JNC TJ9440 2000-004

動的信頼性解析プログラムDYANAの改良

（核燃料サイクル開発機構　契約業務報告書）

2000年3月

株式会社CRC総合研究所

JNC TJ9440 2000-004
２０００年３月

動的信頼性解析プログラムDYANAの改良

（核燃料サイクル開発機構　契約業務報告書）

田村一雄*
入谷佳一*

要　旨

確率論的安全評価において、事故シーケンスの発生頻度を求めるために、フォールトツリー/イベントツリー手法が広く使われている。しかし、従来の手法では、運転現場において実際に運転員が対峙している事象推移をダイナミックに取り扱うことができない。

そこで、FBRを対象とした、緊急時運転手順操作とプラントの間のダイナミックな相互作用を扱う動的解析プログラム(DYANA)を作成した。

これまでの開発の中で基本的な解析モデルは固まりつつあるが、計算時間の短縮が課題となっている。

今回作業では計算時間を短縮するためにMPIを用いてDYANAの並列化を実施し、WSクラスタ上でほぼ理想値に近い並列化性能を実現した。

本報告書は、株式会社CRC総合研究所が核燃料サイクル開発機構との契約により実施した業務成果に関するものである。

機構担当部課室：大洗工学センター　システム技術開発部　リスク評価技術開発グループ
＊株式会社CRC総合研究所　エネルギー技術部第２チーム

JNC TJ9440 2000-004
March,2000

Improvement of DYANA —the dynamic analysis program for event transition—.

Kazuo Tamura*
Yoshikazu Iriya*

Abstract

In the probabilistic safety assessment(PSA), the fault tree/event tree technique has been widely used to evaluate accident sequence frequencies. However, event transition which operators actually face can not be dynamically treated by the conventional technique.
Therefore, we have made the dynamic analysis program(DYANA) for event transition for a liquid metal cooled fast breeder reactor.

In the previous development, we made basic model for analysis. However, we have a probrem that calculation time is too long.

At the current term, we made parallelization of DYANA using MPI. So we got good performance on WS claster. It performance is close to ideal one.

This work was performed by CRC Research Institute, Inc. under contract withJapan Nuclear Cycle Development Institute.

JNC Liaison: O-arai Engineering Center, System Engineering Technology Division, Nuclear System Safety Research Group

*: CRC Research Institute, Energy Engineering Department, Section No.2

目　次

1.　概要 1
2.　改良作業 1
2.1　作業環境 1
2.2　DYANA 移植作業 2
2.3　DYANA 並列化作業 3
2.3.1　並列化指針と概要　3
2.3.2　乱数　7
2.4　追加・変更したソースファイル 9
2.5　並列化性能評価 10
2.5.1　評価方法と計算結果　10
2.5.2　評価概要　10
2.5.3　計算結果の検討　10
2.6　並列版 DYANA の計算結果 17
3.　並列化した DYANA の使用方法 20
3.1　コンパイル 20
3.2　実行 21
4.　参考文献 22
付録 A　変更したソースファイルのリスト
付録 B　サンプル計算結果(PLOHS シーケンス)
付録 C　サンプル計算結果(LORL/G シーケンス)
付録 D　近似関数

図目次

図 1　並列化の概要 4
図 2　並列化作業を実施した DYANA のプログラムの流れ[main()] 5
図 3　並列化作業を実施した DYANA のプログラムの流れ[exec_sim()] 6
図 4　$n_{PE} = 3$ のときの leapfrog 法による乱数生成の様子 8
図 5　並列化による速度向上率(並列化部):PLOHS シーケンス 15
図 6　並列化による速度向上率(並列化部):LORL/Gシーケンス 15
図 7　試行あたりの計算時間の変化 16
図 8　試行あたりの計算時間の変化(CRC:UltraSparc(Solaris)機) 16
図 9　PLOHS シーケンスの計算結果 18
図 10　LORL/G シーケンスの計算結果 18

表目次

表 1　現行 UltraSparc(Solaris)機による計算時間 11
表 2　PLOHS シーケンス計算時間 (1) 12
表 3　PLOHS シーケンス計算時間 (2) 12
表 4　PLOHS シーケンス計算時間 (3) 13
表 5　LORL/Gシーケンス計算時間 (1) 13
表 6　LORL/Gシーケンス計算時間 (2) 14
表 7　LORL/Gシーケンス計算時間 (3) 14
表 8　PLOHS シーケンスの計算結果 17
表 9　LORL/Gシーケンスの計算結果 17
表 10　オーダー毎の出現回数と寄与 19

リスト目次

リスト 1　DEC alpha(True 64UNIX)でのコンパイル用 makefile 2
リスト 2　並列化超簡易出力版 DYANA のコンパイル用メイクファイル 20
リスト 3　並列化 DYANA の実行用スクリプト 21

1. 概要

高速炉のレベル1の確率論的安全評価(PSA)[1]によれば，原子炉崩壊熱除去の失敗により炉心が損傷するような事故の発生確率を十分低く抑えるためには、緊急時運転手順に従った運転員の回復操作に成功することが極めて重要であることがわかっている．レベル1－PSA の精度を向上させるためには、原子炉崩壊熱除去失敗時のプラント動特性と運転員の行動を動的に解析することが必要である。このため、核燃料サイクル開発機構では仮想的なプラントの動特性，設備故障，そして運転員の行動を同時に解析するような動的信頼性解析プログラムDYANA[2],[3]を複雑系の解析を対象とするヴァーチャルエンジニアリング研究の一部として実施してきた．これまでの開発の中で基本的な解析モデルは固まりつつあるものの，計算時間の短縮が課題となっている．本作業においては、並列処理技法を DYANA へ導入することによって、計算時間の短縮を図った．

2. 改良作業

2.1 作業環境

現行の DYANA が稼働する環境と並列処理技法を導入して改良を行った DYANA が稼働する環境(これをターゲットマシン環境と呼ぶ)を以下に示す．

現行の DYANA 稼動機

SUN UltraSparc(Solaris)

ターゲットマシン環境

DEC alpha(True 64UNIX) WS クラスター

機器

CPU:Alpha600MHz x 4 +500MHz x 1

ネットワーク:FastEthernet

MPI 環境

MPICH1.1

2.2 DYANA 移植作業

現行の DYANA をターゲットマシン環境へ移植し，DYANA が移植後の環境下で正常に機能することを代表的な解析計算を実施することによって確かめた．なお，代表的な解析計算とは，原子炉容器液位が確保された状態で原子炉停止後に強制循環除熱による崩壊熱除去に失敗する(PLOHS：Protected Loss Of Heat Sink)シーケンスと1次冷却材漏洩時に原子炉が停止するものの1次主循環ポンプが停止失敗することによって原子炉容器液位の確保に失敗する(LORL/G：Loss Of Reactor Level に至るシーケンスの中の G)シーケンスである．

移植後の環境下でロードモジュールを作成するための makefile をリスト1に示す．

```
objs = dy.o dypvrd.o dyinrd.o dy_read.o dy_rstf.o dy_mcsrd.o dy_fmke.o ¥
      dy_cnc.o dy_cmcs.o dy_callm.o dy_mkcdf.o ¥
      dy_et.o dy_ett.o dy_hep.o dy_rflst.o dy_r.o dy_p.o ¥
      dy_ju.o dy_clk.o dy_srch.o

exec = ../dyanaMss

FFLAGS=

CFLAGS=

APXFLIB=../etr/apxF.a

libs =  /usr/lib/libfor.a /usr/lib/libFutil.a /usr/lib/libots.a /usr/lib/libm.a

.f.o :
        f77 $(FFLAGS) $< -c -o $*.o

.c.o :
        cc -c $(CFLAGS) $< -o $*.o

.h.c :
        touch *.c

$(exec) :  $(objs) $(COPDLIB)

        cc $(FFLAGS) $(objs) $(COPDLIB) $(APXFLIB) $(libs) -o $(exec)
```

リスト1 DEC alpha(True 64UNIX)でのコンパイル用 makefile

2.3 DYANA 並列化作業

2.3.1 並列化指針と概要

並列化作業は，ファイルへの計算結果の出力を最低限に削減した DYANA(これを超簡易版 DYANA と呼ぶ)をもとにモンテカルロ計算部について実施し，性能評価を行った．並列化に際しては，以下の指針で作業を行った．

1.モンテカルロ法の exec_sim 関数内の試行ループを並列化する．
2.I/O はライブラリアクセスを除きプロセス 0 が行う．
3.外部変数の初期化はプロセス 0 が行い，全プロセスに放送する．
4.各試行毎の出力の抑制

上記 2,3 は以下の目的のための処置である．

・入力データなどの特定ファイルに一時にアクセスが集中することを防ぐ．
・並列機環境に依存しない．

ただし，アクションツリーデータの読み込みに関しては，並列実行時に同期を取らないためにプロセスごとに読み込むこととした．
4.は WS クラスタのような通信の遅い環境で通信を減らすため，計算に関係ないループ内の通信を無くし，並列化性能を上げるための処置である．

DYANA 並列化の概要を図 1 に示す．

通信は初期の変数の設定と，試行ループ後の `n_CM_occer` の集計でしか現れない．このため，プロセス数にほぼ比例して性能が向上することが期待できる．

外部変数の放送，及び計算結果の集約にはそれぞれ `MPI_Bcast()`，`MPI_Reduce()`という集団通信の関数を使用した．集団通信には

・集団通信はベンダーにより並列機アーキテクチャに最適化した実装がなされるため，並列環境変更時にも最適な通信を行える可能性が高い．
・デッドロックを避けられる．

等の利点がある．また `MPI_Reduce()`は通信と同時に演算処理を行うことができる．計算結果の集約では `MPI_Reduce()`により計算結果をプロセス 0 に送る過程で合計が計算される．

