Manual
MULTILEVEL–version 4.4

Yan Zhao, Jocelyn M. Rodgers, Benjamin J. Lynch, Núria González-García, Patton L. Fast, Jingzhi Pu, Yao-Yuan Chuang, Benjamin A. Ellingson, Rubén Meana-Pañeda, and Donald G. Truhlar
Department of Chemistry and Supercomputer Institute, University of Minnesota, Minneapolis, MN 55455-0431

Program Version: 4.4
Program Version Date: November 13, 2013
Manual Version Date: July 21, 2021

Abstract: MULTILEVEL is a computer program written in FORTRAN90 for performing geometry optimizations and calculating single-point energies, gradients, and/or Hessians using two types of dual-level and multi-level methods: integrated molecular orbital (IMO) methods and linear combination (LC) methods. The IMO option is implemented using the integrated molecular orbital method with harmonic cap (IMOHC), and the LC methods include the linear mixing of Hartree-Fock and molecular orbital (HF||MO) method, the scaling all correlation (SAC) method, the Gaussian-x (Gx) methods, the infinite basis (IB) method, and the multi-coefficient (MC) methods, where the MC methods include the multi-coefficient correlation method (MCCM), the multi-coefficient scaling all correlation (MCSAC) methods, the multi-coefficient Gaussian-x (MCG*x*) methods, the multi-coefficient quadratic configuration interaction with single and double excitations (MC-QCISD) method, and doubly hybrid density functional theory (DHDFT) methods. In the case of DHDFT, the MULTILEVEL-v4.4 program can perform calculations by the MC3BB and MC3MPW methods. The program contains several options for optimizing stationary points (minima and saddle points), and it also contains the Nudged Elastic Band (NEB) method, which can be used both to optimize saddle points and to

calculate reaction paths. The MULTILEVEL-v4.4 program can also use the optimizers in the GAUSSIAN program to perform geometry optimization. Note that the use of MULTILEVEL does not require the GAUSSIAN program, though; it can be used with the output of any electronic structure package. The code contains 30 test runs.

Licensing

Table of Contents

Chapter One

1

1. Introduction

MULTILEVEL is a computer program for calculating optimized geometries, single-point energies, single-point gradients, and/or single-point Hessians using dual-level and multi-level methods. The present version supports two quite different types of calculations, namely integrated molecular orbital (IMO) methods and linear combination (LC) methods. The IMO methods are implemented using the integrated molecular orbital method with harmonic cap (IMOHC) algorithm, and the LC methods include the linear mixing of Hartree-Fock and molecular orbital (HF||MO) methods, the scaling all correlation (SAC) method, the Gaussian-*x* (G*x*) methods, the infinite basis (IB) method, and the multi-coefficient (MC) methods. The MC methods include the multi-coefficient correlation method (MCCM), the multi-coefficient scaling all correlation (MCSAC) method, the multi-coefficient Gaussian-*x* (MCG*x*) methods, the minimal multi-coefficient Gaussian-*x* (MMCG*x*) methods, and the multi-coefficient quadratic configuration interaction with single and double excitations (MC-QCISD) method.

As implied by its name, all calculations carried out by this program involve the combination of two or more levels, in particular, two or more levels of quantum mechanical electronic structure theory or one or more levels of quantum mechanical electronic structure theory combined with a molecular mechanics level. The levels may be *ab initio* or semi-empirical. Each level of calculation is obtained by making a call to an external electronic structure or molecular mechanics package. The external programs enabled in version 4.4 of this code are GAUSSIAN94, GAUSSIAN98, GAUSSIAN03 and GAUSSIAN09, but the structure of the code is very modular so that one could also use other packages with straightforward modifications.

Five kinds of calculations can be done with MULTILEVEL: energies, gradients, Hessians, optimizations using MULTILEVEL routines, and optimizations using the external electronic structure packages. Whole reaction path minimizations can also be performed by using the Nudged Elastic Band (NEB) method as the algorithm. This option is a special case of the optimizations using MULTILEVEL routines because it does not optimize one point but a whole reaction path. The list below indicates the order in which each of the calculations will be done within the program. If the user has not selected one of these calculations then the program jumps to the next calculation in the list.

External optimization. This option obtains a geometry from the external electronic structure package. The use of this option is recommended to give a good starting point geometry. The geometry resulting from this calculation can be used as a starting point for step 2, or, if step 2 is omitted, it is used for step 3, 4, or 5.

MULTILEVEL Optimization. This option optimizes a geometry with one of the MULTILEVEL methods. This optimized geometry is then used for all remaining calculations. If this optimization has been carried out, at least one of the three options below will be exercised, i.e., at the very least an energy will be calculated.

Hessian calculation. Frequencies and normal mode coordinate are also calculated for the Hessian calculation.

Gradient calculation. This is only carried out if a Hessian was not calculated since a Hessian calculation outputs a gradient and an energy as well.

Energy calculation. This is only done if a Hessian or gradient was not calculated because both of these calculations yield an energy as well.

A given run performs step 1 or step 2 only once, although a MULTILEVEL optimization may follow a single-level optimization. However, the Hessian, gradient, and energy calculations have been implemented so that multiple calculations may be carried out in one run of MULTILEVEL. Additionally, as can be seen in the theory section, many of the LC calculations require identical calls to the electronic structure package. Thus in order to minimize the expense of multiple LC calculations, certain groups of calculations

'cooperate' and share electronic structure information from calls made. Specifically, information is currently shared among three possible groups: the G3/MCG3 group, the G2/MCG2 group, and the SAC/MCSAC/IB/EIB/MCCM group. Once the electronic structure program calls necessary for the calculations specifically requested by the user have been made, then all SAC, MCSAC, IB, EIB, and MCCM calculations that are able to be carried out with these values will be calculated with the default coefficients. This sharing of information and calculation of lower-level default values among the SAC, MCSAC, IB, EIB, and MCCM calculations may be turned off by the user if so desired. (See the NOCOOP keyword in the LC input section.)

MULTILEVEL is run from one input file, called ml.inp; and one output file, called ml.out, is created. Chapter 6 describes the contents of the input file, and further description of the files necessary to run MULTILEVEL and the files created by MULTILEVEL can be found in Chapter 7. Chapter 8 provides installation and usage instructions.

There are two general versions, version 1 (v1) and version 2 (v2), of the coefficients for the EIB, SAC, MCSAC, MCCM, and MCG*x* linear combination methods. The two general versions each have subversions depending on how spin-orbit and core-correlation effects are included. The subversions are denoted m when both spin-orbit and core-correlation effects are treated implicitly, denoted s when spin-orbit effects are treated explicitly and core-correlation effects are treated implicitly, and denoted sc when both spin-orbit and core-correlation effects are treated explicitly. The version of the coefficients used in a calculation is controlled by the VERSION keyword for each of the methods. Table 1 shows which versions of the coefficients are available for each of the different methods in MULTILEVEL version 4.4. In this table the presence of an entry in a given row and column indicates that the version does exist for that method, and the value of the entry denotes the number of molecules in the training set used to obtain the coefficients. The reference for each entry is indicated in brackets. Coefficient sets without a reference were determined in the same way as the published coefficients but have not been published; the reference for those coefficients is either version 4.4 of the code or this manual. See method descriptions in Chapter 4 and the VERSION keyword of

Section 6.E for additional information on the different versions of the coefficients that are available.

Scheme 1: Methods available in MULTILEVEL-v4.4

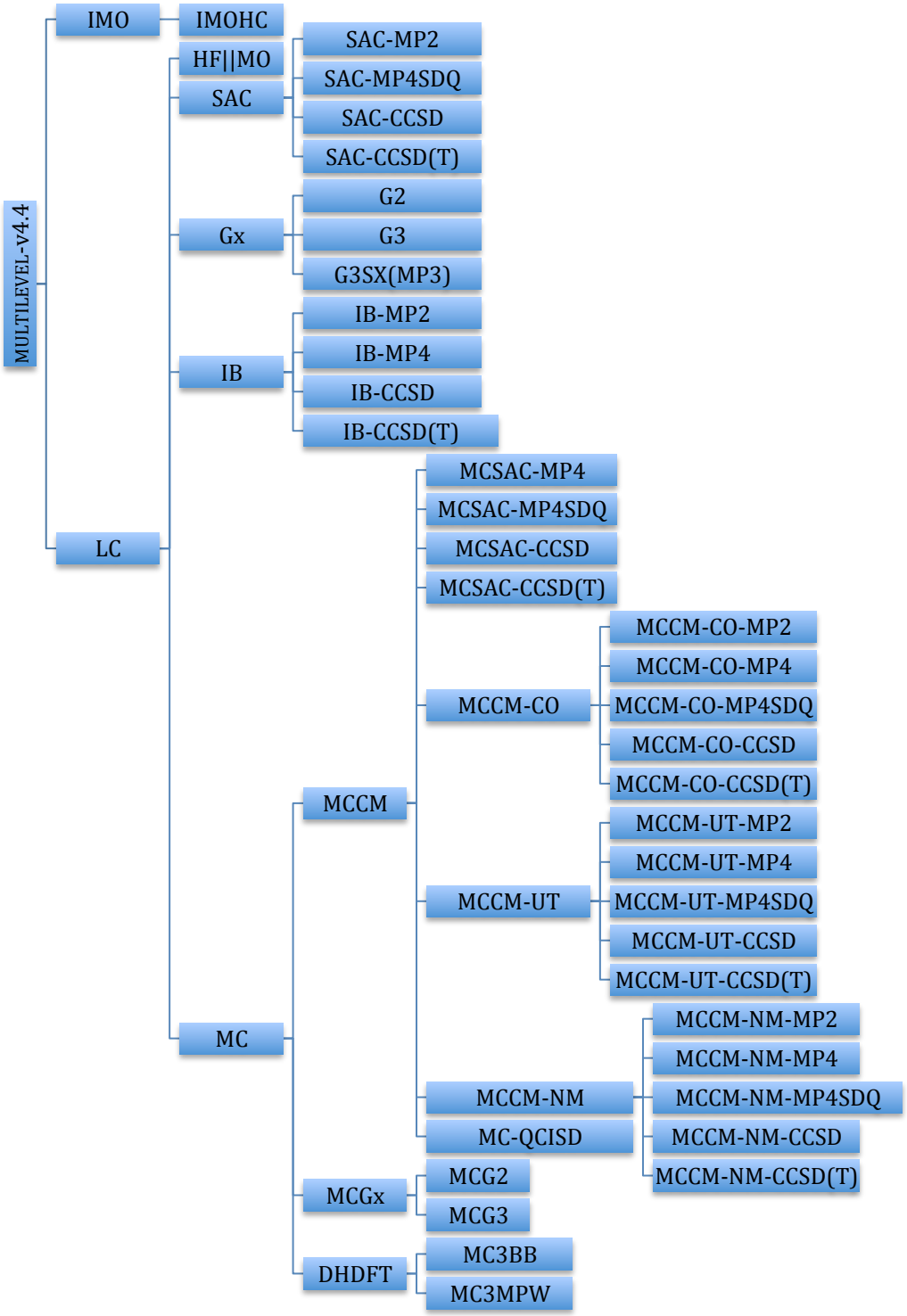


Table 1. Versions of the coefficients available in MULTILEVEL version 4.4 for the SAC, MCSAC, EIB, MCCM, and MCG*x* LC methods.

Method	Version								
	1s	1m	1sc	2m	2s	2sc	HCO-s	v3m	v3s
SAC-MP2/pDZ	[Fa99]		[Fa99b]	82 [Fa00]	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
SAC-MP2/pTZ	[Fa99]		[Fa99b]	82 [Fa00]	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-MP4SDQ/pDZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
SAC-MP4SDQ/pTZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-MP4/pDZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-MP4/pTZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-CCSD/pDZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-CCDS/pTZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-CCSD(T)/pDZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
SAC-CCSD(T)/pTZ	[Fa99]		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-MP2/pDZ			[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-MP2/pTZ			[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-MP4SDQ/pDZ			[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-MP4SDQ/pTZ			[a][Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-MP4/pDZ			[a] [Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-MP4/pTZ			[a] [Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-CCSD/pDZ			[a] [Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-CCDS/pTZ			[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]

MCSAC-CCSD(T)/pDZ		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCSAC-CCSD(T)/pTZ		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-CO-MP2		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
MCCM-CO-MP2; MG3; D+d		[Fa99b]	82 [Fa00]	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
MCCM-UT-MP2			82 [Fa00]			[Ly02]	[Ly02]	[Ly02]
MCCM-CO-MP4SDQ		[a][Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-UT-MP4SDQ		[a] [Fa99b]	82 [Fa00]	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
MCCM-CO-MP4		[a] [Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-UT-MP4		[a] [Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-CO-CCSD		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-UT-CCSD		[Fa99b]	82 [Fa00]	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
MCCM-CO-CCSD(T)		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-UT-CCSD(T)		[Fa99b]	82	82 [Tr99]	82 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCCM-NM-CCSD(T)		[Fa99b]	82	82 [Tr99]	82 [Tr99]			
MCG2	[Fa99c]	[Fa99c]	52	52 [Tr99]	52 [Tr99]	[Ly02]	[Ly02]	[Ly02]
MCG3	[b][Fa99c]	[b][Fa99c]	82 [Fa00]	82 [Tr99]	82 [Tr99]	[Fa01]	[Ly02]	[Ly02]
MC-QCISD			82 [Fa00]			[Fa01]	[Ly02]	[Ly02]

[a] This version involves an MP4D component that has been removed in version 2, and therefore this version can only be run with version 1.0 or 1.5 of MULTILEVEL.

[b] This version involves an MP4/6-31G(2df,p) component that has been removed in version 2 of the coefficients, and therefore this version can only be run with version 1.0 or 1.5 of MULTILEVEL. Version ANLsc is also available in version 1.5 of MULTILEVEL.

Chapter Two

2

2. References for MULTILEVEL Program

The recommended reference for the current version of the code is given below in two styles, first in *J. Chem. Phys.* style, then in *J. Amer. Chem. Soc.* style.

J. Chem. Phys. style if MULTILEVEL is used with GAUSSIAN94:

MULTILEVEL-version 4.4/G94 by Y. Zhao, J. M. Rodgers, B. J. Lynch, N. González-García, P. L. Fast, J. Pu,Y. -Y. Chuang, B. A. Ellingson, R. Meana-Pañeda, and D. G. Truhlar, University of Minnesota, Minneapolis, 2006, based on GAUSSIAN94, by M. J. Frisch, G. W. Trucks, H. B. Schlegel, P. M. W. Gill, B. G. Johnson, M. A. Robb, J. R. Cheeseman, T. Keith, G. A. Petersson, J. A. Montgomery, K. Raghavachari, M. A. Al-Laham, V. G. Zakrzewski, J. V. Ortiz, J. B. Foresman, J. Cioslowski, B. B. Stefanov, A. Nanayakkara, M. Challacombe, C. Y. Peng, P. Y. Ayala, W. Chen, M. W. Wong, J. L. Andres, E. S. Replogle, R. Gomperts, R. L. Martin, D. J. Fox, J. S. Binkley, D. J. Defrees, J. Baker, J. P. P. Stewart, M. Head-Gordon, C. Gonzalez, and J. A. Pople, Gaussian Inc., Pittsburgh, 1995.

J. Chem. Phys. style if MULTILEVEL is used with GAUSSIAN98:

MULTILEVEL-version 4.4/G98 by Y. Zhao, J. M. Rodgers, B. J. Lynch, N. González-García, P. L. Fast, J. Pu, Y.-Y. Chuang, B. A. Ellingson, R. Meana-Pañeda, and D. G. Truhlar, University of Minnesota, Minneapolis, 2006, based on GAUSSIAN98, by M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamos, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K.

Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. G. Johnson, W. Chen, M. W. Wong, J. L. Andres, M. Head-Gordon, E. S. Replogle and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 1998.

J. Chem. Phys. style if MULTILEVEL is used with GAUSSIAN03:

MULTILEVEL-version 4.4/G03 by Y. Zhao, J. M. Rodgers, B. J. Lynch, N. González-García, P. L. Fast, J. Pu, Y.-Y. Chuang, B. A. Ellingson, R. Meana-Pañeda, and D. G. Truhlar, University of Minnesota, Minneapolis, 2006, based on GAUSSIAN03, by M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 2003.

J. Chem. Phys. style if MULTILEVEL is used with GAUSSIAN09:

MULTILEVEL-version 4.4/G09 by Y. Zhao, J. M. Rodgers, B. J. Lynch, N. González-García, P. L. Fast, J. Pu, Y.-Y. Chuang, B. A. Ellingson, R. Meana-Pañeda, and D. G. Truhlar, University of Minnesota, Minneapolis, 2012, based on GAUSSIAN09, by 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. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox, Gaussian, Inc., Wallingford CT, 2009.

J. Amer. Chem. Soc. style if MULTILEVEL is used with GAUSSIAN94:

Zhao, Y.; Rodgers, J. M.; Lynch, B. J.; González-García, N.; Fast, P. L.; Chuang, Y.-Y.; Pu, J.; Ellingson, B.A.; Meana-Pañeda, R.; Truhlar, D. G.; MULTILEVEL-version 4.4/G94; University of Minnesota, Minneapolis, 2005 based on Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Gill, P. M. W.; Johnson, B. G.; Robb, M. A.; Cheeseman, J. R.; Keith, T.; Petersson, G. A.; Montgomery, J. A.; Raghavachari, K.; Al-Laham, M. A.; Zakrzewski, V. G.; Ortiz, J. V.; Foresman, J. B.; Cioslowski, J.; Stefanov, B. B.; Nanayakkara, A.; Challacombe, M.; Peng, C. Y.; Ayala, P. Y.; Chen, W.; Wong, M. W.; Andres, J. L.; Replogle, E. S.; Gomperts, R.; Martin, R. L.; Fox, D. J.; Binkley, J. S.; Defrees, D. J.; Baker, J.; Stewart, J. P. P.; Head-Gordon, M.; Gonzalez, C.; Pople, J. A.; GAUSSIAN94; Gaussian Inc., Pittsburgh, 1995.

J. Amer. Chem. Soc. style if MULTILEVEL is used with GAUSSIAN98:

Zhao, Y.; Rodgers, J. M.; Lynch, B. J.; González-García, N.; Fast, P. L.; Pu, J.;Chuang, Y.-Y.; Ellingson, B.A.; Meana-Pañeda, R.; Truhlar, D. G.; MULTILEVEL-version 4.4 /G98; University of Minnesota, Minneapolis, 2005 based on Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Zakrzewski, V. G.; Montgomery, J . A.; Stratmann, R. E.; Burant, J. C.; Dapprich, S.; Millam, J. M.; Daniels, A. D.; Kudin, K. N.; Strain, M. C.; Farkas, O.; Tomasi, J.; Barone, V.; Cossi, M.; Cammi, R.; Mennucci, B.; Pomelli, C.; Adamos, C.; Clifford, S.; Ochterski, J.; Petersson, G. A.;

Ayala, P. Y.; Cui, Q.; Morokuma, K.; Malick, D. K.; Rabuck, A. D.; Raghavachari, K.; Foresman, J. B.; Cioslowski, J.; Ortiz, J. V.; Stefanov, B. B.; Liu, G.; Liashenko, A.; Piskorz, P.; Komaromi, I.; Gomperts, R.; Martin, R. L.; Fox, D. J.; Keith, T.; Al-Laham, M. A.; Peng, C. Y.; Nanayakkara, A.; Gonzalez, C.; Challacombe, M.; Gill, P. M. W.; Johnson, B. G.; Chen, W.; Wong, M. W.; Andres, J. L.;
Head-Gordon, M.; Replogle, E. S.; Pople, J. A.; GAUSSIAN98; Gaussian, Inc., Pittsburgh PA, 1998.

J. Amer. Chem. Soc. style if MULTILEVEL is used with GAUSSIAN03:

Zhao, Y.; Rodgers, J. M.; Lynch, B. J.; González-García, N.; Fast, P. L.; Pu, J.; Chuang, Y.-Y; Ellingson, B.A.; Meana-Pañeda, R.; Truhlar, D. G.; MULTILEVEL-version 4.4/G03; University of Minnesota, Minneapolis, 2005 based on M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala,
K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford,
J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople.; GAUSSIAN98; Gaussian, Inc.; Pittsburgh PA, 2003.

J. Amer. Chem. Soc. style if MULTILEVEL is used with GAUSSIAN09:

Zhao, Y.; Rodgers, J. M.; Lynch, B. J.; González-García, N.; Fast, P. L.; Pu, J.; Chuang, Y.-Y; Ellingson, B.A.; Meana-Pañeda R.; Truhlar, D. G.; MULTILEVEL-version 4.4/G09; University of Minnesota, Minneapolis, 2012 based on 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. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox, Gaussian, Inc., Wallingford CT, 2009.

Additional references are given in the documentation for GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, and GAUSSIAN09.

Chapter Three

3

3. General Program Description

MULTILEVEL is written in standard FORTRAN90. The program is written using modular subprograms called "hooks." There are four main hooks for MULTILEVEL values: energy hook (MLEHOOK), gradient hook (MLGHOOK), Hessian hook (MLHHOOK), and optimization hook (MLOHOOK). The energy, gradient, and Hessian hooks have been designed in order to allow MULTILEVEL calculations with a variety of methods that are available. They make a minimum number of calls to the electronic structure package in order to carry out these calculations. Each of these three MULTILEVEL hooks makes calls to "subhooks," one for each of the methods enabled in MULTILEVEL.

The optimization hook is structured somewhat differently in that it makes calls to routines for the different optimization algorithms available in MULTILEVEL. When energies, gradients, and/or Hessians are required for the optimization algorithm, three different hooks may be called for this information: OPTEHK, OPTGHK, and OPTHHK respectively. These three hooks differ from other energy, gradient, and Hessian hooks in that they serve only as a front end to the hooks for each type of MULTILEVEL method, calling the appropriate hook for the method that one is using for optimization.

Whenever the program calls MLHOOK, but not when it calls OPTHHK, the program also calculate the harmonic vibrational frequencies and normal mode coordinates and write them to the output file. This calculation is carried out in mass-scaled Cartesian coordinates, and the translations and rotations are projected out before diagonalizing the mass-scaled force constant matrix.

As all of the methods require multiple calls to an electronic structure package, program hooks have been written to make calls to the appropriate electronic structure program and then extract the energies (PROGEHK), gradients (PROGGHK), and Hessians (PROGHHK) from the output of that program. Additionally there is a program hook (MPROGEHK) that, for greater efficiency, will return all the energies available in the output file, not just the calculation specifically requested. Unfortunately the output files produced by GAUSSIAN for gradient and Hessian runs contain only the one gradient and/or Hessian that was requested, so it is not possible to increase the efficiency of those two program hooks.

Besides these four hooks, there is also a program hook to do geometry optimizations (PROGOHK) using the electronic structure methods of the external program to provide a starting geometry for all calls to ML hooks. The MULTILEVEL code currently has program hooks for calling GAUSSIAN94, GAUSSIAN98, GAUSSIAN03 and GAUSSIAN09, but in the future it is planned that it will contain hooks that will call other electronic structure packages, for example, ACESII and/or GAMESS.

The hooks are called in the following order to create the functionality described in the introduction: PROGOHK, MLOHOOK, MLHHOOK, MLGHOOK, MLEHOOK.

See Chapter 7 for a complete listing of all the files that compose the source code of MULTILEVEL, as well as a description of the subprograms and modules within the source code.

Chapter Four

4

4. Theoretical Background

This chapter describes the IMO and LC methods available in this version of the MULTILEVEL code.

4.A. Integrated Molecular Orbital (IMO) Theory

Integrated molecular orbital (IMO) methods combine a calculation with an affordable level of theory and basis set on a large system (typically a system for which a high level of theory and/or a large basis set are prohibitively expensive) with a higher-level calculation on a capped sub-system of the large system (where the capped sub-system is a system for which high levels of theory and/or basis sets are affordable). The higher-level calculation on the capped sub-system contributes higher-level character to the composite calculation on the large system to make up for not doing the higher-level calculation on the large system. [Ma95, Hu96, Co96, No97, Co97, Sv96]

There are several approaches to combining the higher- and lower-level calculations. In MULTILEVEL we have implemented the IMO with harmonic cap (IMOHC) method. [Co98] This method is summarized in the following section.

