OpenFMO Manual Documentation

Release 1.0

Hirotaka KITOH-NISHIOKA

Mar 31, 2018

CONTENTS

GPU-accelerated OpenFMO version 1.0 relaesed. (2018-04-01)

A PDF version of this manual is available: here.

CHAPTER

ONE

OVERVIEW

1.1 What is OpenFMO?

Open-architecture Implementation of Fragment Molecular Orbital Method for Peta-scale Computing (OpenFMO) *[TMO+07][IMH+13]* is an open-architecture program targeting the effective FMO calculations on massive parallel computers, now including post-Peta Scale systems.

Fragment molecular orbital (FMO) method *[KIA+99][NKS+02][FK04]* is a representative method to solve the electronic structures of large bio-molecules including protein, DNA, sugar chain, and so on.

OpenFMO was written from scratch by Inadomi and co-workers *[TMO+07][IMH+13]* in C-language (ca. 54,000 lines) with OpneMP and MPI hybrid programming model. *Figure 1* illustrates the MPI dynamic process management used for OpenFMO program on the basis of the master-worker execution model involving master process, worker groups, and data server.

In addition to FMO calculations, the users can do conventional restricted Hartree-Fock (RHF) calculations using the "skeleton-RHF" code of OpenFMO, which is also OpenMP and MPI hybrid program.

In this released version OpenFMO 1.0, GPU-accelerated FMO-RHF and RHF calculations can be performed on GPU cluster. *[UHS+15b][UHS+15a][UHS+]*

1.2 Capabilities and Limitations

- Single-point Ground-state Energy Calculation
- RHF with *skeleton-RHF* Code
- FMO2-RHF (FMO2: FMO Method with Two-body Correction)
- Minimum and Double-zeta Gaussian Basis Functions; Up to Third-row Atoms (namely, H - Ar)

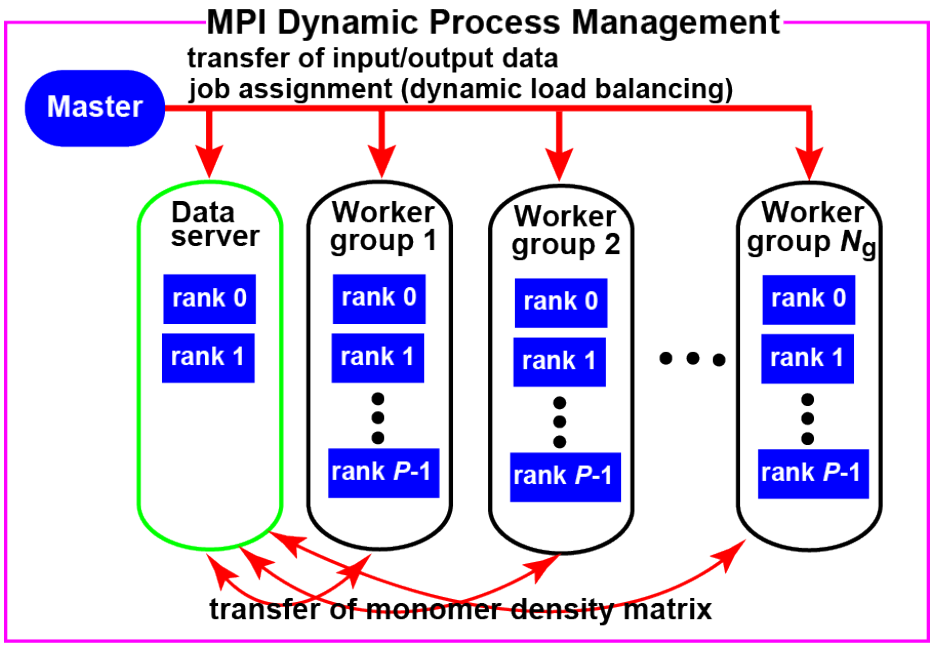

Fig. 1.1: **Figure 1.** Master-worker execution model used for OpenFMO

  - STO-3G, 6-31G, 6-31G(d), 6-31G(d,p)
- MPI + OpenMP Parallelization for RHF and FMO2-RHF
- GPU-accelerated RHF and FMO-RHF with Fermi or Kepler microarchitecture supporting double-precision floating-point operations

1.3 Citing OpenFMO

1. **OpenFMO**

   Please cite the following references *[TMO+07][IMH+13]* for any publication including the scientific works obtained from OpenFMO application.

2. **OpenFMO with Falanx Middleware**

   In addition, please cite the following reference *[TIN+]* for any publication including the scientific works obtained from OpenFMO application with Falanx middleware .

3. **GPU-accelerated RHF**

   Please cite the following references *[UHS+15b][UHS+]* when publishing the results obtained from the GPU-accelerated *skeleton RHF* code in OpenFMO.

4. **GPU-accelerated FMO-RHF**

   The following reference *[UHS+15a][UHS+]* should be cited when publishing the results utilizing the GPU-accelerated FMO-RHF code in OpenFMO.

1.4 License

1. OpenFMO:

   Copyright (c) 2007 Yuichi Inadomi, Toshiya Takami, Hiroaki Honda, Jun Maki, and Mutsumi Aoyagi

   Released under the MIT license

   http://opensource.org/licenses/mit-license.php

   The copy of the license is also included in the distribution as "LICENSE_OpenFMO".

2. GPGPU parts:

   Copyright (c) 2013 Hiroaki Umeda, Yuichi Inadomi, Toshihiro Hanawa, Mitsuo Shoji, Taisuke Boku, and Yasuteru Shigeta

   Released under the MIT license

   http://opensource.org/licenses/mit-license.php

   The copy of the license is also included in the distribution as "LICENSE_GPGPU_parts".

