Westinghouse

Our ref: **LTR-LIS-11-78**
Revision 0

To:	Aaron M. Everhard	Author's Date:	**Feb 3, 2011**
Cc:	K. W. Bonadio		J. Ghergurovich
	R. J. Brown		K. S. Howe
	L. Cao		J. R. Kobelak
	D. M. Crytzer		K. M. Koller
	D. C. DiBasilio		E. A. McGrew
	T. A. Downs		M. E. Nissley
	R. J. Espinosa		R. P. Rossman
	R. L. Fittante		J. A. Rozum
	C. Frepoli		R. R. Schoff
	A. F. Gagnon		M. E. Sheaffer
From:	LOCA Integrated Services I		
Telephone:	412-374-4524 (J.J.Besspiata)		412-374-5196 (K.Ohkawa)
	412-374-5172 (J.Liao)		412-374-2102 (A.J.Colussy)

Subject: **Recommended ifort Compiler Options on Linux**

References:

[1] Karpinski, R., “Paranoia: A Floating-Point Benchmark,” Byte Magazine, Vol.10, No.2, pp. 223-235, 1985.
[2] CN-FSLOCA-09-12, Revision 0, "Software Change Specification and Validation for WCOBRA/TRAC-TF2 Version 1.0-T5 Pre-Linux Updates and Linux Conversion to Create WCOBRA/TRAC-TF2 Version 1.1-T1," Nov 22, 2010.
[3] Intel® Fortran Compiler 11.1 User and Reference Guides, No. 304970-006US, Intel Corporation, 2009.
[4] CCF-102493, “Paranoia Floating-Point Benchmark,” a copy of pdf file is attached.
[5] Monniaux, D., “The Pitfalls of Verifying Floating-Point Computations,” ACM Transactions on Programming Languages and Systems, Vol.30 (3), Article No.12, 2008.

Purpose:

The purpose of this memo is to document the FORTRAN compiler options recommended for the migration of WCOBRA/TRAC-TF2 and similar codes to the Linux/ifort environment.

Motivation:

During the development of the WCOBRA/TRAC-TF2 Version 1.1 computer program (WC/T-TF2) it was observed that a change to add print statements in a single routine caused differences in the overall results of the program. This change should have had no impact on the calculated results. It was assumed, based on prior experience, that the added print statements changed the conditions enough to cause the compiler to optimize that coding differently, or not at all. Additionally, the compliance with the IEEE (Institute of Electrical and Electronics Engineers, Inc.) floating point arithmetic standard was tested with the use of Paranoia program [1] obtained from the Oak Ridge National Laboratory netlib.org site (*http://www.netlib.org/paranoia/dpara.f*). The results with the compiler options selected for the T-configuration version of WC/T-TF2 were not satisfactory.

Current Compiler Options:

During the conversion of WC/T-TF2 from HP (Hewlett-Packard) f90 compiler to Linux ifort compiler [2] a set of general compiler options were selected for ifort:

-align all -assume nounderscore -IPF-fltacc -mp1 -save -traceback -O1

"-O1" selects the first level of advanced optimization to override the less conservative default level of "-O2". "-mp1" option is described in Ref. [3] as providing good accuracy and run-time performance at the expense of less consistent floating-point results. This set of compiler options was further justified by the favorable comparison to the WC/T-TF2 results obtained on HP11.23 version.
The current versions of the ifort compiler and Linux system are as follows:

ifort compiler:	Intel(R) Fortran Intel(R) 64 Compiler Professional for applications running on Intel(R) 64, Version 11.1 Build 20090630 Package ID: l_cprof_p_11.1.046
Linux system:	Linux susedev1 2.6.16.60-0.21-smp #1 SMP Tue May 6 12:41:02 UTC 2008 x86_64 x86_64 x86_64 GNU/Linux

Methodology:

A set of compiler options was sought which satisfies the speed requirement necessary for the Monte Carlo process used by ASTRUM-FS, exhibits the consistent behavior between the debug and optimized versions, and complies with the IEEE floating point arithmetic standard as judged by the Paranoia program. To obtain reliable ifort compiler options for WC/T-TF2 numerous computer runs were made trying different optimization levels and options related to the precision of the floating point calculations.
A relevant subset of these runs is presented in the Attachments:

- Attachment A provides the details of the WC/T-TF2 computer run results.
- Attachment B includes results from the Paranoia computer program compiled with various options to test floating-point behavior.

