week using 40 threads on two Intel Xeon® E7-4850 v2 processors embedded in a single motherboard running at nominal 2.5 GHz core clock speed. Finer grids may be used for smaller molecules,

for they provide a higher level of certainty that no sharp, but narrow minima of F will be skipped.

```
Scan :
coarse
#medium
#fine
NAO :
  C 3
P 3 0.0 0.0 0.2929d0
20.96420000     0.04024870
 4.80331000     0.23759400
 1.45933000     0.81585400
P 1 0.0 0.0 0.5275d0
 0.48345600     1.00000000
P 1 0.0 0.0 0.4343d0
 0.14558500     1.00000000
```

Figure 8. AO and NAO section – *defined for carbon as an element*, basis set: 6-311G.

```
NAO :
1 A 3
P 3 0.0 0.0 0.2929d0
20.96420000     0.04024870
 4.80331000     0.23759400
 1.45933000     0.81585400
P 1 0.0 0.0 0.5275d0
 0.48345600     1.00000000
P 1 0.0 0.0 0.4343d0
 0.14558500     1.00000000
2 A 3
P 3 0.0 0.0 0.2929d0
20.96420000     0.04024870
 4.80331000     0.23759400
 1.45933000     0.81585400
P 1 0.0 0.0 0.5275d0
 0.48345600     1.00000000
P 1 0.0 0.0 0.4343d0
 0.14558500     1.00000000
1 B 3
P 3 0.0 0.0 0.2929d0
20.96420000     0.04024870
 4.80331000     0.23759400
 1.45933000     0.81585400
P 1 0.0 0.0 0.5275d0
 0.48345600     1.00000000
P 1 0.0 0.0 0.4343d0
 0.14558500     1.00000000
```

Figure 9. AO and NAO definition section – *defined per atom*, basis set: 6-311G.

Section following the geometry is the basis set section. *Simple* expands orbitals from the predefined atomic orbital (AO) basis into natural atomic orbitals (NAOs). This basis set of NAOs is further expanded into molecular orbitals (MOs) that are normalized and orthogonalized by an intrinsic subroutine thereafter. All of the expansion coefficients need to be provided, e.g. from the NBO analysis.

There are two possibilities how to construct the NAO part. The easiest one is shown in Fig. 8. User defines the AO and NAO basis set only per element.**

The first line *under* **NAO :** contains 1 string and 1 number, **C** and **3**. The string defines the element by the symbol as in the periodic table of elements. The number defines the amount of basis functions and because it was opted to use the 6-311G basis set, this basis AO basis set is constructed using only 3 P-orbital functions.

The use of specifically P-orbital functions is denoted in the first character of the next line, **P**. The number following it, **3**, is the number of contracted gaussian functions used to define it. The three following floating-point numbers, **0.0 0.0 0.2929d0**, are the NAO expansion coefficients in the AO basis in x-, y-, and z- direction, respectively, obtained from the NBO analysis. Do observe that only z-direction is used for the further construction of HOMO and LUMO (Figs. 8 and 10), the P-orbitals in the other two directions have negligible contribution and can be safely neglected.

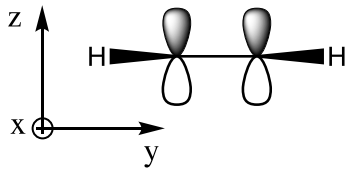


Figure 10. Side-view of the model ethylene molecule with the depiction of the used P-orbitals in the further construction of HOMO and LUMO as defined by the input file.

The AO basis are the following three lines separated into two columns. This copies the way AO bases are defined in Turbomole format in Basis Set Exchange Library (https://bse.pnl.gov/bse/portal) so they can be simply readily copied.

The latter, more complex definition of NAO basis (Fig. 9) is useful for larger molecules where the atoms are not chemically identical, which is not the case of ethylene. It follows identical rules, only the header of each NAO changes. Instead of one string and a number (**C 3** in Fig. 8), one number before the string and one after is used – **1 A 3**. The first number defines the atom number now (**first** atom defined by the geometry section in the example), the string defines the monomer (monomer **A**), and the last number is again the number of contracted gaussian functions. The rest of this section is identical.

**Note that the input file omits hydrogen atoms in the basis set construction. It is mostly unneeded to include them in the construction of neither HOMO, nor LUMO.

```
MOs :
 hA
 1   0.5
 2   0.5
 lA
 1   0.5
 2  -0.5
 hB
 1   0.5
 2   0.5
 lB
 1   0.5
 2  -0.5
```

Figure 11. The definition of the ethylene MOs in the previously defined NAO basis.