3. Falanx parts:

   The Falanx middleware is copyrighted by National Institute Advanced Industrial Science and Technology (AIST), and is licensed under the Apache License, Version 2.0.

   You may obtain a copy of the License at

   http://www.apache.org/licenses/LICENSE-2.0

   which is also included in the distribution as "LICENSE_Falanx".

4. Released Version 1.0:

   Copyright (c) 2018 Hiroaki Umeda, Yuichi Inadomi, Hirotaka Kitoh-Nishioka, and Yasuteru Shigeta

   Released under the MIT license

   http://opensource.org/licenses/mit-license.php

   The copy of the license is also included in the distribution as "LICENSE_Released_Version_1.0".

1.5 Contact Information

If you have a question about OpenFMO or find any bugs, please contact Hirotaka Kitoh-Nishioka at Biological Function and Information Group of Professor Shigeta in Center for Computational Sciences, University of Tsukuba.

1.6 Acknowledgments

JST-CREST: "Development of System Software Technologies for post-Peta Scale High Performance Computing"

CHAPTER

TWO

COMPILING AND INSTALLING

2.1 Prerequisites

- LINUX/UNIX Cluster Machines
- GNU C Compiler
- Intel C Compiler
- MPI Libraries (Default: Intel MPI Library) supproting MPI_Comm_spawn functions.
- Intel MKL(Math Kernel Library)

In addition, **GPU-accelerated OpenFMO** requires:

- NVIDIA Graphics card (Fermi or Kepler microarchitecture) supporting double precision floating point operations
- NVIDIA drivers for GPU

2.2 How to Get

OpenFMO program is available through the repositories hosted on github .

To check out the latest OpenFMO sources:

```
$ git clone https://github.com/OpenFMO/OpenFMO.git OpenFMO
```

2.3 How to Compile

After checking out the release archive of OpenFMO, you should move to its top directory:

```
$ cd OpenFMO
```

Makefile located in the top directory is used to build the OpenFMO executables. By typing make command with the "help" target option, the usage of the make command is printed:

```
$ make help
usage: make <target>

 original    build ofmo-master, ofmo-worker, ofmo-mserv.
 falanx      build ofmo-falanx.
 rhf         build ofmo-rhf.
 clean       remove build files.
 help        print this message.
```

In line with the printed explanation, the following command yields the "skeleton-RHF" executable, *ofmo-rhf* , in the top directory:

```
$ make rhf
```

The following command yields the three executables, *ofmo-master* , *ofmo-master* , and *ofmo-mserv* , in the top directory, which run the FMO calculations with the master-worker execute model (See *Figure 1*):

```
$ make original
```

If it is difficult to run with MPI_Comm_Spawn for your system, you can use Falanx programming middleware .

To build **GPU-accelerated OpenFMO** executables, one should modify the following parts of Makefile:

```
xcCUDAA = KEPLER
xcCUDAA = FERMI
xcCUDA = 0
```

Default value of *xcCUDA* variable is set to zero, which turns off nvcc compilation. To build the codes with nvcc for the Fermi microarchitecture, Makefile should be modified as follows:

```
#xcCUDAA = KEPLER
xcCUDAA = FERMI
#xcCUDA = 0
```

Similarly, for the Kepler microarchitecture, Makefile should be modified as follows:

```
xcCUDAA = KEPLER
#xcCUDAA = FERMI
#xcCUDA = 0
```

For getting optimal performance on your system, you may change dim2e[][] array in cuda/cuda-integ.cu.

CHAPTER

THREE

EXECUTING OPENFMO

3.1 Setup

- Set **OMP_NUM_THREADS** environment variable:

```
# csh, tcsh:
setenv OMP_NUM_THREADS (Number_of_Threads)
```

```
# sh, bash:
export OMP_NUM_THREADS=(Number_of_Threads)
```

- Set **LIBRARY_PATH** environment variable:

```
# csh, tcsh: add to shell
setenv LIBRARY_PATH $LD_LIBRARY_PATH
```

```
# sh, bash: add to shell
export LIBRARY_PATH=$LD_LIBRARY_PATH
```

- Set **OFMOPATH** environment variable that points the directory storing the OpenFMO executables (*ofmo-master* , *ofmo-worker* , and *ofmo-mserv*) or "skeleton-RHF" executable (*skel-rhf*), which is usually the directory where you compile OpenFMO programs (see *How to Compile*); you have to tell OpenFMO the path to this directory using the execution option of *ofmo-master* with *-bindir* . (See the detail in *Command Line Options*)

```
# csh, tcsh:
setenv OFMOPATH /OpenFMO/executables/install/directory
```

```
# sh, bash:
export OFMOPATH=/OpenFMO/executables/install/directory
```

- Set **SCRDIR** environment variable that points the directory storing the temporary "scratch" files for OpenFMO executables; you have to tell OpenFMO the path to this directory using the execution option of *ofmo-master* with *-scrdir* . (See the detail in *Command Line Options*)

```
# csh, tcsh:
setenv SCRDIR /Pass/To/OpenFMO/Scratch/Files/Directory
```

```
# sh, bash:
export SCRDIR=/Pass/To/OpenFMO/Scratch/Files/Directory
```

- Prepare the input files. (See *Input File Format*)

3.2 Command Line Options

By running the "skeleton-RHF" program, *skel-rhf* , with a help command-line argument, *-h* , its usage is printed:

```
$ ${OFMOPATH}/skel-rhf -h
Usage: skel-rhf [-snvh][-B buffer] input [density]
  -B buf: # buffer size (MB, default: 0)
  -s: sync
  -n: dryrun
  -v: verbose
  -h: show this help
Options for GPGPU:
  -d ndev: # devices (default:0)
```

Similarly, by running the OpenFMO program, *ofmo-master* , with a help command-line argument, *-h* , you can see some of its command-line arguments:

```
$ ${OFMOPATH}/ofmo-master -h
Usage: ofmo-master [options] [input [InitDens]]
  -ng #: # groups
  -np #: # total MPI procs
  -B #: buffer size / proc (MB, default: 512)
  -v: verbose
  -h: show this help
 Options for GPGPU:
  -d #: # devices (default:0)
```

Note that OpenFMO should be invoked with -ng and -np command-line arguments.