Conclusions:

Following lists some of the findings of pertinent compiler flags:

- Use of the default "-pc80" and the "-mp1" precision option yields WC/T-TF2 calculated results that change with each optimization level tested (-O0 (with and without "-g"), -O1, and -O2), and fails the Paranoia test.
- The options "–mp1" or "-mp" are considered obsolete and shall be replaced by option "-fp-model <arg>" per Intel Fortran Compiler User Guides [3]. To maintain the numerical precision of a computer program, the option of "-fp-model precise" is recommended.
- With the "-pc64" and "-fp-model precise" options used, and "-mp1" removed, the WC/T-TF2 calculated results are identical at the optimization levels tested (-O0 (with and without "-g"), -O1, and -O2). In addition, the results are identical with and without added print statements.
- With the "-pc64" and "-fp-model precise" options used, Paranoia test result is excellent at the optimization levels tested (-O0 (with and without "-g"), -O1, -O2 and –O3).

Recommendations:

Finally, the recommended compiler options for WC/T-TF2 and similar computer codes are given below, which generates identical numerical results between the optimization levels, and the debug version:

1. The following options are recommended for the optimized WC/T-TF2 version:
   -align all -assume nounderscore -IPF-fltacc -fp-model precise -pc64 -save -traceback -O2
2. The following options are recommended for the debug WC/T-TF2 version:
   -align all -assume nounderscore -IPF-fltacc -fp-model precise -pc64 -save -traceback -O0 -g
3. It is recommended to repeat the Paranoia test before future adjustment of the compiler options related to floating point. The double precision version of Paranoia Fortran program can be obtained from the configuration control (CCF-102493 - Paranoia Version 1.0 [4]).
4. It is recommended that three ifort compiler default options be changed as follows:
   - "-pc64", which sets internal floating-point unit (FPU) precision to 53 bit significand and conforms to the double precision defined in IEEE 754-1985 standard. This should replace the current "-pc80" default.
   - "-O0", which performs minimal optimization. This should replace the current "-O2" default. For other computer programs, it is advisable that a similar optimization study be performed to justify use of a higher optimization level than "-O0".
   - "–fp-model precise", to maintain floating point precision.

For additional information contact the undersigned.

Author[1]: *(Electronically Approved)*
John J. Besspiata
LOCA Integrated Services I

Author[2]: *(Electronically Approved)*
Jun Liao
LOCA Integrated Services I

1. John J. Besspiata authored main body and Attachment A.
2. Jun Liao co-authored main body and authored Attachment B.

Reviewer: *(Electronically Approved)*
Katsuhiro Ohkawa
LOCA Integrated Services I

Approval: *(Electronically Approved)*
Amy J. Colussy
Manager, LOCA Integrated Services I

Attachments: A (pages 4 through 10)
B (pages 11 through 12)

(NOTE: Files are attached to this document in the electronic document management system, as listed in Attachments A and B.)

Attachment A – WC/T-TF2 Results

The results of computer runs made using different WC/T-TF2 executable versions are presented in this attachment.

These versions are built from WC/T-TF2 Version 1.1 T4. The WC/T-TF2 versions used, and the logs of each executable creation, are as follows in Table A-1.
(Compiler Option "-O2" also represents the default value, when no optimization level is explicitly used; and WC/T-TF2 Version containing 01 is base, 02 has added prints.)