並列化作業を実施した DYANA のプログラムの流れを図 2, 3 に示す．

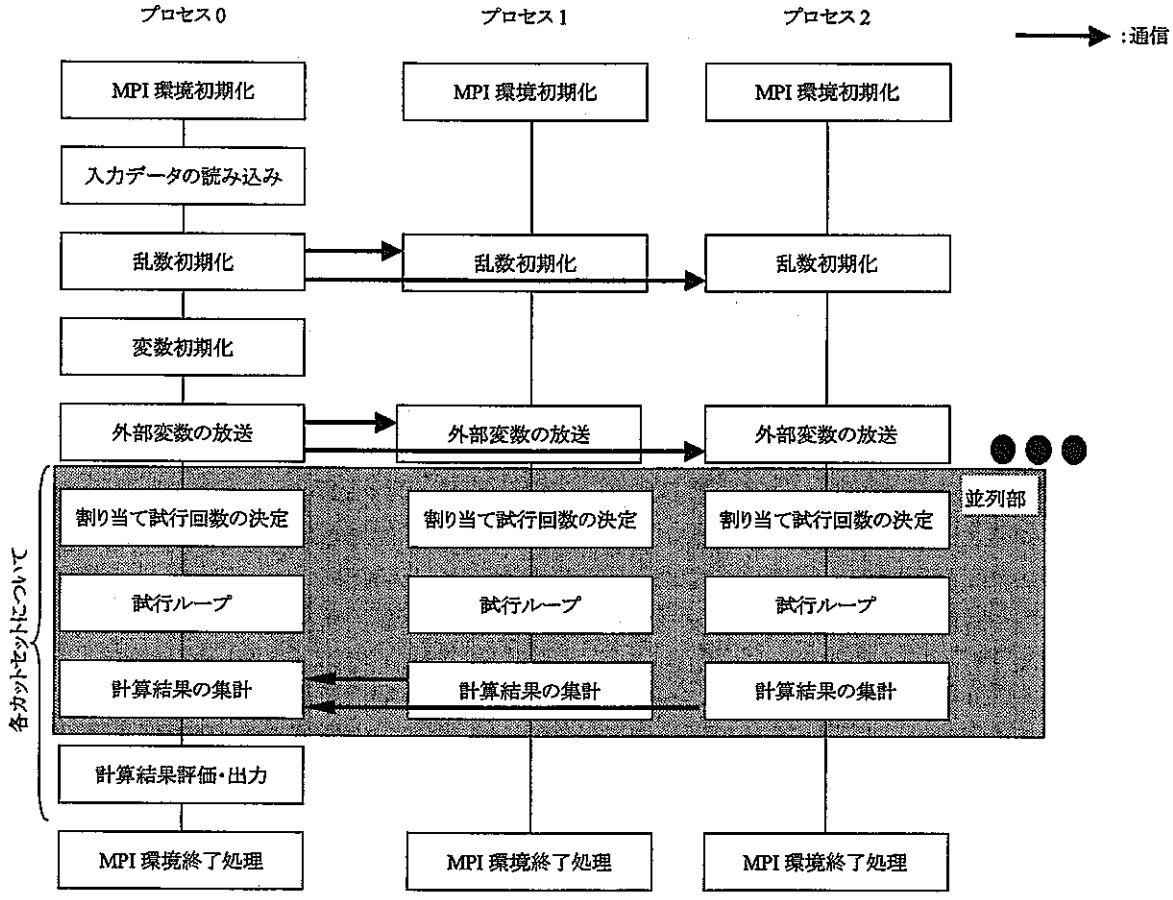
プロセス0
プロセス1
プロセス2
:通信
MPI 環境初期化
入力データの読み込み
乱数初期化
変数初期化
外部変数の放送
割り当て試行回数の決定
試行ループ
計算結果の集計
計算結果評価・出力
MPI 環境終了処理
並列部
各カットセットについて

図1　並列化の概要

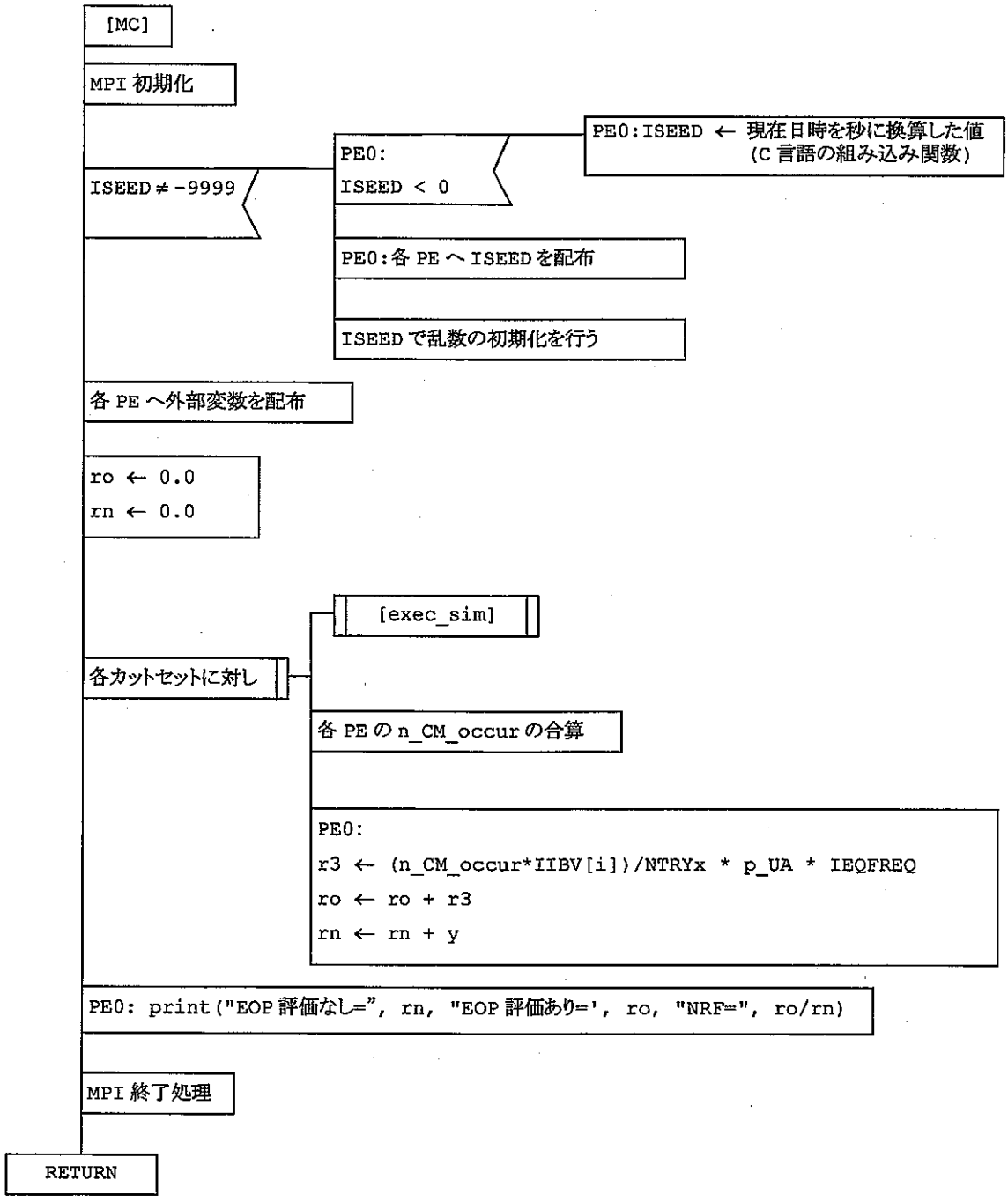

図 2　並列化作業を実施した DYANA のプログラムの流れ[main()]

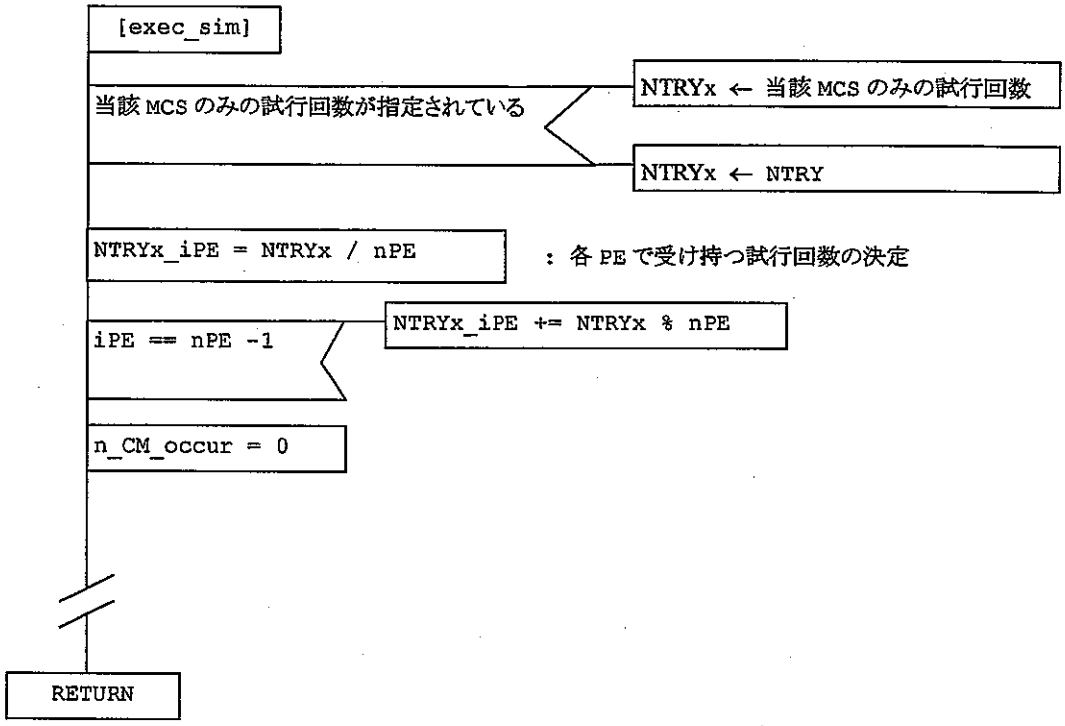

図 3　並列化作業を実施した DYANA のプログラムの流れ[exec_sim()]

2.3.2 乱数

並列計算において，乱数を生成する際にはプロセス毎に独立して生成できること，生成した乱数がプロセス間で無相関であることが重要である．シリアル計算で一般的に準備されている線型合同法を並列計算で用いる際には次の方法が考えられる．

(1) 各プロセスで別個の種より独立した系列で乱数を生成する．

(2) 1 つの種より生成する乱数を各プロセスに分配する．

(1)の方法は系列毎の相関の有無が必ずしも明確ではない．そこでの方法を用いることとした．(2)の方法のうちもっとも基本的なものは leapfrog 法(カエル跳び法)と呼ばれる方法である．今回作業では，leapfrog 法と同様な乱数を用いるようコーディングした．
以下に leapfrog 法の概要を示す[4]．

leapfrog 法

線型合同法は以下の式を用いて初期値 R_1 より疑似乱数列$\{R\}$を生成する．

$$R_{i+1} = (aR_i + c) \mod m \tag{1}$$

係数 a,c 及び除数 m は$\{R\}$が周期性を持たないように選ばれたる．ところで

$$R_{i+n} = \left(a^n R_i + c\frac{a^n - 1}{a-1}\right) \mod m \tag{2}$$

により n 個先の乱数を求めるものとすれば，プロセス 0 において元の乱数系列により他のプロセス k の乱数系列の初期値 $R_1^{(k)}$ を求め，プロセス k では $R_1^{(k)}$ を用いて，

$$R_{i+1}^{(k)} = \left(a^{n_{\mathrm{PE}}} R_i^{(k)} + c\frac{a^{n_{\mathrm{PE}}} - 1}{a-1}\right) \mod m \tag{3}$$