Table 1 lists the command-line arguments to *ofmo-master* . Note that the command-line arguments are given priority over the corresponding ones defined in Input File.

Table 3.1: **Table 1.** Command Line Arguments to OpenFMO

Argument	Acceptable Variables	Explanation	Note
-h, –help		Display the explanation of its command-line arguments	
-np #, -nmaxprocs #	#: positive integer	Total number of MPI processes	Master + Server + Worker MPI processes; 1 + 2 + *Ng* multiplied by *P* , in the case of *Figure 1*
-ng #, -ngroup #	#: positive integer	Number of Worker groups	*Ng* in the case of *Figure 1*
–niogroup #	#: positive integer	Number of server groups (default=1)	1 in the case of *Figure 1*; you can also set *niogroup* variable through $GDDI group in Input File.
–nioprocs #	#: positive integer	Size of each server group (default=1)	2 in the case of *Figure 1*; you can also set *nioprocs* variable through $GDDI group in Input File.
-B #, -buffer #	#: zero or positive integer	buffer size / proc in MB (default=512)	you can also set *buffer* variable as *nintic* through $INTGRL group in Input File.
-bindir	Path to the directory storing OpenFMO executables	OpenFMO executables location (default = Current directory)	Use **OFMOPATH** environment variable set in *Setup*
-scrdir	Path to the directory storing the temporary “scratch” files	Scratch files location (default = Current directory)	Use **SCRDIR** environment variable set in *Setup*
-d	0 or 1	Turn off/on GPGPU (default=0)	

3.3 Multi-thread Execution of “skeleton-RHF”

1. First set **OMP_NUM_THREADS** environment variable (see *Setup*).
2. Next set **OFMOPATH** environment variable (see *Setup*). Then, execute the “skeleton-RHF” program within a single cluster node:

```
$ ${OFMOPATH}/skel-rhf Input_File_Name > Log_File_Name
```

3. Execute the GPGPU-accelerated “skeleton-RHF” program within a single cluster node:

```
$ ${OFMOPATH}/skel-rhf -d 1 Input_File_Name > Log_File_Name
```

3.4 Hybrid Execution of “skeleton-RHF”

1. First set **OMP_NUM_THREADS** , **OFMOPATH** , and **SCRDIR** environment variables (see *Setup*).
2. Execute the “skeleton-RHF” program with *N* MPI processes:

```
$ mpiexec.hydra -np N ${OFMOPATH}/skel-rhf Input_File_Name > Log_File_Name
```

3. To perform GPGPU-accelerated RHF/RKS calculations with *N* MPI processes:

```
$ mpiexec.hydra -np N ${OFMOPATH}/skel-rhf -d 1 Input_File_Name > Log_File_Name
```

3.5 Execution of OpenFMO

Here, we demonstrate an example of the way OpenFMO is executed by using the cluster including the GPU nodes; one node is comprised of Intel Xeon E-5-2680 (2.6 GHz 8 cores) 2 CPUs and NVIDIA Tesla M2090 (Fermi) 4 units.

If the GPU-accelerated FMO-RHF calculation is performed using 8 nodes (2 x 8 x 8 = 128 cores and 4 x 8 = 32 GPU units) with 1 data server of 1 rank and 15 worker groups of 2 ranks (See *Figure 1*), you should run OpenFMO as follows:

1. First, set up **OFMOPATH** and **SCRDIR** environment variables (see *Setup*).
2. **OMP_NUM_THREADS** should be set to 4 (128 cores / 32 MPI processes).
3. Then, execute *ofmo-master* with the proper command-line arguments (see *Command Line Options*):

```
$ mpiexec.hydra -np 1 ${OFMOPATH}/ofmo-master -np 32 -ng 15 -d 1 -bindir $
↪{OFMOPATH} -scrdir ${SCRDIR} Input_File_Name > Log_File_Name
```

The master openMP thread of each MPI rank controls one GPU unit. Therefore, **you had better make the total number of MPI processes be equal to that of the available GPU units** (32 in the above case) in order to bring out the GPU's maximum performance on your cluster machines.

3.5.1 PBI Job File

Next, we demonstrate an example of the way OpenFMO is run on a PBS queuing system. When performing the same GPU-accelerated FMO-RHF calculations described above, you need to write a PBS job file. The minimum example "job.sh" is as follows:

```
#!/bin/sh
#PBS -j oe
#PBS -N JobName
OFMOPATH="/OpenFMO/executables/install/directory"
SCRDIR="/Pass/To/OpenFMO/Scratch/Files/Directory"
LIBRARY_PATH=${LD_LIBRARY_PATH}
cd ${PBS_O_WORKDIR}
OMP_NUM_THREADS=4
opt=""
opt+=" -np 32"
opt+=" -bindir ${OFMOPATH}"
opt+=" -B 0"
opt+=" -ng 15"
opt+=" -scrdir ${SCRDIR}"
date
set -x
mpiexec.hydra -np 1 -print-rank-map ${OFMOPATH}/ofmo-master $opt Input_Fine_Name
set +x
date
```

Then, submit the PBS job:

```
$ qsub job.sh
```

CHAPTER

FOUR

INPUT FILE FORMAT

4.1 "Skeleton-RHF"