Table A-1 - WCT-TF2 Versions, Compiler Options, and Attached Text Log Files

WCT-TF2 Version	*Compiler Options*	*Attached Text Log File*
1.1T4+JJB_01	-O1 -mp1	wx.sh_LOG_1.1T4+JJB_01_1295465333_LTR-LIS-11-78
1.1T4JJB01CONTAIN	-O1 -mp1, except contain.f: -O0 -mp1 -g	wx.sh_LOG_1.1T4JJB01CONTAIN_1295465344_LTR-LIS-11-78
1.1T4+JJB_01_OZERO	-O0 -mp1	wx.sh_LOG_1.1T4+JJB_01_OZERO_1295465351_LTR-LIS-11-78
1.1T4+JJB_01_NOOPT	-mp1	wx.sh_LOG_1.1T4+JJB_01_NOOPT_1295465358_LTR-LIS-11-78
1.1T4JJB01fpreciseO0	-O0 -fp-model precise -pc64	wx.sh_LOG_1.1T4JJB01fpreciseO0_1295465365_LTR-LIS-11-78
1.1T4JJB01fprecise	-O1 -fp-model precise -pc64	wx.sh_LOG_1.1T4JJB01fprecise_1295465373_LTR-LIS-11-78
1.1T4JJB01fpreciseO2	-O2 -fp-model precise -pc64	wx.sh_LOG_1.1T4JJB01fpreciseO2_1295465380_LTR-LIS-11-78
1.1T4JJB01fpO0g	-O0 -fp-model precise -pc64 -g	wx.sh_LOG_1.1T4JJB01fpreciseO0g_1296146954_LTR-LIS-11-78
1.1T4+JJB_02	-O1 -mp1	wx.sh_LOG_1.1T4+JJB_02_1295465392_LTR-LIS-11-78
1.1T4JJB02CONTAIN	-O1 -mp1, except contain.f: -O0 -mp1 -g	wx.sh_LOG_1.1T4JJB02CONTAIN_1295465402_LTR-LIS-11-78
1.1T4+JJB_02_OZERO	-O0 -mp1	wx.sh_LOG_1.1T4+JJB_02_OZERO_1295465409_LTR-LIS-11-78
1.1T4+JJB_02_NOOPT	-mp1	wx.sh_LOG_1.1T4+JJB_02_NOOPT_1295465417_LTR-LIS-11-78
1.1T4JJB02fprecise	-O1 -fp-model precise -pc64	wx.sh_LOG_1.1T4JJB02fprecise_1295465424_LTR-LIS-11-78

Table A-2 presents the WC/T-TF2 results that were obtained by executing the above program versions. The number of time steps (transient stdout final MAJOR EDIT) is used as a check that the calculated results are the same, and is an adequate test for this 650 second run: if the steps differ, then results differ. The system was empty when these runs were made, so the CPU seconds (central processing unit seconds; transient stdout CPU TIME at END OF PROBLEM) presented are appropriate for comparison.

Table A-3(1) and Table A-3(2) provide lists of the WC/T-TF2 stdout files corresponding to the Table A-2 results.

Table A-2 – WC/T-TF2 Results Using Different Compiler Options

Tested Compiler Options	***Number of Time Steps for 650 second Transient***		***CPU Seconds***	
	(1) Base	***(2) Prints in contain.f***	***(1)***	***(2)***
-O1 -mp1	**338802**	**340268**	11674	11598
-O1 -mp1 all; -O0 -mp1 -g contain.f	**340268**	**340268**	11574	11592
-O0 -mp1	**338825**	**338825**	25876	26116
-mp1	339269	339269	10548	10594
-O0 -fp-model precise -pc64	**342515**	N/A	26348	N/A
-O1 -fp-model precise -pc64	**342515**	**342515**	12056	12015
-O2 -fp-model precise -pc64	**342515**	N/A	11596	N/A
-O0 -fp-model precise -pc64 -g	**342515**	N/A	26659	N/A

These results demonstrate that the ifort compiler on the Nuclear Services Linux system:

- does not yield consistent results at different optimization levels when the current "-mp1" option and the "-pc80" default are used
- does give identical results at the -O0 (with and without "-g"), -O1, and -O2 optimization levels when the recommended "-fp-model precise" and "-pc64" options are used

Execute "ifort -V" to see the ifort compiler version:
Intel(R) Fortran Intel(R) 64 Compiler Professional for applications running on Intel(R) 64, Version 11.1 Build 20090630 Package ID: l_cprof_p_11.1.046

Execute "uname -a" to see the Linux system information:
Linux susedev1 2.6.16.60-0.21-smp #1 SMP Tue May 6 12:41:02 UTC 2008 x86_64 x86_64 x86_64 GNU/Linux

A comparison plot of the peak cladding temperature (PCT) results obtained from the computer runs with the "-mp1" option and the "-pc80" default demonstrates that the -O0, -O1, and default (-O2) optimizations yield different results (PCT of 1856, 1848, and 1895 F, respectively), as shown in Figure A-1.

Similarly, a comparison plot of the PCT results obtained from the computer runs with the "-fp-model precise" and "-pc64" options demonstrates that the -O0, -O1, and -O2 optimizations yield identical results (PCT of 1874 F), as shown in Figure A-2. Based on this comparison plot, and to maintain job turnaround, it is recommended that the -O2 optimization level be used for future WC/T-TF2 compilations. The time for compilation is about 4 minutes for both -O1 and -O2.

Figure A-3 provides a comparison between the results obtained from the current compiler option setup and those from the new recommended setup (PCT of 1848 and 1874 F, respectively).

The previous plots are created by using the NSAPLOT version 5.0.4 T5 computer program on Linux. The NSAPLOT execution log and trail files are attached as the following text files:

xp_log_345492266_LTR-LIS-11-78
trail_30091.txt_1295540129_LTR-LIS-11-78

In addition to the time-step and plot comparisons, the diff utility program was used on Linux to compare the various stdout files to support the conclusions.