4.A.1. Integrated Molecular Orbital Method with Harmonic Cap (IMOHC)

For the case of hydrogen being the capping atom, the IMOHC energy is written as

$$E_{\mathrm{ES}}(\mathrm{I}) = E_{\mathrm{CSS}}(\mathrm{HL}) - E_{\mathrm{CSS}}(\mathrm{LL}) + E_{\mathrm{ES}}(\mathrm{LL}) + \frac{1}{2}k_1(R_{\mathrm{XH}} - R_{\mathrm{eq}})^2$$

$$+ \frac{1}{2}k_2(\theta_{\mathrm{XYH}} - \theta_{\mathrm{eq}})^2 + \frac{1}{2}k_3(\phi_{\mathrm{XYZH}} - \phi_{\mathrm{eq}})^2 \qquad (1)$$

where I denotes integrated, HL indicates the higher level of calculation, LL is the lower level of calculation, ES denotes the entire system (or large system), CSS is the capped subsystem, k_1, k_2, k_3 are the force constants for the bond stretching, bend, and torsion introduced by the cap, R_{XH} is the X–H bond length where X is the atom capped by H, R_{eq} is the equilibrium bond length for the X–H bond length, θ_{YXH} and θ_{eq} are the YXH angle and the equilibrium YXH angle, respectively, and ϕ_{ZYXH} and ϕ_{eq} are the torsion angle variable and its equilibrium value for the ZYXH torsion. The components of the gradient and the Hessian are given by the first and second partial derivatives of eq. (1), respectively.

4.B. Linear Combination (LC) Methods

The LC methods take linear combinations of different levels of theory to obtain an extrapolated energy, gradient, or Hessian. There are three different approaches to the extrapolation. The first is the scaling all correlation (SAC) method; this method scales the correlation energy in an attempt to make up for the incompleteness in the level of correlation used as well as the one-electron basis set. The second approach is the infinite basis method (IB); this method uses one level of correlation with two different basis sets and attempts to extrapolate to the infinite basis limit. The third approach is the multi-coefficient correlation method (MCCM); this method combines the approaches of SAC and IB in an attempt to reach the full configuration interaction (CI) limit with an infinite basis set. The combination of full CI (FCI) with an infinite one-electron basis (IB) is called complete CI (CCI). These three approaches are described in more detail below.

Throughout this manual we will use the pipe "|" to represent the energy difference either between two one-electron basis sets B1 and B2, between two many-body levels L1 and L2, e.g., Møller-Plesset second-order perturbation theory and Hartree-Fock theory, or between the level increments obtained with two different basis sets. The energy difference between two basis sets will be represented as

$$\Delta E(\mathrm{L}/\mathrm{B2}\,|\,\mathrm{B1}) \equiv E(\mathrm{L}/\mathrm{B2}) - E(\mathrm{L}/\mathrm{B1}) \qquad (2)$$

where L is a particular electronic structure method, and B1 is smaller than B2. The energy change that occurs upon improving the treatment of the correlation energy will be represented by

$$\Delta E(\mathrm{L2}\,|\,\mathrm{L1}\,/\mathrm{B}) \equiv E(\mathrm{L2}\,/\mathrm{B}) - E(\mathrm{L1}\,/\mathrm{B}) \qquad (3)$$

where L1 is a lower many-electron level of theory than L2, and B is a common basis set. Finally, the change in energy increment due to increasing the many-electron level of the treatment of the correlation energy with one one-electron basis set as compared to the increment obtained with a smaller basis set will be represented as

$$\Delta E(\mathrm{L2}\,|\,\mathrm{L1}\,/\mathrm{B2}\,|\,\mathrm{B1}) \equiv E(\mathrm{L2}\,/\mathrm{B2}) - E(\mathrm{L1}\,/\mathrm{B2}) - [E(\mathrm{L2}\,/\mathrm{B1}) - E(\mathrm{L1}\,/\mathrm{B1})]. \qquad (4)$$

Additionally, some standard abbreviations for electronic structure methods will be used throughout this manual, namely

HF	Hartree-Fock [Ro51, Po54]
MP2	Møller-Plesset (MP) perturbation theory, second order [Mø34, Po77]
MP4D	MP perturbation theory, fourth order, with double excitations [Kr78, Kr80]
MP4SDQ	MP perturbation theory, fourth order, with single, double, and quadruple excitations [Kr78, Kr80]
MP4	full fourth-order MP perturbation theory, i.e., MP4SDQ plus triple excitations (Sometimes, in the older literature, MP4 is called MP4SDTQ.) [Kr78, Kr80]
CCSD	coupled-cluster theory with single and double excitations [Ci69, Pu82, Sc88, Sc89]
CCSD(T)	CCSD plus a quasiperturbative treatment of fourth-order and fifth order connected triple excitations [Ra89]
QCISD	Quadratic configuration interaction with single and double excitations [Po87]
QCISD(T)	QCISD plus a quasiperturbative treatment of fourth-order and fifth order connected triple excitations [Po87]

All of the equations defining the methods will be written in terms of energies. However for all methods, the gradient and Hessian may be determined by applying the analogous equation to each of their components. It should be noted that, in this version of MULTILEVEL, the spin-orbit and core-correlation contributions to the energy, if present in a method, are treated as constants, and as such contribute zero to the gradient and Hessian components.

The following abbreviations for standard basis sets are used throughout this manual:

pDZ	cc-pVDZ [Du89, Wo93]
mpDZ	maug-cc-pVDZ [Pa09, Fa99b, Ke92, Wo94]
pTZ	cc-pVTZ [Du89, Wo93]
Dd	6-31G(d) [He86]
D+d	6-31+G(d) [He86]
D2dfp	6-31G(2df,p) [He86]
Tdp	6-311G(d,p) [He86]
T+dp	6-311+G(d,p) [He86]
T2dfp	6-311G(2df,p) [He86]
T+3df2p	6-311+G(3df,2p) [He86]
MG3	Modified G3large [Fa99d]
MG3S	Semidiffuse MG3 [Ly02]

Note that maug-cc-pVDZ is sometimes called aug″-cc-pVDZ in older papers. The spin-orbit and core-correlation energies (E_{SO} and E_{CC}) referred to in the subsequent equations are simple estimates (requiring negligible computational effort). These methods are described elsewhere. [Tr99, Fa99, Fa99a] Since the coefficients in the equations containing E_{SO} and E_{CC} were parameterized with these values in the equation, the user must supply the correct spin-orbit and core-correlation corrections to obtain an accurate energy for those methods that require E_{SO} and E_{CC}. (See ESO and ECC in the MULTIGEN input section.)

4.B.1. Linear Mixing of Hartree-Fock and Molecular Orbital Methods (HF||MO)

The HF||MO method involves a linear combination of the HF and semiempirical molecular orbital theory (MO) [St90] total electronic energies. The overall HF||MO energy [Ch99] is written as

$$E_{\mathrm{HF||MO}} = xE_{\mathrm{HF}} + (1-x)E_{\mathrm{MO}} \tag{5}$$

where x is the mixing parameter, E_{HF} is the total electronic energy from Hartree-Fock theory, and E_{MO} is the total electronic energy from semiempirical MO theory.

4.B.2. Scaling All Correlation (SAC) Method

The scaling all correlation (SAC) method [Fa99, Go86] may be written [Tr99, Fa99b]

$$E\,(\mathrm{SAC\text{-}L/B}) = E\,(\mathrm{HF/B}) + c_1 \Delta E\,(\mathrm{L\,|HF/B}) + E_{\mathrm{SO}} + E_{\mathrm{CC}} \tag{6}$$

where E_{SO} and E_{CC} are the spin-orbit and core-correlation contributions to the energy, respectively, and c_1 is a constant. The default for the VERSION variable in the SAC keyword of the *LC section is *v2m*. Table 1 gives the value of c_1 for all of the different SAC versions. See the VERSION and COEFF options of the SAC keyword in the *LC section for changing the default value.

Table 1. Available versions of c_1 for the SAC method [Tr99, Fa99, Fa99b,Ly02]

Methods	VERSION							
	v1s	v1sc	v2m	v2s	v2sc	v3m	v3s	HCO-s
SAC-MP2/pDZ	1.2877	1.2768	1.2318	1.2373	1.2181	1.2638	1.2672	1.2660
SAC-MP2/pTZ	1.0578	1.0482	1.0090	1.0138	0.9970	1.0219	1.0255	1.1753
SAC-MP2/6-31G(d)						1.2979	1.3019	1.3577
SAC-MP2/6-31G†						1.3218	1.3258	1.3714
SAC-MP2/6-31G(d,p)						1.1671	1.1707	1.1753
SAC-MP2/6-31+G(d,p)						1.1761	1.1796	1.1888

Methods	VERSION							
	v1s	v1sc	v2m	v2s	v2sc	v3m	v3s	HCO-s
SAC-MP2/6-31+G(2df,p)						1.0530	1.0563	1.0795
SAC-MP2/MG3S						1.0268	1.0300	1.0517
SAC-MP2/6-31G(d,2p)						1.1478	1.1512	1.1526
SAC-MP4SDQ/pDZ	1.4308	1.4189	1.4370	1.4431	1.4209	1.4281	1.4320	1.3980
SAC-MP4SDQ/pTZ	1.1854	1.1747	1.1880	1.1933	1.1737	1.1569	1.1808	1.1569
SAC-MP4/pDZ	1.3355	1.3243	1.3306	1.3362	1.3156	1.3394	1.3430	1.3245
SAC-MP4/pTZ	1.0853	1.0756	1.0739	1.0788	1.0610	1.0766	1.0803	1.0711
SAC-CCSD/pDZ	1.4497	1.4375	1.4665	1.4727	1.4501	1.4573	1.4613	1.4308
SAC-CCSD/pTZ	1.2022	1.1915	1.2125	1.2178	1.1979	1.1997	1.2036	1.1801
SAC-CCSD(T)/pDZ	1.3656	1.3542	1.3716	1.3774	1.3562	1.3753	1.3790	1.3586
SAC-CCSD(T)/pTZ	1.1201	1.1100	1.1181	1.1232	1.1047	1.1156	1.1193	1.1064

Notice that *all* other methods (level of theory and basis set combination) not specified in Table 1 are given a default c_1 value of 1.2500.

4.B.3. Multi-Coefficient SAC (MCSAC)

The overall methodology for SAC is to scale all of the correlation energy that comes from a given level of correlation energy treatment as calculated with a single basis. However the different components of the correlation energy may need different scaling factors. [Tr99, Fa99b] For example, using CCSD theory and the pDZ basis set, we can write

$$E(\text{MCSAC-CCSD}/\text{pDZ}) = E(\text{HF}/\text{pDZ}) + c_1\Delta E(\text{MP2}\,|\,\text{HF}/\text{pDZ}) + c_2\Delta E(\text{CCSD}\,|\,\text{MP2}/\text{pDZ}) + E_{\text{SO}} + E_{\text{CC}} \quad (7)$$

where c_1 and c_2 are constants. In general, we define the multi-coefficient SAC method (MCSAC) for electron correlation level L_n and basis B as

$$E(\text{MCSAC-L}_n/\text{B}) = E(\text{HF}/\text{B}) + \sum_{m=1}^{n} c_m \Delta E(\text{L}_m|\text{L}_{m-1}/\text{B}) + E_{\text{SO}} + E_{\text{CC}} \qquad (8)$$

where the lowest level, L_0, is HF theory, and L_1, L_2, … are the correlated members of the sequence leading up to level L_n. We will define two sequences: the MP sequence and the CC sequence. The MP sequence consists of L_0 = HF, L_1 = MP2, L_2 = MP4SDQ, and L_3 = MP4; and the CC sequence consists of L_0 = HF, L_1 = MP2, L_2 = CCSD, and L_3 = CCSD(T). The default for the VERSION variable in the MCSAC keyword of the *LC section is *v2m*. Table 2 gives the value of c_m for all of the different MCSAC versions. See the VERSION and COEFF options of the MCSAC keyword in the *LC section for changing the default values.

Table 2. Available versions of c_m for the MCSAC method [Tr99, Fa99b]

Methods	VERSION	c_1	c_2	c_3
MCSAC-MP4SDQ/pDZ	v2m	1.3740	0.9849	
	v2s	1.3727	0.9387	
	v2sc	1.3573	0.9644	
	v3m	1.3747	0.9511	
	v3s	1.3734	0.9093	
	HCO-s	1.3885	1.2652	
MCSAC-MP4SDQ/pTZ	v2m	1.1326	0.8114	
	v2s	1.1299	0.7626	
	v2sc	1.1169	0.7875	
	v3m	1.1366	0.8621	
	v3s	1.1334	0.8108	
	HCO-s	1.1228	0.8285	

Methods	VERSION	c_1	c_2	c_3
MCSAC-MP4/pTZ	v2m	1.0899	1.1081	0.9541
	v2s	1.0880	1.0538	0.9363
	v2sc	1.0775	1.0621	0.8829
	v3m	1.1023	1.1119	0.8192
	v3s	1.1000	1.0539	0.7969
	HCO-s	1.1042	1.0377	0.5628
MCSAC-CCSD/pDZ	v1sc	1.4174	1.2403	
	v2m	1.3866	0.9543	
	v2s	1.3852	0.9121	
	v2sc	1.3699	0.9358	
	v3m	1.3900	0.9366	
	v3s	1.3886	0.8993	
	HCO-s	1.4063	1.2066	
MCSAC-CCSD/pTZ	v1sc	1.1406	0.7433	
	v2m	1.1415	0.7796	
	v2s	1.1388	0.7353	
	v2sc	1.1257	0.7569	
	v3m	1.1472	0.8347	
	v3s	1.1440	0.7897	
	HCO-s	1.1323	0.7868	

Table 2. (cont.)

Methods	VERSION	c_1	c_2	c_3
MCSAC-CCSD(T)/pDZ	v1sc	1.3055	1.7800	3.0180
	HCO-s	1.3157	1.8210	3.3330
MCSAC-CCSD(T)/pTZ	v1sc	1.0513	1.2183	2.1835
	v2m	1.0818	1.0123	1.3836
	v2s	1.0769	0.9761	1.4314
	v2sc	1.0712	0.9688	1.2601
	v3m	1.1119	0.9676	0.8427
	v3s	1.1062	0.9321	0.9026
	HCO-s	1.0715	1.1969	1.8880

Notice *all* other methods (level of theory and basis set combination) not specified in Table 2 are given a set of default c_m values of 1.0000.

4.B.4. Infinite Basis (IB) Method

The IB method [Tr98, Fa99e] may be written [Fa99b] as

$$E(\text{IB-L}/\text{B2}\,|\,\text{B1}) = E\,(\text{HF}/\text{B1}) + c_1\Delta E(\text{HF}/\text{B2}\,|\,\text{B1})$$

$$+ \Delta E(\text{L}\,|\,\text{HF}/\text{B1}) + c_2\Delta E(\text{L}\,|\,\text{HF}/\text{B2}\,|\,\text{B1}) + E_{\text{SO}} + E_{\text{CC}} \qquad (5)$$

where, in the original notation, with B1 = pDZ and B2 = pTZ, c_1 is given by $3^{\alpha}/(3^{\alpha} - 2^{\alpha})$ and c_2 is given by $3^{\beta}/(3^{\beta} - 2^{\beta})$, where α and β are parameters. The recommended values (MULTILEVEL default values for the ALPHA and BETA options of the IB keyword in the *LC section) of α and β are given in the Table 3. See the ALPHA and BETA options of the IB keyword in the *LC section for changing the default values.

Table 3. Available versions of α and β for the IB method [Fa99e]

Methods	α	β
IB-MP2/pDZ|pTZ	3.39	1.91
IB-MP4/pDZ|pTZ	3.39	2.08
IB-CCSD/pDZ|pTZ	3.39	1.94
IB-CCSD(T)/pDZ|pTZ	3.39	2.02

Notice *all* other methods (level of theory and basis set combination) not specified in Table 3 are given default values of $\alpha = 3.39$ and $\beta = 2.00$.

4.B.5 Multi-Coefficient Correlation Methods (MCCM)

The MCCM methods involve linear combinations of different levels of theory with the Dunning correlation consistent polarized double and triple zeta (cc-pVDZ, abbreviated pDZ, and cc-pVTZ, abbreviated pTZ) basis sets. [Fa99b] We recognize three different kinds of MCCM methods. The Colorado (CO) kind uses a polarized double zeta and triple zeta basis set for all the correlation calculations that are included as well as for the HF part. The Utah (UT) kind uses pDZ for all HF and correlation calculations but uses pTZ only for HF and MP2. The final kind is New Mexico (NM) which is similar to UT, but adds an additional HF/mpDZ calculation. (The abbreviation mpDZ is shorthand for the maug-cc-pVDZ basis set, which differs from aug-cc-pVDZ in that diffuse functions have been omitted on hydrogen and the diffuse subshell corresponding to the highest angular momentum has been omitted for the heavy atoms.)

4.B.5.a. Colorado Variant

The Colorado (CO) variant of MCCM is formally written as

$$E(\text{MCCM-CO-L}_n) = c_1 E(\text{HF / B1}) + c_2 \Delta E(\text{HF / B2 | B1})$$

$$+ \sum_{m=1}^{n} c_{2m+2}\Delta E(\mathrm{L}_m \mid \mathrm{L}_{m\text{-}1} / \mathrm{B2} \mid \mathrm{B1})$$

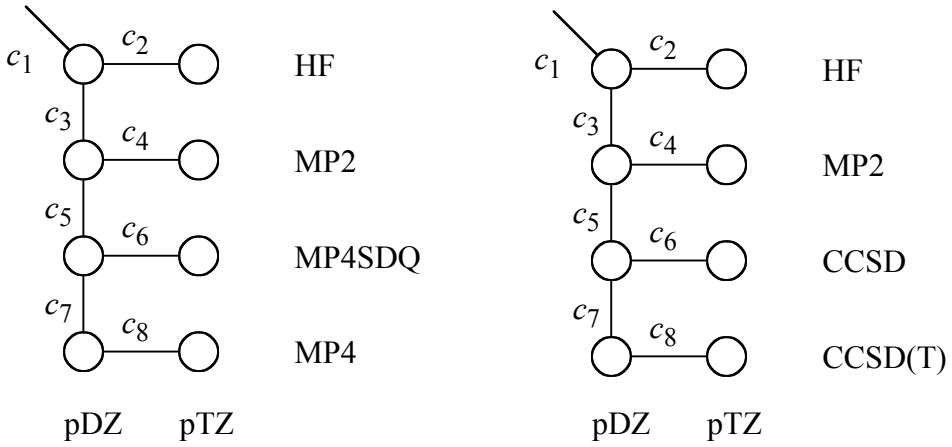
$$+ \sum_{m=1}^{n} c_{2m+1}\Delta E(\mathrm{L}_m \mid \mathrm{L}_{m\text{-}1} / \mathrm{B1}) + E_{\mathrm{SO}} + E_{\mathrm{CC}} \qquad (6)$$

where B1 and B2 are pDZ and pTZ or D+d and MG3, respectively. For clarity let's use MCCM-CO-CCSD as an example,

$$E(\text{MCCM-CO-CCSD}) = c_1 E\,(\mathrm{HF}/\mathrm{pDZ}) + c_2\Delta E\,(\mathrm{HF}/\mathrm{pTZ}\,|\mathrm{pDZ})$$
$$c_3\Delta E\,(\mathrm{MP2}\,|\mathrm{HF}/\mathrm{pDZ}) + c_4\Delta E\,(\mathrm{MP2}\,|\mathrm{HF}/\mathrm{pTZ}\,|\mathrm{pDZ})$$
$$c_5\Delta E\,(\mathrm{CCSD}\,|\mathrm{MP2}/\mathrm{pDZ}) + c_6\Delta E\,(\mathrm{CCSD}\,|\mathrm{MP2}/\mathrm{pTZ}\,|\mathrm{pDZ})$$
$$+ E_{\mathrm{SO}} + E_{\mathrm{CC}} \qquad (7)$$

where c_1 through c_6 are constants. The coefficients, given in Table 5, and the corresponding energy differences can be visualized with Figure 1 given below. The first circle represents E(HF/pDZ); the vertical lines represent level improvements (e.g., ΔE(MP2|HF/ pDZ)), and the horizontal lines represent basis set improvements (e.g., ΔE(MP2|HF/pTZ|pDZ)). Deleting the last row of the CC tree yields the tree corresponding to eq. (7); comparing the CC tree to eq. (7) should make the notation obvious.

Figure 1. Coefficient trees for MCCM-CO-MP4 and MCCM-CO-CCSD(T). Coefficient trees for other symmetric MCCM methods (Colorado variants) are obtained by deleting rows successively from the bottom. (left) MP tree. (right) CC tree. [Tr99, Fa99b]



pDZ ≡ cc-pVDZ
pTZ ≡ cc-pVTZ

Table 5. Available versions of c_m for the Colorado variant of MCCM [Tr99, Fa99b, Fa00]

Methods	VERSION	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
MCCM-CO-MP2	v1sc	0.9971	1.6560	0.7718	2.6398				
	v2m	0.9918	1.0276	0.7833	2.6875				
	v2s	0.9887	1.0828	0.7768	2.7893				
	v2sc	0.9888	1.1177	0.7671	2.7028				
	v3m	1.0000	1.1361	0.7609	2.6099				
	v3s	1.0000	1.1722	0.7648	2.5938				
	HCO-s	1.0349	2.0168	0.7502	1.6960				
MCCM-CO-MP2; MG3; 6-31+G(d)	v2m	0.9724	1.2936	0.8577	2.0067				
MCCM-CO-MP4SDQ	v2m	0.9613	1.3628	1.0093	2.6287	0.5294	3.1443		
	v2s	0.9633	1.3834	0.9872	2.6535	0.5145	2.6409		
	v2sc	0.9615	1.4505	1.0035	2.5157	0.5870	2.8439		
	v3m	1.0000	1.4282	0.9551	1.9690	0.6646	1.1617		
	v3s	1.0000	1.4285	0.9382	2.0131	0.6384	0.7710		
	HCO-s	1.0000	1.8089	1.0015	1.5018	0.5966	1.3778		
MCCM-CO-MP4	v2m	0.9895	1.4888	0.8535	2.1953	1.1825	3.8465	1.6905	3.9165
	v2s	0.9964	1.5157	0.8123	2.1066	1.2808	3.6512	1.8043	5.0707
	v2sc	0.9886	1.5656	0.8568	2.0846	1.2148	3.5897	1.5567	3.9450
	v3m	1.0000	1.5666	0.8071	2.1216	1.3189	1.9612	2.3753	2.1027
	v3s	1.0000	1.5624	0.7855	2.1292	1.3351	1.6939	2.3521	2.7357
	HCO-s	1.0000	1.6997	0.9601	1.5852	0.8631	1.4685	0.7742	0.7237
MCCM-CO-CCSD	v1sc	0.9703	1.6636	1.1206	1.7329	0.7127	2.6136		

	v2m	0.9618	1.3810	1.0475	2.4491	0.5782	2.8382		
	v2s	0.9635	1.4008	1.0217	2.5017	0.5515	2.4227		
	v2sc	0.9629	1.4633	1.0392	2.3290	0.6254	2.5222		
	v3m	1.0000	1.4306	0.9709	1.9298	0.7020	0.8383		
	v3s	1.0000	1.4321	0.9555	1.9709	0.6752	0.5276		
	HCO-s	1.0000	1.6159	1.0392	1.4306	0.6187	1.2756		
MCCM-CO-CCSD(T)	v1sc	0.9887	1.5377	1.0048	1.5208	1.0106	1.5695	1.7202	0.9124
	v2m	0.9929	1.5121	0.9479	1.7902	0.9462	1.7687	1.9810	0.8222
	v2s	0.9981	1.5432	0.9097	1.7613	0.9684	1.3340	2.1823	1.2716
	v2sc	0.9914	1.5839	0.9477	1.7238	0.9633	1.5365	1.8210	0.7451
	v3m	1.0000	1.5652	0.8370	2.0959	0.9349	1.3857	2.2431	1.1679
	v3s	1.0000	1.5613	0.8168	2.1349	0.9169	1.0760	2.2619	1.4593
	HCO-s	1.0000	1.5895	1.0026	1.3615	0.9520	1.4874	1.5912	0.8156

Table 6. Available versions of c_m for the Utah and New Mexico variants of MCCM [Tr99, Fa99b]