Here, we show an example of a "skeleton-RHF" input file used for the RHF energy calculation of one water molecule with the 6-31G(d) basis sets:

```
6-31G*
0 3
O  10.438  -22.339  1.220
H  11.395  -22.339  1.220
H  10.198  -21.412  1.220
```

The input file format is schematically written by

```
Name_of_Basis_Sets
Molecular_Charge Number_of_Atoms
Atomic_Name X Y Z
```

```
Atomic_Name X Y Z
Atomic_Name X Y Z
...
```

Table 4.1: **Table 2.** Explanation of Input File Format of "Skeleton-RHF" Code

Line	Column	Description	Acceptable Variables	Note
1	1	Name of Basis Sets	sto-3g, 6-31g, 6-31g*, 6-31g**	Both capital and lower-case letters are O.K.
2	1	Molecular charge	Integer values	Only closed-shell
2	2	Number of atoms of molecule	Integer values	
4-	1	Atomic name	Up to Third-row Atoms (namely, H - Ar)	Both capital and lower-case letters are O.K.
4-	2-4	Cartesian coordinates	Floating-point numbers	In Angstrom

4.2 OpenFMO

4.2.1 Introduction

OpenFMO adopts the same input-file format as the FMO calculations implemented in GAMESS *ab initio* quantum chemistry package. Since some of the input groups used in GAMESS are directory used in the OpenFMO, the GAMESS documentations, Input Description and Further Information , are also useful for the users of OpenFMO. *Table 3* lists the input groups used in OpenFMO. Both lower-case and capital letters can be used to write the input groups and their options in the input files for OpenFMO.

Table 4.2: **Table 3.** Input Groups and Their Options Used in OpenFMO

name	function	options
$CONTRL	chemical control data	maxit, nprint, itol, icut
$BASIS	basis set	ngauss, gbasis, ndfunc, npfunc
$INTGRL	two-electron integrals	nintmx, nintic
$GDDI	MPI dynamic process management	ngroup, niogroup, nioprocs
$SCF	SCF control	diis, npunch, conv
$FMOPRP	FMO convergers	sccconv, maxscc
$FMOXYZ	atomic coordinates for FMO	Name, ZNUC, X, Y, Z
$FMO	Define FMO fragments	nfrag, icharg, indat
$FMOLMO	localized MO for FMO boundaries	Refer to "HMOs.txt" distributed by GAMESS
$FMOBND	FMO bond cleavage definition	

4.2.2 Simple Example (Glycylglycine)

Here, we show an example of an OpenFMO input file used for the FMO-RHF/STO-3G energy calculation of one glycylglycine. *Figure 2* illustrates the structure and its fragmentation points:

- Fragment 1 includes N1, H2, H3, C4, H5, and H6 atoms
- Fragment 2 includes C7, O8, N9, H10, C11, H12, and H13 atoms.
- Fragment 3 includes C14, O15, O16, and H17 atoms.

- The alpha-carbon atoms (C4 and C11) are treated as bond-detached atoms (BDAs).
- Thus, the detached bonds are assigned to the atoms on the other side (C7 and C14), which are called bond-attached atoms (BAAs).

$FMOBND group in the *input file* tells OpenFMO program the way of the bond detachments. In the *input file*, the number of server groups (See *Figure 1*) is set to 1 through the *niogroup* option in $GDDI group. As mentioned in *Command Line Options*, the command-line argument *-niogroup* can also set up the number of server groups. Similarly, the command-line argument *-nioprocs* can also set up the size of each server groups, which is defined by the *nioprocs* option in $GDDI group.

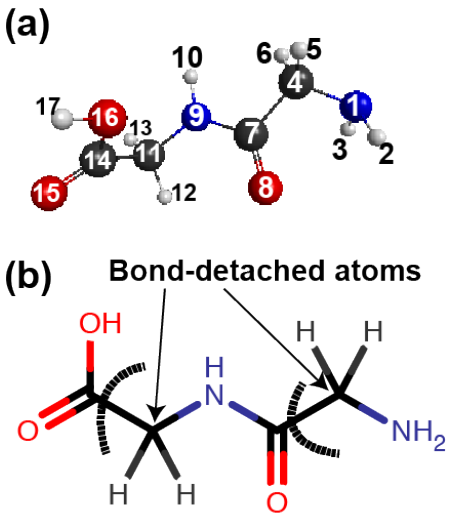

Fig. 4.1: **Figure 2.** Glycylglycine: (a) Structure and Atom Numbering (b) Fragmentation

```
$gddi niogroup=1 nioprocs=1 $end
$fmo nfrag=3 ICHARG(1)= 0,0,0
INDAT(1)=0, 1, -6, 0, 7, -13, 0, 14, -17,0  $end
$basis gbasis=sto ngauss=3 $end
$fmoxyz
N  7.0   3.5584    0.0170    0.1638
H  1.0   3.6446   -0.8687   -0.3332
H  1.0   3.4912   -0.2124    1.1546
C  6.0   2.3540    0.7121   -0.2674
H  1.0   2.2350    1.6486    0.2858
H  1.0   2.4304    0.9444   -1.3339
C  6.0   1.1558   -0.1725    0.0097
O  8.0   1.1192   -0.9807    0.9350
N  7.0   0.1194    0.0665   -0.8809
H  1.0   0.2322    0.7805   -1.5946
C  6.0  -1.1505   -0.6217   -0.8231
H  1.0  -0.9953   -1.6290   -0.4254
H  1.0  -1.5383   -0.6729   -1.8442
C  6.0  -2.1620    0.0850    0.0422
O  8.0  -3.2962   -0.3376    0.2304
O  8.0  -1.6980    1.2320    0.5903
H  1.0  -2.3743    1.6726    1.1478
$end
$FMOLMO
STO-3G 5 5
1 0  -0.117784    0.542251    0.000000    0.000000    0.850774
```

```
0 1  -0.117787    0.542269    0.802107    0.000000   -0.283586
0 1  -0.117787    0.542269   -0.401054   -0.694646   -0.283586
0 1  -0.117787    0.542269   -0.401054    0.694646   -0.283586
0 1   1.003621   -0.015003    0.000000    0.000000    0.000000
$end
$FMOBND
  -4    7 STO-3G
 -11   14 STO-3G
$end
```