The next step is to define the MOs in the NAO basis. That is carried out in the **MOs :** section of the input (Fig. 11). The first line *under* the header reads **hA**. That stands for the HOMO (**h**) of monomer A (**A**). The lines *under* specify the MO expansion coefficients in the NAO basis obtained from the NBO analysis. They are ordered in two columns, *second* column being the expansion coefficient itself. The *first* column specifies the number of the NAO (*not of the atom*) that is expanded by the coefficient in the second column. This is repeated for LUMO A (**lA**), HOMO B (**hB**), and LUMO B (**lB**).

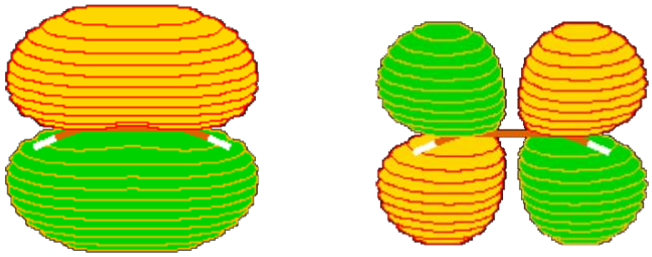


Figure 12. HOMO (left) and LUMO (right) as constructed by *Simple* from the example input file, visualized by *MOLDEN*.[9]

```
lB
 1   0.5
 2  -0.5
dE(CT) : 1000 meV
EAC : 1.00
ECA : 1.00
lambda : 1.5
```

Figure 13. Last part of the input file, CT state energy and reorganization energy for Marcus theory.

The last two steps are to define the energies of charge transfer (CT) states and reorganization energy for Marcus theory approach.

The energy difference between S_1 (T_1) and CT state is written *next* to **dE(CT) :** keyword. It can be either left as default 1 eV (Fig. 13) – the rationale behind this particular number is described in the referenced article,[1] or it can be calculated. The latter option might not be advisable because the crystal CT state energy is going to possibly differ greatly from the calculated dimer CT state energy. Should one want to define two different energies for a case where E(AC) ≠ E(CA), e.g. when monomer A and B are different species, use **EAC :** and **ECA :** keywords, respectively.

The reorganization energy (λ) can be, for the sake of the example, crudely assessed from a simple formula in Eq. (4), where E(A,B) stands for the energy of a state A in the geometry of a state B. Both energies can be put in in either eV (**eV**), meV (**meV**), or kcal/mol (**kcal**), the first being the default and therefore not needing any specification of the units (**lambda : 1.5** is therefore in eV).

$$\lambda = E(S_1,T_1) + E(S_0,T_1) - E(S_1,S_1) - E(S_0,S_0) \qquad (4)$$

2.2 Automatic Generation of *Simple* Input

The generation of an input file for a larger and possibly non-planar molecule could prove challenging and frankly tiresome. For this reason, an interface between NBO analysis and *Simple* was written and is distributed under the name *Simput*, within the folder Tools. Fig. 15 shows the input file for this program.

In order to run, *Simput* needs the output files of NBO analysis of version 5.9 or higher run within *Gaussian* or *ORCA* with **$NBO CMO AONAO=W NAOMO=W PLOT $end** appended to the calculation input. This results in more than one output file excluding the standard output or the log file of the calculation. The files of interest are FILE.31 through FILE.41, and FILE.46 and FILE.49. These should be kept in a separate folder (i.e. **Bu/orbs/** as in Fig. 14).

It is advised to keep the molecule oriented in a way that the planar part of it, if present, is in the xy-plane (Figs. 7 and 10). This reduces the number of orbitals needed to define HOMO and LUMO speeding up the calculation dramatically.

```
HOMO A:109
LUMO A:110
HOMO B:109
LUMO B:110
DIR A:Bu/orbs/
DIR B:Bu/orbs/
Basis Set:6-311G.basis
Title:TDPP
Method:rate
Grid:
T X: 9.067430
T Y: 0.437786
T Z: 5.127527
R X: 181.216595
R Y: 364.563490
R Z: 359.951523
EAC:1.0
ECA:1.0
Planar:no
Processors:48
Cutoff value:0.05
dE(CT): 1.0
Reduce: 1.0
Lambda: 0.5
omit
```

Figure 14. *Simput* input file structure.