Methods	VERSION	c_1	c_2	c_3	c_4	c_5	c_6	c_7
MCCM-UT-MP4SDQ	v2m	0.9934	1.2606	1.0363	1.6307	0.8541		
	v2s	0.9903	1.2975	1.0099	1.8153	0.7872		
	2sc	0.9905	1.3579	1.0280	1.6130	0.8807		
	v3m	1.0000	1.3772	0.9318	2.0071	0.7505		
	v3s	1.0000	1.3949	0.9231	2.0361	0.6956		
	HCO-s	1.0273	1.6167	0.9268	1.1904	0.4652		
MCCM-UT-MP4	v2m	1.0040	2.5726	0.9136	2.8707	1.1605	1.7108	
	v2s	1.0009	2.5466	0.8862	2.9916	1.0962	1.7254	
	v2sc	1.0002	2.6548	0.9165	2.8444	1.1591	1.5546	
	v3m	1.0000	1.5394	0.7735	2.3270	1.2431	2.5328	
	v3s	1.0000	1.5544	0.7645	2.3611	1.1834	2.5178	
	HCO-s	1.0000	1.6043	0.9108	1.7591	0.8669	1.0151	
MCCM-UT-CCSD	v1sc	0.9949	1.6872	1.1422	0.8323	1.0040		
	v2m	0.9969	1.2625	1.0610	1.4813	0.8312		
	v2s	0.9935	1.2997	1.0332	1.6755	0.7675		
	v2sc	0.9941	1.3580	1.0512	1.4689	0.8502		
	v3m	1.0000	1.3800	0.9505	1.9749	0.7531		
	v3s	1.0000	1.4008	0.9425	1.9995	0.7073		
	HCO-s	1.0325	1.8885	0.7867	1.6133	0.0687		

Methods	VERSION	c_1	c_2	c_3	c_4	c_5	c_6	c_7
MCCM-UT-CCSD(T)	v1sc	1.0002	1.4852	1.0026	1.1447	1.1869	2.1343	
	v2m	1.0143	1.4894	0.9402	1.2493	1.1061	2.3805	
	v2s	1.0112	1.5307	0.9101	1.4394	1.0473	2.4238	
	v2sc	1.0099	1.5644	0.9413	1.2580	1.1002	2.1652	
	v3m	1.0000	1.5118	0.8062	2.2306	0.9670	2.2989	
	v3s	1.0000	1.5360	0.7937	2.2635	0.9254	2.3616	
	HCO-s	1.0000	1.3069	0.9276	1.6755	0.9415	1.9394	
MCCM-NM-CCSD(T)	v1sc	1.0013	1.4127	1.0173	1.0516	1.2159	2.1967	0.3790
	v2m	1.0164	1.4933	0.9482	1.1781	1.0779	2.2702	0.3115
	v2s	1.0126	1.5334	0.9156	1.3905	1.0280	2.3481	0.2140
	v2sc	1.0114	1.5672	0.9471	1.2062	1.0798	2.0852	0.2262

4.B.5.b. Utah Variant

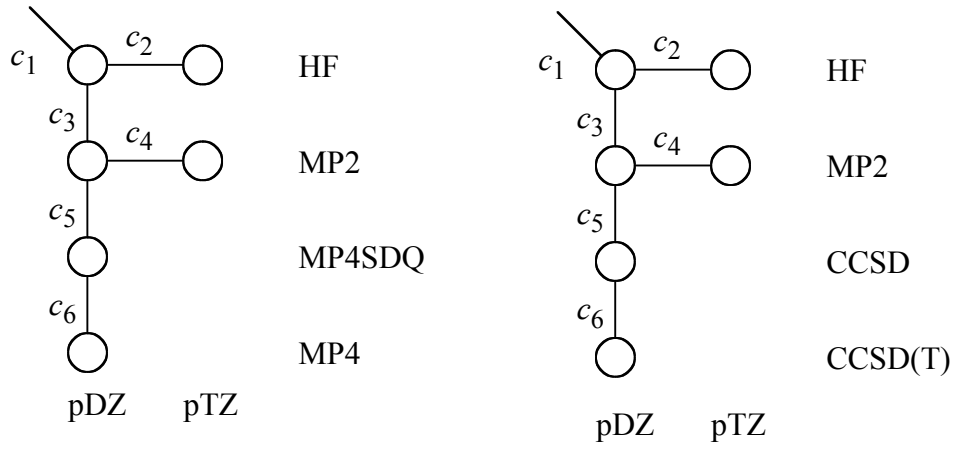
The Utah (UT) variant of MCCM truncates the pTZ part of the coefficient tree (shown in Figure 2) at the MP2 level. This is done for efficiency purposes. The UT version MCCM is formally written as

$$\begin{aligned} E(\text{MCCM-UT-L}_{\upsilon}) &= c_1 E(\text{HF} / \text{B1}) + c_2 \Delta E(\text{HF} / \text{B2} \mid \text{B1}) \\ &+ \sum_{m=1}^{n} c_{2m+2} \Delta E(\text{L}_m \mid \text{L}_{m-1} / \text{B2} \mid \text{B1}) \\ &+ \sum_{m=1}^{n} c_{2m+1} \Delta E(\text{L}_m \mid \text{L}_{m-1} / \text{B1}) \\ &+ \sum_{m=n+1}^{\upsilon} c_{m+n+2} \Delta E(\text{L}_m \mid \text{L}_{m-1} / \text{B1}) + E_{\text{SO}} + E_{\text{CC}} \end{aligned} \tag{8}$$

By definition $\upsilon \geq n + 1$. For the suggested values of c_m (given in Table 6), B1 is always pDZ, and B2 is always pTZ. For example,

$$\begin{aligned} E(\text{MCCM-UT-CCSD}) &= c_1 E\,(\text{HF} / \text{pDZ}) + c_2 \Delta E\,(\text{HF} / \text{pTZ} \mid \text{pDZ}) \\ &+ c_3 \Delta E\,(\text{MP2} \mid \text{HF} / \text{pDZ}) + c_4 \Delta E\,(\text{MP2} \mid \text{HF} / \text{pTZ} \mid \text{pDZ}) \\ &+ c_5 \Delta E\,(\text{CCSD} \mid \text{MP2} / \text{pDZ}) + E_{\text{SO}} + E_{\text{CC}} \end{aligned} \tag{9}$$

Figure 2. Coefficient tree for asymmetric MCCM-UT-MP4 and MCCM-UT-CCSD(T). Coefficient trees for other asymmetric MCCM methods (Utah variants) are obtained by deleting rows successively from the bottom. (left) MP tree. (right) CC tree. [Tr99, Fa99b]



pDZ ≡ cc-pVDZ
pTZ ≡ cc-pVTZ

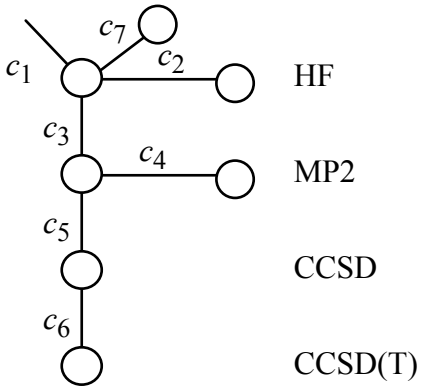
4.B.5.c. New Mexico Variant

The New Mexico (NM) variant of MCCM (shown in Figure 3) adds a correction for diffuse functions to the Utah variant and is formally written as

$$\mathrm{E(MCCM\text{-}NM\text{-}L}_{\upsilon}) = E(\mathrm{MCCM\text{-}L}_{\upsilon};\mathrm{L}_{n})$$
$$+ c_{n+\upsilon+3}\Delta E(\mathrm{HF} / \mathrm{mpDZ} \mid \mathrm{pDZ}) \qquad (10)$$

The default values of c_m for the NM variant are given in Table 6.

Figure 3. Coefficient tree for asymmetric MCCM-NM-CCSD(T). The only New Mexico variant available.



4.B.6. Gaussian-*x* (G*x*) and Multi-Coefficient Gaussian-*x* (MCG*x*) Methods

The G2 and G3 methods were designed to yield accurate thermochemistry, including vibrational contributions. [Po89, Cu91, Cu98] However the implementation of these two methods in MULTILEVEL deals only in the electronic energies of the molecules. These methods involve attempts to get the accuracy of a larger basis set QCISD(T) calculation without the full expense. In addition they have an empirical 'higher level correction'.

Multi-coefficient variations on G2 and G3, using a subset of energies have been designed to scale the correlation energy and extrapolate to an infinite basis set. [Tr99, Fa99c, Fa99d] As has been noted before, gradients and Hessians maybe obtained by applying the linear combinations given below to the gradient and Hessian components. The spin-orbit and core-correlation contributions to the multi-coefficient energies are required to be zero in the gradient and Hessian since they are treated as constants. Similarly the spin-orbit and higher level corrections in G2 and G3 also must be zero.

4.B.6.a. Gaussian-2 (G2)

In essence, the G2 method is an empirically corrected approximation to a QCISD(T) / 6-311+G(3df,2p) calculation obtained from several smaller calculations, in particular, from QCISD(T) / 6-311G(d,p), MP4 / 6-311+G(d,p), MP4 / 6-311G(2df,p), and MP2 / 6-311+G(3df,2p) energy calculations. For convenience we will abbreviate the basis sets used in these calculations as Tdp, T+dp, T2dfp, and T+3df2p, respectively. The electronic energy (including nuclear repulsion but not vibration or rotation) in Gaussian-2 (G2) is given by

$$E(\mathrm{G2}) = E(\mathrm{QCISD(T)/Tdp}) + \Delta E(+) + \Delta E(\mathrm{T2dfp}) + \Delta E(\mathrm{T+3df2p}) + \Delta E(\mathrm{HLC}) \qquad (11)$$

where

$$\Delta E(+) = E(\mathrm{MP4/T+dp}) - E(\mathrm{MP4/Tdp})$$

$$\Delta E(2df) = E(\mathrm{MP4/T2dfp}) - E(\mathrm{MP4/Tdp})$$

$$\Delta E(\mathrm{T+3df2p}) = E(\mathrm{MP2/T+3df2p}) - E(\mathrm{MP2/T2dfp}) - E(\mathrm{MP2/T+dp}) + E(\mathrm{MP2/Tdp})$$

and the higher level correction ΔE(HLC) is defined elsewhere. [Po89, Cu91]

When optimization are carried out using the G2 energy expression, the result should be labeled G2//G2 since G2 alone implies MP2(full)//6-431G(d) geometries [Ro00].

4.B.6.b. Multi-Coefficient Gaussian-2 (MCG2)

The multi-coefficient G2 (MCG2) method [Tr99, Fa99c] does not attempt to approximate a QCISD(T) / T+3df2p calculation, but rather attempts to scale the correlation energy and extrapolate to an infinite basis set to try to approximate a complete configuration interaction calculation. Both scaling of the correlation energy and extrapolation to an infinite basis set involve taking linear combinations of various differences of energy components, all found in the components of the G2 calculation itself. With these functional forms as motivation, we define MCG2 by

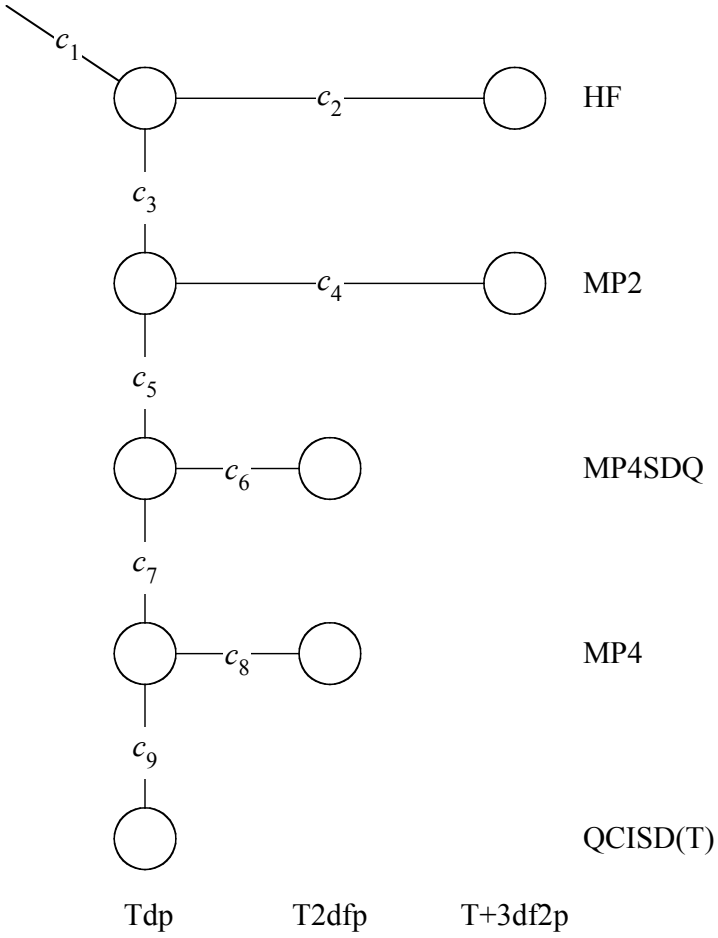
$$\begin{aligned} E(\text{MCG2}) = {} & c_1 E(\text{HF} / \text{Tdp}) + c_2 \Delta E(\text{HF} / \text{T+3df2p} \mid \text{Tdp}) \\ & + c_3 \Delta E(\text{MP2} \mid \text{HF} / (\text{Tdp}) + c_4 \Delta E(\text{MP2} \mid \text{HF} / \text{T+3df2p} \mid \text{Tdp}) \\ & + c_5 \Delta E(\text{MP4SDQ} \mid \text{MP2} / \text{Tdp}) + c_6 \Delta E(\text{MP4SDQ} \mid \text{MP2} / \text{T2dfp} \mid \text{Tdp}) \\ & + c_7 \Delta E(\text{MP4} \mid \text{MP4SDQ} / \text{Tdp}) + c_8 \Delta E(\text{MP4} \mid \text{MP4SDQ} / \text{T2dfp} \mid \text{Tdp}) \\ & + c_9 \Delta E(\text{QCISD(T)} \mid \text{MP4} / \text{Tdp}) + E_{\text{SO}} + E_{\text{CC}} \end{aligned} \tag{12}$$

where the coefficients $\{c_i\}$ are constants. In Ref. Fa99c we presented two parameterizations different parameterizations of MCG2, one called MCG2 that included spin-orbit and core-correlation, and one called minimal MCG2 (MMCG2) that did not included either of these effects. These parameterizations will be referred to as MCG2v1sc instead of MCG2 and MCG2v1m instead of MMCG2. The methods will be denoted in a similar fashion when using the version 2 coefficients [Tr99]. Figure 4 provides a diagram that helps one to visualize the terms in eq. (12) and Table 7 gives the values of the coefficients.

Table 7. Available versions of c_m for the MCG2 method [Tr99, Fa99c]

Method	VERSION	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
MCG2	v1m	0.9949	0.9462	1.1414	1.0396	1.0784	3.6766	0.6666	3.3428	1.1427
	v1sc	0.9911	1.0329	1.1498	1.0160	1.0242	3.4914	0.3824	3.1698	1.0826
	v2m	0.9926	0.6149	1.1703	0.9968	1.0233	4.6485	0.5703	4.3440	1.2560
	v2s	0.9932	0.6787	1.1695	0.9901	0.9980	4.4804	0.5096	4.2274	1.2598
	v2sc	0.9900	0.7229	1.1715	0.9504	1.0366	4.1810	0.4366	3.8767	1.2064
	v3m	1.0144	1.1576	1.0266	1.1630	1.3435	1.4462	1.6410	1.2324	1.1482
	v3s	1.0146	1.1567	1.0258	1.1589	1.3331	1.2197	1.5563	1.6337	1.1578
	HCO-s	0.9922	0.5822	1.1349	1.3589	0.8900	4.4271	0.4245	3.4042	1.1790

Figure 4. Coefficient tree for MCG2 [Fa99c]



Tdp ≡ 6-311G(d,p)
T2dfp ≡ 6-311G(2df,p)
T+3df2p ≡ 6-311+G(3df,2p)

4.B.6.c. Gaussian-3 (G3)

The G3 method [Cu98] involves three essential changes from G2, namely (i) substitution of a polarized 6-31G basis set for the polarized 6-311G basis set in some of the steps, (ii) a more balanced treatment of contracted *s* and *p* functions and valence polarization functions for first- and second-row atoms, and (iii) a final-step calculation involving core correlation and core polarization functions. Improvement (i) lowers the cost more than improvements (ii) and (iii) raise it, and the gain in accuracy from improvements (ii) and (iii) outweigh the potential loss in accuracy from modification (i).

The G3 method essentially provides an approximation to a QCISD(T)/G3large calculation from several smaller calculations, namely, from QCISD(T)/6-31G(d), MP4/6-31+G(d), MP4/6-31G(2df,p), and MP2(full)/G3large energy calculations, where the notation is standard, and the frozen-core (FC) approximation is implied except where we indicate "full." In order to condense the notation in eq. (13) below, we will abbreviate the first through third basis sets used in these calculations as Dd, D+d, and D2dfp, respectively. The electronic energy (including nuclear repulsion but not vibration or rotation) in Gaussian-3 (G3) is given by

$$E(\mathrm{G3}) = E[\mathrm{QCISD(T)/Dd}] + \Delta E(+) + \Delta E(\mathrm{D2df}) + \Delta E(\mathrm{G3large}) + \Delta E(\mathrm{SO}) + E(\mathrm{HLC}) \qquad (13)$$

where

$$\Delta E(+) = E(\mathrm{MP4/D+d}) - E(\mathrm{MP4/Dd})$$

$$\Delta E(2df) = E(\mathrm{MP4/D2dfp}) - E(\mathrm{MP4/Dd})$$

$$\Delta E(\mathrm{G3large}) = E(\mathrm{MP2/G3large}) - E(\mathrm{MP2/D2dfp}) - E(\mathrm{MP2/D+d}) + E(\mathrm{MP2/Dd})$$

and where G3large is a new basis set containing core polarization functions and ΔE(SO) and ΔE(HLC) are defined elsewhere. [Cu98]

When optimization are carried out using the G2 energy expression, the result should be labeled G2//G2 since G2 alone implies MP2(full)//6-431G(d) geometries [Ro00].

4.B.6.d. Gaussian-3 with Reduced-order Møller-Plesset Perturbation Theory Method (G3SX(MP3))

The G3SX with reduced-order Møller-Plesset perturbation theory method (G3SX(MP3)) [Cu01] is defined as :

$$E(\text{G3SX(MP3)}) = E[\text{HF / Dd}] + c_1\Delta E[\text{MP4|HF / Dd}]$$
$$+ c_2\Delta E[\text{QCISD(T)} \mid \text{MP4 / Dd}] + c_3\Delta E[\text{ HF / G3XLarge} \mid \text{Dd}]$$
$$+ c_4\,(\Delta E[\text{MP2(full)} \mid \text{HF / G3Large}] - \Delta E[\text{MP2} \mid \text{HF / Dd}])$$
$$+ c_5\Delta E[\text{MP3} \mid \text{MP2 /D2dfp| Dd}] \quad + E_{\text{SO}}$$

The coefficients [Cu01] are:

c_1	1.050200
c_2	1.185400
c_3	1.080300
c_4	1.202700
c_5	1.008100

Note that G*x* methods (include G3SX(MP3)) include spin-orbit energy only for atoms whereas MCCM methods that include explicit spin-orbit terms for any open-shell species for which E_{SO} is nonzero.

4.B.6.f. Multi-Coefficient Gaussian-3 (MCG3)

MCG3 [Tr99, Fa99d] may be obtained from most of the same energy calculations required for G3. For future reference we note that performing a QCISD(T) energy calculation with a given basis set also yields lower-level Hartree-Fock (HF), MP2, MP4SDQ, and MP4 results for that basis at no additional cost because they are all part of the overall calculation. Similarly MP4 calculations include MP4SDQ as a subset, and MP2 calculations include HF. These facts become important as we define the MCG3 method.

The MCG3 method is written as

$$E(\text{MCG3}) = c_1 E[\text{HF}/\text{Dd}] + c_2 \Delta E[\text{HF}/\text{MG3}\,|\,\text{Dd}]$$
$$+ c_3 \Delta E[\text{MP2}\,|\,\text{HF}/\text{Dd}] + c_4 \Delta E[\text{MP2}\,|\,\text{HF}/\text{MG3}\,|\,\text{Dd}]$$
$$+ c_5 \Delta E[\text{MP4SDQ}\,|\,\text{MP2}/\text{Dd}] + c_6 \Delta E[\text{MP4SDQ}\,|\,\text{MP2}/\text{D2dfp}|\,\text{Dd}]$$
$$+ c_7 \Delta E[\text{MP4}\,|\,\text{MP4SDQ}/\text{Dd}] + c_8 \Delta E[\text{QCISD(T)}\,|\,\text{MP4}/\text{Dd}] + E_{\text{SO}} + E_{\text{CC}} \qquad (14)$$

where the pipe "|" notation is defined above, c_m are constants (defaults given in Table 8), and the MG3 (modified G3) basis set denotes the G3large basis set without the core polarization functions. Note that we perform an MP2/MG3 calculation instead of the MP2(full)/G3large calculation used in the G3 method. This is because we obtain the core-correlation effects by a simple estimate (E_{CC}) instead of by the more expensive electronic structure calculation. Also note that the E_{SO} in eq. (14) is different from ΔE(SO) used in G3. [Fa99] In Ref. Fa99d we presented two different parameterizations of MCG3, one called MCG3 that included spin-orbit and core-correlation, and one called minimal MCG3 (MMCG3) that did not included either of these effects. These parameterizations will be referred to as MCG3v1sc instead of MCG3 and MCG3v1m instead of MMCG3. The methods will be denoted in a similar fashion when using the version 2 coefficients, namely MCG3v2sc and MCG3v2m [Tr99]. Figure 5 provides a diagram that helps one to visualize the terms in eq. (14). The values of the coefficients are given in Table 8.

MCG3 version 3s [Ly02] and version 3m are slightly different from previous versions of MCG3. In version 3, the full MP4 energy is no longer calculated and the MG3S basis set replaces the MG3 basis set. The MCG3 version 3s is also referred to as MCG3/3. The MCG3/3 energy can be expressed as:

$$\begin{aligned} E(\text{MCG3/3}) = {} & c_1 E[\text{HF}/\text{Dd}] + c_2 \Delta E[\text{HF}/\text{MG3S}\,|\,\text{Dd}] \\ & + c_3 \Delta E[\text{MP2}\,|\,\text{HF}/\text{Dd}] + c_4 \Delta E[\text{MP2}\,|\,\text{HF}/\text{MG3S}\,|\,\text{Dd}] \\ & + c_5 \Delta E[\text{MP4SDQ}\,|\,\text{MP2}/\text{Dd}] + c_6 \Delta E[\text{MP4SDQ}\,|\,\text{MP2}/\text{D2dfp}|\,\text{Dd}] \\ & + c_7 \Delta E[\text{QCISD(T)}\,|\,\text{MP4SDQ}/\text{Dd}] + E_{\text{SO}} \end{aligned} \tag{15}$$

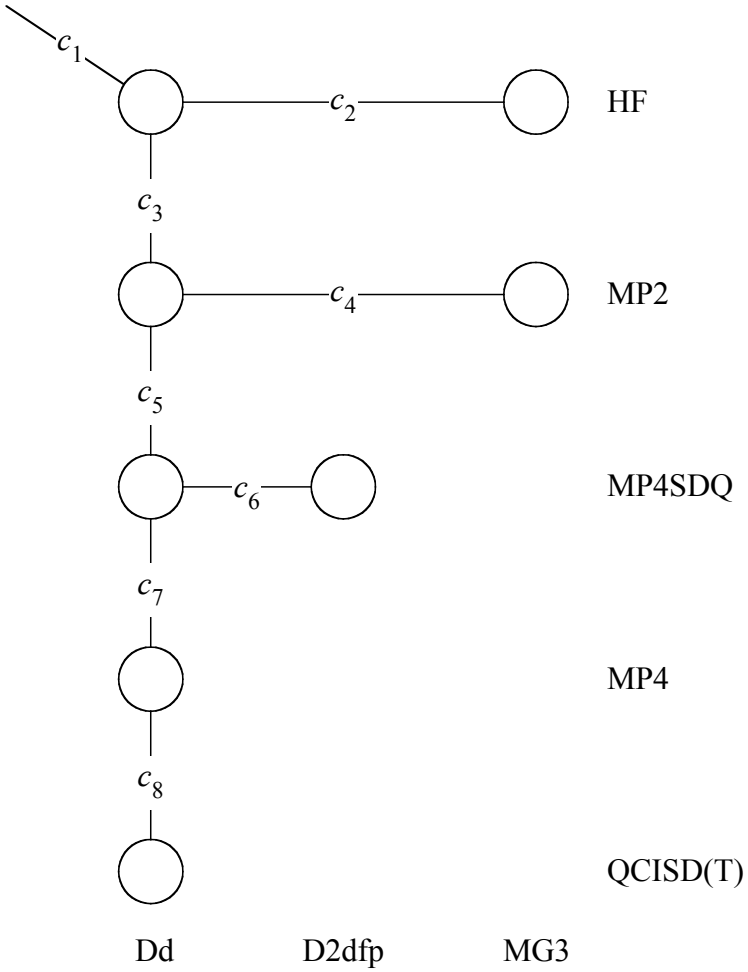
When a MCG3 energy calculation is performed, the MCQCISD energy can be obtained without any other additional electron structure calculation. Thus the MCQCISD energy is also reported in the MCG3 energy calculation output file.