により乱数を生成することによりプロセス毎に独立に乱数を生成することができる．ここで n_{PE} はプロセス数である．乱数の生成の様子を図 4 に示す．

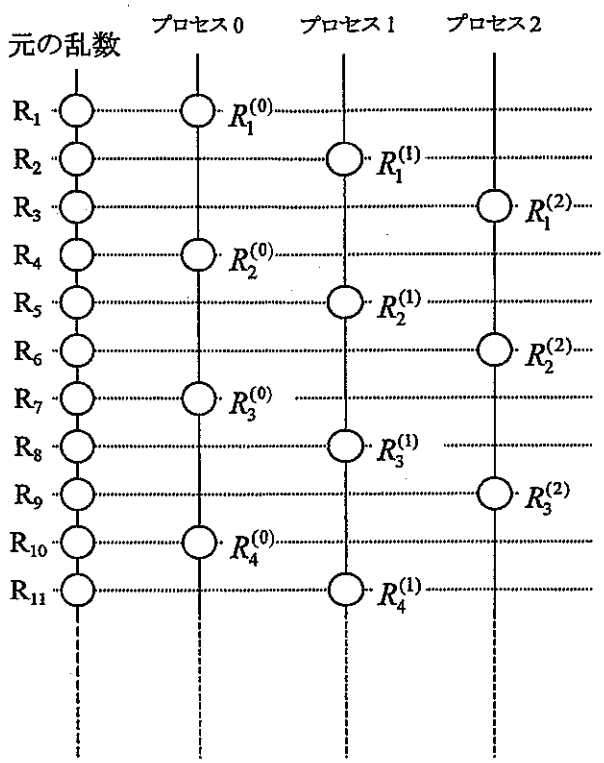

図 4　$n_{\mathrm{PE}} = 3$ のときの leapfrog 法による乱数生成の様子

実装

実際の作業では，すぐに使用できる長周期のライブラリがなかったため，leapfrog 法に相当する乱数を生成するルーチンをコーディングした．上記方法で leapfrog 法を実装するには，2^{48},2^{64} 等の長周期の実装のためにそれなりのコーディングが必要であること，係数の調整，乱数のテストなどが必要なことなど時間的に困難である．そこで DYANA での乱数の実装に際しては，leapfrog 法で生成した場合と同じ乱数を生成するよう，次のような実装を行った．

- 関数は従来どおり drand48( )を使用する．
- プロセス 0 は各プロセスに種を放送する．
- 各プロセスは放送された種を用い drand48( )を用いて乱数を生成し，$n_{\mathrm{PE}} - 1$ 回ダミー呼び出しを行う．

ダミーの呼び出しにより各プロセス数が増えるに従い乱数生成のためのオーバーヘッドが増加するが，DYANA では計算量における乱数生成の比率は十分小さいため大きな影響はないと判断した．

また, leapfrog 法では各プロセス数が偶数のときに乱数に相関が現れるかことがあるといわれている*. そのため, 各プロセス数が偶数のときには $n_{PE}+1$ のときと同じ乱数を生成するようコーディングすることによりこの問題を回避した.

2.4 追加・変更したソースファイル

追加したソースファイル

ソースファイル名:dy_dst.c

関数:void dist_init_data(void)

内容:外部変数の放送

ソースファイル名:dy_rand.c

関数:double rand_lf(void)

内容:drand48()を用いた leapfrog 法のエミュレーション.

変更したソースファイル

ソースファイル名:dy.c

関数:int main(int argc,char *argv[])

関数:void exec_sim(int tgt_mcs_num)

関数:oid set_UR_occur(void)

変更内容:並列化, 乱数呼び出し変更

ソースファイル名:dy_hep.c

関数:double rLHS(int *lc, int kkk)

関数:void calc_D_hep(TYPE_OF_ITEM *x, double *hep, int depend)

関数:void calc_O_hep(TYPE_OF_ITEM *x, double *hep, int tgtsubpath)

変更内容:乱数の呼び出し変更

---

* Makino.J :Comput.Phys.Commun.,78,105(1993)

2.5 並列化性能評価

2.5.1 評価方法と計算結果

PLOHS シーケンスおよび LORL/G シーケンスを用いてプロセス数 1-5 でテスト計算を行い，並列化性能評価を実施した．テストは，DYANA 以外のジョブについて制限をしない環境で，試行回数 1000 回，10000 回，20000 回について実施，DYANA 以外のジョブを排除した環境で試行回数 20000 回について実施した．

使用した WS は，プロセス 0 について 500Mz，それ以外のプロセスについては 600Mz である．並列処理の計算時間の計測では，通信等を考慮したシステムタイムを計測するツールがないため，プロセス 0 の実時間を計測し，図 1 における並列部とそれ以外のシリアル部を分離して評価した．並列部の時間計測には，実際に並列計算を行っているループ部だけでなく，計算結果を合計する処理までを含んでいる．

計算時間の計測は，他ジョブの影響がある程度排除できるよう各ケースにつき 2 回計測し，計算時間の短い方をもとに性能を評価した．ただし 20000 回の試行については他ジョブを排除したときの計測も含め 3 回とした．UltraSparc(Solaris)機での計算時間を表 1 に,WS クラスタ環境での計算時間を表 2 から表 7 に示す．表中の速度向上率は

(速度向上率) = (1 プロセスでの実行時間) / (複数プロセスでの実行時間)

として計算した．

2.5.2 評価概要

全体の傾向としては，以下のことがいえる．

1. 試行回数が増えると速度向上率があがる．
2. プロセス数に比例して速度が向上する．このとき比例定数は 1 より大きい．

結果として，WS クラスタ環境では UltraSparc(Solaris)機に比して最大で 10 倍近い計算速度を得ることができた．

2.5.3 計算結果の検討

評価概要 1.については以下のように解釈される．試行回数 1000 回では，ループの回数が少ないため，プロセスを増やすとプロセス毎のループにかかる計算時間がループ以外の部分の計算時間と同程度以下になり，速度向上率はプロセス数が増えるにしたがって低下して行く．プロセス数 5 ではそれが顕著に見られる．しかし試行回数が増えれば並列化ループ部の計算時間の割合が多くなるため次第に線型な並列化速度向上率に近づいていく．図 5,6 からは試行回数 2 万回では十分線型な並列化速度向上率となっていることがわかる．

評価概要 2.については次のように解釈できる．プロセス数にほぼ比例して速度が向上することは図 5 及び図 6 より確認できるが，試行回数 10000 回，20000 回では PLOHS シーケンス，

LORL/G シーケンス共に比例係数が 1 より大きくなっている. プロセッサ台数効果以上に速度が向上するのは, シーケンシャルの DYANA で, 試行回数が増えるにつれて 1 試行あたりの計算時間が増加するためである.

試行回数が増えるにつれて 1 試行あたりの計算時間が増加する傾向を PLOHS シーケンスについて検討した. 図 7 に alpha 機における 1 プロセスで計算した 1 試行あたりの計算時間の変化を執行回数 1000 回, 10000 回, 20000 回について示す. さらに図 8 に UltraSparc (Solaris) 機において試行 2000 回刻みで測定した 1 試行あたりの計算時間の変化を示す. 機種により計算時間の増加割合は異なるが, どちらの環境でも試行回数が増えるにしたがって 1 試行あたりの計算時間が増加する傾向が見られる. 機種が違うため直接比較はできないが, 同様の傾向を示すと仮定して alpha 機の結果を対数関数 $a\ln(n+b)$ を用いてフィッティングした. ここで n は試行回数である. フィッティングの結果は a=0.00454, b=139.85803 であった. 図中の実線がフィッティングした関数の値である.

この関数値にプロセスあたりの試行回数をかけたものを PLOHS シーケンス試行回数 20000 回の予測値として図 5 に一点鎖線で示した. 1 プロセスの計算結果から求めた速度向上率の予測値は複数プロセスでの速度向上率と良く一致している. このことから PLOHS シーケンス試行回数 20000 回では CPU の台数に見合った効果が十分得られていると結論できる.

表 1　現行 UltraSparc (Solaris) 機による計算時間 (秒)

	試行回数	1000	10000	20000
PLOHS	1回目	118	881	1756
	2回目	117	885	1765
	min	117	881	1756
LORL/G	1回目	107	774	1472
	2回目	104	771	1540
	min	104	771	1472

表 2　PLOHS シーケンス計算時間 (秒) (1)

PLOHS　試行回数 :1000回

	プロセス数	1	2	3	4	5
1回目	全体	59	44	39	39	35
	シリアル部	27	27	27	27	27
	並列部	32	17	12	12	8
	速度向上率(全体)	1.00	1.34	1.51	1.51	1.69
	速度向上率(並列部)	1.00	1.88	2.67	2.67	4.00
2回目	全体	57	42	37	34	33
	シリアル部	25	25	25	25	25
	並列部	32	17	12	9	8
	速度向上率(全体)	1.00	1.36	1.54	1.68	1.73
	速度向上率(並列部)	1.00	1.88	2.67	3.56	4.00
min	全体	57	42	37	34	33
	シリアル部	25	25	25	25	25
	並列部	32	17	12	9	8
	速度向上率(全体)	1.00	1.36	1.54	1.68	1.73
	速度向上率(並列部)	1.00	1.88	2.67	3.56	4.00

表 3　PLOHS シーケンス計算時間 (秒) (2)

PLOHS　試行回数 :10000回

	プロセス数	1	2	3	4	5
1回目	全体	439	226	156	122	111
	シリアル部	27	25	25	25	25
	並列部	412	201	131	97	86
	速度向上率(全体)	1.00	1.94	2.81	3.60	3.95
	速度向上率(並列部)	1.00	2.05	3.15	4.25	4.79
2回目	全体	443	225	153	117	99
	シリアル部	25	25	25	25	25
	並列部	418	200	128	92	74
	速度向上率(全体)	1.00	1.97	2.90	3.79	4.47
	速度向上率(並列部)	1.00	2.09	3.27	4.54	5.65
min	全体	439	225	153	117	99
	シリアル部	25	25	25	25	25
	並列部	412	200	128	92	74
	速度向上率(全体)	1.00	1.95	2.87	3.75	4.43
	速度向上率(並列部)	1.00	2.06	3.22	4.48	5.57

表4　PLOHSシーケンス計算時間 (秒) (3)

PLOHS　試行回数 :20000回

	プロセス数	1	2	3	4	5
1回目	全体	1021	461	309	223	182
	シリアル部	27	26	25	25	25
	並列部	994	435	284	198	157
	速度向上率(全体)	1.00	2.21	3.30	4.58	5.61
	速度向上率(並列部)	1.00	2.29	3.50	5.02	6.33
2回目	全体	969	441	292	223	185
	シリアル部	25	24	24	25	25
	並列部	944	417	268	198	160
	速度向上率(全体)	1.00	2.20	3.32	4.35	5.24
	速度向上率(並列部)	1.00	2.26	3.52	4.77	5.90
3回目	全体	937	459	294	224	185
(他ジョブを	シリアル部	24	25	25	25	25
排除)	並列部	913	434	269	199	160
	速度向上率(全体)	1.00	2.04	3.19	4.18	5.06
	速度向上率(並列部)	1.00	2.10	3.39	4.59	5.71
min	全体	937	441	292	223	182
	シリアル部	24	24	24	25	25
	並列部	913	417	268	198	157
	速度向上率(全体)	1.00	2.12	3.21	4.20	5.15
	速度向上率(並列部)	1.00	2.19	3.41	4.61	5.82

表 5　LORL/Gシーケンス計算時間 (秒) (1)