The input file (Fig. 14) starts the definition of MOs. The number *next* to **`HOMO A:`** defines that the 109th orbital in the NBO analysis is the sought HOMO of the monomer A. The same is then defined for LUMO and MOs of monomer B.

A string *next* to **`DIR A:`** keyword specifies the folder, where the above-described output files of the calculation are kept.

A string *next* to **`Basis Set:`** keyword specifies the file that is used to keep the basis set information. This file is in the *Turbomole* format, which is shown in Fig. 15. As of now, the program supports only two basis sets – Pople's 6-311G (for B, C, N, O, F, S), and Dunning et al. cc-pVDZ (for H, B, C, N, O, F).

`Planar:` keyword specifies whether *Simput* should (**`yes`**) or should not (**`no`**) trim the P-orbitals in x and y axes. *Does not work with cc-pVDZ basis set.* (See below for a similar approach.)

Similarly to that, **Cutoff value:** keyword specifies the lowest possible MO expansion coefficient that should be still used in the input. Modest use of this feature can cut the computational time drastically, while forfeiting negligible accuracy. The value of 0.05 proved to be useful for the scans of the 6D space.

For optimizations it is recommended not to use this feature (let it be set to 0.0).

This keyword works similarly to **Planar:** for cc-pVDZ basis set (or for the future basis sets with diffuse d-orbitals). If the molecule is properly oriented (the planar part is in the xy-plane), some D- and P- orbitals "pointing" in direction that is unfavorable for mixing into HOMO or LUMO will therefore have insignificant contribution to the construction of these MOs and can be safely neglected.

The rest of the keywords work similarly to the keywords in *Simple* itself as defined in the previous subchapter. Should there be a need to use one of the predefined grids, this is specified after **Grid:** keyword.

The program is called in the following way,

```
> Simput Simput_input [Simple_input]
```

where **Simput_input** is the name of the *Simput* input file. **[Simple_input]** is the optional name of the formed *Simple* input file, should it not be specified **Simple.inp** is used. If the output file bears the same name as an already existing file, the old one is *overwritten*.

An important note: *The program itself, as it is in its programming infancy, does not handle some errors in reading and does not stop when it encounters one. The input building takes no longer than* ***10*** *seconds. Should nothing happen after this short period of time, stop the program and recheck the NBO output files, basis set definition, and* Simput *input file.*

This feature is to be resolved in the future iterations of Simple*.*

```
$basis
*
c   6-311G
*
    6  s
   4563.2400000              0.00196665
    682.0240000              0.0152306
    154.9730000              0.0761269
     44.4553000              0.2608010
     13.0290000              0.6164620
      1.8277300              0.2210060
    3  s
     20.9642000              0.1146600
      4.8033100              0.9199990
      1.4593300             -0.00303068
    1  s
      0.4834560              1.0000000
    1  s
      0.1455850              1.0000000
    3  p
     20.9642000              0.0402487
      4.8033100              0.2375940
      1.4593300              0.8158540
    1  p
      0.4834560              1.0000000
    1  p
      0.1455850              1.0000000
*
$end
```

Figure 15. *Turbomole* formatted basis set definition of carbon as taken from ESML Basis Set Exchange Library (https://bse.pnl.gov/bse/portal).

3. Running *Simple*

With the input file ready, the run of the program suite is just a few keystrokes away. The user only needs to know what the purpose of his/her calculation is.

3.1 Single Point Calculations and Scans – *Simple*

These calculations are handled by program *Simple* and/or *SimplePar.* The latter is the OpenMP parallelized version of the former and it is recommended to be used when performing **scans**.

Single point calculations are to be performed by *Simple* as it writes all the important and additional information about the dimer at hand.

Both of the programs are called in the following way,

```
> Simple inputfile.inp
> SimplePar inputfile_par.inp
```

where **inputfile.inp** and **inputfile_par.inp** are the names of the input files for *Simple* and parallelized version of it, respectively.

a. Scan Results

The scan procedure if ran with user's own grid or with the predefined one results in one file next to the standard output – **Minima3** (Fig. 16).

This file contains all the found points possibly close to minima ordered from the lowest value of F (points with the maximum of T^2 or k) to the highest (points usually governed by the repulsion and of little concern to the user). The usual amount of points found by the scan is around 20,000, but it can exceed this number. As stated in the parentheses, not all of them are of interest to the user.