4.2.3 $CONTRL Group

Table 4.3: **Table 4.** Options of $CONTRL Group

Option	Acceptable Variables	Explanation	Note
maxit =	positive integer	Maximum number of SCF iteration cycles	default: maxit = 30
nprint =	integer value	Print control flag	default: nprint = 0
itol =	positive integer	primitive cutoff factor 10**(-n)	default: itol = 20
icut =	positive integer	2e-integral cutoff factor 10**(-n)	default: icut = 12

4.2.4 $BASIS Group

OpenFMO can treat minimum and double-zeta Gaussian basis functions of up to third-row atoms, including STO-3G, 6-31G, 6-31G(d) and 6-31G(d,p).

Table 4.4: **Table 5.** Options of $BASIS Group

Option	Acceptable Variables	Explanation	Note
ngauss =	3 (for STO-3G) or 6 (for 6-31G)	the number of Gaussians	default: ngauss = 3
gbasis =	sto (STO-3G) or n31 (for 6-31G)	basis function	default: gbasis=sto
ndfunc =	0 (for STO-3G and 6-31G) or 1 (for 6-31G* and 6-31G**)	number of heavy atom polarization functions to be used	default: ndfunc = 0
npfunc =	0 (for STO-3G, 6-31G, and 6-31G*) or 1 (for 6-31G**)	number of light atom, p type polarization	default: npfunc = 1

4.2.5 $INTGRL Group

Table 4.5: **Table 6.** Options of $INTGRL Group

Option	Acceptable Variables	Explanation	Note
nintmx =	positive integer	Maximum no. of integrals in a record block	default: nintmx = 15000
nintic =	0 or positive integer	buffer size / processes in MB	default: nintic = 512; See the command-line argument *-B* in *Command Line Options*.

4.2.6 $GDDI Group

$GDDI group tells the OpenFMO program how to manage the MPI dynamic processes in FMO calculations. However, as described in *Command Line Options*, users had better use the command-line options, *-ng*, *-niogroup*, and *-nioprocs* rather than $GDDI group in the input file in order to manage the MPI dynamic processes.

Table 4.6: **Table 7.** Options of $GDDI Group

Option	Acceptable Variables	Explanation	Note
ngroup	positive integer	Number of worker groups	default: ngroup = 1; OpenFMO prefers this option to the *-ng* command-line argument
niogroup	positive integer	Number of server groups	default: niogroup=1; OpenFMO prefers this option to the *-niogroup* command-line argument
nioprocs	positive integer	Size of each server group	default: nioprocs=1; OpenFMO prefers this option to the *-nioprocs* command-line argument

4.2.7 $SCF Group

Table 4.7: **Table 8.** Options of $SCF Group

Option	Acceptable Variables	Explanation	Note
diis =	false or true	Pulay's DIIS interpolation	default: diis=true
npunch =	0, 1, or 2	option for output	
conv =	positive integer	SCF density convergence criteria 10**(-n)	default: conv=7

4.2.8 $FMOPRP Group

Table 4.8: **Table 9.** Options of $FMOPRP Group

Option	Acceptable Variables	Explanation	Note
sccconv =	positive integer	monomer SCF energy convergence criterion; 10**(-n)	default: sccconf = 7
maxscc =	positive integer	the maximum number of monomer SCF iterations	default: maxscc=30

4.2.9 $FMOXYZ Group

$FMOXYZ group defines the molecular data as follows:

```
$fmoxyz
Name ZNUC X Y Z
Name ZNUC X Y Z
Name ZNUC X Y Z
...
$end
```

- NAME = atomic name
- ZNUNC = nuclear charge
- X, Y, Z = Cartesian coordinates in Angstrom

4.2.10 $FMO Group

Table 4.9: **Table 10.** Options of $FMO Group

Option	Acceptable Variables	Explanation	Note
nfrag =	positive integer	number of distinct fragments present	default: nfrag = 1
icharg(1) =	an array of positive integers	an array of charges on the fragments	default: all 0 charges
indat(1) =	Described below	an array assigning atoms to fragments	

The *indat* option in $FMO group tells OpenFMO program an array assigning atoms to fragments. We would like to cite its explanation from the GAMESS documentation, Input Description :

> INDAT(i)=m assigns atom i is to fragment m.
> INDAT(i) must be given for each atom.
> An element is one of the following:
> I or I -J
> where I means atom I, and a pair I, -J means
> the range of atoms I-J. There must be no space
> after the "-"!
> Example:
> indat(1)=1,1,1,2,2,1 is equivalent to
> indat(1)=0, 1,-3,6,0, 4,5,0
> Both assign atoms 1,2,3 and 6 to fragment 1,
> and 4,5 to fragment 2.

4.2.11 $FMOBND Group

The atom indices involved in the bond detachment are given, in pairs for each bond. Bonds are always detached between fragments.

```
$fmobnd
-I1  J1  NAM1
-I2  J2  NAM2
-I3  J3  NAM3
...
$end
```

I and J are positive integers giving absolute atom indices. NAMs are hybrid orbital set names, defined in $FMOLMO group. Each line is allowed to have different set of NAMs, which can happen if different type of bonds are detached.