Table 8. Available versions of c_m for the MCG3 method [Tr99, Fa99d]

Method	version	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
MCG3	v2m	1.0121	1.2047	1.0646	1.0975	1.1859	0.8139	1.4470	1.4142
	v2s	1.0096	1.2246	1.0637	1.1363	1.1506	0.5224	1.3219	1.3980
	v2sc	1.0080	1.2750	1.0661	1.0998	1.1330	0.8006	1.1689	1.3192
	cho	1.0054	0.9060	1.0527	1.2504	0.9272	0.4746	1.0165	1.5713

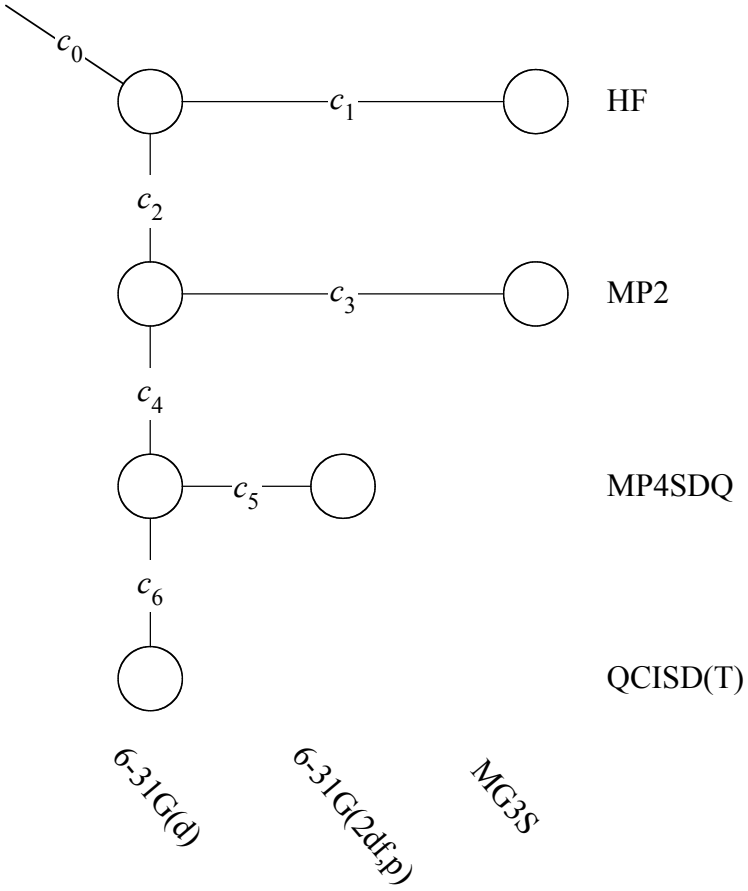
MCG3	version	c_0	c_1	c_2	c_3	c_4	c_5	c_6
	v3s	1.0067	1.1249	1.0585	1.2027	1.1369	0.5024	1.2666
	v3m	1.0073	1.1172	1.0588	1.1951	1.1212	0.8412	1.3058

Figure 5. Coefficient tree for MCG3v2 [Tr99].



Dd ≡ 6-31G(d)
D2dfp ≡ 6-31G(2df,p)
MG3 ≡ Modified G3large

Figure 6. Coefficient tree for MCG3/3 [Ly02].



4.B.7. Multi-Coefficient QCISD (MC-QCISD)

Multi-coefficient quadratic interaction with single and double excitations (MC-QCISD) is obtained from a subset of the energy calculations used in obtaining the MCG3 energy.

The MC-QCISD energy is defined by:

$$\begin{aligned} E(\text{MC-QCISD}) = {} & c_1 E(\text{HF/6-31G(d)}) + c_2 \Delta E(\text{MP2}|\text{HF/6-31G(d)}) \\ & + c_3 \Delta E(\text{MP2/MG3}|\text{6-31G(d)}) \\ & + c_4 \Delta E(\text{QCISD}|\text{MP2/6-31G(d)}) \end{aligned} \quad (16)$$

where the pipe "|" notation is defined above [eqs. (2), (3), and (4)], c_m are constants (defaults given in Table 9), and the MG3 (modified G3) basis set denotes the G3large basis set without the core polarization functions. We do not include core-correlation, spin-orbit, or scalar relativistic contributions explicitly, and thus they are implicit in the model. This approach is called a "minimal" model in previous work[Fa99b, Fa99c, Fa99d]. Figure 7 gives the coefficient tree for the MC-QCISD v2m method.

The v3m and v3s versions of MC-QCISD are slightly different from version 2. MC-QCISD version 3 (also referred to as MC-QCISD/3) replaces the MG3 basis with the MG3S basis. It also scales the energy difference between basis sets at the HF level and no longer scales the HF/6-31G(d) energy. The MC-QCISD/3 energy is defined by:

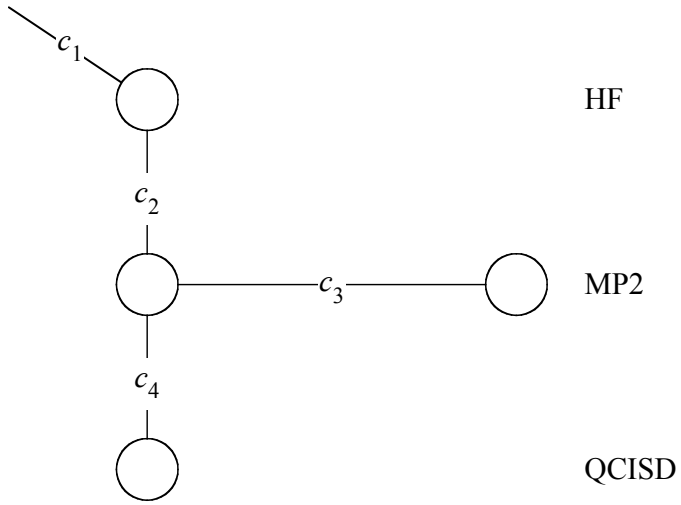
$$\begin{aligned} E(\text{MC-QCISD}) = {} & E(\text{HF/6-31G(d)}) + c_1 \Delta E(\text{HF/MG3S}|\text{6-31G(d)}) \\ & + c_2 \Delta E(\text{MP2}|\text{HF/6-31G(d)}) \\ & + c_3 \Delta E(\text{MP2}|\text{HF/MG3S}|\text{6-31G(d)}) \\ & + c_4 \Delta E(\text{QCISD}|\text{MP2/6-31G(d)}) \end{aligned} \quad (17)$$

Figure 8 illustrates the coefficient tree for the MC-QCISD v3 methods.

Table 9. Available versions of c_m for the MC-QCISD methods [Fa00, Ly02]

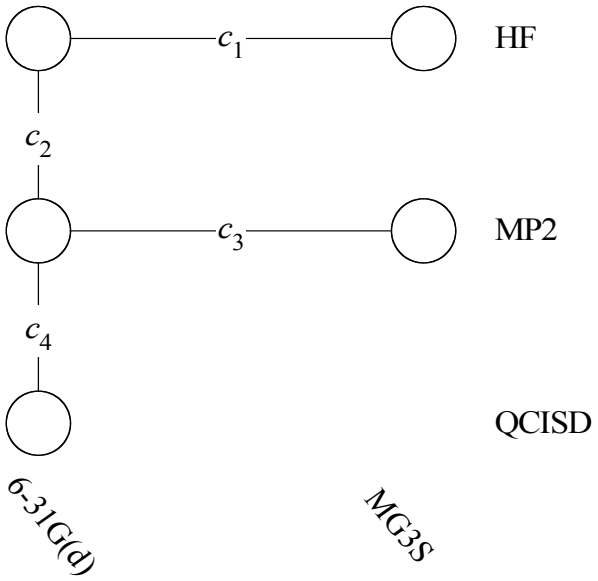
Method	c_1	c_2	c_3	c_4
MC-QCISD - v2m	1.0038	1.0940	1.2047	1.0441
MC-QCISD - cho	1.0173	0.9304	1.3959	0.4421
MC-QCISD - v3s	1.0452	1.1305	1.2302	1.1673
MC-QCISD - v3m	1.0325	1.1357	1.2226	1.2208

Figure 7. MC-QCISD coefficient tree [Fa00]



Dd ≡ 6-31G(d)
MG3 ≡ modified G3large

Figure 8. MC-QCISD/3 coefficient tree [Ly02]



4.B.7. Doubly Hybrid Density Functional Theory (DHDFT)

Doubly hybrid density functional methods [Zh04] are obtained by mixing the SAC method with hybrid density functional methods. In particular, there are two methods in the current version of MULTILEVEL; they are the multi-coefficient three-parameter Becke88-Becke95 (MC3BB) method and the multi-coefficient three-parameter modified Perdew-Wang (MC3MPW) method.

The MC3BB method is defined in eq. (18):

$$E(\text{MC3BB}) = c_2\,[E(\text{HF/DIDZ}) + c_1 \Delta E(\text{ MP2|HF/DIDZ})] + (1-c_2)\,E(\text{BB}X\text{/MG3S}) \qquad (18)$$

where BBX is same as BB1K except that the percentage, X, of HF exchange was determined by parametrization.

The MC3MPW method is defined in eq. (19):

$$E(\text{MC3MPW}) = c_2\,[E(\text{HF/DIDZ}) + c_1 \Delta E(\text{ MP2|HF/DIDZ})] + (1-c_2)\,E(\text{MPW}X\text{/MG3S}) \qquad (19)$$

where MPWX is same as MPW1K except that the percentage, X, of HF exchange was determined by parametrization.

The parameters for MC3BB and MC3MPW are listed in Table 10.

Table 10. Parameters for MC3BB and MC3MPW [Zh04]

Methods	c_1	c_2	X
MC3BB	1.332	0.205	39
MC3MPW	1.339	0.266	38

Chapter Five
5

5. Optimization Procedures

This chapter describes the optimization procedures available in this version of the MULTILEVEL code.

5.A. Optimization Algorithms

Currently there are two kinds of geometry optimization algorithms available in MULTILEVEL: (1) Newton-Raphson algorithm and (2) the eigenvector following (EF) algorithm.
The difference among the Newton-Raphson algorithms are in the treatment of the Hessian in those steps where it is not recalculated with an HHOOK call. Each of the three algorithms takes a Newton-Raphson (NR) step with Brent line minimization at every iteration of the optimization. The NR step is calculated by solving the following linear equation for $\boldsymbol{x}$:

$$\boldsymbol{H}\,\boldsymbol{x} = -\boldsymbol{g} \qquad (20)$$

where $\boldsymbol{H}$ is the Hessian matrix, $\boldsymbol{x}$ is a vector consisting of the Cartesian components of the step, and $\boldsymbol{g}$ is the gradient vector. The Hessian and gradient are both in unscaled Cartesian coordinates.

Brent line minimization [Pr92] then scales the geometry step $\boldsymbol{x}$ to a magnitude which minimizes the energy by the greatest amount. This line minimization avoids taking steps that are too large or too small.

The first Newton-Raphson algorithm (to be called just plain NR from now on) does not alter the Hessian in steps where the Hessian is not recalculated with an HHOOK. Equation (20) is simply solved using the most recently calculated Hessian. The remaining two

algorithms are quasi-Newton methods; in the steps between Hessian recalculations, eq. (20) is not solved using a true Hessian from a previous iteration, but rather using an approximate inverse Hessian created from a variable metric update. The second algorithm uses Broyden-Fletcher-Goldfarb-Shanno (BFGS) Hessian updates, and the third uses Davidon-Fletcher-Powell (DFP) Hessian updates [Pr92a]. Both of these updating algorithms take advantage of information contained in the change in the gradient from one iteration to the next in order to build up corrections to the inverse Hessian matrix, which should in principal tend to converge to the true inverse Hessian, at least for the important components. In fact the two types of inverse Hessian updates differ from one another by only one term.

One might notice that both updating schemes operate on the inverse Hessian. Due to this, in the implementation of the BFGS and DFP algorithms the step size is obtained with another equation instead of eq. (20):

$$\boldsymbol{x} = -\boldsymbol{H}^{-1}\boldsymbol{g} \tag{21}$$

This manner of solving for the Newton-Raphson step is somewhat more memory intensive than just solving eq. (20) in that it involves explicitly finding the inverse of the Hessian, the NR algorithm does not actually compute $\boldsymbol{H}^{-1}$ but rather solves eq. (20). Since the BFGS and DFP algorithms require the inverse Hessian, utilization of eq. (21) is most reasonable for them.

EF is an optimization routine based on Jack Simons P-RFO algorithm as implemented by Jon Baker [Ba85]. Step scaling to keep the step size within the trust radius is taken from Culot et al. [Cu92]. The trust radius is automatically updated dynamically by the method of Fletcher[Fl87]. The EF step is calculated by the following linear equation for $\boldsymbol{x}$:

$$\boldsymbol{x} = (s\boldsymbol{I}-\boldsymbol{H})^{-1}\boldsymbol{g} \tag{22}$$

where s is a shift factor which ensures that the step length is within or on a hypersphere, and $\boldsymbol{I}$ is the unit vector. If the Hessian has one and only one negative eigenvalue, the shift factor is set equal to zero. If this step is longer than the trust radius, a P-RFO step is attempted. If this is also too long, then the best step on the hypersphere is made via the

QA formula. Both P-RFO and QA steps are obtained with eq. (22), but these methods use different formulas for s.

Using the step calculated, a new geometry is obtained, at which a new energy and gradient are evaluated. If it is a TS search, two criteria are used in determining whether the step is accepted. The ratio between the actual and predicted energy change should ideally be 1. If it deviates substantially from this value, the second order Taylor expansion is no longer accurate. If the ratio is outside the interval defined by the RMIN and RMAX limits, the step is rejected, the trust radius reduced by a factor of two, and a new step is determined. The second criterion is that the eigenvector along which the energy is being maximized should not change substantially between iterations. The minimum overlap of the TS eigenvector with that of the previous iteration should be larger than OMIN; otherwise the step is rejected.

In the EF routine, there are three Hessian update options which are specified by the keyword IUPD. The BFGS and DFP updating schemes are included in the EF routine. The third option is to reuse the Hessian without updating, i.e., to freeze the Hessian. The BFGS update is generally regarded as the best update to use for optimizing to a minimum energy structure, but it tends to preserve positive definiteness, i.e., if the Hessian before the update is positive definite (all the eigenvalues are positive), then the updated Hessian will also have this property. For this reason, BFGS is not recommended for a transition state search. The DFP update has no particular bias towards positive definiteness. Thus the DFP updating scheme is recommended for transition state optimization.

Historical note: As discussed elsewhere [Po71, Yp95], a more appropriate name for the Newton-Raphson method would be "Newton-Raphson-Simpson method."

5.B. Hessians Obtained with OPTHHK

As the above equations make clear, each optimization algorithm calculates the step size based upon the Hessian or some update of the Hessian inverse. For smaller molecules,

using some of the less expensive methods within MULTILEVEL, the use of the true Hessian is reasonable. Yet many of the methods contained in this program are not inexpensive, and, due to this, calculating a Hessian every few steps in an optimization may not be feasible. Therefore, OPTHHK may employ 3 different types of Hessian strategies depending on the user's specifications. Option 1 is to use the high level or true Hessian. Option 2 is that one may use a Hessian calculated at a lower level of electronic structure theory. And the third option is to use a unit matrix scaled to the approximate magnitude of the components of the true Hessian. While the last two types of Hessians do not contain information on the exact second derivatives of the potential energy surface, it has been shown that any type of Hessian results in faster geometry convergence than simply following the gradient.

In particular, we have found that optimizations using lower level Hessians recalculated every several steps converge in as few if not fewer iterations than optimizations using a 'true' Hessian. And one must remember that not only does the optimization with the lower level Hessian converge in a comparable number of iterations, but it is also costs considerably less than an optimization with the higher level Hessian.

5.C. Optimization with GAUSSIAN09/03's Optimizers

If one uses MULTILEVEL in conjunction with GAUSSIAN09/03 (where GAUSSIAN09/03 is shorthand for GAUSSIAN09 or GAUSSIAN03) one can use the algorithm keyword *gauext* to specify that geometry optimization is to be carried out by using the optimizers in GAUSSIAN09/03. The overall control for this procedure is:

MULTILEVEL ↔ GAUSSIAN09/03 ↔ *Gau_External* ↔ MULTILEVEL

If one uses *gauext* as the optimization algorithm keyword, the primary MULTILEVEL calculation will call a GAUSSIAN09/03 optimization with the *external* keyword. This GAUSSIAN09/03 calculation calls an external PERL scripts *Gau_Externa*l, which will provide the MULTILEVEL energy, gradient and Hessian needed for optimization. *Gau_External* will call a secondary MULTILEVEL calculation and pass the secondary MULTILEVEL results to GAUSSIAN09/03. When GAUSSIAN09/03 finishes the optimization,

it will return the optimized geometry to the primary MULTILEVEL calculation. Note that in the GAUSSIAN09/03 manual, it says *Gau_External* can pass energy, gradient, and Hessian, but our tests show that in GAUSSIAN09/03, Gau_External can only pass energy and gradient, and GAUSSIAN09/03 will use numerical differentiation to calculate the Hessian needed for optimization.

5.C.1 Note about Gau_External script

The default Gau_External script in MULTILVEL-V4.4 is developed only to work with GAUSSIAN09 and with the Revisions D01 and D02 of GAUSSIAN03. The script directory also contains the script Gau_External_old, which works with older revisions of GAUSSIAN03. In order to use the old version of Gau_External the user has to replace the default version of the script by the older version (e.g. using the unix command: *cp Gau_External_old Gau_External*). It should be noted that the default geometry optimization algorithm in GAUSSIAN09 is different than that which was used in earlier versions (The use of the microiterations scheme for geometry optimizations has been made by default in GAUSSIAN09 when the "External" keyword is used). The optimization keywords 'Opt=NoMicro' must be included in the route section of the Gaussian input file in order to use the previous optimization algorithm.

5.D. Optimization with Nudged Elastic Band Method

If one uses Nudged Elastic Band (NEB) method (Go06) for optimization, the reaction path is described by a discrete sequence of images, consisting of two fixed end points and *n* intermediate movable images, that form the chain or the elastic band.
Spring interactions are added between adjacent images so the total (also called the adjusted) force acting on each image is the sum of the spring force $\vec{F}_i^s$ and the force $\vec{F}_i^t$ from the potential energy surface (which will be called true force). The band is optimized, minimizing the total force acting on each image.
The adjusted force acting on an image, *i*, is given by:

$$\vec{F}_i = \vec{F}^s_{i,\|} + \vec{F}^t_{i,\perp}$$

which is the sum of the spring force along the tangent to the chain and the true force perpendicular to the chain. The perpendicular component of the true force is obtained by:

$$\vec{F}^t_{i,\perp} = \vec{F}^t_i - \vec{F}^t_i \hat{\tau}_i$$

The parallel component of the spring force as well as the tangent can be estimated by different procedures depending on the version of the NEB algorithm.
The versions available in the MULTILEVEL code are: the Bisection or B-NEB,[Mi94,Mi95] Improved Tangent or IT-NEB, [He00] and the Climbing Image or CI-NEB.[He00,He00a] The last algorithm (the CI-NEB) is a modification of the NEB code introduced for the purpose of using an NEB calculation to converge a saddle point. This method modifies the definition of the total force acting on the highest-energy image after a few iterations (in MULTILEVEL code, five iterations). After identifying this image as point i_{max}, the force on i_{max} is given by:

$$\vec{F}_{i_{\max}} = -\nabla V(\vec{R}_i) + 2\nabla V(\vec{R}_i)\,|_{\|} = -\nabla V(\vec{R}_i) + 2(\nabla(\vec{R}_i)\cdot\hat{\tau}_i)\hat{\tau}_i$$

The CI-NEB version of the algorithm was introduced for the purpose of using an NEB calculation to converge a saddle point.
After evaluating all the adjusted force vectors acting on each movable image, a global vector $\vec{F}$, which collects the M (where M is the number of movable images) $\vec{F}_i$, is obtained. The components of the resulting vector $\vec{F}$, of order $M\times 3N$ (where N is the number of atoms of the system), will be minimized using one of the available quasi-Newton methods.

5.E. Comments on Optimizing Geometries in Cartesian Coordinates

Cartesian coordinates are currently the only way to specify geometries in MULTILEVEL, and the geometry optimizations are performed in Cartesian coordinates.

One issue which must be addressed is that Cartesian coordinates have η more degrees of freedom than are needed to fully describe the system (η = 6 for non-linear systems and

$\eta = 5$ for linear systems). If all Cartesian coordinates are allowed to vary, the optimization becomes unstable because the changes in geometry correspond not only to movement of the atoms relative to one another but also to translations of the entire molecule across space and rotations of the whole molecule. Thus either 6 or 5 Cartesian coordinates are fixed during the optimization; the default constant coordinates are *x*, *y*, and *z* for the first atom, *y* and *z* for the second atom, and if the molecule is non-linear *z* for the third atom. These may be changed (see the CONSTANT keyword in the MULTIGEN section), yet the remainder of this discussion will be carried out using the default. If different coordinates are held constant, the treatment described below should change only superficially in terms of axes and coordinates.

It is not sufficient though to only hold the designated six coordinates constant. The molecule must be oriented in a certain way in order not to lose any generality in the optimization. The *y* and *z* coordinates of the second atom must be the same as the *y* and *z* coordinates of the first atom. And for non-linear molecules, the *z* coordinate of the third atom must be the same the *z* coordinate for the first and second atoms. However the user is not required to enter a geometry, which adheres to these requirements. The optimization routine automatically reorients the molecule to adhere to these requirements (or requirements applicable to the constant coordinates specified by the user). This is achieved by first translating the molecule so that the first atom is at the origin. Then the molecule is rotated about the *z*-axis so that the *y*-coordinate of the second atom is 0. Subsequently a similar rotation is made about the *y*-axis in order to set the *z*-coordinate of the second atom to 0. Finally (for non-linear molecules) the molecule is rotated about the *x*-axis in order to force the *z*-coordinate of the third atom to 0.

Since this reoriented geometry may be undesirable to the user, at the end of the optimization these rotations and translations are reversed to place the first three atoms back in their original plane. (See NOREORIENT in the MULTIGEN section to switch this off.) Note that this reorientation requires only that the first atom's coordinates all remain the same. The coordinates of the other two atoms most likely will have changed during the optimization. Yet these three atoms should still define the same plane as before the

optimization. In our (limited) experience, this reorientation at the end usually results in a negligible energy change of 10^{-12} hartrees. In certain cases however a change as large as 10^{-9} hartrees has been observed, so the user may want to pay attention to the energy before and after the reorientation.

The freezing of the coordinates has very little effect on the algorithms actually employed during the optimization as described in Section 4.A. The portions of the Hessian and the gradient that are specific to these frozen coordinates are ignored during the calculation of the geometry steps, thus allowing the step for each of the frozen coordinates to be zero.

Chapter Six

6

6. Input Description

The input file (ml.inp) is divided into six sections namely, the *MULTIGEN section, the *EXTOPT section, the *MULTIOPT section, the *IMO section, the *LC section, and the *TEST section. The *MULTIGEN section must be first in the input file, and each section *must* be preceded by the asterisk as shown above. The description for each of these sections is given below. There are three types of keywords: switches, variables, and lists. The syntax for each type of keyword is as follows:

Switch
……
Variable *Value*
……
List
.
.
.
End

List keywords must be terminated by END, and they may contain other keywords within their bodies (when this is the case, it shall be indicated in the description of the keyword). All keywords are case insensitive. The value for a variable should be on the same line as a variable keyword, though, and the contents of a list keyword should be on the lines between the list keyword and its terminating END but not on those two lines. Also, all list keywords specifically designated for a title or for electronic structure program options are constrained to a maximum content of 5 lines.