	F	T Z	T Y	T X	R Z	R Y	R X
1	-0.6380486E-01	3.7500000	0.0000000	-0.7500000	0.0000000	160.0000000	0.0000000
2	-0.6352576E-01	3.7500000	-0.7500000	0.0000000	20.0000000	40.0000000	20.0000000
3	-0.6282679E-01	3.7500000	-0.7500000	0.0000000	100.0000000	340.0000000	0.0000000
4	-0.5587001E-01	3.0000000	-2.2500000	0.0000000	80.0000000	0.0000000	0.0000000
5	-0.3113526E-01	3.0000000	-2.2500000	-0.7500000	20.0000000	160.0000000	140.0000000

Figure 16. First five structures of the model ethylene molecule as calculated and written by *SimplePar* into **Minima3**, as specified in Fig. 2, grid: coarse.

b. Single Point Calculation Results

The single point calculation results in two files and a standard output that is recommended to be diverted into a file (Fig. 17). The two files are **`MOAB-OrthNorm.molden`** and **`Molecule.json`**.

The first file is formatted to be readily readable by *MOLDEN* molecule viewer and it includes information on the MOs, which can be viewed by the same tool as well. This can prove helpful should the user be unsure whether the orbitals are constructed in the desired way. The .json file is a file containing formatted results of the calculation so they can be easily readable by user-created program.

```
      PERTURBATION THEORY - NO SINGLET MIXING
      TA=   -1.2205709E-03 eV     TA^2=    1.4897933E-06 eV^2       LJ(Rep)=    7.7254042E-09 eV^2    Mixed LJ-
TA^2=   -1.4820679E-06 eV^2
      TB=    1.1474257E-03 eV     TB^2=    1.3165857E-06 eV^2       LJ(Rep)=    7.7254042E-09 eV^2    Mixed LJ-
TB^2=   -1.3088603E-06 eV^2
      3x3 and 3x3 DIAGONALIZATION - NO SINGLET MIXING
      TA=   -1.2149345E-03 eV     TA^2=    1.4760657E-06 eV^2
      TB=   -1.1421532E-03 eV     TB^2=    1.3045140E-06 eV^2
      4x4 and 3x3 DIAGONALIZATION - MIXING OF S1S0 and S0S1
      T(S+)= 5.3463654E-05 eV     T(S+)^2= 2.8583623E-09 eV^2
      T(S-)=-1.6107495E-03 eV     T(S-)^2= 2.5945140E-06 eV^2
      MARCUS THEORY
      k=  8.0967617E+08 s-1    Mixed LJ-k= -8.0967617E+08 s-1

      DETAILED RESULTS
   Excitonic:

      (hAlA|hBlB)         =      36.096158 meV

      (hAhA|lBlB)         =   -1893.627068 meV

   Mixing LE states in |S+>
      Mixing(rad)         =       0.785398
      Mixing(deg)         =      44.999998

   Mixing LE states in |S->
      Mixing(rad)         =       0.785398
      Mixing(deg)         =      45.000002

   Davydov Splitting:
      dE(S- - S+)         =    -144.600383 meV

      T(S+)^2             =       2.858362E-03 meV^2
      Mixed LJ-Trp(S+)^2  =       4.867042E-03 meV^2
      dE(TT - S+)         =     -68.731237 meV
      S+  phase           =      + (in phase)

      T(S-)^2             =       2.594514E+00 meV^2
      Mixed LJ-Trp(S-)^2  =      -2.586789E+00 meV^2
      dE(TT - S-)         =      75.869145 meV
      S-  phase           =      - (out of phase)


   Biexciton Binding Energy:
      dE(T1T1 - TT)       =       0.605625 meV

   Endoergicity:
      dE(process)         =      75.958197 meV

   Boltzmann weighting:
      w(S+)               =       0.003572
      w(S-)               =       0.996428

   Marcus theory:
      lambda(reorg.)      =       0.300000 eV
      dE(internal)        =       0.000000 eV
      Rate const. k       =       8.096762E+08 s^-1
      Lifetime tau        =    1235.062 ps

      CSV:
  -1.2205709E-03;   1.1474257E-03;  -1.2149345E-03;  -1.1421532E-03;   5.3463654E-05;  -1.6107495E-03;
8.0967617E+08;   7.2192316E-02;  -6.8731237E-02;   7.5869145E-02;  -1.4460038E-01;   6.0562466E-04;
7.5958197E-02
```

Figure 17. Results section of the standard output of *Simple* single point calculation.