Note that OpenFMO code definitely reads $FMOBND group. If your FMO calculation does NOT need any bond detachement, its input file has to include a blank line sandwiched between $fmobnd and $end keywords (see subsection *TCNE-(Benzene)8-TCNE* in *Samples*.):

```
$fmobnd

$end
```

4.2.12 $FMOLMO or $FMOHYB Group

Hybrid orbitals are used to describe bond detachment when dividing a molecule into fragments. These are the familiar sp3 orbitals for Carbon atom, plus the 1s core orbital.

OpenFMO can treat STO-3G, 6-31G, 6-31G(d), and 6-31G(d,p) basis sets. Therefore, you can extract the orbitals (NAM=STO-3G, 6-31G, 6-31G*, or 6-31G**) to be put into $FMOLMO group taken from "HMOs.txt" distributed by GAMESS.

Note that OpenFMO code definitely reads $FMOLMO or $FMOHYB group. Even if your FMO calculation does NOT need any bond detachement, you have to write some hybrid molecular orbitals (HMOs) taken from "HMOs.txt" for $FMOLMO/$FMOHYB group in its input file. However, the HMOs are anything O.K. and the OpenFMO code does not use them in the FMO calculation. See subsection *TCNE-(Benzene)8-TCNE* in *Samples*.

CHAPTER

FIVE

SAMPLES

5.1 Introduction

In this section we show the results of several test calculations obtained from the "skeleton-RHF" and OpenFMO codes. To this end, we used the ACC nodes of System B of Institute for Solid State Physics (ISSP) supercomputer at the University of Tokyo; the configuration of ACC node is as follows:

- CPU: 2 Intel Xeon E5-2680 v3 2.5GHz (12core * 2)
- GPU: 2 nVIDIA Tesla K40 (2880 CUDA core * 2)
- MEM: DDR4-2133 128GB
- FDR InfiniBand

The compilers and libraries used are as follows:

- icc version 16.0.4 (gcc version 4.8.5 compatibility)
- mpiicc for the Intel(R) MPI Library 5.1.3 for Linux*
- Cuda compilation tools, release 7.0, V7.0.27

5.2 Glycylglycine

We again treat the FMO-RHF/STO-3G calculation of one glycylglycine illustrated in *Figure 2*, whose input file is explained in *Simple Example (Glycylglycine)*. You can download the input (`digly-ofmo-rhf.inp`), output (`digly-ofmo-rhf.out`), and PBS job (`job.sh`) files used. For the calculation, we used one ACC node with 1 data server of 1 rank and 2 worker groups of 1 rank.

We also performed the RHF/STO-3G calculation with "skeleton-RHF" code on the same glycyglycine. You can download the input (`digly-rhf.inp`), output (`digly-rhf.inp`), and PBS job (`job2.sh`) files used. For the calculation, we performed 24 threads execution of "skeleton-RHF" code with one GPU unit.

We can see the FMO-RHF/STO-3G calculation reproduces the energy of the molecule obtained from the RHF/STO-3G calculation, as listed in *Table 11*

Table 5.1: **Table 11.** Total SCF Energy (in hartree) of Glycylglycine

RHF	FMO
-483.23779373	-483.23772776

5.3 ala10 in alpha-helix conformation

We demonstrate the FMO-RHF/6-31G(d) calculation of the alanine polypeptide in an ideal alpha-helix conformation (called ala10 hereafter) plotted in *Figure 3*. The alpha-carbon atoms, as marked by the arrows in *Figure 3* (b), are treated as bond-detached atoms (BDAs). The coordinate is taken from the previous study *[NA11]*. The input (`ala10-ofmo-rhf.inp`), output (`ala10-ofmo-rhf.out`), and PBS job (`job3.sh`) files used can be downloaded. For the calculation, we used one ACC node with 1 data server of 1 rank and 2 worker groups of 1 rank.

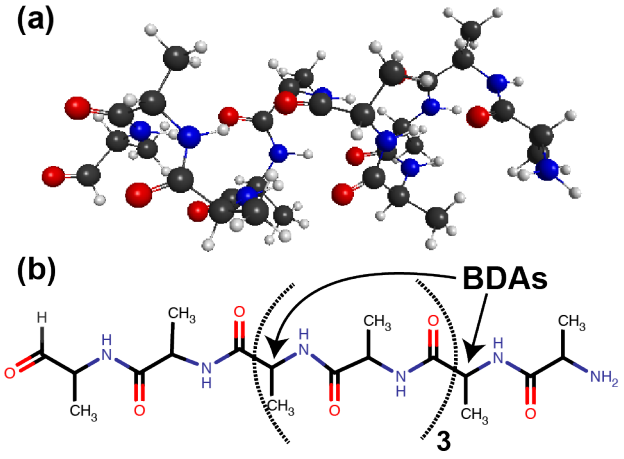

Fig. 5.1: **Figure 3.** ala10: (a) Structure (b) Fragmentation

We also performed the RHF/6-31G(d) calculation with "skeleton-RHF" code on the same ala10. You can download the input (`ala10-rhf.inp`), output (`ala10-rhf.out`), and PBS job (`job4.sh`) files used. For the calculation, we performed 24 threads execution of "skeleton-RHF" code with one GPU unit.

Table 12 compares the resultant FMO-RHF/6-31G(d) energy with the RHF/6-31G(d) one. *Table 12* also lists the RHF/6-31G(d) and FMO-RHF/6-31G(d) energies of ala10 calculated with GAMESS; you can download the input files (`Ala10AlphaRhfN31d.inp` and `Ala10AlphaFmoRhfN31d.inp`) used for GAMESS.