LORL/G　試行回数 :1000回

	プロセス数	1	2	3	4	5
1回目	全体	54	41	36	33	33
	シリアル部	25	25	25	25	25
	並列部	29	16	11	8	8
	速度向上率(全体)	1.00	1.32	1.50	1.64	1.64
	速度向上率(並列部)	1.00	1.81	2.64	3.63	3.63
2回目	全体	54	41	36	34	32
	シリアル部	25	25	25	25	25
	並列部	29	16	11	9	7
	速度向上率(全体)	1.00	1.32	1.50	1.59	1.69
	速度向上率(並列部)	1.00	1.81	2.64	3.22	4.14
min	全体	54	41	36	33	32
	シリアル部	25	25	25	25	25
	並列部	29	16	11	8	7
	速度向上率(全体)	1.00	1.32	1.50	1.64	1.69
	速度向上率(並列部)	1.00	1.81	2.64	3.63	4.14

表 6　G シーケンス計算時間 (秒) (2)

LORL/G　試行回数 :10000回

	プロセス数	1	2	3	4	5
1回目	全体	358	193	137	109	92
	シリアル部	27	25	25	25	25
	並列部	331	168	112	84	67
	速度向上率(全体)	1.00	1.85	2.61	3.28	3.89
	速度向上率(並列部)	1.00	1.97	2.96	3.94	4.94
2回目	全体	362	197	132	120	94
	シリアル部	25	25	25	25	25
	並列部	337	172	107	95	69
	速度向上率(全体)	1.00	1.84	2.74	3.02	3.85
	速度向上率(並列部)	1.00	1.96	3.15	3.55	4.88
平均	全体	358	193	132	109	92
	シリアル部	25	25	25	25	25
	並列部	331	168	107	84	67
	速度向上率(全体)	1.00	1.85	2.71	3.28	3.89
	速度向上率(並列部)	1.00	1.97	3.09	3.94	4.94

表 7　LORL/Gシーケンス計算時間 (秒) (3)

LORL/G　試行回数 :20000回

	プロセス数	1	2	3	4	5
1回目	全体	853	448	272	201	170
	シリアル部	26	26	38	26	26
	並列部	827	422	234	175	144
	速度向上率(全体)	1.00	1.90	3.14	4.24	5.02
	速度向上率(並列部)	1.00	1.96	3.53	4.73	5.74
2回目	全体	780	384	260	197	164
	シリアル部	25	25	25	24	26
	並列部	755	359	235	173	138
	速度向上率(全体)	1.00	2.03	3.00	3.96	4.76
	速度向上率(並列部)	1.00	2.10	3.21	4.36	5.47
3回目	全体	787	389	264	198	164
(他ジョブを	シリアル部	24	25	25	26	25
排除)	並列部	763	364	239	172	139
	速度向上率(全体)	1.00	2.02	2.98	3.97	4.80
	速度向上率(並列部)	1.00	2.10	3.19	4.44	5.49
平均	全体	780	384	260	197	164
	シリアル部	24	25	25	24	25
	並列部	755	359	234	172	138
	速度向上率(全体)	1.00	2.03	3.00	3.96	4.76
	速度向上率(並列部)	1.00	2.10	3.23	4.39	5.47

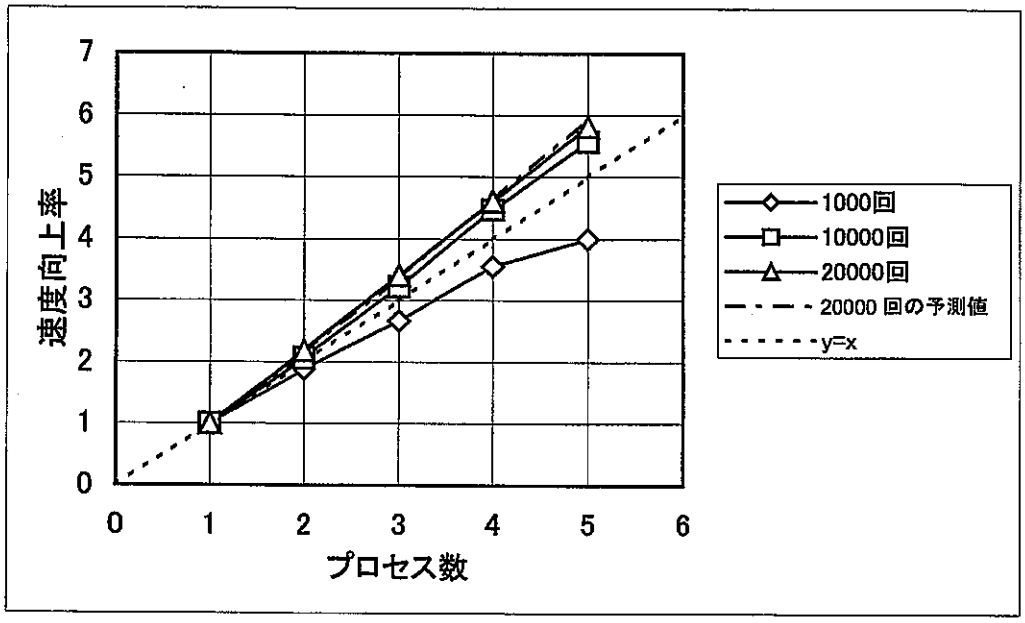

図 5　並列化による速度向上率(並列化部):PLOHS シーケンス

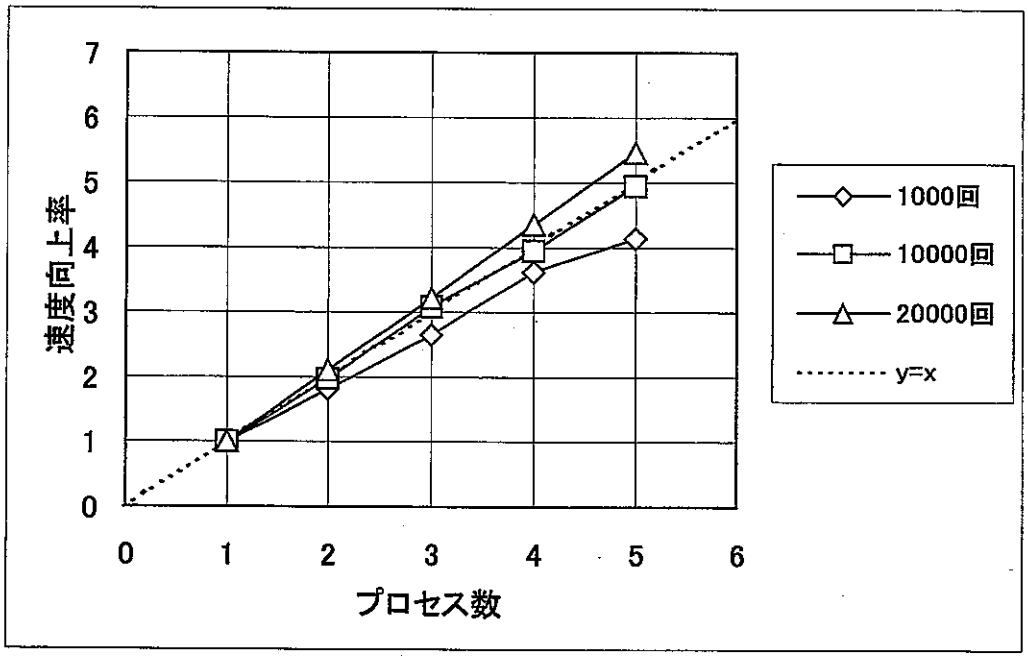

図 6　並列化による速度向上率(並列化部):LORL/G シーケンス

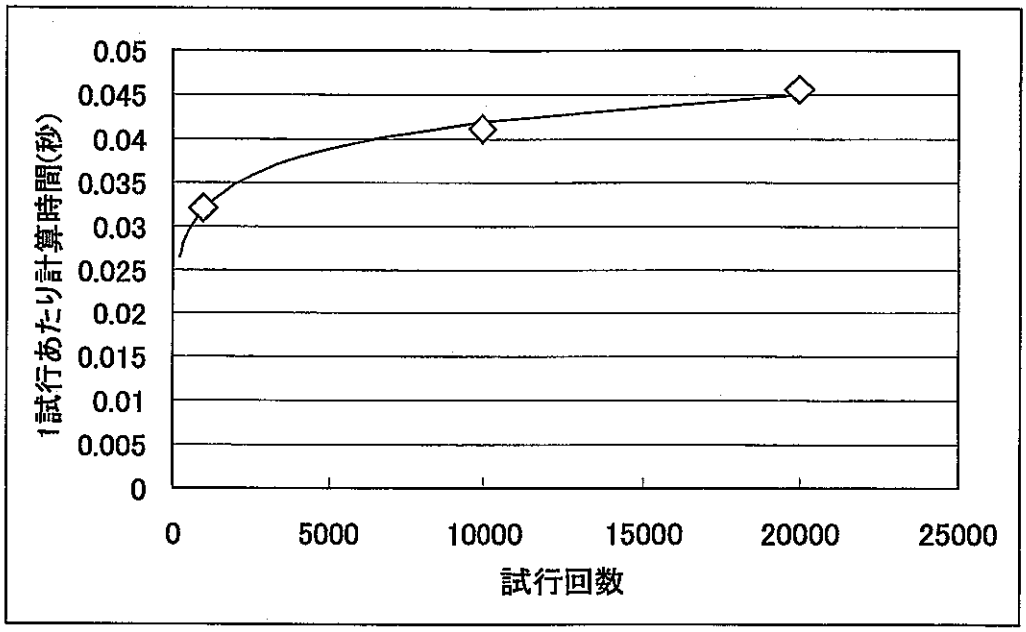

図 7　試行あたりの計算時間の変化

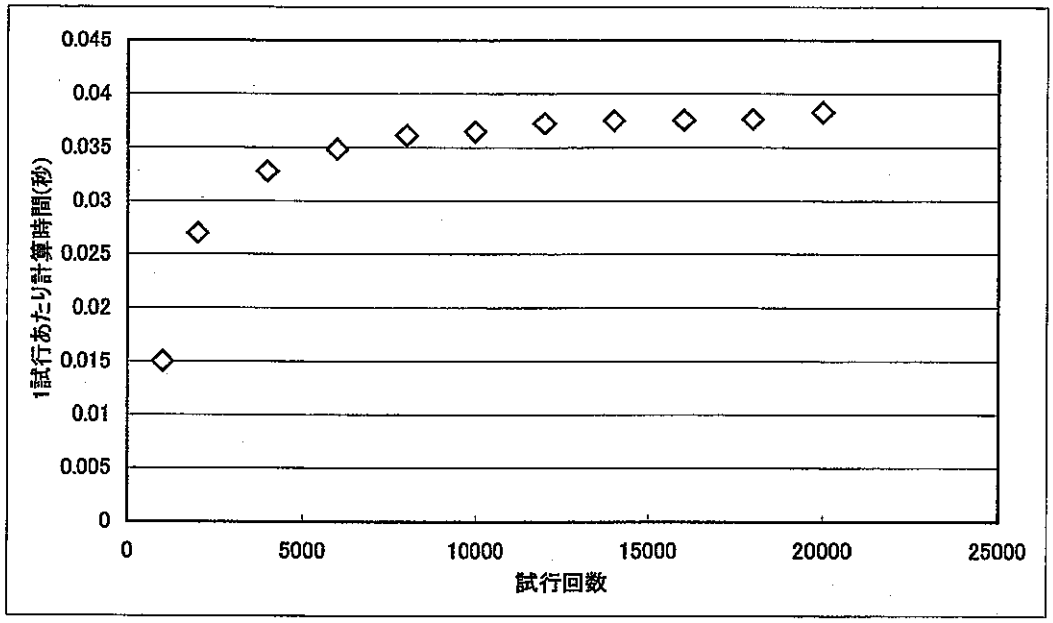

図 8　試行あたりの計算時間の変化(CRC:UltraSparc(Solaris)機)

2.6 並列版 DYANA の計算結果

並列版 DYANA のテスト計算の結果として得られる NRF(Non-Recovery Factor:PLOHS 等の事象の発生を想定したときに運転員の回復操作が失敗することによって炉心損傷に至る条件付き確率)の値を表 8, 表 9, 図 9, 図 10 に示す. プロセス数が変わると計算結果も変化しているが, これは 1 つの乱数系列をプロセス毎に順に使っているために, 1 つのプロセスに着目すれば元の乱数系列とは別の系列の乱数を使って計算していることになるためである. そのため, 試行回数が増えれば変化の幅も小さくなることが期待される.