The following conclusions related to the ifort compiler have been made based on these results:

- Use of the default “-pc80” and the “-mp1” precision option yields WC/T-TF2 calculated results which change with each optimization level tested (-O0, -O1, and -O2).
- The default optimization level is “-O2”. It would be prudent to have the default be “-O0”, which turns off all optimization.
- The default significand option for FPU is “-pc80” does not conform to the IEEE standard. It would be wise to have the default be “-pc64” which is consistent with the IEEE standard.
- The options “–mp1” or “-mp” are considered obsolete and shall be replaced by option “-fp-model <arg>” [2]. To maintain the numerical precision of a computer program, the option of “-fp-model precise” is recommended.
- With the “-pc64” and "-fp-model precise" options used, and “-mp1” removed, the program calculated results are identical at the optimization levels tested (-O0 (with and without "-g"), -O1, and -O2). In addition, the program results are identical with and without added print statements.

Table A-3(1) – WC/T-TF2 Attached stdout Text Files for Base (1) Versions

Tested Compiler Options	*Text Files Attached for Table A-2 Base (1) Code Versions Results*
-O1 -mp1	SS_O1+mp1_base_stdout_1295533022_LTR-LIS-11-78 TR_O1+mp1_base_stdout_1295533030_LTR-LIS-11-78
-O1 -mp1 all; -O0 -mp1 -g contain.f	SS_O1+mp1_gcontain_base_stdout_1295533038_LTR-LIS-11-78 TR_O1+mp1_gcontain_base_stdout_1295533046_LTR-LIS-11-78
-O0 -mp1	SS_O0+mp1_base_stdout_1295533055_LTR-LIS-11-78 TR_O0+mp1_base_stdout_1295533063_LTR-LIS-11-78
-mp1	SS_mp1_base_stdout_1295533071_LTR-LIS-11-78 TR_mp1_base_stdout_1295533080_LTR-LIS-11-78
-O0 -fp-model precise -pc64	SS_O0+frpecise+pc64_base_stdout_1295533088_LTR-LIS-11-78 TR_O0+frpecise+pc64_base_stdout_1295533097_LTR-LIS-11-78
-O1 -fp-model precise -pc64	SS_O1+frpecise+pc64_base_stdout_1295533105_LTR-LIS-11-78 TR_O1+frpecise+pc64_base_stdout_1295533113_LTR-LIS-11-78
-O2 -fp-model precise -pc64	SS_O2+frpecise+pc64_base_stdout_1295533122_LTR-LIS-11-78 TR_O2+frpecise+pc64_base_stdout_1295533130_LTR-LIS-11-78
-O0 -fp-model precise -pc64 -g	SS_O0g+frpecise+pc64_base_stdout_1296146961_LTR-LIS-11-78 TR_O0g+frpecise+pc64_base_stdout_1296146970_LTR-LIS-11-78

Table A-3(2) – WC/T-TF2 Attached stdout Text Files for Prints in contain.f. (2) Versions

Tested Compiler Options	*Text Files Attached for Table A-2 Prints in contain.f (2) Code Versions Results*
-O1 -mp1	SS_O1+mp1_prints_stdout_1295537023_LTR-LIS-11-78 TR_O1+mp1_prints_stdout_1295537031_LTR-LIS-11-78
-O1 -mp1 all; -O0 -mp1 -g contain.f	SS_O1+mp1_gcontain_prints_stdout_1295537040_LTR-LIS-11-78 TR_O1+mp1_gcontain_prints_stdout_1295537048_LTR-LIS-11-78
-O0 -mp1	SS_O0+mp1_prints_stdout_1295537056_LTR-LIS-11-78 TR_O0+mp1_prints_stdout_1295537064_LTR-LIS-11-78
-mp1	SS_mp1_prints_stdout_1295537072_LTR-LIS-11-78 TR_mp1_prints_stdout_1295537081_LTR-LIS-11-78
-O0 -fp-model precise -pc64	N/A
-O1 -fp-model precise -pc64	SS_O1+frpecise+pc64_prints_stdout_1295537089_LTR-LIS-11-78 TR_O1+frpecise+pc64_prints_stdout_1295537097_LTR-LIS-11-78
-O2 -fp-model precise -pc64	N/A
-O0 -fp-model precise -pc64 -g	N/A

Figure A-1 - Comparison of Optimization Level Results With “-mp1” and “-pc80”