In the sections below, each keyword is in bold, and directly following the keyword is both its type and its default value.

6.A. MULTIGEN Section

The MULTIGEN section contains keywords that are needed for any calculation; in other words, they are not specific to an IMO or LC type calculation. The keywords are:

CHARGE VARIABLE *0*

The CHARGE keyword is used to specify the charge of the entire system.

ENERGY/NOENERGY SWITCH *ENERGY*

The ENERGY keyword is used to specify a single-point energy calculation of the system defined by the GEOM keyword.

ESO, ECC VARIABLE *0.0000*

The ESO and ECC keywords are used to input the values of the spin-orbit correction to the energy and the core-correlation correction to the energy. The suggested values may be found in References [Fa99] and [Fa99a]. The values should be in atomic units. Accurate values should be given when the following methods will be used: IB, EIB, SAC, MCSAC, MCCM, MCG2, and MCG3.

GEOM LIST *No Default.*

The GEOM keyword specifies the geometry of the entire system. It is required, and the following is an example of the input format, although there are no strict requirements on spacing or number formats so long as an atom and its 3 coordinates are on the same line.

```
GEOM
      O      0.4515E+00    -0.3543E+00    0.0000E+00
      H      0.4853E+00     0.6115E+00    0.0000E+00
      H     -0.4788E+00    -0.6161E+00    0.0000E+00
END
```

GEOMTYPE VARIABLE *cartesian*

The GEOMTYPE keyword is used to specify the format of the geometry. Currently Cartesian coordinates are the only valid type of geometry specification.

GEOMUNIT VARIABLE *ang*

The GEOMUNIT keyword is used to specify the units of the geometry.

ang: Angstroms
au: atomic units (bohrs)

GRADIENT/NOGRADIENT SWITCH *NOGRADIENT*

The GRADIENT keyword is used to specify a single-point gradient calculation of the system defined by the GEOM keyword. The energy is also calculated.

HESSIAN/NOHESSIAN SWITCH *NOHESSIAN*

The HESSIAN keyword is used to specify a single-point Hessian calculation of the system defined by the GEOM keyword. The energy and gradient are also calculated, and the program will also calculate the harmonic vibrational frequencies and the normal mode eigenvectors in the mass-scaled coordinates. The eigenvalues are printed (including the six zero eigenvalues corresponding to translations and rotations), and the eigenvectors are printed both in mass-scaled Cartesians and in unscaled Cartesians.

MULTIPLICITY VARIABLE *1*

The MULTIPLICITY keyword is used to specify the multiplicity of the system, i.e., *1* for singlet, *2* for doublet, etc.

NATOMS VARIABLE *No Default.*

The NATOMS keyword is used to specify the total number of atoms in the system. When requesting an IMO calculation, NATOMS should include the cap atom (number of atoms in the entire system plus the cap atom). This keyword is required for all calculations.

PRSUM/NOPRSUM SWITCH *NOPRSUM*

The PRSUM keyword is used to specify whether a summary file is printed.

TITLE LIST *No Default.*

The TITLE keyword allows the user to give up to a five-line description of the calculation.

6.B. EXTOPT Section

The EXTOPT section contains keywords that are needed for an optimization with an external electronic structure program. This section is useful for creating good starting point geometries. The initial guess geometry for the external optimization should be supplied by the GEOM keyword in the MULTIGEN section. The keywords in the EXTOPT section are:

BASIS VARIABLE *cc-pvdz*

Keyword that indicates the basis set to be used for the optimization. Note that for certain chemistry models the basis set is already predefined.

GENECP VARIABLE *No Default.*

This keyword indicates the use of effective core potentials. The argument must be the name of the file that contains the specifications of the basis set and pseudopotentials for all the atoms in *Gaussian* format. That file must be put into the basis subdirectory of MULTILEVEL. This keyword must be used in combination with the variable *gen* in the **BASIS** keyword.

METHOD VARIABLE *mp2*

This keyword specifies the electronic structure theory level at which to carry out the optimization.

OPTIONS LIST *No Default.*

This keyword is used to give the options the user desires for the external optimization. There may be a maximum of 5 lines of options, yet there must at the very least be one line specifying the optimization algorithm. For example to request a transition state Berny optimization in GAUSSIAN one must give the following OPTIONS:

Options
Opt=TS

End

To specify the memory requirements and the number of processors to be used when calling GAUSSIAN, simply add Link1 commands to this list. Link1 commands (% commands) must come first in the list of options as shown below:

Options
%nproc=2
%mem=800mb
scf=(tight,maxcycle=1000) Opt=(ts,noeigentest)
End

PROGRAM VARIABLE *g03*

The PROGRAM keyword specifies the electronic structure package to be used to optimize the geometry. Currently only GAUSSIAN09 (*g09*), GAUSSIAN03 (*g03*) and GAUSSIAN98 (*g98*) and GAUSSIAN94 (*g94*) are valid input values.

6.C. MULTIOPT Section

The MULTIOPT section contains keywords to specify a geometry optimization of the system via MULTILEVEL algorithms and electronic structure methods. If an external optimization has been carried out that optimized geometry will be used as a starting point for this optimization. Otherwise, the initial geometry is obtained from the GEOM keyword in the MULTIOPT section. If MULTIOPT is used, the geometry is subsequently redefined for all remaining calculations as the resulting multilevel optimized geometry. Six of the keywords in this section, inparticular DDMAX, DDMAXTS, IUPD, OMIN, RMAX and RMIN are used only with the EF algorithm. The valid options are:

ALGORITHM VARIABLE *nr*

This keyword specifies the optimization algorithm to be used. The algorithms currently available are Newton-Raphson with Brent line minimization (*nr*), NR with Broyden-Fletcher-Goldfarb-Shanno updates on the Hessian (*bfgs*), NR with Davidon-Fletcher-Powell updates to the Hessian (*dfp*), EF with three Hessian update scheme, Berny algorithm using redundant internal coordinates in GAUSSIAN09/03 (*gauext*), and the Nudged Elastic Band method (neb).

CONSTANT LIST *See below.*

The CONSTANT keyword indicates which coordinates will be frozen during the optimization. The default is that after reorientation the first atom will be frozen at the origin (i.e. its *x*, *y*, and *z* coordinates remain fixed). In addition, the second atom will be fixed on the *x*-axis and the *y* and *z* coordinates of the second atom will be held constant. If the molecule is non-linear, the third atom will be in the *xz*-plane and its *z*-coordinate will remain fixed. The user may change which coordinates will be fixed, but it is strongly recommended that these coordinates be chosen in such a way that one atom has 3 coordinates fixed, a second atom has 2 coordinates fixed, and (if the system is non-linear) a third atom has 1 coordinate fixed. Additionally the coordinate held constant for this third atom should be one of the two coordinates held constant for the second atom. As long as these guidelines are adhered to, the system will be neither over- nor under-

constrained during an optimization. An example of the use of the CONSTANT keyword is given below. The first integer on each line is the number of an atom, and the letters following specify which coordinates will be frozen for that atom.

```
CONSTANT
   3          x
   7          x  y  z
   2          x  z
END
```

DDMAX VARIABLE 0.5

DDMAXTS VARIABLE 0.3

The DDMAX and DDMAXTS variables are used to set the maximum of the trust radius (in angstroms); the DDMAX variable is used for minima (equilibrium structures), and the DDMAXTS variable is used for saddle points (transition statex).

GAUEXTOPTIONS LIST *No Default.*

This keyword is used to give the options the user desires for the external optimization. There may be a maximum of 5 lines of options, and there must be at least one line specifying the optimization algorithm. For example to request a transition state Berny optimization in GAUSSIAN one can give the following OPTIONS:

```
Gauextoptions
   Opt= (ts,noeigentest)
End
```

GCOMP VARIABLE *10e-3*

This keyword gives the convergence criteria for the optimization. Once the component of the gradient with the maximum magnitude falls below this value, the structure is considered optimized. The value is in atomic units.

HBAS VARIABLE *6-31G**

The HBAS keyword indicates the basis set to be used for the lower level Hessian calculations. Note: this keyword is only valid when the HESSIAN keyword value is *lowlev*.

HESSIAN VARIABLE *lowlev*

The HESSIAN keyword specifies the type of Hessian to be used in the optimization algorithm. In this version the three options are a scaled unit matrix (*unitmat*), a Hessian at a lower level of electronic structure theory (*lowlev*), and a Hessian at the same level as the optimization method chosen (*highlev*).

HMETH VARIABLE *hf*

This keyword gives the method for the lower level Hessian to be calculated. Note: this keyword is only valid when the HESSIAN keyword value is *lowlev*.

HPROG VARIABLE *g98*

This keyword indicates the program to be used to gather the low level Hessian.

HREC VARIABLE *10*

This keyword indicates the number of iterations between recalculation of the Hessian.

HSCALE VARIABLE *10e-5*

The HSCALE keyword gives the value by which the unit matrix used for a Hessian is scaled. Note: this keyword is only significant when the unit matrix is chosen for the HESSIAN keyword or INITHESS is set to *off*.

INITHESS VARIABLE *on*

This keyword tells whether a Hessian should be calculated before the first step of the optimization or a scaled unit matrix should be used initially as the Hessian.

IUPD VARIABLE 0

IUPD = n selects the Hessian updating scheme in Eigenvector Following optimizations.

IUPD = 0 No updating

IUPD = 1 Powell updating scheme

IUPD = 2 BFGS updating scheme

METHOD VARIABLE *sac*

This keyword specifies the MULTILEVEL theory level at which to carry out the optimization. The valid values for the variable are *imohc*, *hfmo*, *sac*, *mcsac*, *ib, mccmco*, *mcomp2*, *mccmut*, *mccmnm*, *g2*, *mcg2*, *g3*, *mcg3*, *mcqcisd*, or *test*. In order to specify the details of these theories such as theory levels and basis sets to be used, see the TEST section (for a TEST optimization) or the keywords in the LC and IMO sections corresponding to the method names above.

MOLTYPE VARIABLE *nonlin*

This keyword indicates the type of molecule to be optimized. Currently the four valid values for this variable are *lin, nonlin, lints,* or *nonlints,* for a linear reactant/product, a non-linear reactant/product, a linear saddle point, or a non-linear saddle point, respectively. Currently this keyword has two effects. The first is that if the molecule is a saddle point, Brent line minimization is turned off. The second is that the program will freeze 5 coordinates during the optimization for a linear species, as opposed to 6 for a non-linear species.

NITER VARIABLE *50*

This keyword gives the maximum number of iterations in the optimization.

OMIN VARIABLE *0.8*

During transition state optimizations, the EF algorithm calculates the dot product between the previously followed direction and the Hessian eigenvectors. The new step will be along the direction defined by the eigenvector for which this dot product is maximum, if this value is greater than OMIN.

REORIENT/NOREORIENT SWITCH *REORIENT*

As stated above, for an optimization the molecule's orientation is changed in such a way that one atom will remain at the origin, another will remain on an axis, and a third will remain in a plane. REORIENT returns these three atoms to their original plane once the optimization is done. If the user prefers to leave the molecule in the orientation it had during the optimization for any further energy, gradient, and Hessian calculations, this may be achieved by switching on NOREORIENT.

RETRY VARIABLE *on*

This keyword tells whether the optimization routine should switch to HREC=1 if the optimization fails with regards to STPTOL.

RMIN VARIABLE *0.0*
RMAX VARIABLE *4.0*

For an Eigenvector Following step to be accepted, the value of the ratio of the calculated energy change to the predicted energy change must be bracketed by the values of RMIN and RMAX. Default values are RMIN = 0 and RMAX = 4.

SCALE VARIABLE *1.0*

This keyword gives the maximum value (in bohrs) of the square root of the sum of the squares of the components of the calculated step in the geometry. Should the step exceed this value, every component of the step is scaled smaller to yield a sum of this size.

STPTOL VARIABLE *10e-5*

The STPTOL keyword specifies the failure criteria for an optimization. If the maximum component of the calculated step is smaller than this value, the optimization will fail unless RETRY is on. The reason for this is that when a geometry step is of such a small magnitude, there is very little change in either the energy or the gradient. The overall effect of this situation is a useless geometry step. And since the gradient has not changed at all, the next geometry step will have the exact same result. So the net effect of such a small geometry step is a stalled optimization.

VERSION VARIABLE *v2m*

The VERSION keyword specifies the version of the coefficients that will be used with multilevel method chosen for the multilevel optimization. Please see the particular method in Section 4.B. for the available versions of the coefficients. The default is *v2m*.

NEB LIST *No Default*

This keyword indicates all the options for the NEB algorithm. Its keywords are:

REACT LIST *No Default.*

The REACT keyword specifies the geometry of the first fixed end point. It is required, and it follows the same rules as the the GEOM keyword in the MULTIGEN section.

PROD LIST *No Default.*

The PROD keyword specifies the geometry of the second fixed end point. It is required, and it follows the same rules as the the GEOM keyword in the MULTIGEN section.

NIMG VARIABLE *No default.*

The NIMG keyword indicates the number of movable images to be used in the band.

NEBITER VARIABLE 40

The NEBITER keyword gives the maximum number of iterations in the NEB optimization.

INTERP VARIABLE LINEAR

The INTERP keyword defines the interpolation method to be used to generate the initial chain of images. The only method available in MULTILEVEL-V4.1 is linear interpolation between the two fixed end points (*linear*).

IMGFILE VARIABLE *No default.*

The IMGFILE keyword gives the name of the file that will be created or already exists which contains the initial chain of images (this file also contains the two fixed end points)

KSPR VARIABLE 0.01

The KSPR keyword specifies the value for the spring constant to be used to evaluate the spring force (in a.u.).

NEBALG VARIABLE *No default.*

The NEBALG keyword specifies the version of the NEB method to be used when computing the tangent and the spring forces. The available values for this keyword are: *b-neb* (for the Bisect NEB), *it-neb* (for the Improved Tangent NEB), and *ci-neb* (for the Climbing Image NEB).

OPTM VARIABLE *No default.*

The OPTM keyword gives the optimization algorithm to be used for the NEB minimization. All the options are quasi-Newton methods and they differ in the way of evaluating (recalculate or update) the Hessian along the minimization cycles. The options available for this keyword are: *bfgs* (for the Broyden-Fletcher-Goldfarb-Shanno update of the Hessian), *dfp* (for the Davidson-Fletcher-Powell update scheme), *nr* (for recalculating the Hessian using the hhooks subroutines), and *vb* (for the Modified Broyden method).

Recommended values for all of these keywords can be found in reference [Go06].

6.D. IMO Section

This section contains keywords that are specific to the IMO methods.

IMOHC LIST *Not used.*

This keyword indicates the all the options for the IMOHC method. Its keywords are:

CAPATOM VARIABLE *No default.*

The CAPATOM keyword is a variable keyword that specifies the atom number of the capping atom. Required.

CSS LIST *No default.*

The CSS keyword is a keyword that specifies the atom numbers (in GEOM) of the atoms in the small system, excluding the capping atom. It must be specified.

HC LIST *No default.*

The HC keyword is a keyword that specifies the atoms involved in the three harmonic interaction terms as well as the force constants. The general format is as follows:

HC
k1 R_0 *atom1 atom2*
k2 A_0 *atom1 atom2 atom3*
k3 D_0 *atom1 atom2 atom3 atom4*
END

for example,

```
HC
   0.10  1.079    1   2
   0.02  0.000    1   2   3
   0.01  3.141    1   2   3   7
END
```

The force constants and the equilibrium bond length should be in atomic units, and the equilibrium angles should be in degrees. The atoms are indicated by their atom numbers (from GEOM).

HLKEY LIST

This keyword specifies the options for the high-level calculations to be done. The following are its valid options:

BASIS VARIABLE *No default.*

The BASIS keyword indicates the basis set for the high level calculation.

METHOD VARIABLE *mp2 / hf*

The METHOD keyword specifies the method for the high level calculation.

OPTIONS LIST *No default.*

This keyword indicates the options for the high level electronic structure program call.

PROGRAM VARIABLE *g03*

This keyword indicates the electronic structure package to be used for the high level calculations. The four valid options are *g09*, *g03*, *g94* and *g98*.

LLKEY LIST

This keyword specifies the options for the low-level calculations to be done. The following are its valid options:

BASIS VARIABLE *No default.*

The BASIS keyword indicates the basis set for the low level calculation.

METHOD VARIABLE *mp2 / hf*

The METHOD keyword specifies the method for the low level calculation.

OPTIONS LIST *No default.*

This keyword indicates the options for the low level electronic structure program call.

PROGRAM VARIABLE *g03*

This keyword indicates the electronic structure package to be used for the low level calculations. The four valid options are *g09*, *g03*, *g94* and *g98*.

6.E. LC Section

This section contains keywords that are specific to the linear combination (LC) methods. Its keywords are as follows:

COOP/NOCOOP SWITCH *NOCOOP*

This switch keyword indicates whether electronic structure information will be shared between SAC, MCSAC, IB, MCCMCO, MCCMUT, and MCCMNM method, and whether all possible default calculations will be carried out with the available information.

G2 LIST *Not used.*

The G2 keyword lists the options for a G2 calculation. The valid keywords are as follows:

ALPHA VARIABLE *0*

BETA VARIABLE *0*

These two keywords indicate the number of α and β electrons, respectively, in the molecule. These values must be specified to obtain a valid G2 energy. Note that $\alpha > \beta$.

G2OPTIONS VARIABLE *No default.*

This keyword specifies any options required for the electronic structure program for the G2 and MCG2 calculations.

G2PROGRAM VARIABLE *g03*

This keyword indicates the electronic structure program to be called to gather electronic structure information for the G2 and MCG2 calculations. Currently *g09*, *g03*, *g94* and *g98* are the supported values, which correspond to the GAUSSIAN09, GAUSSIAN03, GAUSSIAN98, and GAUSSIAN94 electronic structure programs, respectively.

G3PROGRAM VARIABLE *g98*

This keyword indicates the electronic structure program to be called to gather electronic structure information for the G3 and MCG3 calculations. Currently g09, g03, *g98* and *g94* are the supported values for serial mode, which correspond to the GAUSSIAN09, GAUSSIAN03, GAUSSIAN98 and GAUSSIAN94 electronic structure programs, respectively. If the program has been compiled for parallel execution, the value pg98 will run MCG3 energies and gradients in parallel using GAUSSIAN98. For MCG3 energies, one thread calculates the MP2/MG3 component, and the second thread calculates the MP4SDQ/6-31G(2df,p) and QCISD(T)/6-31G(d) components. For MCG3 Gradients, one thread calculates the MP2/MG3 and MP4SDQ/6-31G(2df,p) components, and the second thread calculates the QCISD(T)/6-31G(d) component. See section 6.G. for more information on parallel operation.

G3OPTIONS LIST *No default.*

This keyword specifies any options required for the electronic structure program for the G3 calculations.

G3 LIST *Not used.*

The G3 keyword lists the options for a G3 calculation. The valid keywords are as follows:

ALPHA VARIABLE *0*

BETA VARIABLE *0*

These two keywords indicate the number of α and β electrons, respectively, in the molecule. These values must be specified to obtain a valid G3 energy.

G3SXMP3

The G3SXMP3 keyword specifies the G3SX(MP3) method. The valid options are:

COEFFS LIST PAGE 42

The COEFFS keyword indicates the coefficients for the given G3SX(MP3) method.

G3SX(MP3)OPTIONS LIST *No default.*

This keyword specifies any options required for the electronic structure program for the G3SX(MP3) calculations.

HFMO LIST *Not used.*

The HFMO keyword is a list keyword used to specify an HF||MO calculation and the options for the calculation. The valid options are:

HFKEY LIST *Not used.*

This keyword specifies the options for the HF calculation. The valid options are:

PROGRAM VARIABLE *g98*

The PROGRAM keyword indicates the electronic structure program to be used for the HF calculation.

BASIS VARIABLE *sto-3g*

The BASIS keyword indicates the basis set for the HF calculation.

OPTIONS LIST *Not used.*

This keyword is to list any special program options for the HF calculations.

MOKEY **LIST** **NOT USED.**

The mokey keyword is used to indicate the options for the MO calculation. Its valid options are:

METHOD VARIABLE *am1*

This keyword indicates the MO method for the calculation.

OPTIONS LIST *Not used.*

This keyword indicates any options for use in the electronic structure package.

PROGRAM VARIABLE *g98*

The PROGRAM keyword indicates the electronic structure program to be used for the MO calculation.

FRACHF VARIABLE *0.25*

The FRACHF keyword specifies the fraction *x* of the HF contribution in the HF||MO method.

IB LIST *Not used.*

The IB keyword is a list keyword that specifies the options for IB calculations. The following are the valid keyword options for the list:

ALPHA VARIABLE *3.39*

The ALPHA keyword specifies the IB α parameter, corresponding to the HF contribution.

BETA VARIABLE [*see Table 3*]

The BETA keyword specifies the IB β parameter, corresponding to the correlation contribution. The default value is set appropriately based on the IB method and basis set combination chosen.

HLBASIS VARIABLE *cc-pvtz*

This keyword indicates the higher-level basis set.

LLBASIS VARIABLE *cc-pvdz*

This keyword specifies the lower-level basis set.

METHOD VARIABLE *mp2*

The METHOD keyword indicates the IB method to be used.

MC3BB

The MC3BB keyword specifies the multi-coefficient three-parameter Becke-Becke95 method. The valid options are:

COEFFS LIST [*see Table 10*]

The COEFFS keyword indicates the coefficients for the given MC3BB method.

MC3BBOPTIONS LIST *No default.*

This keyword specifies any options required for the electronic structure program for the MC3BB calculations.

MC3MPW

The MC3MPW keyword specifies the multi-coefficient three-parameter modified Perdew-Wang method. The valid options are:

COEFFS LIST [*see Table 10*]

The COEFFS keyword indicates the coefficients for the given MC3MPW method.

MC3MPWOPTIONS LIST *No default.*

This keyword specifies any options required for the electronic structure program for the MC3MPW calculations.

MCCMCO

The MCCMCO keyword is a list keyword that specifies the options for the Colorado variant of the MCCM (MCCM-CO) method. The following are the valid options for the list:

METHOD VARIABLE *mp2*

The METHOD keyword indicates the MCCM-CO method to be used.

LLBASIS VARIABLE *cc-pvdz*

This keyword specifies the lower-level basis set for the calculation. Currently default coefficients are only available for cc-pVDZ as the lower-level basis set.

HLBASIS VARIABLE *cc-pvtz*

This keyword specifies the higher-level basis set for the calculation. In this version, default coefficients are only available for cc-pVTZ as the higher-level basis set.

COEFFS LIST [*see Table 5*]

The COEFFS keyword indicates the coefficients for the given MCCM-CO method.

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MCCM-CO method. The valid options are given in Table 5.

MCCMUT

The MCCMUT keyword is a list keyword that specifies the options for the Utah variant of the MCCM (MCCM-UT) method. The following are the valid options for the list:

METHOD VARIABLE *ccsd(t)*

The METHOD keyword indicates the MCCM-UT method to be used.

LLBASIS VARIABLE *cc-pvdz*

This keyword specifies the lower-level basis set for the calculation. Currently default coefficients are only available for cc-pVDZ as the lower-level basis set.

HLBASIS VARIABLE *cc-pvtz*

This keyword specifies the higher-level basis set for the calculation. In this version, default coefficients are only available for cc-pVTZ as the higher-level basis set.

COEFFS LIST [*see Table 6*]

The COEFFS keyword indicates the coefficients for the given MCCM-UT method.

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MCCM-UT method. The valid options are given in Table 6.

MCCMNM

The MCCMNM keyword is a list keyword that specifies the options for the New Mexico variant of the MCCM (MCCM-NM) method. The following are the valid options for the list:

METHOD VARIABLE *ccsd(t)*

The METHOD keyword indicates the MCCM-NM method to be used.

LLBASIS VARIABLE *cc-pvdz*

This keyword specifies the lower-level basis set for the calculation. Currently default coefficients are only available for cc-pVDZ as the lower-level basis set.

MLBASIS VARIABLE *pdz+*

This keyword specifies the middle-level basis set for the calculation. Currently default coefficients are only available for pDZ+ as the lower-level basis set.