PLOHS シーケンス試行回数 20000 回のケースについて, 各プロセスにおける 1 回の試行の n_CM_occur への寄与を表 10 にまとめた. 1 回毎の寄与のオーダー毎に見るとプロセス数が異なってもほぼ同じ出現傾向にある事が分かる. したがって乱数の系列による差は小さいことが期待できる. しかしながら n_CM_occur の値への寄与は 20000 回の試行の中で 3,4 回しか現れない最も上のクラスが 70% 前後の寄与をしている. 3,4 回程度の出現回数で 3 回と 4 回の違いは大きく, 3 回では n_CM_occur への寄与は 60%強, 4 回では 70%前後となっている. したがって乱数系列によるこのクラスの出現回数の違いが無視できる程度まで試行回数を増やすことが望ましいと判断できる.

表 8　PLOHS シーケンスの計算結果

PLOHS	1000	10000	20000	100000
1プロセス	6.95E−05	2.73E−04	2.82E−04	
2プロセス	7.47E−05	3.93E−04	2.90E−04	
3プロセス	8.12E−05	2.85E−04	2.38E−04	
4プロセス	1.36E−04	1.89E−04	2.36E−04	
5プロセス	1.07E−04	1.76E−04	2.28E−04	2.25E−04

表 9　LORL/Gシーケンスの計算結果

LORL/G	1000	10000	20000
1プロセス	8.93E−02	7.87E−02	7.92E−02
2プロセス	9.32E−02	7.92E−02	8.05E−02
3プロセス	9.22E−02	8.01E−02	7.90E−02
4プロセス	8.54E−02	8.06E−02	8.02E−02
5プロセス	8.54E−02	8.16E−02	7.98E−02

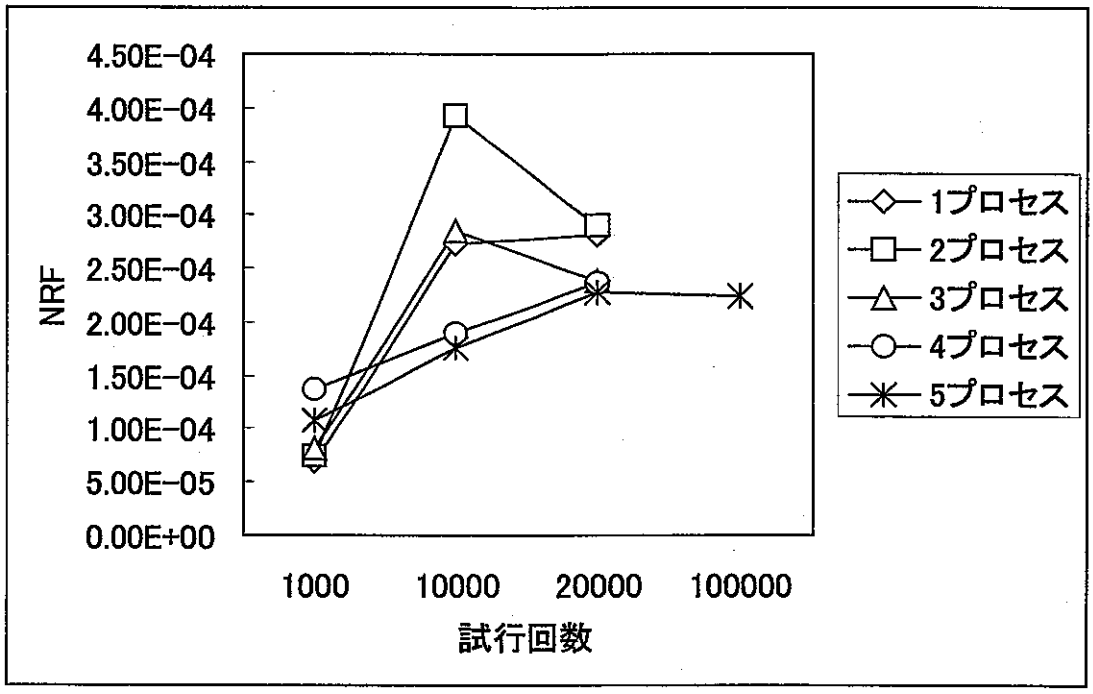

図 9　PLOHS シーケンスの計算結果

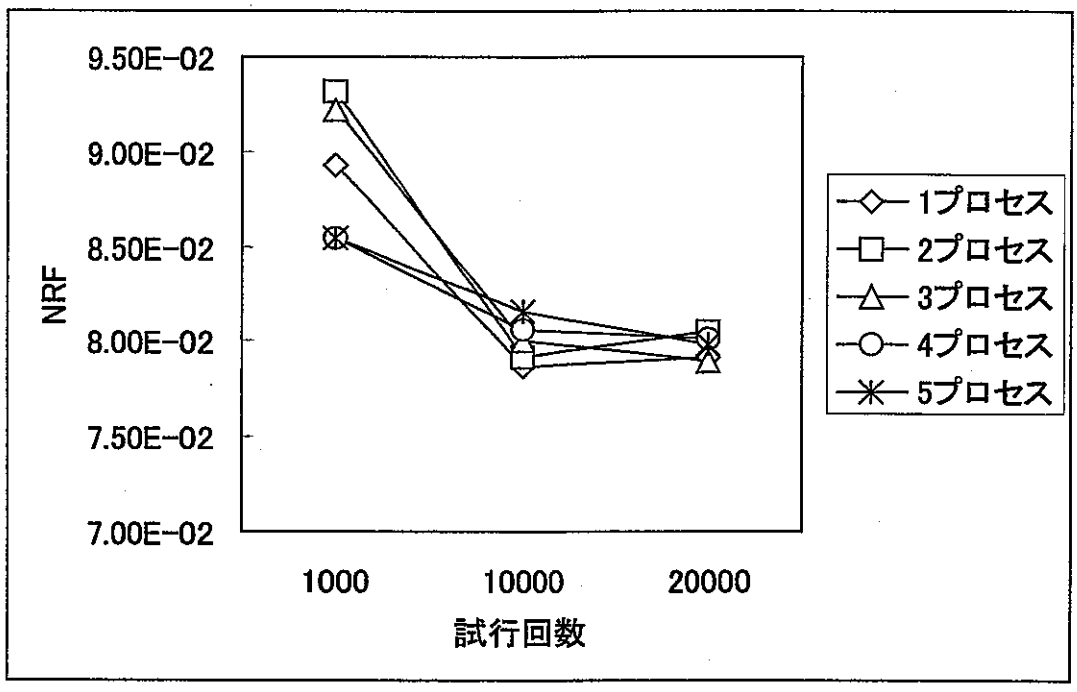

図 10　LORL/Gシーケンスの計算結果

表 10　オーダー毎の出現回数と寄与(PLOHS シーケンス)

	評価値のオーダー	出現回数							n_CM_occurへの寄与	
1プロセス		プロセス0	プロセス1	プロセス2	プロセス3	プロセス4	合計	%	値	比率(%)
	10^{-1} ~ 10^{0}	4					4	0.02	4.00E+00	70.99%
	10^{-2} ~ 10^{-1}	49					49	0.25	1.14E+00	20.15%
	10^{-3} ~ 10^{-2}	4					4	0.02	4.72E-03	0.08%
	10^{-4} ~ 10^{-3}	615					615	3.08	3.37E-01	5.98%
	10^{-5} ~ 10^{-4}	3951					3951	19.76	4.98E-02	0.88%
	10^{-6} ~ 10^{-5}	15377					15377	76.89	1.08E-01	1.92%
2プロセス		プロセス0	プロセス1	プロセス2	プロセス3	プロセス4	合計	%	値	比率(%)
	10^{-1} ~ 10^{0}	2	2				4	0.02	4.00E+00	68.95%
	10^{-2} ~ 10^{-1}	37	34				71	0.36	1.32E+00	22.78%
	10^{-3} ~ 10^{-2}	4	5				9	0.05	9.76E-03	0.17%
	10^{-4} ~ 10^{-3}	286	282				568	2.84	3.13E-01	5.40%
	10^{-5} ~ 10^{-4}	1888	1936				3824	19.12	4.78E-02	0.82%
	10^{-6} ~ 10^{-5}	7783	7741				15524	77.62	1.09E-01	1.88%
3プロセス		プロセス0	プロセス1	プロセス2	プロセス3	プロセス4	合計	%	値	比率(%)
	10^{-1} ~ 10^{0}	1	2	0			3	0.02	3.00E+00	63.00%
	10^{-2} ~ 10^{-1}	27	22	17			66	0.33	1.30E+00	27.21%
	10^{-3} ~ 10^{-2}	0	5	1			6	0.03	6.30E-03	0.13%
	10^{-4} ~ 10^{-3}	189	179	191			559	2.80	3.03E-01	6.36%
	10^{-5} ~ 10^{-4}	1273	1282	1306			3862	19.31	4.85E-02	1.02%
	10^{-6} ~ 10^{-5}	5176	5176	5151			15504	77.52	1.09E-01	2.29%
4プロセス		プロセス0	プロセス1	プロセス2	プロセス3	プロセス4	合計	%	値	比率(%)
	10^{-1} ~ 10^{0}	0	0	2	1		3	0.02	3.00E+00	63.69%
	10^{-2} ~ 10^{-1}	14	20	15	11		60	0.30	1.22E+00	25.89%
	10^{-3} ~ 10^{-2}	2	1	1	4		8	0.04	8.55E-03	0.18%
	10^{-4} ~ 10^{-3}	145	152	162	134		593	2.97	3.25E-01	6.91%
	10^{-5} ~ 10^{-4}	931	948	998	958		3835	19.18	4.78E-02	1.02%
	10^{-6} ~ 10^{-5}	3908	3879	3822	3892		15501	77.51	1.09E-01	2.31%
5プロセス		プロセス0	プロセス1	プロセス2	プロセス3	プロセス4	合計	%	値	比率(%)
	10^{-1} ~ 10^{0}	0	0	1	1	1	3	0.02	3.00E+00	65.71%
	10^{-2} ~ 10^{-1}	10	13	11	11	11	56	0.28	1.09E+00	23.89%
	10^{-3} ~ 10^{-2}	2	1	1	3	1	8	0.04	8.76E-03	0.19%
	10^{-4} ~ 10^{-3}	107	124	134	101	101	567	2.84	3.09E-01	6.76%
	10^{-5} ~ 10^{-4}	736	769	798	777	806	3886	19.43	4.85E-02	1.06%
	10^{-6} ~ 10^{-5}	3145	3093	3055	3107	3080	15480	77.40	1.09E-01	2.39%