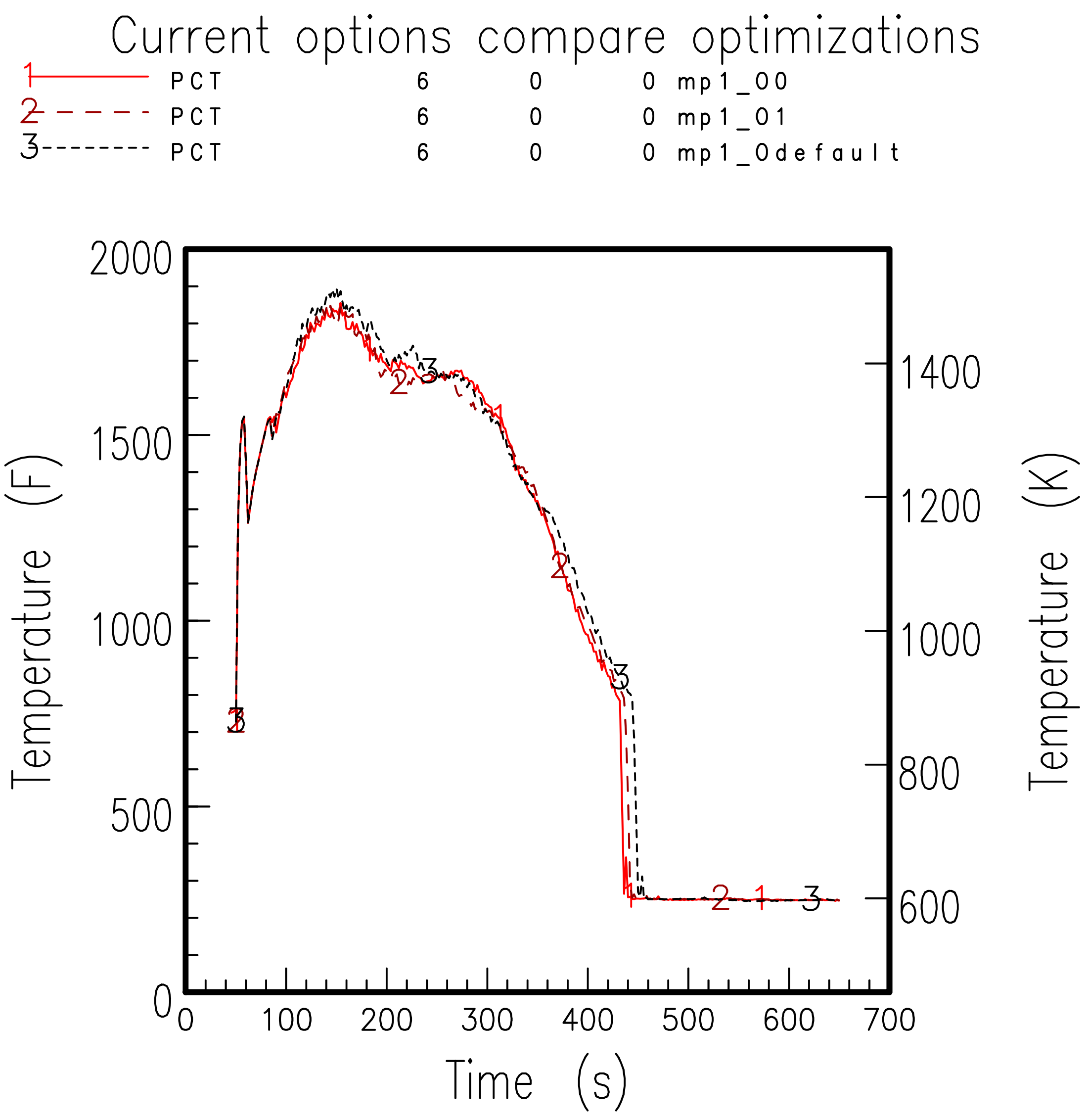


Figure A-2 - Comparison of Optimization Level Results With “-fp-model precise” and “-pc64”