HLBASIS VARIABLE *cc-pvtz*

This keyword specifies the higher-level basis set for the calculation. In this version, default coefficients are only available for cc-pVTZ as the higher-level basis set.

COEFFS LIST [*see Table 6*]

The COEFFS keyword indicates the coefficients for the given MCCM-NM method.

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MCCM-NM method. The valid options are given in Table 6.

MCG2

The MCG2 keyword specifies the multi-coefficient Gaussian-2 (MCG2) method. The valid options are:

COEFFS LIST [*see Table 7*]

The COEFFS keyword indicates the coefficients for the given MCG2 method.

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MCG2 method. The valid options are given in Table 7.

MCG3

The MCG3 keyword specifies the multi-coefficient Gaussian-3 (MCG3) method. The valid options are:

COEFFS LIST [*see Table 8*]

The COEFFS keyword indicates the coefficients for the given MCG3 method. The coefficient tree used will be a version 3 tree (see figure 6).

VERSION VARIABLE *v2m*

The version keyword indicates the version of the coefficients for the given MCG3 method. The valid options are given in Table 8.

MCG3OPTIONS LIST *No default.*

This keyword specifies any options required for the electronic structure program for the MCG3 calculations.

MCOMP2

The MCOMP2 keyword specifies an MP2 variant of the MCCM method, in particular MCCM-CO; MG3; 6-31+G(d). Note that this method can be obtained using the MCCMCO keyword by specifying the 2 basis sets and the coefficients in Table 9. The valid options are:

COEFFS LIST [*see Table 9*]

The COEFFS keyword indicates the coefficients for the given MCOMP2 method. The coefficient tree used will be a version 3 tree using the MG3S basis set.

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MCCM-CO; MG3; 6-31+G(d) method. The valid options are given in Table 9.

MCOPTIONS LIST *Not used.*

This list keyword is used to indicate any special options for the electronic structure package for calculations employinh SAC, MC-SAC, IB, MCCM-CO, MCCM-UT, and MCCM-NM.

MCQCISD

The MCQCISD keyword specifies the multi-coefficient quadratic configuration interaction with single and double excitations (MC-QCISD) method. The valid options are:

COEFFS LIST [*see Table 9*]

The COEFFS keyword indicates the coefficients for the given MC-QCISD method. The coefficient tree used will be a version 3 tree (see figure 8).

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MC-QCISD method. The valid options are given in Table 9.

MCQCISDOPTIONS LIST *Not used.*

This list keyword is used to indicate any special options for the electronic structure package for calculations employing MC-QCISD.

MCPROGRAM VARIABLE *g98*

This keyword indicates the electronic structure program to be called to gather electronic structure information for SAC, MCSAC, IB, MCCMCO, MCCMUT, and MCCMNM calculations. Currently only GAUSSIAN09 (*g09*), GAUSSIAN03 (*g03*), GAUSSIAN98 (*g98*) and GAUSSIAN94 (*g94*) are supported values.

MCSAC

The MCSAC keyword is a list keyword that specifies the options for the MCSAC method. The following keywords are the valid options for the list:

METHOD VARIABLE *ccsd*

The METHOD keyword indicates the MCSAC method to be used.

BASIS VARIABLE *cc-pvdz*

This keyword specifies the basis set for the calculation.

COEFFS LIST [*see Table 2*]

The COEFFS keyword indicates the coefficients for the given MCSAC method.

VERSION VARIABLE *v2m*

The VERSION keyword indicates the version of the coefficients for the given MCSAC method. The valid options are given in Table 2.

SAC

The SAC keyword is a list keyword that specifies the options for the SAC method. The following are the valid options for the list:

BASIS VARIABLE *cc-pvdz*

This keyword specifies the basis set for the calculation.

COEFFS LIST [*see Table 1*]

The COEFFS keyword indicates the coefficients for the given SAC method.

METHOD VARIABLE *mp2*

The METHOD keyword indicates the SAC method to be used.

VERSION VARIABLE *v2m*

The version keyword indicates the version of the coefficients for the given SAC method. The valid options are given in Table 1.

6.F. TEST Section

This section contains keywords that are specific to the testing function of MULTILEVEL. This function allows the user to gather energies, gradients, and Hessians at only one level of electronic structure theory. The user may also optimize at the specified level using the algorithms enabled in MULTILEVEL. This may be helpful in testing starting point geometries and Hessian levels to be used in an optimization, before attempting to optimize with the more expensive MULTILEVEL methods. Its keywords are as follows:

BASIS VARIABLE *cc-pvdz*

The BASIS keyword indicates the basis set to be used in the calculation. Note that for certain chemistry models the basis set is already predefined.

GENECP VARIABLE *No Default.*

This keyword indicates the use of effective core potentials. The argument must be the name of the file that contains the specifications of the basis set and pseudopotentials for all the atoms. That file must be put into the basis subdirectory of MULTILEVEL. This keyword must be used in combination with the variable *gen* in the **BASIS** keyword.

METHOD VARIABLE *mp2*

This keyword specifies the electronic structure method to be used.

OPTIONS LIST *No default*

The OPTIONS keyword may be used to specify any options necessary for the electronic structure program. As in previous sections, this keyword can be use to specify also the memory requirements and the number of processors to be used.

PROGRAM VARIABLE *g03*

This keyword indicates the electronic structure program to be called for the TEST energies, gradients, and Hessians. Currently *g09*, g03, *g98,* and *g94* are the supported

values, which correspond to the GAUSSIAN09, GAUSSIAN03, GAUSSIAN98, and GAUSSIAN94 electronic structure programs, respectively.

6.G. Running MULTILEVEL in Parallel

There are two ways in which MULTILEVEL calculations can be run in parallel. The first, and most simple, is to run GAUSSIAN in parallel by passing link1 commands to GAUSSIAN09/03/98 through the program option. See page 47 and testrun 11. The second method of parallel execution involves spawning multiple threads within MULTILEVEL itself. MULTILEVEL can be compiled with the appropriate OpenMP libraries to offer limited parallel capabilities. The parallel operation of multilevel allows multiple GAUSSIAN09/03 calculations to be run simultaneously. Currently, only the calculation of MCG3 energies and gradients will operate in parallel mode.

Chapter Seven

7

7. Description of Files in MULTILEVEL

This chapter gives a description of the files involved in the compiling and running of MULTILEVEL, such as: the source code needed to compile the program, files required to run MULTILEVEL, files created during a run of MULTILEVEL, and a script supplied to simplify the running of MULTILEVEL.

7.A. Source Code

MULTILEVEL source code is composed of eight FORTRAN90 files. In the following subsections, 7.A.1 describes each of the files in the source code, and 7.A.2 gives an alphabetical listing and description of each subprogram in MULTILEVEL.

7.A.1. Source Code Files

display.F
This file contains the subprograms for displaying most of the MULTILEVEL output.

ef.F
This file contains the driver for the eigenvector following algorithm.

ehooks.F
This file contains the subprograms for carrying out energy calculations as well as the formula subroutines.

freq.F
This file contains the subprograms for calculating the harmonic vibrational frequencies and normal mode coordinates.

gau_ext_opt.F
This file contains the subprograms for calling GAUSSIAN09/03's Berny optimizer.

ghooks.F
This file contains the subprograms for carrying out gradient calculations.

hhooks.F
This file contains the subprograms for conducting Hessian calculations.

main.F
This file contains the driver for the MULTILEVEL program.

ml.F
This file contains the subprograms for parsing the MULTILEVEL input file, *ml.inp*.

module.F
This file contains all the modules defining parameters for the program, defining the default coefficients for multi-coefficient methods, creating structures for the input information, and defining common blocks for the energies, gradients, and Hessians to be calculated.

ohooks.F
This file contains the subprograms for carrying out optimizations with multilevel.

nebmin.F
This file contains the main driver for the NEB algorithm.

neb.F
This file contain all the subprograms needed for the NEB minimization.

7.A.2. Subprogram List

The listings in this section have the subprogram name in bold. On that same line is the type of subprogram and the file that contains the subprogram. The following lines then give a short description of the subprogram.

ADX, ADX2 subroutine ghooks.F
Calculates the components of the harmonic angle correction to the IMOHC gradient.

ATMINFO module module.F
Contains the information used to confirm that the atomic symbols given in the input geometry are valid and to assign an atomic mass based on this symbol.

BRENT function ohooks.F
Performs Brent line minimization given three points bracketing a minimum.

CALG subroutine ehooks.F
Calculates a G2 or G3 energy, gradient component, or Hessian component.

CALG3SXMP3 subroutine ehooks.F
Calculates a G3SX(MP3) energy, gradient component, or Hessian component.

CALHFMO subroutine ehooks.F
Calculates an HF||MO energy, gradient component, or Hessian component.

CALIB subroutine ehooks.F
Calculates an IB energy, gradient component, or Hessian component.

CALMC3BB subroutine ehooks.F
Calculates an MC3BB energy, gradient component, or Hessian component.

CALMC3MPW subroutine ehooks.F
Calculates an MC3MPW energy, gradient component, or Hessian component.

CALMCO subroutine ehooks.F
Calculates an MCCM-CO energy, gradient component, or Hessian component.

CALMCS subroutine ehooks.F
Calculates an MCSAC energy, gradient component, or Hessian component.

CALMG subroutine ehooks.F
Calculates an MCG2 or MCG3 energy, gradient component, or Hessian component.

CALMNM subroutine ehooks.F
Calculates an MCCM-NM energy, gradient component, or Hessian component.

CALMQC subroutine ehooks.F
Calculates an MC-QCISD energy, gradient component, or Hessian component.

CALMUT subroutine ehooks.F
Calculates an MCCM-UT energy, gradient component, or Hessian component.

CALTYP module module.F
Contains parameters for identifying the type of calculation (i.e., optimization, energy, gradient, or Hessian calculation)

CALSAC subroutine ehooks.F
Calculates an SAC energy, gradient component, or Hessian component.

CASE function ml.F
Converts input strings to all lower case letters for case consistency.

CFLOAT function ml.F
Converts a string to a double precision number.

CFLTFMT function ehooks.F
Converts a string from a formatted checkpoint file to a double precision number. Takes advantage of the known formats of numbers in the checkpoint files.

CHKLN subroutine ml.F
Checks a line for special characters such as a comment, a section start, or a list keyword end.

D2XX, D2XY subroutine hhooks.F
Calculates the components of the harmonic bond correction to the IMOHC Hessian.

DATTIM subroutine display.F
Gets the date and time for the output file.

DAX1X1, DAX1X2, DAX1X3, DAX2X2, DAX1Y1, DAX1Y2, DAX1Y3, DAX2Y2
subroutine hhooks.F
Calculates the components of the harmonic angle correction to the IMOHC Hessian.

DAXPY subroutine freq.F
BLAS routine calculates c**x+y** vector.

DEFGEN subroutine ml.F
Sets the defaults for the MULTIGEN section input information.

DEFIMO subroutine ml.F
Sets the defaults for the IMO section input information.

DEFLC subroutine ml.F
Sets the defaults for the LC section input information.

DEFPROG subroutine ml.F
Sets the defaults for the electronic structure programs to be called for different MULTILEVEL calculations.

DEFTEST subroutine ml.F
Sets the defaults for the TEST section input information.

DGEDI subroutine freq.F
Computes the determinant and inverse of a matrix using the factors computed by **DGEFA**.

DGEFA subroutine freq.F
Factors a double precision matrix by Gaussian elimination.

DSWAP subroutine freq.F
BLAS routine interchanges two vectors.

DX1X1, DX1X2, DX1X3, DX1X4, DX2X2, DX2X3, DX1Y1, DX1Y2, DX1Y3, DX1Y4, DX2Y2, DX2Y3
subroutine hhooks.F
Calculates the components of the harmonic torsion correction to the IMOHC Hessian.

EF subroutine ef.F
Eigenvector following driver.

EFOVLP subroutine ef.F
Determines the overlap of the geometry steps in the eigenvector following algorithm.

ENERGY module module.F
Defines all the COMMON block energy variables.

ESGPARSE subroutine ghooks.F
Makes all the calls to PROGGHK for the SAC, MCSAC, IB, and MCCM methods when COOP is on.

ESHPARSE subroutine hhooks.F
Makes all the calls to PROGHHK for the SAC, MCSAC, IB, and MCCM methods when COOP is on.

ESPARSE subroutine ehooks.F
Makes all the calls to PROGEHK for the SAC, MCSAC, IB, and MCCM methods when COOP is on.

F1DIM function ohooks.F
A pseudo 1-dimensional function for the energy of the molecule used to perform Brent line minimization of the step size during optimization.

FCHAR subroutine ml.F
Finds the next character on a line.

FILES module module.F
Contains the definitions of the file handles, names, and locations used throughout MULTILEVEL. May be modified to change file handle numbers, basis set file locations, or memory allocations used in GAUSSIAN input files. But the filenames themselves should not be changed.

FREQCAL subroutine freq.F
Calculates the harmonic vibrational frequencies and normal mode coordinates.

FSPACE subroutine ml.F
Finds the next blank space on a line.

G2DISP subroutine display.F
Displays the G2 output information.

G2EHK subroutine ehooks.F
Gets the G2 energy, using information from MCG2 calculations, if done.

G2GHK subroutine ghooks.F
Gets the G2 gradient and energy, using information from MCG2 calculations, if performed.

G2HHK subroutine hhooks.F
Gets the G2 Hessian, gradient, and energy, using information from MCG2 calculations, if performed.

G3DISP subroutine display.F
Displays the G3 output information.

G3SXMP3DISP subroutine display.F
Displays the G3SX(MP3) output information.

G3EHK subroutine ehooks.F
Gets the G3 energy, using information from MCG3/MMCG3 calculations, if done.

G3GHK subroutine ghooks.F
Gets the G3 gradient and energy, using information from MCG3 calculations, if performed.

G3HHK subroutine hhooks.F
Gets the G3 Hessian, gradient, and energy, using information from MCG3 calculations, if performed.

G3SXMP3EHK subroutine ehooks.F
Carries out G3SX(MP3) energy calculations.

G3SXMP3GHK subroutine ghooks.F
Carries out G3SX(MP3) gradient and energy calculations.

G3SXMP3HHK subroutine hhooks.F
Carries out G3SX(MP3) Hessian, gradient, and energy calculations.

G98INP subroutine ehooks.F
Creates input files for GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, and GAUSSIAN09.

G98OUTE subroutine ehooks.F
Reads the energy from a GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09 formatted checkpoint file.

G98OUTG subroutine ghooks.F
Reads the gradient from GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09 formatted checkpoint file.

G98OUTH subroutine hhooks.F
Reads the Hessian from GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09 formatted checkpoint file.

G98OUTO subroutine ohooks.F
Reads the optimized geometry, energy, and gradient from a GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09 formatted checkpoint file.

GAU_EXT_OPT subroutine gau_ext_opt.F
Gau_ext_opt is an interface subroutine which create an input file for GAUSSIAN09/03 to perform multilevel optimization by using the GAUSSIAN09/03's Berny optimizer.

GETEIB subroutine ehooks.F
Gets an IB energy using previously calculated electronic structure information when COOP is on.

GETEMCO subroutine ehooks.F
Gets an MCCM-CO energy using previously calculated electronic structure information when COOP is on.

GETEMCS subroutine ehooks.F
Gets an MCSAC energy using previously calculated electronic structure information when COOP is on.

GETEMNM subroutine ehooks.F
Gets an MCCM-NM energy using previously calculated electronic structure information when COOP is on.

GETEMUT subroutine ehooks.F
Gets an MCCM-UT energy using previously calculated electronic structure information when COOP is on.

GETESAC subroutine ehooks.F
Gets a SAC energy using previously calculated electronic structure information when COOP is on.

GETGIB subroutine ghooks.F
Gets an IB gradient using previously calculated electronic structure information when COOP is on.

GETGMCO subroutine ghooks.F
Gets an MCCM-CO gradient using previously calculated electronic structure information when COOP is on.

GETGMCS subroutine ghooks.F
Gets an MCSAC gradient using previously calculated electronic structure information when COOP is on.

GETGMNM subroutine ghooks.F
Gets an MCCM-NM gradient using previously calculated electronic structure information when COOP is on.

GETGMUT subroutine ghooks.F
Gets an MCCM-UT gradient using previously calculated electronic structure information when COOP is on.

GETGSAC subroutine ghooks.F
Gets a SAC gradient using previously calculated electronic structure information when COOP is on.

GETHIB subroutine hhooks.F
Gets an IB Hessian using previously calculated electronic structure information when COOP is on.

GETHLC function ehooks.F
Calculates the higher level energy correction for G2 or G3.

GETHMCO subroutine hhooks.F
Gets an MCCM-CO Hessian using previously calculated electronic structure information when COOP is on.

GETHMCS subroutine hhooks.F
Gets an MCSAC Hessian using previously calculated electronic structure information when COOP is on.

GETHMNM subroutine hhooks.F
Gets an MCCM-NM Hessian using previously calculated electronic structure information when COOP is on.

GETHMUT subroutine hhooks.F
Gets an MCCM-UT Hessian using previously calculated electronic structure information when COOP is on.

GETHSAC subroutine hhooks.F
Gets a SAC Hessian using previously calculated electronic structure information when COOP is on.

GETSO function ehooks.F
Calculates the spin-orbit energy correction for G3.

GPARAM module module.F
Contains the parameters and names for the components of G2, MCG2, G3, and MCG3, as well as all default coefficients for MCG2 and MCG3.

GRADIENT module module.F
Defines all the COMMON block gradient variables.

HCGEOM subroutine ehooks.F
Creates geometries for the small and large systems in IMOHC based on the atoms specified with CAPATOM and CSS.

HESSIAN module module.F
Defines all the COMMON block Hessian variables.

HFMODISP subroutine display.F
Displays all the HF||MO output.

HFMOEHK subroutine ehooks.F
Gets the HF||MO energy.

HFMOGHK subroutine ghooks.F
Gets the HF||MO energy and gradient.

HFMOHHK subroutine hhooks.F
Gets the HF||MO energy, gradient, and Hessian.

HGRAD subroutine ghooks.F
Adds the harmonic correction to the IMOHC gradient.

HHESS subroutine hhooks.F
Adds the harmonic correction to the IMOHC Hessian.

HNRG subroutine ehooks.F
Adds the harmonic correction to the IMOHC energy.

IBCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested IB gradient or Hessian calculation when COOP is on.

IBDISP subroutine display.F
Displays the IB output from IBEHK, IBGHK, or IBHHK.

IBEHK subroutine ehooks.F
Gets an IB energy when NOCOOP is on.

IBGHK subroutine ghooks.F
Gets an IB gradient and energy when NOCOOP is on.

IBHHK subroutine hhooks.F
Gets an IB Hessian, gradient, and energy when NOCOOP is on.

IBMCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested IB energy calculation when COOP is on.

ICINT function ml.F
Converts a string to an integer.

IDAMAX subroutine freq.F
BLAS routine finds the index of element having maximum absolute value.

IMOHCDISP subroutine display.F
Displays the IMOHC output.

IMOHCEHK subroutine ehooks.F
Gets an IMOHC energy.

IMOHCGHK subroutine ghooks.F
Gets an IMOHC gradient and energy.

IMOHCHHK subroutine hhooks.F
Gets an IMOHC Hessian, gradient, and energy.

INPUT module module.F
Contains the structure types into which all the user input information is placed.

INSUMRY subroutine display.F
Prints out a summary of the user input information.

LINMN subroutine ohooks.F
Optimizes the geometry step during an optimization via Brent line minimization.

LOWDISP subroutine display.F
Displays the low level Hessian calculated for a geometry optimization.

LUDCMP subroutine ohooks.F
Conducts LU decomposition on a matrix.

LUBKSB subroutine ohooks.F
Back substitutes into an LU decomposed matrix to find a solution to a linear equation.

MC3BBEHK subroutine ehooks.F
Carries out MCG3BB energy calculations.

MC3BBGHK subroutine ghooks.F
Carries out MCG3BB gradient and energy calculations.

MC3BBHHK subroutine hhooks.F
Carries out MC3BB Hessian, gradient, and energy calculations.

MC3MPWEHK subroutine ehooks.F
Carries out MCG3MPW energy calculations.

MC3MPWGHK subroutine ghooks.F
Carries out MCG3MPW gradient and energy calculations.

MC3MPWHHK subroutine hhooks.F
Carries out MC3MPW Hessian, gradient, and energy calculations.

MCCHK subroutine ehooks.F
Determines all possible SAC, MCSAC, IB, and MCCM calculations that are able to be done based on the electronic structure program calls being made when COOP is on.

MCDISP subroutine display.F
Displays all SAC, MCSAC, IB, and MCCM output when COOP is on.

MCEHK subroutine ehooks.F
Serves as a front end for all SAC, MCSAC, IB, and MCCM energy calculations when COOP is on.

MCGHK subroutine ghooks.F
Serves as a front end for all SAC, MCSAC, IB, and MCCM gradient and energy calculations when COOP is on.

MCHHK subroutine hhooks.F
Serves as a front end for all SAC, MCSAC, IB, and MCCM Hessian, gradient, and energy calculations when COOP is on.

MCOCHK subroutine ehooks.F

Determines which electronic structure program calls are to be made for a requested MCCM-CO gradient or Hessian calculation when COOP is on.

MCODISP subroutine display.F
Displays the MCCM-CO output from MCOEHK, MCOGHK, or MCOHHK.

MCOEHK subroutine ehooks.F
Gets an MCCM-CO energy when NOCOOP is on.

MCOGHK subroutine ghooks.F
Gets an MCCM-CO gradient and energy when NOCOOP is on.

MCOHHK subroutine hhooks.F
Gets an MCCM-CO Hessian, gradient, and energy when NOCOOP is on.

MCOMCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested MCCM-CO energy calculation when COOP is on.

MCSCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested MCSAC gradient or Hessian calculation when COOP is on.

MCSDISP subroutine display.F
Displays the MCSAC output from MCSEHK, MCSGHK, or MCSHHK.

MCSEHK subroutine ehooks.F
Gets an MCSAC energy when NOCOOP is on.

MCSGHK subroutine ghooks.F
Gets an MCSAC gradient and energy when NOCOOP is on.

MG2EHK subroutine ehooks.F
Carries out MCG2 energy calculations.

MG2GHK subroutine ghooks.F
Carries out MCG2 gradient and energy calculations.

MG2HHK subroutine hhooks.F
Carries out MCG2 Hessian, gradient, and energy calculations.

MG3DISP subroutine display.F
Displays MCG3 output.

MG3EHK subroutine ehooks.F
Carries out MCG3 energy calculations.

MG3GHK subroutine ghooks.F
Carries out MCG3 gradient and energy calculations.

MG3HHK subroutine hhooks.F
Carries out MCG3 Hessian, gradient, and energy calculations.

MG98OUTE subroutine ehooks.F
Extracts multiple energies from either a GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09 formatted checkpoint file.

MLEHOOK subroutine ehooks.F
Serves as a front end for all MULTILEVEL energy calculations.

MLGHOOK subroutine ghooks.F
Serves as a front end for all MULTILEVEL gradient and energy calculations.

MLHEDR subroutine display.F
Prints out the program header in the output file.

MLHHOOK subroutine hhooks.F
Serves as a front end for all MULTILEVEL Hessian calculations.

MLOHOOK subroutine ohooks.F
Serves as a front end for all MULTILEVEL optimization algorithms.

MNBRAK subroutine ohooks.F
Brackets a minimum of a 1-dimensional function with three points.

MNMCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested MCCM-NM gradient or Hessian calculation when COOP is on.

MNMDISP subroutine display.F
Displays the MCCM-NM output from MNMEHK, MNMGHK, or MNMHHK.

MNMEHK subroutine ehooks.F
Gets an MCCM-NM energy when NOCOOP is on.

MNMGHK subroutine ghooks.F
Gets an MCCM-NM gradient and energy when NOCOOP is on.

MNMHHK subroutine hhooks.F
Gets an MCCM-NM Hessian, gradient, and energy when NOCOOP is on.

MNMMCHK subroutine ehooks.F

Determines which electronic structure program calls are to be made for a requested MCCM-NM energy calculation when COOP is on.

MPROGEHK subroutine ehooks.F
Carries out an energy call with the specified electronic structure program, extracting multiple energies.

MQCDISP subroutine display.F
Displays the MC-QCISD output from MQCEHK, MQCGHK, or MQCHHK.

MQCEHK subroutine ehooks.F
Gets an MC-QCISD energy.

MQCGHK subroutine ghooks.F
Gets an MC-QCISD gradient and energy.