オーダー:1 試行の炉心損傷評価値のオーダー

出現回数:20000 回試行したとき, 1 試行の評価値が該当オーダーであった試行の回数

n_CM_occur への寄与

値:試行毎の評価値のオーダー別の和

比率:(試行毎の評価値のオーダー別の和) / (試行毎の評価値の総和)

3. 並列化したDYANAの使用方法

3.1 コンパイル

変数 PARALLEL を定義してコンパイルすることにより並列版の超簡易出力版 DYANA が生成され，変数 PARALLEL を定義せずにコンパイルすればシリアル版の超簡易出力版 DYANA が生成される．ただしこのときにも結果出力は抑制されている．結果出力の抑制を解くには変数 DYNORMO を定義してコンパイルすれば良い．並列化超簡易出力版 DYANA のコンパイル用メイクファイルをリスト 2 に示す．

```
F77=mpif77
CC=mpicc
objs = dy.o dypvrd.o dyinrd.o dy_read.o dy_rstf.o dy_mcsrd.o dy_fmke.o ¥
      dy_cnc.o dy_cmcs.o dy_callm.o dy_mkcdf.o ¥
      dy_et.o dy_ett.o dy_hep.o dy_rflst.o dy_r.o dy_p.o ¥
      dy_ju.o dy_clk.o dy_srch.o dy_dst.o dy_rand.o

exec = ../dyanaMss

FFLAGS= -align commons -O1
CFLAGS=-DPARALLEL -O1

APXFLIB=../etr/apxF.a

libs = /usr/lib/libfor.a /usr/lib/libFutil.a /usr/lib/libots.a /usr/lib/libm.a

.f.o :
        $(F77) $(FFLAGS) $< -c -o $*.o
.c.o :
        $(CC) -c $(CFLAGS) $< -o $*.o
.h.c :
        touch *.c
$(exec) : $(objs) $(COPDLIB)
        $(CC) $(CFLAGS) $(objs) $(COPDLIB) $(APXFLIB) $(libs) -o $(exec)
```

変数 PARALLEL を定義

結果出力の抑制を解くにはここで-DDYNORMO とする

リスト 2　並列化超簡易出力版 DYANA のコンパイル用メイクファイル

3.2 実行

サイクル機構所有の DEC alpha 機クラスタ環境における実行手順を示す.

並列化超簡易出力版 DYANA の実行はシリアル版と同様に実行用スクリプト drun_m を用いる. ただし drun_m の引数として使用プロセス数を指定しなければならない.

リスト 3 に並列化 DYANA の実行用スクリプトを示す.

実行手順は以下のとおりである.

(1) DEC alpha 機に login

(2) 並列化 DYANA ディレクトリに移動

> cd dy99p↵

(3) データ準備

データファイルはシリアル版 DYANA と並列版 DYANA の間に相違はない.

(4) 実行

> drun_m プロセス数 データファイル名↵

(5) 計算結果の確認

計算結果は output/dym.lst に出力される.

```
#!/bin/csh -x
rm -f output/dym.lst
rm -f output/dym.out
date > output/dym.lst
setenv DRFPLOHS   ./dmy1
setenv DRFPLOHSNH ./dmy2
setenv DRFLORL    ./dmy3
(timex mpirun -np $1 dyanaMss $2 >> output/dym.out) >& error.lst
echo ' ' >> output/dym.lst
echo '*****************************************************' >> output/dym.lst
echo ' ' >> output/dym.lst
matome2 output/dym.out >> output/dym.lst
echo '*****************************************************' >> output/dym.lst
echo ' ' >> output/dym.lst
uname -a >> output/dym.lst
echo ' ' >> output/dym.lst
echo '*****************************************************' >> output/dym.lst
date >> output/dym.lst
```

リスト 3　並列化 DYANA の実行用スクリプト

4. 参考文献

[1] 可児吉男、中井良大、佐久間孝志、日置一雅、栗坂健一、「高速増殖炉原型炉のレベル1-PSA」、PNC ZN9410 91-381、1991

[2] K. Hioki, "Development of Accident Sequence Dynamic Analyis Method,"Proceedings of Probabilistic Safety Assessment Methodology and Applications,PSA'95, Seoul, Korea, 26-30 November, 1995