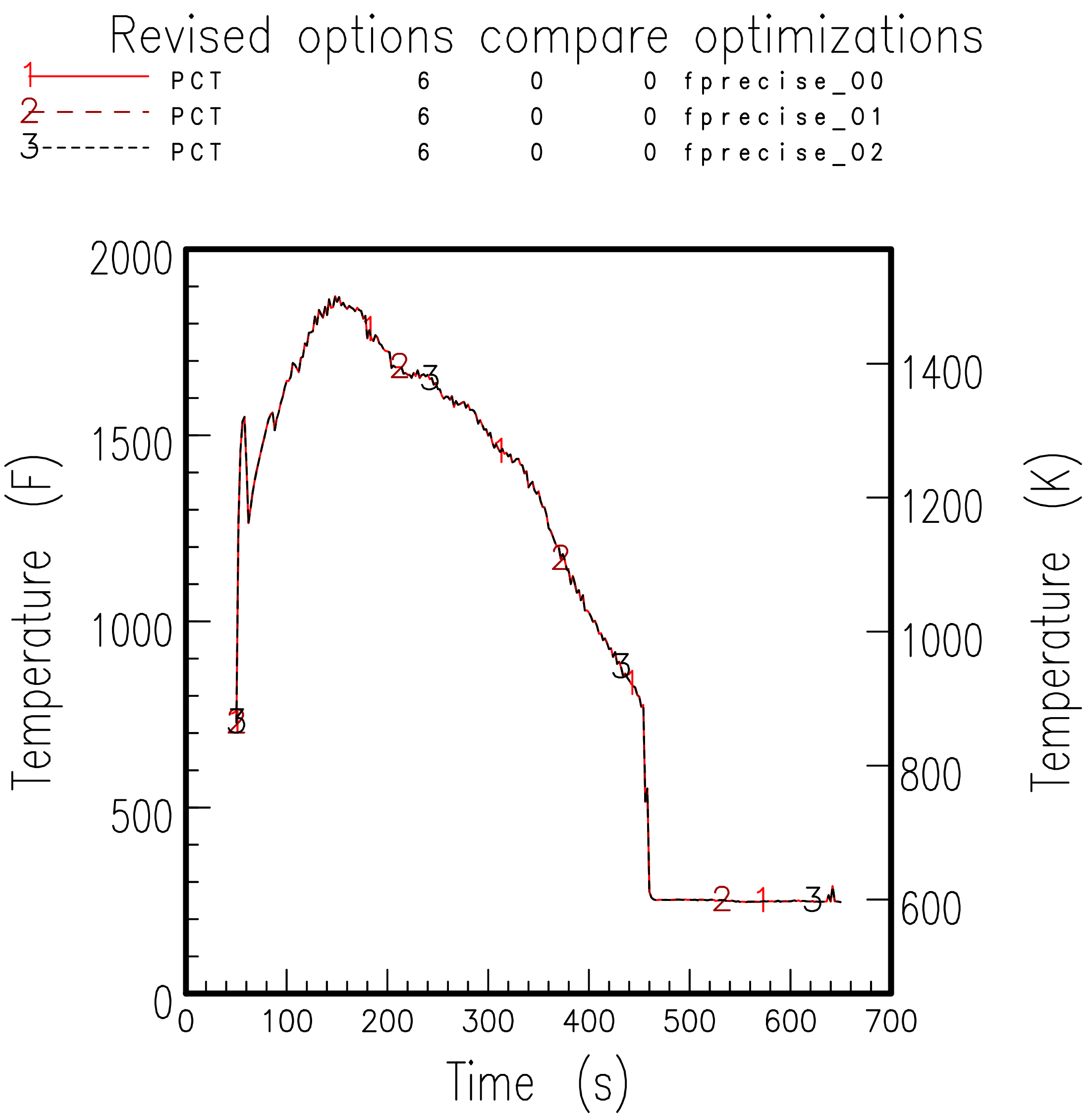
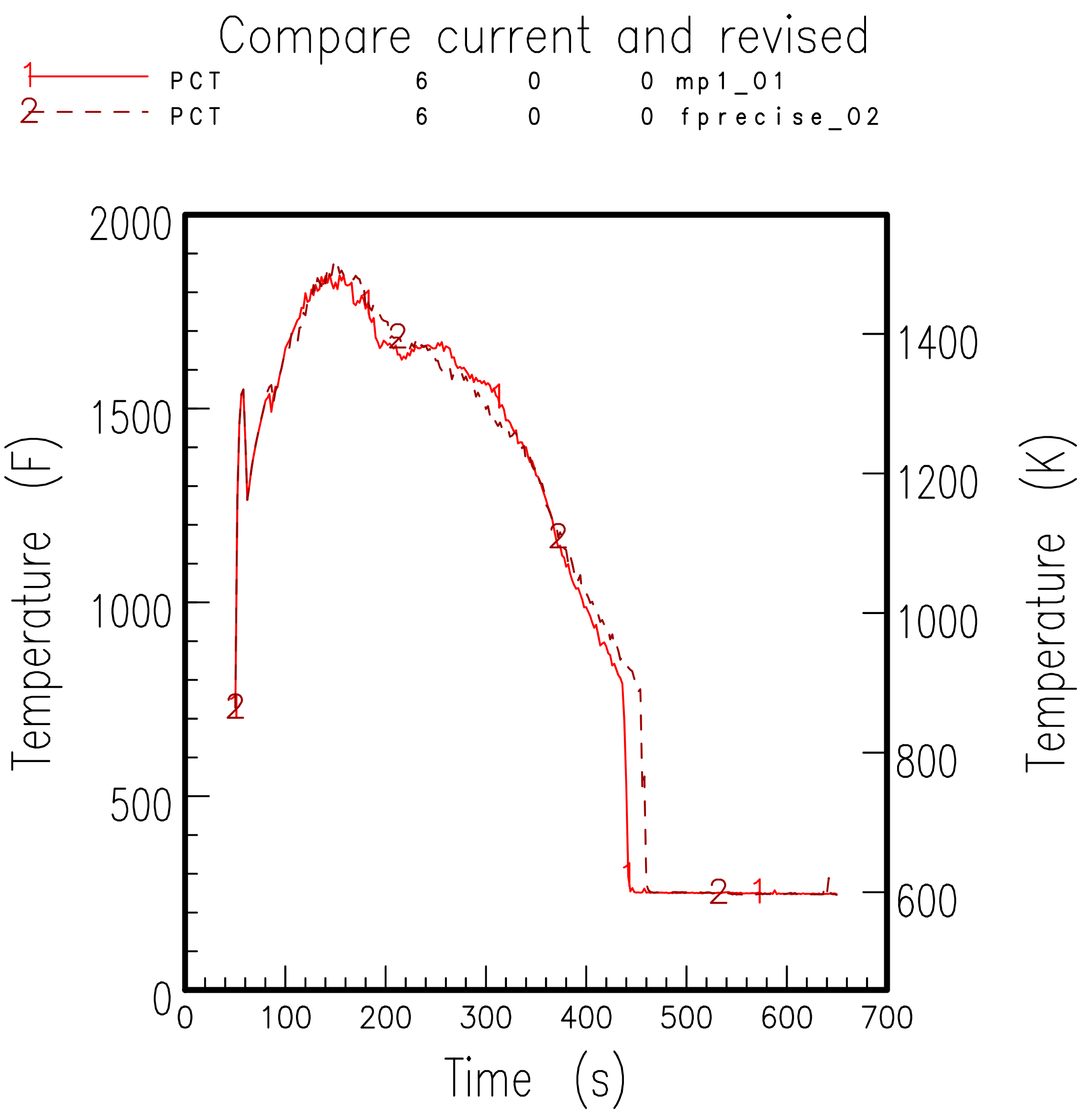