MQCHHK subroutine hhooks.F
Gets an MC-QCISD Hessian, gradient, and energy.

MTOLTM subroutine hhooks.F
Converts a symmetric matrix to a 1-dimensional array containing the lower triangular portion of the matrix.

MUTCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested MCCM-UT gradient or Hessian calculation when COOP is on.

MUTDISP subroutine display.F
Displays the MCCM-UT output from MUTEHK, MUTGHK, or MUTHHK.

MUTEHK subroutine ehooks.F
Gets an MCCM-UT energy when NOCOOP is on.

MUTGHK subroutine ghooks.F
Gets an MCCM-UT gradient and energy when NOCOOP is on.

MUTHHK subroutine hhooks.F
Gets an MCCM-UT Hessian, gradient, and energy when NOCOOP is on.

MUTMCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested MCCM-UT energy calculation when COOP is on.

MXLNEQ subroutine ohooks.F
Calculates the inverse of a matrix.

NEWT subroutine ohooks.F
Carries out Newton-Raphson optimization with Brent line minimization using either a high level Hessian, a low level Hessian, or a scaled unit matrix, kept frozen when not recalculated.

NEWT2 subroutine ohooks.F
Carries out Newton-Raphson optimization with Brent line minimization using either a high level Hessian, a low level Hessian, or a scaled unit matrix with either BFGS or DFP updates, when not recalculated.

NUMFMT module module.F
Contains information on the formats of numbers in the GAUSSIAN94, GAUSSIAN98, GAUSSIAN03 and GAUSSIAN09 formatted checkpoint files.

OPTDISP subroutine ohooks.F
Displays a step in the optimization.

OPTEHK subroutine ohooks.F
Gets the appropriate energy during an optimization.

OPTGHK subroutine ohooks.F
Gets the appropriate gradient and energy during an optimization.

OPTHHK subroutine ohooks.F
Gets the appropriate Hessian (and possibly gradient and energy, as well) during an optimization.

OPTIMIZE module module.F
Contains COMMON block variables used during optimization.

PARAM module module.F
Contains the parameters and names for components of SAC, MCSAC, IB, and MCCM methods as well as some general program parameters.

PRJFC subroutine ehooks.F
Calculates the projected force constant matrix.

PROGDEF module module.F
Contains the default electronic structure programs called by the various parts of MULTILEVEL. Automatically all default programs are set to GAUSSIAN09, but the user may alter some or all to another supported electronic structure package.

PROGEHK subroutine ehooks.F
Carries out a single energy call to the specified electronic structure program.

PROGGHK subroutine ghooks.F
Carries out a gradient call to the specified electronic structure program.

PROGHHK subroutine hhooks.F
Carries out a Hessian call to the specified electronic structure program.

PROGOHK subroutine ohooks.F
Carries out an optimization call to the specified electronic structure program.

RCOEF subroutine ml.F
Reads the coefficients from the COEFFS keywords in the input file.

RCONST subroutine ml.F
Reads the CONSTANT list keyword from the MULTIGEN section.

RCSS subroutine ml.F
Reads the CSS list keyword from the IMO section.

RDX subroutine ghooks.F
Calculates the components of the harmonic bond correction to the IMOHC gradient.

READ5 subroutine ml.F
Reads the MULTILEVEL input file.

REXTOPT subroutine ml.F
Reads the EXTOPT list keyword from the MULTIGEN section.

RG2 subroutine ml.F
Reads the G3 list keyword from the LC section.

RG3 subroutine ml.F
Reads the G3 list keyword from the LC section.

RG3SXMP3 subroutine ml.F
Reads the G3SX(MP3) list keyword from the LC section.

RGEOM subroutine ml.F
Reads the RGEOM list keyword from the MULTIGEN section.

RHC subroutine ml.F
Reads the RHC list keyword from the IMOHC section.

RHFMO subroutine ml.F
Reads the HFMO list keyword from the LC section.

RIB subroutine ml.F

Reads the IB list keyword from the LC section.

RLINE subroutine ml.F
Reads the next non-blank and non-comment line of the input file.

RLIST subroutine ml.F
Reads the OPTIONS list keywords.

RMC3BB subroutine ml.F
Reads the MC3BB list keyword from the LC section.

RMC3MPW subroutine ml.F
Reads the MC3MPW list keyword from the LC section.

RMCCMCO subroutine ml.F
Reads the MCCMCO list keyword from the LC section.

RMCCMNM subroutine ml.F
Reads the MCCMNM list keyword from the LC section.

RMCCMUT subroutine ml.F
Reads the MCCMUT list keyword from the LC section.

RMCG2 subroutine ml.F
Reads the MCG2 list keyword from the LC section.

RMCG3 subroutine ml.F
Reads the MCG3 list keyword from the LC section.

RMCSAC subroutine ml.F
Reads the MCSAC list keyword from the LC section.

RMLGEN subroutine ml.F
Reads the MULTIGEN section.

RMLOPT subroutine ml.F
Reads the MLOPT list keyword from the MULTIGEN section.

RMQCISD subroutine ml.F
Reads the MCQCISD list keyword from the LC section.

ROTCOL subroutine ohooks.F
Given the geometry coordinates for 2 axes, rotates those coordinates about the third axis by a specified angle.

ROTMOL subroutine ohooks.F

Either rotates a molecule to place appropriate atoms at the origin, on an axis, and in a plane for an optimization, or undoes these rotations afterwards.

RSAC subroutine ml.F
Reads the SAC list keyword from the LC section.

RSP subroutine ef.F
EISPACK diagonalization routine. Finds the eigenvalues and eigenvectors of a real symmetric packed matrix.

RTEST subroutine ml.F
Reads the TEST section.

RVAR function ml.F
Returns the string following a variable keyword.

RWORD subroutine ml.F
Reads the next word on a line.

SACCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested SAC gradient or Hessian calculation when COOP is on.

SACDISP subroutine display.F
Displays the SAC output from SACEHK, SACGHK, or SACHHK.

SACEHK subroutine ehooks.F
Gets a SAC energy when NOCOOP is on.

SACGHK subroutine ghooks.F
Gets a SAC gradient and energy when NOCOOP is on.

SACHHK subroutine hhooks.F
Gets a SAC Hessian, gradient, and energy when NOCOOP is on.

SACMCHK subroutine ehooks.F
Determines which electronic structure program calls are to be made for a requested SAC energy calculation when COOP is on.

SETIB subroutine ehooks.F
Determines all possible IB calculations that are able to be done with available electronic structure information if COOP is on.

SETMCO subroutine ehooks.F
Determines all possible MCCM-CO calculations that are able to be done with available electronic structure information if COOP is on.

SETMCS subroutine ehooks.F
Determines all possible MCSAC calculations that are able to be done with available electronic structure information if COOP is on.

SETMCSW subroutine ehooks.F
Sets a new calculation in the switch array storing the possible SAC, IB, MCSAC, and MCCM calculations.

SETMNM subroutine ehooks.F
Determines all possible MCCM-NM calculations that are able to be done with available electronic structure information if COOP is on.

SETMUT subroutine ehooks.F
Determines all possible MCCM-UT calculations that are able to be done with available electronic structure information if COOP is on.

SETROT subroutine ohooks.F
Sets the 3 rotations to be performed based on the atoms and coordinates specified in CONSTANT.

SETSAC subroutine ehooks.F
Determines all possible SAC calculations that are able to be done with available electronic structure information if COOP is on.

SUMDISP subroutine display.F
Displays the summary output file ml.sum.

TESTDISP subroutine display.F
Displays the output from a TEST calculation.

TDX, TDX2 subroutine ghooks.F
Calculates the components of the harmonic torsion correction to the IMOHC gradient.

TQL2 subroutine ef.F
Computes the eigenvalues and eigenvectors of a symmetric tridiagonal matrix by the QL method [H. Bowdler, R. S. Martin, C. H. Reinsch, and J. H. Wilkinson, Num. Math 11 (1968) 293].

TQLRAT subroutine ef.F
Finds the eigenvalues of a symmetric tridiagonal matrix by the rational QL method [C. H. Reinsch, Comm. ACM 16 (1973) 689].

TRBAK3 subroutine ef.F
Forms the eigenvectors of a real symmetric matrix by back transformation of the eigenvectors of the similar symmetric tridiagonal matrix.

TRED3 Newton-Raphson subroutine ef.F
Reduces a real symmetric matrix to a symmetric tridiagonal matrix.

NEBNR subroutine nebmin.F
Performs the NEB minimization when the optimizer chosen is BFGS or DFP.

NEBNR2 subroutine nebmin.F
Performs the NEB minimization when the optimizer chosen is NR (do not updatethe Hessian along the minimization).

NEBVB subroutine nebmin.F
Performs the NEB minimization when the optimizer chosen is VB.

RMINV subroutine nebmin.F
Computes the inverse of a Hessian

NEBHHK subroutine nebmin.F
Computes the Hessian matrix along the NEB minimization.

NEBGST subroutine nebmin.F
Computes the adjusted force acting only on the adjacent images

NEBGHK subroutine nebmin.F
Computes the total force vector containing the M (where M is the number of images) adjusted force vectors acting on each image.

REORIENT subroutine nebmin.F
Does the initial reorientation between the two fixed end points.

INTERP subroutine nebmin.F
Performs the linear interpolation between the two previously reoriented fixed end points, and generates the initial chain of structures.

TANGCALC subroutine nebmin.F
Computes the tangent vector for each image

FORCALC subroutine nebmin.F
Computes the total force vector acting on each image

OPTNEBDISP subroutine nebmin.F
Displays NEB information in the output file

7.B. Files Required to Run MULTILEVEL

Aside from the executable itself, there are three types of files which are necessary to run MULTILEVEL: the input file, the basis set files, and shuttle scripts. All such files are required to be in the directory in which the user is running the program. (See 7.D for a script that handles this requirement.) The following two subsections give the details of the basis set files and the shuttle scripts. The input file, which must be named ml.inp, has been described in detail in the previous chapter.

7.B.1. Basis Set Files

Due to the fact that several of the MULTILEVEL calculations require electronic structure program calls for non-standard basis sets, one or more of the three non-standard GAUSSIAN basis set files may be required. The basis set files and the MULTILEVEL methods which require them are listed below:

Basis Set:	File Name:	Methods Required For:
mpDZ	mpdz.gbs	MCCM-NM
pDZ+	pdz+.gbs	MCCM-NM
G3Large	g3large.gbs	G3
G3XLarge	g3xlarge.gbs	G3SX(MP3)
Modified G3 (MG3)	mg3.gbs	MCG3, MC-QCISD, MCOMP2
MG3-semidiffuse (MG3S)	mg3s.gbs	MCG3/3, MC-QCISD/3, MCOMP2/3
Basis sets and ECPs	lanl2dz.gbs	6-31+G(d,p)

7.B.2. Electronic Structure Program Shuttle Scripts

MULTILEVEL makes many calls to the specified electronic structure package throughout the course of a run. In order to minimize the system calls made within the program, a C-shell shuttle script has been provided for each of the electronic structure packages supported in MULTILEVEL. The only system calls made are directly to these shuttles. The usage of each shuttle script is: *shuttle input-file output-file*. The scripts in this version of MULTILEVEL are named g94shuttle, g98shuttle g03shuttle, and g09shuttle, and they are designed for calling GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, and GAUSSIAN09.

Within each script, the user must modify at least two variables – the one specifying the path of GAUSSIAN and the one for the system scratch directory. The user may also alter the handling of the GAUSSIAN input, output, and scratch files. It is important to keep in mind though that the formatted checkpoint file must remain after the shuttle script has finished, for the GAUSSIAN output is read by MULTILEVEL from that file.

To use the Berny optimizer in GAUSSIAN09/03, a PERL script Gau_External is needed for calling a secondary MULTILEVEL calculation of energy, gradient, and Hessian and passing them to GAUSSIAN09/03.

We note that a subdirectory is created within the user specified scratch directory for the handling of scratch files; this subdirectory is named by the process ID to ensure a unique name. All GAUSSIAN scratch files are stored in that subdirectory. At the end of the GAUSSIAN job, the subdirectory itself is removed. This process allows the user to run more than one MULTILEVEL job at once and not worry about the removal of the scratch files from another GAUSSIAN job. This may be altered at the user's discretion.

7.C. Files Created During a MULTILEVEL Run

Several files are created in the process of running MULTILEVEL which may be of importance to the user.

7.C.1. The Output File: *ml.out*

The output file is created under the name ml.out. The first portion of the file summarizes the options specified in the input file. The remainder details the results of the external optimization, each step in a MULTILEVEL optimization, and all energies, gradients, and Hessians calculated. All values should be clearly identified within the output.

7.C.2. The Electronic Structure Program Input and Output Files

Currently there are seven files created that fall under this category. Each will be described below. These files should indicate to the user which job is currently being run. A line is also written to standard error for each electronic structure job run by MULTILEVEL.

g94.inp, g98.inp, g03.inp, and g09.inp

These are the input files for GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, and GAUSSIAN09 respectively for all energy, gradient, and Hessian calculations made with the electronic structure package. They are rewritten for each call to the electronic structure program. The method and basis set are those required for the current MULTILEVEL calculation. The two options specified in every input file are *FChk=All* (in order to read the GAUSSIAN output) and *NoSymm* (in order to avoid calculation failures due to a changing of the molecular symmetry). The three types of calculations that may be specified in these files are *SP*, *Force*, and *Freq* to obtain the energy, gradient, and Hessian respectively. The option *Force=EnOnly* is always chosen for those methods that do not have analytic gradients, due to the fact that GAUSSIAN98 fails otherwise. All calculations employing

CCSD(T) specify it as *ccsd(t)=e4t* so that MP4SDTQ energies will be available in the formatted checkpoint file.

The user may specify additional options in the MULTILEVEL input file. However the user may not alter the Link0 command lines in the input files. The only Link0 command specified in these two files is the memory allocation line, and MULTILEVEL writes no other Link0 lines. The memory requirements are hardwired into the program, to the values of 100MB for energy calculations, 250MB for gradient calculations, and 500MB for both Hessian calculations and geometry optimizations. If these memory allocations are undesirable for some reason, the user may alter these constant amounts in the module FILES, so long as the amount is in a format accepted by GAUSSIAN. In addition, the only non-standard basis sets able to be used in any calculation are the three listed in 6.B.1. They should be named in the MULTILEVEL input file as pDZ+, G3large, and MG3; otherwise MULTILEVEL will not recognize the basis.

g94.out, g98.out, g03.out, and g09.out

These are the output files resulting from each GAUSSIAN run of *g94.inp*, *g98.inp, g03.inp,* and *g09inp*. As is true for the input files, these two files are overwritten with each run of the electronic structure package. They are not used in the running of MULTILEVEL. However, should the user encounter an error reading the checkpoint file, examination of these output files may prove useful in identifying the problem.

extopt.inp

This is the input file for the geometry optimization with an external electronic structure program. While this file and extopt.out (described later) are both technically going to be either GAUSSIAN94, GAUSSIAN98 GAUSSIAN03, or GAUSSIAN09 files, these two files have unique names so that they will not be overwritten, allowing the user to examine them at a later time. As was true for g94.inp, g98.inp, g03.inp, and g09.inp, the options *FChk=All* and *NoSymm* are always activated. And the memory allocation specified in Link0 is

500MB (although this may be altered as well in the module FILES). While for the other input files, the user input options are not necessary, the user *must* specify options for an external geometry optimization (*Opt* at the very minimum). Otherwise GAUSSIAN will not carry out an optimization.

extopt.out

Unlike the other output files from GAUSSIAN, MULTILEVEL does access this file to ensure that the optimization was successful before reading data from Test.FChk. But the user may want to examine the results of the geometry as well; thus this file will not be overwritten.

Test.FChk

This is the formatted checkpoint file created by either GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09. It is integral to the operation of MULTILEVEL in that all energies, gradients, Hessians, and optimized geometries output by the electronic structure packages are read from this file.

7.C.3. The Summary Output File: *ml.sum*

Whether a summary output file is printed is specified by the switch keyword PRSUM/NOPRSUM in the MULTIGEN section. The summary output file is created under the name ml.sum. It lists the final geometry, gradients and Hessians obtained from MULTILEVEL calculations.

7.D. The C Shell Script *run.ml*

While MULTILEVEL may be run by simply typing *multilevel.exe*, there is a fair amount of file bookkeeping that is necessary. The shuttles and basis set files must be in the current working directory. And in addition since MULTILEVEL's input and output always use the same filenames, sequential runs of MULTILEVEL require a large amount of file handling. Thus a sample script *run.ml* has been provided to illustrate these procedures.

Within this script the user *must* alter one line: the initial line of the *foreach* loop. The list that the *foreach* loop parses should contain the prefixes of each MULTILEVEL input file the user desires to run. Each of these input files should be named *prefix*.dat.

The usage of the script is *run.ml directory*, where *directory* is the location of the input files to be run. If this argument is omitted, the input files are assumed to be in the current working directory. With the script, the only file handling required of the user is to have the MULTILEVEL input files in the directory specified. The script then copies the shuttle scripts and the basis set files from the directory containing the MULTILEVEL executable to the current working directory. Then for each *prefix* listed in *run.ml*, *prefix*.dat is copied to *ml.inp* and MULTILEVEL is called. At the end of each run of MULTILEVEL the output files are transferred as follows:

$$\begin{aligned}
\textit{ml.out} &\Rightarrow \textit{prefix}.\text{out} \\
\textit{extopt.out} &\Rightarrow \textit{prefix}.\text{extopt} \\
\textit{g09.inp} &\Rightarrow \textit{prefix}.\text{g09inp} \\
\textit{g09.out} &\Rightarrow \textit{prefix}.\text{g09out} \\
\textit{g03.inp} &\Rightarrow \textit{prefix}.\text{g03inp} \\
\textit{g03.out} &\Rightarrow \textit{prefix}.\text{g03out} \\
\textit{g98.inp} &\Rightarrow \textit{prefix}.\text{g98inp} \\
\textit{g98.out} &\Rightarrow \textit{prefix}.\text{g98out} \\
\textit{g94.inp} &\Rightarrow \textit{prefix}.\text{g94inp} \\
\textit{g94.out} &\Rightarrow \textit{prefix}.\text{g94out}
\end{aligned}$$

Test.FChk ⇒ *prefix*.fchk

This way the user may observe the MULTILEVEL output, the external optimization output, and the final GAUSSIAN input, output, and formatted checkpoint files for each MULTILEVEL run.

There is also one additional file created by the script for each *prefix* in the list: *prefix*.jobs. This file contains all the standard error messages which include a list of all the electronic structure package jobs called. Also the final file line gives the system and user time spent on that MULTILEVEL run.

At the end of the script when all the MULTILEVEL calls have been made, all files are removed with the exception of those beginning with one of the *prefixes*.

The user may alter the handling of the input and output files but should keep in mind which files are required for the running of MULTILEVEL. *Run.ml* is provided as a template script which observes these considerations.

Chapter Eight

8

8. Installing and Using MULTILEVEL

A step-by-step procedure for installing MULTILEVEL on a Unix computer and testing it is given here. Compilation of the code can be accomplished with the script named *configure*. The test runs should illustrate the proper way in which to use this program.

8.A. Installation Instructions

Step 1:

The MULTILEVEL program should have been obtained in the tar format with the following file name: *multilevel4.4.tar.gz.* This file should be placed in the directory in which the user wishes to install MULTILEVEL, and then the following two commands should be executed:

tar –xzvf multilevel4.4.tar.gz

Once these two commands have been executed the directory structure on the next page should have been created in the directory in which MULTILEVEL was untarred. Please verify that this is true.

Step 2:

Verify that the files have been placed into the directory structure above as follows.
In the multilevel4.4 directory:

basis/ configure exe/ script/ src/ testo/ testrun/

In the src directory:

Makefile ef.F freq.F ghooks.F main.F module.F nebmin.F display.F ehooks.F gau_ext_opt.F hhooks.F ml.F neb.F ohooks.F

In the script directory:

Gau_External Gau_External_old ex_shuttle g09shuttle g03shuttle g94shuttle g98shuttle mlcompile mmol pg98shuttle

In the basis directory:

g3large.gbs	*g3xlarge.gbs*	*mg3.gbs*	*mg3s.gbs*
pdz+.gbs	*lanl2dz.gbs*		

In the testrun directory:

test_g94/	*test_g98/*	*test_g03/*	*test_g09/*

In the testrun/test_g94 directory:

test1.dat	*test1.ml*	*test2.dat*	*test2.ml*
test3.dat	*test3.ml*	*test4.dat*	*test4.ml*
test5.dat	*test5.ml*	*test6.dat*	*test6.ml*
test7.dat	*test7.ml*	*test8.dat*	*test8.ml*
test9.dat	*test9.ml*	*test10.dat*	*test10.ml*

In the testrun/test_g98 directory:

test1.dat	*test1.ml*	*test2.dat*	*test2.ml*
test3.dat	*test3.ml*	*test4.dat*	*test4.ml*
test5.dat	*test5.ml*	*test6.dat*	*test6.ml*
test7.dat	*test7.ml*	*test8.dat*	*test8.ml*
test9.dat	*test9.ml*	*test10.dat*	*test10.ml*
test11.dat	*test11.ml*	*test12dat*	*test12.ml*
test13.dat	*test13.ml*	*test14dat*	*test14.ml*
test15.dat	*test15.ml*	*test16dat*	*test16.ml*
test17.dat	*test17.ml*	*test18.dat*	*test18.ml*
test19.dat	*test19.ml*	*test20.dat*	*test20.ml*
test21.dat	*test21.ml*	*test22.dat*	*test22.ml*

test23.dat	*test23.ml*	*test24.dat*	*test24.ml*

In the testrun/test_g03 and testrun/test_g09 directory:

test1.dat	*test1.ml*	*test2.dat*	*test2.ml*
test3.dat	*test3.ml*	*test4.dat*	*test4.ml*
test5.dat	*test5.ml*	*test6.dat*	*test6.ml*
test7.dat	*test7.ml*	*test8.dat*	*test8.ml*
test9.dat	*test9.ml*	*test10.dat*	*test10.ml*
test11.dat	*test11.ml*	*test12dat*	*test12.ml*
test13.dat	*test13.ml*	*test14dat*	*test14.ml*
test15.dat	*test15.ml*	*test16dat*	*test16.ml*
test17.dat	*test17.ml*	*test18.dat*	*test18.ml*
test19.dat	*test19.ml*	*test20.dat*	*test20.ml*
test21.dat	*test21.ml*	*test22.dat*	*test22.ml*
test23.dat	*test23.ml*	*test24.dat*	*test24.ml*
test25.dat	*test25.ml*	*test26.dat*	*test26.ml*
test27.dat	*test27.ml*	*test28.dat*	*test28.ml*
test29.dat	*test29.ml*	*test28.img*	*test30.ml*

In the testo/g98, testo/g03, or testo/g09 directory:

test1.out	*test2.out*	*test3.out*	*test4.out*
test5.out	*test6.out*	*test7.out*	*test8.out*
test9.out	*test10.out*	*test11.out*	*test12.out*
test13.out	*test14.out*	*test15.out*	*test16.out*
test17.out	*test18.out*	*test19.out*	*test20.out*
test21.out	*test22.out*	*test23.out*	*test24.out*
test25.out	*test26.out*	*test27.out*	*test28.out*
test29.out	*test30.out*		

Step 3:

Change the working directory to the multilevel4.4 directory, and run the script *configure* by typing ./configure <Return>. This script will create a file in the home directory named *.multi_path* stating where the MULTILEVEL directory structure is located. This file is used by other scripts to locate MULTILEVEL on the user's system.

This script also attempts to find the system's f90 compiler and creates the Makefile. If a compiler is found, the script tells the Makefile to compile the program. If a compiler is not found, the user must fill in the correct compiler on the first line of the Makefile. Once the correct compiler is supplied, the code may be compiled by typing gmake MULTILEVEL <Return> from within the src/ directory.

Please note that the Makefile uses the timestamps on the files to determine if anything needs to be recompiled. If the user wishes to use a different compiler after source files have been compiled, the *touch* command should be used to update the timestamps on the source files before attempting to remake the executable.