The first part of the standard output of *Simple* shows the steps of the calculation, the user can find warnings, geometry after transformation, and matrices and eigenvectors used to calculate the coupling elements through diagonalization.

The results themselves are printed out as in Fig. 17. They start with **PERTURBATION THEORY - NO SINGLET MIXING** headline and the first line (starting with **TA=**) shows the results of the perturbation theory approach, where **T** stands for the coupling element T of SF and **A** stands for the initial excitation of monomer A – therefore the initial state is constructed from S_1S_0 and charge transfer states. No mixing of S_0S_1 state is introduced. **B** stands for the opposite way of constructing of the initial state – S_0S_1 and charge transfer states are used. The final state is constituted of T_1T_1 and charge transfer states.

The following paragraph **3x3 and 3x3 DIAGONALIZATION - NO SINGLET MIXING** shows the results of a similar calculation with the same initial states, only performed by diagonalization subroutine, hence not by the analytical formula.

The last paragraph of this sub-block (**4x4 and 3x3 DIAGONALIZATION - MIXING OF S1S0 and S0S1**) introduces the mixing of locally excited states S_1S_0 and S_0S_1. **S+** and **S-** are the states formed by a linear combination of initial states – S_1S_0, S_0S_1, D^+D^-, and D^-D^+, where the +/– is the sign between the excitonic state coefficients within the base state basis vector. Coupling elements between these initial states and the final state are denoted **T(S+)** and **T(S-)**.

The next result is the V coupling element between S_1S_0 and S_0S_1 calculated within *Simple* theory as the 2-electron integral denoted **(hAlA|hBlB)**. The following part defines the mixing angle between S_1S_0 and S_0S_1 states.

The section under this is calculated using diagonalization subroutine and it informs the user of the energetics of the phenomenon – **Davydov Splitting:** (or more precisely **dE(S- - S+)**, the energy difference between S^+ and S^- states), **Biexciton Binding Energy:**, and **Endoergicity:**.

Skipping the two other parts, the user arrives to the final part of new results giving him/her the information about the kinetics of SF, including lifetime of triplet formation.

The `csv:` headline introduces the final, but condensed section of results. It is written in the format of "Comma Separated Values" that is recognized by spreadsheet editing programs – such as *Microsoft Excel* ®. This line can be directly copied to a spreadsheet or more presumably if the user runs a long batch of points it can be grafted into a larger .CSV file by a script. All the points can then be viewed at once as the data are ordered in the following way:

T^{A} (pert.); T^{B}(pert.); T^{A} (diag.); T^{B}(diag.); T (S+); T (S–); k (Marcus eq.); $2(h_{A}l_{A}\,|\,h_{B}l_{B})$; ΔE (TT – S+); ΔE (TT – S–); ΔE (Davydov spl.); ΔE (Biexciton binding energy); ΔE (Overall endoergicity)

The last two lines inform of the successful completion of the calculation and the elapsed CPU time – Fig. 17 shows the time needed for a single point calculation of ethylene on a single thread of Intel Xeon® E7-4850 v2 processor.

3.2 Optimizations of Dimer Structures – *SimOpt*

These calculations are handled by program *SimOpt* and *AutoOpt*. The latter being parallelized using OpenMP and intended to be used for larger batches of points – e.g. for the results of *Simple* scan. It uses Davidon-Fletcher-Powell method to find the minimum of *F*.

The programs are used to optimize the "crude" minima found by setting up a simple grid and are called in the following way,

```
> SimOpt [optional flag] inputfile.inp
> AutoOpt inputfile.inp structure_list [number of threads]
```

where **inputfile.inp** is the input file used for *Simple* calculations, **structure_list** is a file containing the points to be optimized and is formatted as **Minima3** file (Fig. 16), and **[number of threads]** is an option to specify the amount of threads to be used (note: **Processors :** specification within the input file is omitted here, **[number of threads]** is used instead).

Another option that is SimOpt specific is the use of flags **[optional flag]**.

a. Single Optimization Run Results

The optimization procedure results in a single *MOLDEN*-readable file (**inputfile.molden**) next to the standard output, for which it is recommended to be diverted into a file. The *MOLDEN*-readable output can be suppressed by the use of **-nm** flag.