Table 5.2: **Table 12.** Total SCF Energy (in hartree) of ala10 from OpenFMO and GAMESS

OpenFMO		GAMESS	
RHF	FMO	RHF	FMO
-2459.51658600	-2459.51865445	-2459.51658597	-2459.51863956

5.4 TCNE-(Benzene)8-TCNE

We, here, demonstrate a FMO-RHF/6-31G(d,p) calculation of the model system TCNE-(Benzene)8-TCNE, where eight perfectly eclipsed-stack benzene molecules are sandwiched by tetracyanoethylene (TCNE), as plotted in *Figure 4*. You can download the input (`TcneBenzen8Tcne-ofmo-rhf.inp`), output (`TcneBenzen8Tcne-ofmo-rhf.out`), and PBS job (`job5.sh`) files. The coordinate is taken from the previous study *[NA11]*. The FMO calculation treats each isolated molecule as a fragment, thereby involving no bond detachement. **Note that the input file for OpenFMO has to define $FMOBND and $FMOLMO/$FMOHYB groups if its FMO calculation involves no bond detachment** , as `TcneBenzen8Tcne-ofmo-rhf.inp`; in that case, the OpenFMO code does NOT use the hybrid molecular orbitals defined by $FMOLMO/$FMOHYB group in the calculation and read the blank line defined by $FMOBND group. See *$FMOLMO or $FMOHYB Group* and *$FMOBND Group*. For the calculation, we used 8 ACC nodes with 1 data server of 1 rank and 6 worker groups of 1 rank.

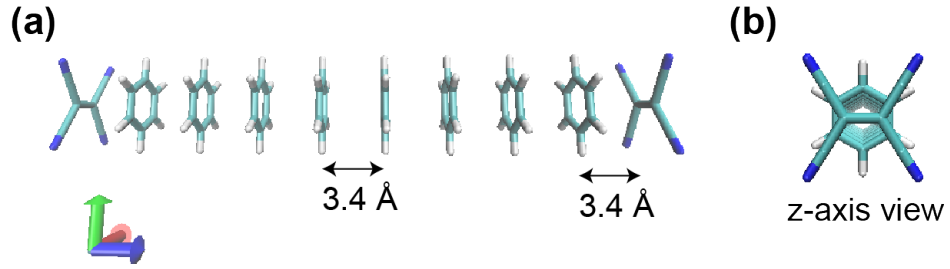

Fig. 5.2: **Figure 4.** Structure of TCNE-(Benzene)8-TCNE possessing D2h molecular symmetry

In addition, You can download the input (`TcneBenzen8Tcne-rhf.inp`), output (`TcneBenzen8Tcne-rhf.out`), and PBS job (`job6.sh`) files used for the RHF/6-31G(d,p) calculation with "skeleton-RHF" code. *Table 13* compares the resultant FMO-RHF/6-31G(d,p) energy with the RHF/6-31G(d,p) one.

Table 5.3: **Table 13.** Total SCF Energy (in hartree) of TCNE-(Benzene)8-TCNE

RHF	FMO
-2735.45391614	-2735.45313547

5.5 DNA

This subsection shows an example of the FMO-RHF/6-31G(d) calculation of dephosphorilated GTTTG B-DNA oligomer (dGTTTG) neutralized by 8 Na+ ions, as illustrated in *Figure 5* (a). You can download the input (`dna-ofmo-rhf.inp`), output (`dna-ofmo-rhf.out`), and PBS job (`job7.sh`) files. We constructed the structure of dGTTTG by using the NBA program from AmberTools package in line with the previous study *[RP14]*. *Figure 5* (b) shows how to place Na+ ion, which follows the previous study *[FWK+14]*. *Figure 5* (c) shows that the FMO calculation treats the cabon atoms at the 5' and 4' positions as BDA (bond-detached atom) and BAA (bond-attached atom), respectively. For the calculation, we used 8 ACC nodes with 1 data server of 1 rank and 6 worker groups of 1 rank.

For comparison, we performed the same calculation using OpenFMO turning off its GPU acceleration; you can see the detail of the results in the output file `dna-ofmo-rhf-offgpu.out`. *Table 14* lists the total SCF energy and computational time obtained using OpenFMO with/without GPU units. Note that the computational times taken from the output files, which are printed as "total etime = ...", are crude, but are probably useful information in this test calculation. We can confirm that the usage of GPU units does NOT affect the resultant SCF energy. The GPU-accelerated OpenFMO halves the computational time calculated with the OpenFMO even though the system size is

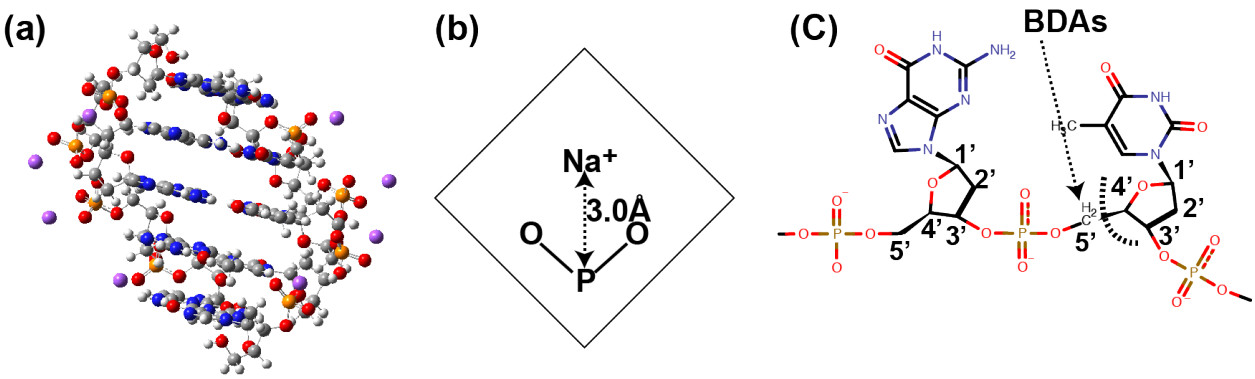

Fig. 5.3: **Figure 5.** dGTTTG with 8 Na+ ions: (a) Structure, (b) Placement of Na+ ion, and (c) Fragmentation

modest and the computational settings, such as the types of the two-electron integral, are not optimized.