Figure A-3 - Comparison of Results Using Current and Recommended Options



Attachment B - Paranoia Results

Paranoia (Ref. [1]) is the name for a program written by William M. Kahan in the early 80's. It was designed to characterize floating-point behavior of computer systems. The source code is obtained from NETLIB repository at the Oak Ridge National Lab. The Fortran version of Paranoia program is downloadable from following URL:

http://www.netlib.org/paranoia/dpara.f

The source code is captured inside Westinghouse with configuration control form in Ref. [4]. The CCF is not released when the letter is signed off, but a copy of CCF is attached to EDMS with the file name of

CCF-102493.pdf_1296743530_LTR-LIS-11-78

A copy of the source code dpara.f is also attached to EDMS with the file name of

dpara.f_1295623853_LTR-LIS-11-78

The source code is compiled on GNU/Linux 2.6 using ifort compiler. Following command on Linux creates an executable program dpara.x. The options in the brace are the compiler flags to be studied (see Table B-1).

ifort [options] dpara.f –o dpara.x

Once the program dpara.x is executed on GNU/Linux 2.6, please press "0" key followed by pressing "enter" key exactly 13 times. The results of Paranoia tests will be shown on screen.

The compiler options explored in this study are captured in Table B-1 together with the summary of test results. The Paranoia test with option "-O0", which is the lowest level of optimization, shows one flaw. With the higher level of optimization "-O1", 1 failure, 2 serious defects, 1 defect, and 3 flaws are discovered. The option of "O2" leads to more failures. In general, none of these options produces result in compliance with IEEE standard 754-1985.