The final section of standard *SimOpt* output that differs from *Simple* standard output is shown in Fig. 18.

```
Initial structure:
R X     180.000000
R Y     180.000000
R Z     180.000000
T X      -1.750000
T Y      -4.750000
T Z       4.250000

          ================== OPTIMIZATION ================                                                  Convergence: geom
grad
  0  F= -4.4970570E-07  geom=  180.000000  180.000000  180.000000   -1.750000   -4.750000    4.250000          ----------------------
-
  1  F= -1.0201295E-06  geom=  181.065190  177.961497  176.974956   -1.951956   -4.411287    4.144960  gNorm=  4.153925E-06   F    F
  2  F= -1.0377620E-06  geom=  181.283707  177.399811  176.231759   -1.980170   -4.356306    4.141029  gNorm=  1.816179E-10   T    T
     ---- Final Cycle ----
  3  F= -1.0377620E-06  geom=  181.283709  177.399793  176.231739   -1.980171   -4.356305    4.141029  gNorm=  1.779853E-12   T    T
CONVERGED in    3  cycles

Optimized structure:
R X     181.283709
R Y     177.399793
R Z     176.231739
T X      -1.980171
T Y      -4.356305
T Z       4.141029

k=  1.0949268E-06  E-Rep=  5.7164768E-08  mix*Rep-k= -1.0377620E-06  k(s^-1)=  1.6634867E+09

Program SimOpt finished.
Elapsed CPU time:   0 days   0 hours   6 minutes  19.356 seconds
```

Figure 18. Standard output of *SimOpt*, results part of the optimization.

This section contains the information on the initial geometry translation and rotation vector (under **Initial structure:**). Under this, each step of the optimization is printed out. Next to the search function value (**F**) and translation and rotation vector (**geom=**), user finds the value of the norm of the gradient (**gNorm=**) and the value of the repulsion potential. Next to it, the user finds **T** or **F** (True or False) flags that tell whether some of the criteria were met. Two criteria of reaching the optimal point are employed and are denoted in the headline **Convergence: geom grad**.

To satisfy the **geom** (geometry) criterion it is needed for the succeeding points not to differ in geometry more than a set margin. The satisfaction of the **grad** (norm of the gradient) criterion comes from the need of the finding of the minimum of the search function *F*. For such cases the norm of the gradient needs to be equal to zero. Therefore, if the norm gets within a set margin close to zero, the criterion is satisfied.

After the optimization is done, *SimOpt* prints out the number of steps it took to do so, optimized geometry translation and rotation vector (under **`Optimized structure:`**), and the values of search function, gradient and its norm.

The final section once again informs of the successful completion of the calculation and the amount of CPU time needed for it.

b. Batch Optimization Runs

The batch optimization using *AutoOpt* results in three files next to the standard output – **`OptResults`**, **`OptResultsSorted`**, and **`OptResultsReduced`**. The standard output may be disregarded by the user, for it only states the currently calculated point.

The following three files are the results of the optimization, first (**`OptResults`**) being all the results without apparent sorting, the second is sorted according to *F* (**`OptResultsSorted`**), and the last (**`OptResultsReduced`**) has identical/symmetrical results deleted (Fig. 19).

	F	T Z	T Y	T X	R Z	R Y	R X
1	-0.1734330E+00	-3.5056273	0.0000000	-0.6529431	0.0000000	11.4077840	0.0000000
2	-0.1591969E+00	-3.5403081	-0.6544088	0.0000000	89.9999972	14.4285549	0.0000002
3	-0.1971021E-01	-3.3849260	-2.0334458	0.0000000	0.0000000	90.0000000	90.0000000
4	-0.2114531E-02	-4.0665539	-0.0000001	0.0000000	90.0000911	89.9999883	89.9999085
5	-0.8242650E-03	-3.9696650	0.0000000	-1.7140479	90.0000000	90.0000000	90.0000000

Figure 19. First five structures of the model ethylene molecule as optimized and written by *AutoOpt* into **`OptResultsReduced`**, input files as specified in Fig. 2 and structures input **`Minima3`** taken from the scan calculation with coarse grid.

3.3 Tools for Geometry Extraction – *Merkur* and *Inertia*

Excluding *Simput* described in chapter 2, *Simple* comes bundled with two additional tools – *Merkur*, a tool that is used to extract dimer structures from an .xyz file containing a crystal structure, and *Inertia*, a tool that converts .xyz file of a dimer to translation and rotation vectors used in *Simple* input files (Fig. 4). Both of them can be found in the Tools folder together with *Simput*.