Note: As MULTILEVEL is distributed, GAUSSIAN09 is the default electronic structure program for all PROGRAM keywords options. If only GAUSSIAN98 is available to the user, this default may be changed within the code now. To do so, edit the *module.F* file in the src directory, so that every 'g09 ' in the module PROGDEF becomes 'g98 '. Indeed, the distributed program can handle up to 50 atoms. For larger systems the value of the parameter 'maxatm' must be modified in all modules included in *module.F* file in the src directory. This can be easily done by executing the UNIX command:
sed -i 's/maxatm\ =\ 50/maxatm\ =\ 150/g' module.F

Step 4:

While still in the script directory, edit the *shuttle* scripts present, so that the paths indicated for electronic structure packages within these scripts are accurate for the

computer system on which MULTILEVEL has been installed. In addition, the user should specify the scratch directory to be used by GAUSSIAN by properly setting the scratchdir environmental variable in the script. Version g09.d01 is recommended if the Gaussian external options of testruns 25-27 are used.

In order to use the GAUSSIAN optimizer of GAUSSIAN09/03 the "./" must be added to the $PATH environment variable (e.g.: using the bash shell command: 'export PATH=$PATH:.'). It should be noted that in the $PATH environment variable must not contain any the location of any Gau_External executable.

8.B. The MULTILEVEL Test Suite

The test suite has been designed to give the user a sample of the MULTILEVEL capabilities and input files and to provide examples of test input and output. It does not give examples of everything that can be done with the program, but each test run demonstrates a key feature of the program. These test runs also allow the user to familiarize himself or herself with the MULTILEVEL output format.

In order to use the test suite, one must change the test run directory in the multilevel4.4 directory. Within this directory there are 4 subdirectories – test_g94, test_g98, test_g03, and test_g09. The *only* difference between the input files in these three directories is that they run with GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09. Each of these subdirectories contains MULTILEVEL input files.

While it is recommended that the user run all test runs, he or she may select certain portions of the test suite to run. This may be done by changing to either the testrun/test_g 09, testrun/test_g 03, testrun/test_g98 or the testrun/test_g94 directory and running the test#.ml script for the specific test run. In this case, the usage of the script for test3, for example, would be *test3.ml* <Return>.

A portion of the output that these test runs should produce can be found in the directory testo. This directory contains both the MULTILEVEL output files test#.out, the external optimization output file test1.extopt, and the test#.jobs files detailing the run time and the electronic structure jobs called by MULTILEVEL. For the most part, the output the user generates should be identical to that in the testo directory (whether running with GAUSSIAN94, GAUSSIAN98, GAUSSIAN03, or GAUSSIAN09). Some minor numerical differences may arise due to round-off issues, and the run time listed in *test#.jobs* files will in all likelihood be different. But most output values should have similar values between your test runs and the sample test run output supplied.

When examining the output in both testrun/test_g#/ and testo/, the file of most interest to the user will probably be the output file from MULTILEVEL. These files will be named *test#.out*. Examining the *test#.jobs* files will give the user a clearer view of all the calls made to the electronic structure program throughout a MULTILEVEL run.

8.B.1. Test 1

Test run 1 performs an MCCM-CO-CCSD. The coefficients used are v3s.

8.B.2. Test 2

Test run 2 is an MP2/cc-pVDZ gradient calculation of the water molecule using the TEST option in MULTILEVEL

8.B.3. Test 3

Test run 3 is an MCCM-CO-MP2 optimization of the OH molecule. HF/6-31G* Hessians are calculated every third step during the optimizations. The optimization uses the Newton-Raphson algorithm in MULTILEVEL. The coefficients used are the v3s coefficients.

8.B.4. Test 4

Test run 4 carries out MCSAC-CCSD/cc-pVDZ and MCG3 energy calculations for OH. The v3s coefficients are used for the MCG3 calculation, and user specified coefficients are used for the MCSAC calculation by using the COEFFS keyword. Since the NOCOOP keyword has been specified, only the MCSAC- CCSD/cc-pVDZ and MCG3 energies are calculated.

8.B.5. Test 5

Test run 5 performs a single-point HF||MO calculation of the water molecule using HF/3-21G and PM3. The fraction of the HF energy is given by FRACHF = 0.213.

8.B.6. Test 6

Test run 6 performs an MCCM-UT-CCSD geometry optimization of water utilizing the Newton-Raphson algorithm in MULTILEVEL, in which lower-level HF/6-31G* Hessians are calculated at every third optimization cycle. The coefficients used are v3s.

8.B.7. Test 7

Test run 7 performs a single-point energy calculation of ethanol using the IMOHC method with the low level being UFF and the high level being MP2/STO-3G.

8.B.8. Test 8

Test run 8 performs a single-point energy calculation of ethanol using the IMOHC method with AM1 as the low level and B3LYP/STO-3G as the high level.

8.B.9. Test 9

Test run 9 is a geometry optimization of the water molecule using the MC-QCISD method. The Newton-Raphson algorithm in MULTILEVEL is used, and HF/6-31G* Hessians are calculated every third optimization cycle. The coefficients for the MC-QCISD optimization are the version 2m coefficients.

8.B.10. Test 10

Test run 10 is a transition state optimization of the $H' + H_2 \rightarrow H'H + H$ reaction using the MC-QCISD method. The Newton-Raphson algorithm in MULTILEVEL is used, and HF/6-31G* Hessians are calculated every third optimization cycle. The coefficients for the MC-QCISD optimization are the version 2m coefficients.

8.B.11. Test 11

Test run 11 is an MCG3 single-point energy calculation on the $H' + H_2 \rightarrow H'H + H$ transition state. It runs GAUSSIAN in parallel by requesting two processors with Link1 commands through the G3OPTIONS keyword.

8.B.12. Test 12

Test run 12 is an MCG3 single point energy calculation on the H3 transition state. It runs MULTILEVEL in parallel mode. If MULTILEVEL has not been compiled with the parallel options, this test will run in serial mode. The script to run this test (*test12.ml*) is slightly different than the script for serial jobs. The script creates an extra subdirectory named par to work in.

8.B.13. Test 13

Test run 13 is a calculation of the energy, gradient, Hessian, vibrational frequencies, and normal modes of H_2 using MCOMP2 method with SRP coefficients. The input for this run is a geometry previously optimized for H_2O at the same level.

8.B.14. Test 14

Test run 14 is a calculation of the energy, gradient, Hessian, vibrational frequencies, and normal modes of H_2O for MCOMP2/3. The input for this run is a geometry previously optimized for H_2O at the MCOMP2/3 level.

8.B.15. Test 15

Test run 15 is a calculation of the energy, gradient, Hessian, vibrational frequencies, and normal modes of H_2O for MC-QCISD/3. The input for this run is a geometry previously optimized for H_2O at the MC-QCISD/3 level. (Test 9)

8.B.16. Test 16

Test run 16 is a calculation of the energy, gradient, Hessian, vibrational frequencies, and normal modes of OH for MCG3/3m. The input for this run is a geometry previously optimized for OH at the MCG3/3m level.

8.B.17. Test 17

Test run 17 is an energy calculation on OH with the new cho parameters for MC-QCISD.

8.B.18. Test 18

Test run 18 is a transition state optimization of the H′ + H_2 → H′H + H reaction using the MC-QCISD method. The Eigenvector Following (EF) algorithm in MULTILEVEL is used, and MC-QCISD Hessians are updated by the DFP scheme.

8.B.19. Test 19

Test run 19 is a transition state optimization of the H′ + H_2O→ H′H + OH reaction using the MC-QCISD method. The Eigenvector Following(EF) algorithm in MULTILEVEL is used, and MC-QCISD Hessians are updated by the DFP scheme.

8.B.20. Test 20

Test run 20 is a SAC/3 energy calculation on OH.

8.B.21. Test 21

Test run 21 is a MP2-SAC version 3m calculation on OH. MCOPTIONS controls the memory in GAUSSIAN09/03.

8.B.22. Test 22

Test run 22 is a MCCO/3m energy calculation on OH.

8.B.23. Test 23

Test run 23 is an MP2/MG3S-SAC calculation with a user-defined coefficient.

8.B.24. Test 24

Test run 24 is a MCG3/3, MCCO/3 and SAC/3 energy calculation on OH. The spin-orbit energy is defined as 0.23kcal/mol (0.0003665 Hartrees).

8.B.25. Test 25

Test run 25 is a G3SX(MP3) optimization of H_2 using the Berny optimizer in GAUSSIAN09/03.

8.B.26. Test 26

Test run 26 is a MC3BB optimization of the transition state of the H + H_2O reaction using the Berny optimizer in GAUSSIAN09/03 with keyword *calcfc*.

8.B.27. Test 27

Test run 27 is a MC3MPW optimization of the transition state of the OH + CH_4 reaction using the Berny optimizer in GAUSSIAN09/03.

8.B.28. Test 28

Test run 28 is an MPW1K NEB optimization of the OH + CH_4 reaction path using the following NEB options:

INITHESS	off
HSCALE	10
NIMG	11
NEBITER	20
INTERP	linear
IMGFILE	(external image file already built)
NEBALG	ci-neb
OPTM	bfgs

8.B.29. Test 29

Test run 29 is an MPW1K NEB optimization of the decomposition of CH_3SO_2 to CH_3 + SO_2, using the following NEB options:

INITHESS	off
HSCALE	10
NIMG	5
NEBITER	20
INTERP	linear
IMGFILE	(image file built by the program)
NEBALG	it-neb

OPTM vb

8.B.30. Test 30

Test run 30 is an MPW1K NEB optimization of the transition state of the decomposition of CH_3SO_2 to CH_3 + SO_2, using the options of test29 and the following basis sets: 6-31G(d) for C, H, and O and the LANL2DZ ECP for S.

8.C. Viewing MULTILEVEL Output

The script *mmol* will parse the output of MULTILEVEL geometry optimizations, so that they are viewable with the program MOLDEN. To use this script you must have a version of MOLDEN 3.6. A single line in the beginning of the script needs to be changed to the location of the MOLDEN executable. Example of script use:

mmol test6.out

8.D. Computers, Operating Systems, and FORTRAN Compilers on Which the Code Has Been Tested

In each case we give the MULTILEVEL version number and the computers and operating system on which MULTILEVEL was tested. For each computer and operating system we also specify the FORTRAN compiler that was used for testing.

MULTILEVEL4.2

Computer	Operating System	FORTRAN Compiler
IBM SP	AIX 4.3.3	XL Fortran for AIX
IBM Power4 Regatta	AIX 5.1 XL	Fortran for AIX
SGI Octane2	IRIX 6.4	MIPSpro 7
Netfinity Cluster	RedHat Linux 9.0	PGHPF® Workstation
SunBlade 1000	SunOS 5.8	Sun WorkShop 6

MULTILEVEL4.3

Computer	Operating System	FORTRAN Compiler
HP Proliant BL280C G6 Blade	CentOS 6.4	ifort/12.1
SGI Altix XE 1300	CentOS 6.3	ifort/12.1
SGI Altix UV1000	SUSE Linux Enterprise Server 11 SP1	ifort12.1
Sun Fire X4600 Linux cluster	CentOS 6.3	ifort/11.1

MULTILEVEL4.4

Computer	Operating System	FORTRAN Compiler
HP Proliant BL280C G6 Blade	CentOS 6.4	ifort/13.1.3
SGI Altix XE 1300	CentOS 6.3	ifort/13.1.3

SGI Altix UV1000	SUSE Linux Enterprise Server 11 SP2	ifort/13.1

Chapter Nine

9

9. Bibliography

[Ba91] J. Baker, *J. Comp. Chem.* **7** (1985), 385.

[Ch99] Y.-Y. Chuang, M. L. Radhakrishnan, P. L. Fast, C. J. Cramer, and D. G. Truhlar, *J. Phys. Chem. A* **103** (1999) 4893.

[Ci69] J. Cizek, *Adv. Chem. Phys.* **14** (1969) 35.

[Co96] E. L. Coitiño, D. G. Truhlar, and K. Morokuma, *Chem. Phys. Lett.* **259** (1996) 159.

[Co97] E. L. Coitiño and D. G. Truhlar, *J. Phys. Chem. A* **101** (1997) 4641.

[Co98] J. C. Corchado and D. G. Truhlar, *J. Phys. Chem. A* **102** (1998) 1895.

[Cu91] L. A. Curtiss, K. Raghavachari, G. W. Trucks, and J. A. Pople, *J. Chem. Phys.* **94** (1991) 7221.

[Cu92] P. Culot, G. Dive, V. H. Nguyen and J. M. Ghuysen, *Theor. Chim. Acta* **82** (1992), 189.

[Cu98] L. A. Curtiss, K. Raghavachari, P. C. Redfern, V. Rassolov, and J. A. Pople, *J. Chem. Phys.* **109** (1998) 7764.

[Cu99] L. A. Curtiss, private communication of parameterization mentioned on page 1129 of L. A. Curtiss, K. Raghavachari, P. Redfern, and J. A. Pople, *J. Chem. Phys*. **112** (2000) 1125.

[Cu01] L. A. Curtiss, P. C. Redfern, K. Raghavachari, and J. A. Pople, *J. Chem. Phys.* **114** (2001) 108.

[Du89] T. H. Dunning, Jr., *J. Chem. Phys.* **90** (1989) 1007.

[Fa99] P. L. Fast, J. Corchado, M. L. Sanchez, and D. G. Truhlar, *J. Phys. Chem. A* **103** (1999) 3139.

[Fa99a] P. L. Fast and D. G. Truhlar, *J. Phys. Chem. A* **103** (1999) 3802.

[Fa99b] P. L. Fast, J. C. Corchado, M. L. Sanchez, and D. G. Truhlar, *J. Phys. Chem. A* **103** (1999) 5129.

[Fa99c] P. L. Fast, M. L. Sanchez, J. C. Corchado, and D. G. Truhlar, *J. Chem. Phys.* **110** (1999) 11679.

[Fa99d] P. L. Fast, M. L. Sanchez, and D. G. Truhlar, *Chem. Phys. Lett.* **306** (1999) 407.

[Fa99e] P. L. Fast, M. L. Sanchez, and D. G. Truhlar, *J. Chem. Phys.* **111** (1999) 2921.

[Fa00] P. L. Fast and D. G. Truhlar, *J. Phys. Chem. A*. **104** (2000) 6111.

[Fa01] P. L. Fast, N. E. Schultz and D. G. Truhlar. *J. Phys. Chem. A*. **105** (2001) 4143.

[Fl87] R. Fletcher, Practical methods of optimization, Chichester ; New York: Wiley, c1987

[Go06] N. González-García, J. Pu, A. González-Lafont, J. M. Lluch , and D. G. Truhlar, *J. Chem. Theory Comput.* **2** (4) (2006) 895.

[Go86] M. S. Gordon and D. G. Truhlar, *J. Am. Chem. Soc.* **108** (1986) 5412.

[He00] G. Henkelman, and H. Jónsson, *J. Chem. Phys.* **113** (2000) 9978.

[He00a] G. Henkelman, B. P. Uberuaga, and H. Jónsson, *J. Chem. Phys.* **113** (2000) 9901.

[He86] W. J. Hehre, L. Radom, P. v. R. Schleyer, and J. A. Pople, *Ab Initio Molecular Orbital Theory*, Wiley, New York, 1986.

[Hu96] S. Humbel, S. Sieber, and K. Morokuma, *J. Chem. Phys.* **105** (1996) 1959.

[Ke92] R. A. Kendall, T. H. Dunning, Jr., *J. Chem. Phys.* **96** (1992) 6796.

[Kr78] R. Krishnan and J. A. Pople, *Int. J. Quant. Chem.* **14** (1978) 91.

[Kr80] R. Krishnan, M. J. Frisch, and J. A. Pople, *J. Chem. Phys.* **72** (1980) 4244.

[Ly02] B. J. Lynch and D. G. Truhlar, *J. Phys. Chem. A* **107** (19) (2003) 3898–3906.

[Ma95] F. Maseras and K. Morokuma, *J. Comput.. Chem.* **16** (1995) 1170.

[Mi94] G. Mills, and H. Jónsson, *Phys. Rev. Lett.* **72** (1994) 1124.

[Mi95] G. Mills, H. Jónsson, and G. K. Schenter, *Surf. Sci.* **324** (1995) 305.

[Mø34] C. Møller and M. S. Plesset, *Phys. Rev*. **46** (1934) 618.v

[No97] M. Noland, E. L. Coitiño, and D. G. Truhlar, *J. Phys. Chem. A* **101** (1997) 1193.

[Pa09] E. Papajak, H. R. Leverentz, J. Zheng and D. G. Truhlar *J. Chem. Theory Comput.*, 2009, **5** (5), (2009) 1197.

[Po54] J. A. Pople and R. K. Nesbet, *J. Chem. Phys.* **22** (1954) 571.

[Po71] M. S. D. Powell, *J. Inst. Maths. Applics.* **7** (1971) 21.

[Po77] J. A. Pople, R. Seeger, and R. Krishnan, *Int. J. Quant. Chem. Symp.* **11** (1977) 149.

[Po87] J. A. Pople, M. Head-Gordon, and K. Raghavachari, *J. Chem. Phys.* **87** (1987) 5968.

[Po89] J. A. Pople, M. Head-Gordon, and D. J. Fox, *J. Chem. Phys.* **90** (1989) 5622.

[Pr92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran77* (Cambridge University Press, Cambridge, 1992) 406.

[Pr92a] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran77* (Cambridge University Press, Cambridge, 1992) 418.

[Pu82] G. D. Purvis and R. J. Bartlett, *J. Chem. Phys.* **76** (1982) 1910.

[Ra89] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, *Chem. Phys. Lett.* **157** (1989) 479.

[Ro51] C. C. J. Roothaan, *Rev. Mod. Phys.* **23** (1951) 69.

[Ro00] J. M. Rodgers, P. L. Fast, and D. G. Truhlar, *J. Chem. Phys.* **112** (2000) 3141.

[Sc88] G. E. Scuseria, C. L. Jansen, and H. F. Schaefer, *J. Chem. Phys.* **89** (1988) 7382.

[Sc89] G. E. Scuseria and H. F. Schaefer, *J. Chem. Phys.* **90** (1989) 3700.

[St90] J. J. P. Stewart, *Rev. Comp. Chem.* **1** (1990) 45.

[Sv96] M. Svensson, S. Humbel, R. D. J. Froese, T. Matsubara, S. Sieber, and K. Morokuma, *J. Phys. Chem.* **100** (1996) 19375.

[Tr98] D. G. Truhlar, *Chem. Phys. Lett.* **294** (1998) 45.

[Tr99] C. M. Tratz, P. L. Fast, and D. G. Truhlar, *Phys .Chem. Comm.* **2** (1999) Article 14.

[Yp95] T. J. Ypma, *SIAM Rev*. **37** (1995) 531.

[Wo93] D. E. Woon and T. H. Dunning, Jr., *J. Chem. Phys.* **98** (1993) 1358.

[Wo94] D. E. Woon and T. H. Dunning, Jr., *J. Chem. Phys.* **100** (1994) 2975.

[Zh04] Y. Zhao, B. J. Lynch, and D. G. Truhlar, *J. Phys. Chem.* **108** (2004) 4786.

Chapter Ten

10

10. Revision History

10. A. Version 1.0

First distributed version

10. B. Version 1.5

1. The ANLsc [Cu99] version of MCG3 has been added.

10. C. Version 2.0

1. Versions of methods containing MP4D components were removed.
2. The VERSION keyword has been added.
3. The MP4/6-31G(2df,p) component of MCG3 has been removed. Therefore versions 1m, 1sc, and ANLsc of MCG3 are no longer available.

10. D. Version 2.0.1

1. Several bugs that were causing errors when interfacing with GAUSSIAN have been fixed.
2. A bug in the MC-QCISD hessian has been fixed
3. Several FORMAT statements have been fixed to prevent warnings with some compilers.

10. E. Version 2.1

1. The MCCM-CO-MP2; MG3; 6-31+G(d) method was given it's own keyword [Fa00].

10. F. Version 2.1.1

1. A bug that prevented calculations with charged species was fixed.

10. G. Version 2.2

1. MCG3 energies and gradients can be calculated in parallel when MULTILEVEL is compiled with OpenMP
2. The script *mmol* was added to interface with the MOLDEN GUI.
3. Memory and processor requirements for GAUSSIAN jobs can now be specified in MULTILEVEL input.
4. The maximum number of iterations for the triples calculation in QCISD(T) calculations has been increased to 100.

10. H. Version 2.3

1. A switch keyword PRSUM/NOPRSUM is added to MULTIGEN section to specify whether a summary output file ml.sum is printed. This summary file will be used by the direct dynamics calculation interface program MULTILEVELRATE.

10. I. Version 2.4

1. The version 3 coefficients and methods have been implemented for MCG3, MC-QCISD, and MCOMP2 methods. The CHO-SGP parameters have also been added for MCG3 and MC-QCISD.
2. The MG3S basis set is included in this distribution and can be used in any user-defined multilevel calculation.

10. J. Version 2.5

1. The EF optimizer has been added to the program.
2. MQCOPTIONS was changed to MCQCISDOPTIONS, and MCG3OPTIONS was added.
3. The MCQCISD energy is now calculated and printed in every MCG3 energy calculation.

10. K. Version 2.5.1

1. Changes were made to prevent 60 warnings on some compilers

10. L. Version 3.0

1. The SAC/3 and MCCO/3 methods have been added to the program.
2. Version 3s, 3m, and CHO coefficients have been added for most other MCCM methods.

10. M. Version 3.0.1

1. The coefficients in MCG3 v3s and v3m have been changed to the correct values. This error affected MCG3 v3s and v3m in versions 2.4–3.0 of MULTILEVEL.

10. N. Version 3.1

1. Harmonic vibrational frequencies and normal mode coordinates calculations were implemented in the program.
2. MULTILEVEL has been updated so that it now works with GAUSSIAN03 as well as with gaussian98 and gaussian94.

10. O. Version 4.0

1. MULTILEVEL-v4.0 can use the optimizers in the GAUSSIAN03 program to perform geometry optimization.
2. The G3SX(MP3), MC3BB, and MC3MPW methods have been added to the program.

10. P. Version 4.1

1. MULTILEVEL-v4.1 has the Nudged Elastic Band (NEB) method to perform global path minimizations for gas-phase systems.

10. Q. Version 4.1.1 (not released)

1. Minor bug fix in neb.F related to formatting output.
2. Scripts corrected to install correctly and display proper version information.
3. Added keyword ‘calcfc’ to test27 due to transition state optimization.

10. R. Version 4.2

Authors: Yan Zhao, Jocelyn M. Rodgers, Benjamin J. Lynch, Núria González-García, Patton L. Fast, Jingzhi Pu, Yao-Yuan Chuang, Benjamin A. Ellingson, and Donald G. Truhlar

1. A Makefile is now used for installation.
2. The location file has been changed to generic .multi_path for easy updating.
3. All system calls now include "./", which makes the code portable to environments that do not include the current working directory in the path.

10. S. Version 4.3

Authors: Yan Zhao, Jocelyn M. Rodgers, Benjamin J. Lynch, Núria González-García, Patton L. Fast, Jingzhi Pu, Yao-Yuan Chuang, Benjamin A. Ellingson, Rubén Meana-Pañeda, and Donald G. Truhlar

1. The code has been updated so that now it works with GAUSSIAN09 as well as GAUSSIAN03, GAUSSIAN98, and GAUSSIAN94.
2. Due to the fact that the default optimization algorithm in GAUSSIAN09 is different than that which was used in ealier versions, the optimization keywords 'opt=NoMicro' have been included in the test files test25.dat, test26.dat, and test27.dat in order to use the optimization algorithm used by the previous versions of MULTILEVEL.
3. Several FORMAT statements have been fixed to prevent warnings with some compilers, and now it is possible to compile the code with the GNU Fortran compiler (gfortran).
4. A bug corresponding to the value of the percentage *X* of HF exchange in the MPW*X* density functional has been corrected.
5. The version of the code and the basis sets are now properly displayed in the output file.
6. Errors in two of the input files and in one of the output files have been fixed. In test12.dat the version number has been fixed. In test20.dat the coefficient and basis set have been fixed. The output file test14.out has been corrected according to the options that appear in the manual.

7. The csh –f option has been added to all the csh scripts to avoid reading the *.chsrc* file of the user home directory.

10.S. Version 4.4

Authors: Yan Zhao, Jocelyn M. Rodgers, Benjamin J. Lynch, Núria González-García, Patton L. Fast, Jingzhi Pu, Yao-Yuan Chuang, Benjamin A. Ellingson, Rubén Meana-Pañeda, and Donald G. Truhlar

1. The keyword **GENECP** that allows the use of ECPs has been added. The code includes now a new test file for the use of ECPs in a NEB optimization and its corresponding file in the *basis* subdirectory.

_____

End of Manual