In origin, the numerical problem was noticed after the WC/T-TF2 computer program was migrated from the HP-UX operation system to the Linux operation system. It is noted in Nuclear Services the Linux operation system runs on Intel processors and WC/T-TF2 is compiled using Intel Fortran compiler ifort. Thus, the problem is related to Intel architecture and Intel Fortran compiler ifort. This Paranoia test confirmed that the computational program complied with the default options of the ifort compiler on Linux does not conform to IEEE standard 754-1985.

The non-compliance is likely attributed to numeric pitfalls in Intel architectures (Ref. [5]). For example, the Intel FPU (floating-point unit), also called x87 processor, has 80-bit registers in "double extended" format by default, while the floating point data in WC/T-TF2 are uniformly defined as double precision, which is 64-bit. The inconsistency between 64-bit floating point data and 80-bit FPU sometimes is problematic for computations relying on exact FPU precision for correct operation. The details of Intel FPU can be found on Wikipedia webpage, http://en.wikipedia.org/wiki/X87. The compiler option of "-pc80" is the default option of the Intel compiler which sets the internal FPU precision to 80-bit [2], but it is desired to be 64-bit for floating point.

Historically, there are noticeable amount of comparisons between two double precision floating points in subroutines of WC/T-TF2, which relies on precise FPU operations. Consequently, WC/T-TF2 computation is sensitive to the compiler options. To prevent the numerical pitfalls and find the appropriate compiler options for WC/T-TF2, the compiler options of ifort related to floating point are investigated.

The study in the main body of this letter suggests using options "-fp-model precise" and "-pc64" to avoid the numerical pitfalls. The option of "-fp-model precise -pc64" leads to numerical results conform to IEEE standard with the optimization levels from "-O0" to "-O3" as shown in Table B-1.In addition, the debug option "-g" together with option "-fp-model precise -pc64" is tested. Note, the optimization option is set to"-O0" for the debug option. The results show that the debug version of Paranoia conforms to IEEE standard. Finally, the recommended compiler options for the WC/T-TF2 computer program, "-align all -assume nounderscore -IPF-fltacc -fp-model precise -pc64 -save -traceback -O2", also passed the Paranoia test.

In summary, with the "-pc64" and "-fp-model precise" options used, Paranoia test result is excellent at the optimization levels tested (-O0, -O1, -O2 and –O3), in the debug version (-O0 –g), or with the recommended compiler options for the WC/T-TF2 computer program, "-align all -assume nounderscore -IPF-fltacc -fp-model precise -pc64 -save -traceback -O2".

Table B-1 – Summary of Paranoia Results

Tested Compiler Options	***Summary of Paranoia Tests***
-O0	The number of FLAWs discovered = 1 The arithmetic diagnosed seems Satisfactory though flawed.
-O1	The number of FAILUREs encountered = 1 The number of SERIOUS DEFECTs discovered = 2 The number of DEFECTs discovered = 1 The number of FLAWs discovered = 3 The arithmetic diagnosed has unacceptable Serious Defects. Potentially fatal FAILURE may have spoiled this program's subsequent diagnoses.
-O2	The number of FAILUREs encountered = 2 The number of SERIOUS DEFECTs discovered = 2 The number of DEFECTs discovered = 1 The number of FLAWs discovered = 3 The arithmetic diagnosed has unacceptable Serious Defects. Potentially fatal FAILURE may have spoiled this program's subsequent diagnoses.
-O0 -fp-model precise -pc64	No failures, defects nor flaws have been discovered. Rounding appears to conform to the proposed IEEE standard P754 The arithmetic diagnosed appears to be Excellent!
-O1 -fp-model precise -pc64	No failures, defects nor flaws have been discovered. Rounding appears to conform to the proposed IEEE standard P754 The arithmetic diagnosed appears to be Excellent!
-O2 -fp-model precise -pc64	No failures, defects nor flaws have been discovered. Rounding appears to conform to the proposed IEEE standard P754 The arithmetic diagnosed appears to be Excellent!
-O3 -fp-model precise -pc64	No failures, defects nor flaws have been discovered. Rounding appears to conform to the proposed IEEE standard P754 The arithmetic diagnosed appears to be Excellent!
-g –O0 -fp-model precise -pc64	No failures, defects nor flaws have been discovered. Rounding appears to conform to the proposed IEEE standard P754 The arithmetic diagnosed appears to be Excellent!
-align all -assume nounderscore -IPF-fltacc -fp-model precise –pc64 -save -traceback -O2	No failures, defects nor flaws have been discovered. Rounding appears to conform to the proposed IEEE standard P754 The arithmetic diagnosed appears to be Excellent!