This also cuts down the time needed for NBO analysis, for it allows the user to only do this calculation once and let *Simple* rotate and translate the orbitals to the desired location of the monomer.

a. Merkur – Extraction Tool

The use of Merkur is simple, it only needs a suitable .xyz file containing the desired crystal structure to dissect.

The program is then called in the following way,

```
> Merkur crystal.xyz logfile.log
```

where **`crystal.xyz`** is the .xyz file containing the data on the crystal and **`logfile.log`** is the file that is to contain the resulting xyz-coordinates of the monomers.

The standard output also prints out the number of atoms and molecules in the **`crystal.xyz`**, the center of molecules, as well as the distance matrix of the molecules. This can be used to cut down the needed number of dimers to be calculated by setting up a perimeter around the first molecule. The first molecule is the molecule set into the center of the dissected crystal.

b. Inertia – A Way of Obtaining T and R

Inertia is used to convert the .xyz format to the format used by *Simple* – the vector of translation and rotation of one of the monomers (Figs. 4 and 7). It needs two .xyz files as an input, the monomer A and the monomer B, which could be for example extracted from the results of *Merkur* tool.

The program is called in the following way,

```
> Inertia MolA.xyz MolB.xyz
```

where **MolA.xyz** and **MolB.xyz** are the .xyz files defining the geometry of the initial (A) and oriented (B) molecule, respectively.

The results of the program are 5 files, where one is a premade *Simple* input file **SF.inp**, which contains only geometry related info, and the rest are *MOLDEN* files tracking the progress of *Inertia*. User can compare the result of the translation and rotation with the original .xyz file by opening **molden1.dat** (xyz) and **molden4.dat** (translation and rotation of molecule B) files. The standard input also prints out the distance between expected and optimized structure next to **structure distance:** keyword at the end, as well as the xyz-coordinates of the expected and optimized structure.

4. The *Simple* Program Suite

4.1 Obtaining the Program Suite

To obtain the source files, please visit: https://cloud.uochb.cas.cz/simple

4.2 Installing the Program Suite

Please, bear in mind that the program code as of version 3.0 has been tested only on Intel *ifort* compiler (ver. 16.0.1) on Linux distributions.

The procedure to install the *Simple* program is following:

a. Extract the .tar file containing *Simple* using **`tar -xzvf`** command.
b. Open the folder `Simple_Programs`, find `Makefile` file and open it in text processor of your choice.
c. Change the target folder **`Target=`** to a folder of your choice, save the file and close the text processor.
d. Enter command **`Make`** (this makefile points to every other makefile in all the subfolders).
e. *Done*.

4.3 Changelog ver. 3.0

Should you as a user encounter an unwanted "feature" that is not reported in the following two subsections, please do send an email to Alexandr Zaykov (see below, section **4.4. Contacts**).

a. Fixed Bugs

b. Known Bugs

c. Added Features

4.4 Contacts

Alexandr Zaykov

Institution: IOCB Prague AS CR

Email: alexandr.zaykov@uochb.cas.cz

Dr. Zdeněk Havlas

Institution: IOCB Prague AS CR

Email: havlas@uochb.cas.cz

5. References

[1] Z. Havlas, J. Michl, Isr. J. Chem. **2016**, *56*, 96.

[2] Eric A. Buchanan, Zdeněk Havlas, J. Michl, in *Adv. Quantum Chem.*, *Vol. 75* (Eds.: J. R. Sabin, E. J. Brändas), Academic Press, Burlington, **2017**, pp. 175-227.

[3] P. A. M. Dirac, Proc. R. Soc. London, A **1927**, *114*, 243.

[4] R. A. Marcus, Annu. Rev. Phys. Chem. **1964**, *15*, 155.

[5] R. A. Marcus, N. Sutin, Biochim. Biophys. Acta **1985**, *811*, 265.

[6] E. D. Glendening, J. K. Badenhoop, A. E. Reed, J. E. Carpenter, J. A. Bohmann, C. M. Morales, F. Weinhold, NBO 5.9., Theoretical Chemistry Institute, University of Wisconsin, Madison, WI, **2013**.

[7] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. J. A. Montgomery, *Gaussian 09 Revision D.01*, Gaussian Inc., Wallingford CT, **2009**.

[8] F. Neese, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2012**, *2*, 73.

[9] G. Schaftenaar, J. H. Noordik, J. Comput-Aided Mol. Des. **2000**, *14*.