Table 5.4: **Table 14.** Total SCF Energy (in hartree) and Computational Time (seconds) of dGTTTG with 8 Na+ ions obtained from OpenFMO

	SCF Energy	Computational Time
CPU + GPU	-14004.75470797	1049.715826
Only CPU	-14004.75470797	2154.053616

BIBLIOGRAPHY

[FK04] Dmitri G. Fedorov and Kazuo Kitaura. The importance of three-body terms in the fragment molecular orbital method. *J. Chem. Phys.*, 120(15):6832–6840, 2004. URL: http://aip.scitation.org/doi/abs/10.1063/1.1687334, arXiv:http://aip.scitation.org/doi/pdf/10.1063/1.1687334, doi:10.1063/1.1687334.

[IMH+13] Yuichi Inadomi, Jun Maki, Hiroaki Honda, Toshiya Takami, Taizo Kobayashi, Mutsumi Aoyagi, and Kasuo Minami. Performance tuning of parallel fragment molecular orbital program (openfmo) for effective execution on k-computer. *J. Comput. Chem., Jpn.*, 12(2):145–155, 2013. doi:10.2477/jccj.2012-0031.

[KIA+99] Kazuo Kitaura, Eiji Ikeo, Toshio Asada, Tatsuya Nakano, and Masami Uebayasi. Fragment molecular orbital method: an approximate computational method for large molecules. *Chem. Phys. Lett.*, 313(3–4):701 – 706, 1999. URL: http://www.sciencedirect.com/science/article/pii/S000926149900874X, doi:http://dx.doi.org/10.1016/S0009-2614(99)00874-X.

[NKS+02] Tatsuya Nakano, Tsuguchika Kaminuma, Toshiyuki Sato, Kaori Fukuzawa, Yutaka Akiyama, Masami Uebayasi, and Kazuo Kitaura. Fragment molecular orbital method: use of approximate electrostatic potential. *Chem. Phys. Lett.*, 351(5–6):475 – 480, 2002. URL: http://www.sciencedirect.com/science/article/pii/S0009261401014166, doi:http://dx.doi.org/10.1016/S0009-2614(01)01416-6.

[TMO+07] Toshiya Takami, Jun Maki, Jun-ichi Ooba, Yuichi Inadomi, Hiroaki Honda, Taizo Kobayashi, Rie Nogita, and Mutsumi Aoyagi. Open-architecture implementation of fragment molecular orbital method for peta-scale computing. *CoRR*, 2007. URL: http://arxiv.org/abs/cs/0701075, arXiv:cs/0701075.

[TIN+] Atsuko Takefusa, Tsutomu Ikegami, Hidemoto Nakada, Ryousei Takano, Takayuki Tozawa, and Yoshio Tanaka. Scalable and highly available fault resilient programming middleware for exascale computing. The International Conference for High Performance Computing, Networking, Storage and Analysis: SC14. URL: http://sc14.supercomputing.org/sites/all/themes/sc14/files/archive/tech_poster/poster_files/post217s2-file2.pdf.

[UHS+] Hiroaki Umeda, Toshihiro Hanawa, Mitsuo Shoji, Taisuke Boku, and Yasuteru Shigeta. Large-scale mo calculation with gpu-accelerated fmo program. The International Conference for High Performance Computing, Networking, Storage and Analysis: SC15. URL: http://sc15.supercomputing.org/sites/all/themes/SC15images/tech_poster/tech_poster_pages/post198.html.

[UHS+15a] Hiroaki Umeda, Toshihiro Hanawa, Mitsuo Shoji, Taisuke Boku, and Yasuteru Shigeta. Gpu-accelerated fmo calculation with openfmo: four-center inter-fragment coulomb interaction. *J. Comput. Chem., Jpn.*, 14(3):69–70, 2015. doi:10.2477/jccj.2015-0041.

[UHS+15b] Hiroaki Umeda, Toshihiro Hanawa, Mitsuo Shoji, Taisuke Boku, and Yasuteru Shigeta. Performance benchmark of fmo calculation with gpu-accelerated fock matrix preparation routine. *J. Comput. Chem., Jpn.*, 13(6):323–324, 2015. doi:10.2477/jccj.2014-0053.

[FWK+14] Kaori Fukuzawa, Chiduru Watanabe, Ikuo Kurisaki, Naoki Taguchi, Yuji Mochizuki, Tatsuya Nakano, Shigenori Tanaka, and Yuto Komeiji. Accuracy of the fragment molecular orbital (fmo) calculations for dna: total energy, molecular orbital, and inter-fragment interaction energy. *Comput. Theor.*

Chem., 1034:7 – 16, 2014. URL: http://www.sciencedirect.com/science/article/pii/S2210271X14000577, doi:https://doi.org/10.1016/j.comptc.2014.02.002.

[NA11] Hirotaka Nishioka and Koji Ando. Electronic coupling calculation and pathway analysis of electron transfer reaction using ab initio fragment-based method. i. fmo–lcmo approach. *J. Chem. Phys.*, 134(20):204109, 2011. URL: http://dx.doi.org/10.1063/1.3594100, arXiv:http://dx.doi.org/10.1063/1.3594100, doi:10.1063/1.3594100.

[RP14] Pablo Ramos and Michele Pavanello. Quantifying environmental effects on the decay of hole transfer couplings in biosystems. *J. Chem. Theory Comput.*, 10(6):2546–2556, 2014. URL: https://doi.org/10.1021/ct400921r, arXiv:https://doi.org/10.1021/ct400921r, doi:10.1021/ct400921r.