[3]日置一雅,「事象推移の動的解析手法の開発」, 日本原子力学会 1995 年秋の大会, D41,1995

[4] 澤田孝夫,「モンテカルロ法とシミュレーション」, 培風館, 1990

付録 A

変更したソースファイルのリスト

A.1 ソースプログラム dy.c

A.2 ソースプログラム dy_dst.c

A.3 ソースプログラム dy_hep.c

A.4 ソースプログラム dy_rand.c

A.1 ソースプログラム dy.c

```
dy.c     1:
dy.c     2: /* DYANA9803 : dy.c <M>                    */
dy.c     3: /* 1998.9                                  */
dy.c     4:
dy.c     5: #define _MAIN_MODULE_
dy.c     6:
dy.c     7: #include "mpi.h"
dy.c     8:
dy.c     9: #include "dy.h"
dy.c    10:     double rand_lf(void);
dy.c    11:     int g_ntryx;
dy.c    12:
dy.c    13: /*-----------------------------------------------------------------*/
dy.c    14: void err_msg(char *msg)                          /* Error Messege Display */
dy.c    15: {
dy.c    16:     printf("%s¥n¥n",msg);
dy.c    17:     fprintf(stderr,"%s¥n¥n",msg);
dy.c    18:     exit(1);
dy.c    19: }
dy.c    20:
dy.c    21: /*-----------------------------------------------------------------*/
dy.c    22: char *trim(char *s)
dy.c    23: { int i;
dy.c    24:     if ( ( i = strlen(s) ) > 0 ){
dy.c    25:         for( i--; i >= 0; i-- ){
dy.c    26:             if ( s[i] != ' ' && s[i] != '¥t' ) break;
dy.c    27:         }
dy.c    28:         s[i+1]=NULL;
dy.c    29:     }
dy.c    30:     return s;
dy.c    31: }
dy.c    32:
dy.c    33: /*-----------------------------------------------------------------*/
dy.c    34: void set_case_id(char *eostat, double aprob, double hepvalue)
dy.c    35: {
dy.c    36:     sprintf(case_id,"%5s%10.2e%10.2e",
```

```
dy.c   37:                          eostat,
dy.c   38:                          aprob * current_mcs->p_U * ie_freqncy,
dy.c   39:                          hepvalue);
dy.c   40: }
dy.c   41:
dy.c   42: /*-----------------------------------------------------------------*/
dy.c   43: void set_rstfile_number(int n, unsigned int *tgt)
dy.c   44: {
dy.c   45:     *tgt = n*100 + 1;
dy.c   46: }
dy.c   47:
dy.c   48: /*-----------------------------------------------------------------*/
dy.c   49: void out_path(char *p)
dy.c   50: {
dy.c   51: #ifdef DYDEBUG
dy.c   52:     if ( *p != 0 ){
dy.c   53:         fprintf(fp_sd,"%s¥n¥n",p);
dy.c   54:         printf("%s¥n",p);
dy.c   55:     }
dy.c   56:     else{
dy.c   57:         fprintf(fp_sd,"<¥n");
dy.c   58:     }
dy.c   59: #endif
dy.c   60: }
dy.c   61:
dy.c   62: /*-----------------------------------------------------------------*/
dy.c   63: void out_status_endtrace(int stat)
dy.c   64: {
dy.c   65: #ifdef DYNORMO
dy.c   66:     switch(stat){
dy.c   67:         case -1:
dy.c   68:             printf("¥n シーケンス計算終了 ： 低温設定下限値を下回りました.¥n¥n");
dy.c   69:             break;
dy.c   70:         case  1:
dy.c   71:             printf("¥n シーケンス計算終了 ： 高温設定上限値を超過しました.¥n¥n");
dy.c   72:             break;
dy.c   73:     }
dy.c   74: #endif
dy.c   75: #ifdef DYNORMO
dy.c   76:     printf("##########################################################################¥n¥n");
dy.c   77: #endif
dy.c   78: }
```

```
dy.c   79:
dy.c   80: /*-----------------------------------------------------------------*/
dy.c   81: void sdf_file_open(int tgt_mcs_num)
dy.c   82: {
dy.c   83:     char sdfname[32];
dy.c   84:
dy.c   85:     sprintf(sdfname,"output/dy.sdf.%-5d",tgt_mcs_num);
dy.c   86:
dy.c   87:     trim(sdfname);
dy.c   88:
dy.c   89:     if ( ( fp_sd = fopen(sdfname,"w+") ) == NULL ){
dy.c   90:         printf("file create error!!! [%s]¥n",sdfname);
dy.c   91:         exit(1);
dy.c   92:     };
dy.c   93:
dy.c   94: }
dy.c   95:
dy.c   96: /*-----------------------------------------------------------------*/
dy.c   97: void ec_sort_ocr(int n, int ocr[], int ocr_index[])
dy.c   98: { /*  発生時刻順に並べ換えるための index を作る  */
dy.c   99:     int i,j,k;
dy.c  100:     for(i=0;i<n;i++) ocr_index[i] = i;
dy.c  101:     for(i=0;i<n-1;i++){
dy.c  102:         for(j=i+1;j<n;j++){
dy.c  103:             if ( ocr[ocr_index[i]] > ocr[ocr_index[j]] ){
dy.c  104:                 k = ocr_index[i];
dy.c  105:                 ocr_index[i] = ocr_index[j];
dy.c  106:                 ocr_index[j] = k;
dy.c  107:             }
dy.c  108:         }
dy.c  109:     }
dy.c  110: }
dy.c  111:
dy.c  112: /*-----------------------------------------------------------------*/
dy.c  113: void init_copd_param_set(void)
dy.c  114: {
dy.c  115:     air_rate        = DEF_AIR_RATE;
dy.c  116:     prev_taRVNaO = 529.0;
dy.c  117:
dy.c  118:     /* -----   time step setting   -----  それぞれ 4.0sec が最大  */
dy.c  119:     copd_calc_deltaAA = 0.2;
dy.c  120:     copd_calc_deltaFN = 0.2;
```

```
dy.c  121:     copd_calc_deltaRF = 0.2;
dy.c  122:
dy.c  123:     copd_mid_sim_width = MID_SIM_WIDTH;   /* 秒 */
dy.c  124:
dy.c  125:     /*acs_nc_na_out_fac = 1.15;*/
dy.c  126:     acs_nc_na_out_fac = 0.016;
dy.c  127: }
dy.c  128:
dy.c  129: /*-----------------------------------------------------------------*/
dy.c  130: void set_UR_occur(void)
dy.c  131: {
dy.c  132:     double v, px[4], pMAX;
dy.c  133:     int       i, j, n, k;
dy.c  134:
dy.c  135:     /*pMAX = -1.0;*/
dy.c  136:
dy.c  137:     for(i=0;i<current_mcs->nUR;i++){
dy.c  138:        px[i]       = current_mcs->probUR[i];
dy.c  139:        biasv[i] = mis_time * px[i];
dy.c  140:        printf("    Failure Rate of UR #%1d=%9.2e(biasv=%9.2e)¥n",
dy.c  141:                                                    i+1, px[i], biasv[i]);
dy.c  142:     }
dy.c  143:     printf("¥n");
dy.c  144:
dy.c  145:     for(j=0;j<ntryx;j++){
dy.c  146:
dy.c  147:        for(i=0;i<current_mcs->nUR;i++){
dy.c  148:           v = rand_lf() * biasv[i];
dy.c  149: /*      v = drand48() * biasv[i]; */
dy.c  150:           /* UR が発生 */
dy.c  151:           n = (int)( 60.0 * ( v / px[i] ) );
dy.c  152:           t_UR_occur[j][i] = n;
dy.c  153:        }
dy.c  154:
dy.c  155:        n_MCS_occur++;
dy.c  156:
dy.c  157:        /* current_mcs->nUR 個すべて起こったときに mcs 成立 */
dy.c  158:        /* if ( k == current_mcs->nUR ){   */
dy.c  159:        /*     n_MCS_occur++;                  */
dy.c  160:        /* }                                   */
dy.c  161:        /*   n_MCS_not_occur++;                */
dy.c  162:        /*   t_UR_occur[j][0] = -1;          */
```

```
dy.c  163:
dy.c  164:        t_UR_occur_v[j] = current_mcs->p_U;   /* IE 頻度は含まれていない,UA 確率は含まれている */
dy.c  165:
dy.c  166:     }
dy.c  167: }
dy.c  168:
dy.c  169: /*-----------------------------------------------------------------*/
dy.c  170: void exec_sim(int tgt_mcs_num)
dy.c  171: {
dy.c  172:     int       i, j, simcount, i_try_loop, tgt_UR_num;
dy.c  173:     double sim_time;
dy.c  174:     double sim_start_time;
dy.c  175:     double sim_width;
dy.c  176:     double r;
dy.c  177:     double old_n_CM_occur;
dy.c  178:     FILE     *fp_x;
dy.c  179:
dy.c  180:     printf("¥n I am %d. exec_sim start ¥n¥n",myid);
dy.c  181:
dy.c  182:     current_mcs = &(mcs_set[m_indx[tgt_mcs_num]]);
dy.c  183:
dy.c  184:     n_CM_occur = 0.0;
dy.c  185:
dy.c  186:     pr_info_1(tgt_mcs_num);
dy.c  187:
dy.c  188:     if ( current_mcs->p_U * ie_freqncy <= prblimt ){
dy.c  189: #ifndef PARALLEL
dy.c  190:        printf("打ち切り確率値以下なので次に進みます.¥n");
dy.c  191: #endif
dy.c  192:        return;
dy.c  193:     }
dy.c  194:
dy.c  195:     srch_ec_cat();            /* mcs_cat_code の設定        */
dy.c  196:
dy.c  197:     if ( current_mcs->n_TRY > 0 ){
dy.c  198:        ntryx = current_mcs->n_TRY;
dy.c  199:     }
dy.c  200:     else{
dy.c  201:        ntryx = n_try;
dy.c  202:     }
dy.c  203:
dy.c  204:     g_ntryx = ntryx;
```

```
dy.c  205:     ntryx = g_ntryx / numprocs;
dy.c  206: #ifdef PARALLEL
dy.c  207:     if(myid == numprocs-1) ntryx += g_ntryx - ntryx * numprocs;
dy.c  208:     printf("I am %d : ntryx= %d, g_ntryx = %d ¥n", myid, ntryx, g_ntryx);
dy.c  209: #endif
dy.c  210:
dy.c  211:     pr_info_2();       /* UR の発生時刻を決定する */
dy.c  212:
dy.c  213:     r = 1.0;
dy.c  214:     for(i=0;i<current_mcs->nUR;i++) r *= biasv[i];
dy.c  215:     frq_calc_factor = r * current_mcs->p_UA * ie_freqncy / (double)ntryx;;
dy.c  216:
dy.c  217:     if(myid == hostid){
dy.c  218:        printf("(frq_calc_factor=%9.2e)¥n",frq_calc_factor);
dy.c  219:     }
dy.c  220:
dy.c  221:     old_n_CM_occur = 0.0;
dy.c  222:
dy.c  223:     /* 試行回数のループ */
dy.c  224:     printf("¥n I am %d. exec_sim:ntryx= %d¥n¥n",myid,ntryx);
dy.c  225:     for( i_try_loop = 0 ; i_try_loop < ntryx ; i_try_loop++ ){
dy.c  226:
dy.c  227:        if ( t_UR_occur[i_try_loop][0] == -1 ) continue;
dy.c  228:
dy.c  229: #ifndef PARALLEL
dy.c  230:        system("rm -f copd/data/ft40.dat copd/data/ft10.dat");
dy.c  231: #endif
dy.c  232:
dy.c  233:        i__try = i_try_loop;
dy.c  234:
dy.c  235:        t_mark2 = time(NULL);
dy.c  236: #ifndef PARALLEL
dy.c  237:        printf("¥n>>>>>>>>>>>> TRY#%-5d%20s 現在までの CPU 経過時間 = %d¥n¥n",
dy.c  238:                    i_try_loop+1, "", t_mark2-t_mark1);
dy.c  239: #endif
dy.c  240:
dy.c  241:        time_of_MCSvalid   = -999.0;       /* Min. Cut Set:UR が成立した時刻 */
dy.c  242:        time_of_MCSvalid   = -999.0;       /* UR が成立した時刻                */
dy.c  243:        time_of_NHSstart   = -999.0;
dy.c  244:        time_of_NHSstart   = -999.0;
dy.c  245:        time_of_NCstart    = -999.0;
dy.c  246:        time_of_MCSstart   = -999.0;
```

```
dy.c  247:        time_of_PMTstart   = -999.0;
dy.c  248:        time_of_FC1start   = -999.0;
dy.c  249:        time_of_PSOSstart = -999.0;
dy.c  250:        time_of_KTstart    = -999.0;
dy.c  251:        time_of_VCstart    = -999.0;
dy.c  252:
dy.c  253:        /* n_CM_occur に加算した current_aprob の値のリストに登録されている値の数 */
dy.c  254:        n_x_CM_o_list = 0;
dy.c  255:
dy.c  256:        /* Print UR information */
dy.c  257:        pr_info_3(i_try_loop, ocr_indexG[i_try_loop]);
dy.c  258:
dy.c  259:        init_copd_param_set();   /* -> dy.c */
dy.c  260:
dy.c  261:        current_mcs->stat[1] = judge_loop_status(); /* P)LOHS,L)ORL      */
dy.c  262:        current_mcs->stat[2] = 0;
dy.c  263:
dy.c  264:        flgJstart = 0;
dy.c  265:        flgPSOSon = 0; /* 一度でも作動させたら1 */
dy.c  266:
dy.c  267:
dy.c  268: #ifndef RSTF
dy.c  269:        /* select and copy the initial restart file -> dy_rstf.c */
dy.c  270:        rstfile_setup_0();
dy.c  271: #endif
dy.c  272:
dy.c  273:        /* -----   count of copd execution ----- */
dy.c  274:        simcount = 0;
dy.c  275:
dy.c  276:        pv_hist_stkptr = 0;
dy.c  277:
dy.c  278:        tgt_UR_num = current_mcs->nUR - 1;
dy.c  279:
dy.c  280: #ifdef DYDEBUG
dy.c  281:          printf("(tgt_UR_num=%d , loop_status=%s)¥n" , tgt_UR_num , loop_status );
dy.c  282: #endif
dy.c  283:
dy.c  284:        if( ( current_mcs->nUR==0 ) || ( current_mcs->stat[1]=='L' ) ){
dy.c  285:          /* LORL の時は最初から追跡 */
dy.c  286:          sim_time                  =   0.0;
dy.c  287:          sim_width                 = 10.0;
dy.c  288:          time_of_NHSstart          =   0.0;    /* NHS とは限らないが、とりあえず */
```

```
dy.c  289:          sim_time_start_point =   0.0;
dy.c  290:       }
dy.c  291:       else{
dy.c  292:          sim_start_time          = 60.0 * (double)t_UR_occur[i_try_loop][ocr_indexG[i_try_loop][tgt_UR_num]];
dy.c  293:          time_of_NHSstart        = sim_start_time; /* NHS とは限らないが、とりあえず */
dy.c  294:          sim_time_start_point = sim_start_time; /*  とりあえず、初期化                    */
dy.c  295:          sim_time   = floor(sim_start_time/600.0)*600.0;
dy.c  296:          sim_width = sim_start_time - sim_time;                /* 600sec 毎に記録されているので差の分を走らせる */
dy.c  297: #ifdef DYDEBUG
dy.c  298:          printf("(i_try_loop=%d)¥n",i_try_loop);
dy.c  299:          printf("(ocr_indexG[i_try_loop][tgt_UR_num]=%d)¥n",ocr_indexG[i_try_loop][tgt_UR_num]);
dy.c  300:          printf("(sim_start_time=%e)¥n",sim_start_time);
dy.c  301: #endif
dy.c  302:          if ( sim_start_time == 0.0 ){
dy.c  303:             sim_time   =   0.0;
dy.c  304:             sim_width = 10.0;
dy.c  305:          }
dy.c  306:          else{
dy.c  307:             if ( sim_start_time == sim_time ){
dy.c  308:                sim_time -= 600.0;
dy.c  309:                sim_width = 600.0;
dy.c  310:             }
dy.c  311:          }
dy.c  312:       }
dy.c  313:
dy.c  314: #ifndef RSTF
dy.c  315:       rstinpdat05_setup_init(sim_time, sim_width);/* make ft05.dat    */
dy.c  316: #endif
dy.c  317:
dy.c  318:       /* -1:Success(Low temp),    1:Failure(High Temp) */
dy.c  319:       if ( ( i = run_copd(simcount++, &sim_time, sim_width, 1, "", 0) ) != 0 ) goto LOOP_END;
dy.c  320:
dy.c  321: #ifndef RSTF
dy.c  322:       /* ft10.dat -> fort.11, ft40.dat -> fort.41                    */
dy.c  323:       rstfile_setup();
dy.c  324: #endif
dy.c  325:
dy.c  326:       sim_width = (sim_width==10.0) ? 20.0 : 60.0;
dy.c  327:
dy.c  328: #ifndef RSTF
dy.c  329:       rstinpdat05_setup(loop_status, sim_time, sim_width);/* make ft05.dat    */
dy.c  330: #endif
```

```
dy.c  331:
dy.c  332:        /*============= Simulation loop ==============*/
dy.c  333:        while(1){
dy.c  334:
dy.c  335:          if ( ( i = run_copd(simcount++, &sim_time, sim_width, 0, "", 0) ) != 0 ) break; /* -1:Success(Low temp),   1:Failure(High Temp) */
dy.c  336:
dy.c  337:          if ( ( i = judge_ope_act_go() ) > ' ' ){
dy.c  338:
dy.c  339:            /* i->ｼﾅﾘｵ番号   'P', 'J', 'G', 'T'   */
dy.c  340:
dy.c  341: #ifndef PARALLEL
dy.c  342:            printf("¥n 導入条件が満たされた操作手順ｺｰﾄﾞ : %c¥n¥n",(char)i);
dy.c  343: #endif
dy.c  344:
dy.c  345: #ifndef RSTF
dy.c  346:            cpy_init_rstfile(sim_time); /* make RstFile from fort.41 & fort.11 */
dy.c  347: #endif
dy.c  348:
dy.c  349:            et_trace_main(i, simcount, sim_time, sim_width);
dy.c  350:
dy.c  351:            i = 0;
dy.c  352:
dy.c  353:            break;
dy.c  354:          }
dy.c  355:          else{
dy.c  356:
dy.c  357:            sim_width = MID_SIM_WIDTH;
dy.c  358:            if ( ( i = run_copd_continue("", sim_time, sim_width) ) != 0 ) break;
dy.c  359:
dy.c  360:          }
dy.c  361:
dy.c  362:        } /* while(1) */
dy.c  363:
dy.c  364: #ifndef RSTF
dy.c  365:        rstfile_remove_eo();
dy.c  366: #endif
dy.c  367:
dy.c  368:        LOOP_END:;
dy.c  369:
dy.c  370:        /*
dy.c  371:        printf("(current_mcs->p_U=%9.2e ,ie_freqncy=%9.2e)¥n¥n",
dy.c  372:                current_mcs->p_U, ie_freqncy);
```

```
dy.c  373:        */
dy.c  374:
dy.c  375: #ifndef PARALLEL
dy.c  376:        if ( old_n_CM_occur != n_CM_occur ){
dy.c  377:           /* 変化があったら、current_aprob を書く */
dy.c  378:           printf("#%5d 炉心損傷回数(n_CM_occur)=%10.3e (current_aprob=",
dy.c  379:                   i_try_loop+1,
dy.c  380:                   n_CM_occur);
dy.c  381:           for(j=0;j<n_x_CM_o_list-1;j++){
dy.c  382:              printf("%10.3e,",x_CM_o_list[j]);
dy.c  383:           }
dy.c  384:           printf("%10.3e)¥n",x_CM_o_list[n_x_CM_o_list-1]);
dy.c  385:        }
dy.c  386:        else{
dy.c  387:           printf("#%5d 炉心損傷回数(n_CM_occur)=%10.3e¥n",i_try_loop+1,n_CM_occur);
dy.c  388:        }
dy.c  389: #endif
dy.c  390:
dy.c  391:        old_n_CM_occur = n_CM_occur;
dy.c  392:
dy.c  393: #ifdef DYNORMO
dy.c  394:        printf("¥f");
dy.c  395: #endif
dy.c  396:
dy.c  397:     } /* for(n_try) */
dy.c  398:           printf("I am %d, #%5d 炉心損傷回数(n_CM_occur)=%10.3e¥n",myid ,i_try_loop+1,n_CM_occur);
dy.c  399:
dy.c  400: }
dy.c  401:
dy.c  402: /*-----------------------------------------------------------------*/
dy.c  403: void rm_old_sdf_file(void)
dy.c  404: {
dy.c  405:     int i;
dy.c  406:
dy.c  407:     i = system("rm -f output/dy.sdf.???");
dy.c  408:     if ( i != 0 ){
dy.c  409:        printf("Remove the file[dy.sdf.???] error. Error: error code=%2d¥n",i);
dy.c  410:        exit(1);
dy.c  411:     }
dy.c  412:
dy.c  413:     i = system("rm -f output/dy.sdf.*");
dy.c  414:     if ( i != 0 ){
```

```
dy.c  415:        printf("Remove the file[dy.sdf.*] error. Error: error code=%2d¥n",i);
dy.c  416:        exit(1);
dy.c  417:    }
dy.c  418:
dy.c  419:    i = system("rm -f copd/copd.stderr");
dy.c  420:    if ( i != 0 ){
dy.c  421:        printf("Remove the file[copd.stderr] error. Error: error code=%2d¥n",i);
dy.c  422:        exit(1);
dy.c  423:    }
dy.c  424:
dy.c  425: }
dy.c  426:
dy.c  427: /*-----------------------------------------------------------------*/
dy.c  428: int main(int argc,char *argv[])
dy.c  429: {
dy.c  430:    FILE     *fp_sdfx;
dy.c  431:    int      i,j;
dy.c  432:    char     argv80[80];
dy.c  433:    time_t timeX, t_m4;
dy.c  434:    double v,x,r1,r2,r3,y,ro,rn,org_value;
dy.c  435:    double tot_n_CM_occur;
dy.c  436:
dy.c  437: #ifdef PARALLEL
dy.c  438:        MPI_Init(&argc,&argv);
dy.c  439:        MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
dy.c  440:        MPI_Comm_rank(MPI_COMM_WORLD,&myid);
dy.c  441:        printf("num.of process = %d, I am process%d ¥n", numprocs, myid);
dy.c  442:        if(numprocs % 2 == 0){
dy.c  443:            nproc1 = numprocs +1;
dy.c  444:        }else{
dy.c  445:            nproc1 = numprocs;
dy.c  446:      }
dy.c  447: #else
dy.c  448:        myid=0;
dy.c  449:        numprocs=1;
dy.c  450:        nproc1=1;
dy.c  451: #endif
dy.c  452:
dy.c  453:    t_mark1 = time(&timeX);
dy.c  454:
dy.c  455:    if(myid==hostid){
dy.c  456:          printf("¥n------ DYANA9803 : Dynamic Analysis Program for Event Transition. ------        計算開始日時 %s¥n¥n",
```

```
dy.c  457:                    ctime(&t_mark1) );
dy.c  458:
dy.c  459:            printf("(1999/01/31 17:00)¥n");
dy.c  460:      }
dy.c  461:
dy.c  462:      strcpy(sf_str[0],"失敗"); /* #define FAILURE 0 */
dy.c  463:      strcpy(sf_str[1],"成功"); /* #define SUCCESS 1 */
dy.c  464:      strcpy(sf_str[2],"----"); /* #define ROOT       2 */
dy.c  465:      strcpy(sf_str[3],"無効"); /* #define INVALID 3 */
dy.c  466:
dy.c  467:      if(myid==hostid){
dy.c  468:            printf("¥n モンテカルロ法¥n¥n");
dy.c  469:      }
dy.c  470:
dy.c  471:      current_method = 'M';
dy.c  472:
dy.c  473:      if(myid==hostid){
dy.c  474: #ifdef DYNORMO
dy.c  475: #ifdef DYDEBUG
dy.c  476:            printf("詳細版¥n¥n");
dy.c  477: #endif
dy.c  478: #endif
dy.c  479:
dy.c  480: #ifdef DYNORMO
dy.c  481: #ifndef DYDEBUG
dy.c  482:            printf("簡易版¥n¥n");
dy.c  483: #endif
dy.c  484: #endif
dy.c  485:
dy.c  486: #ifndef DYNORMO
dy.c  487: #ifndef DYDEBUG
dy.c  488:            printf("超簡易版¥n¥n");
dy.c  489: #endif
dy.c  490: #endif
dy.c  491:      }
dy.c  492:
dy.c  493:      if(myid==hostid){
dy.c  494:
dy.c  495:            rst_file_PLOHS[0]     = 0;
dy.c  496:            rst_file_PLOHSNH[0] = 0;
dy.c  497:            rst_file_LORL[0]      = 0;
dy.c  498:
```

```
dy.c  499:        strcpy(rst_file_PLOHS     , getenv("DRFPLOHS"));
dy.c  500:        if ( strlen(rst_file_PLOHS) == 0 ){
dy.c  501:           strcpy(rst_file_PLOHS,"copd/data/PLOHS.rstf");
dy.c  502:        }
dy.c  503:
dy.c  504:        strcpy(rst_file_PLOHSNH , getenv("DRFPLOHSNH"));
dy.c  505:        if ( strlen(rst_file_PLOHSNH) == 0 ){
dy.c  506:           strcpy(rst_file_PLOHSNH,"copd/data/PLOHS.NHrstf");
dy.c  507:        }
dy.c  508:
dy.c  509:        strcpy(rst_file_LORL      , getenv("DRFLORL"));
dy.c  510:        if ( strlen(rst_file_LORL) == 0 ){
dy.c  511:           strcpy(rst_file_LORL,"copd/data/LORL.rstf");
dy.c  512:        }
dy.c  513:
dy.c  514:        printf("時刻(秒)=...              :ﾘｽﾀｰﾄﾌｧｲﾙに記録されている時刻(秒).起因事象発生時刻を 0 秒とする.¥n");
dy.c  515:        printf("ｼﾐｭﾚｰｼｮﾝ時刻              :事象推移で問題にしている時刻(秒).起因事象発生時刻を 0 秒とする.¥n");
dy.c  516:        printf("PSF1                      :漏洩時に炉内 Na 液位から求められた PSF.¥n");
dy.c  517:        printf("PSF2                      :R/V 出口 Na 温度から求められた PSF.¥n");
dy.c  518:        printf("LHS                       :ﾗﾃﾝﾊｲﾊﾟｰｷｭｰﾌﾞｻﾝﾌﾟﾘﾝｸﾞを使用する(Y),使用しない(N).¥n");
dy.c  519:        printf("ISEED                     :乱数の種. -1 で日付から求められる.¥n");
dy.c  520:     }
dy.c  521:
dy.c  522:     if(myid==hostid){
dy.c  523:        if ( argc > 1 ) strcpy(argv80,argv[1]);
dy.c  524:        trim(argv80);
dy.c  525:
dy.c  526:        printf("¥n");
dy.c  527:
dy.c  528:        /***** read IE,CUTOFF PROB,etc.          *****/
dy.c  529:        read_namlst_1st(argc,argv80);          /* -> dy_read.c   */
dy.c  530:     }
dy.c  531:
dy.c  532:     if ( iseed == -9999 ){
dy.c  533:        if(myid==hostid){
dy.c  534:           printf("-9999 が指定されたので、乱数の種を設定しませんでした.¥n",iseed);
dy.c  535:        }
dy.c  536:     }
dy.c  537:     else{
dy.c  538:        if ( iseed < 0 ){
dy.c  539:           if(myid==hostid){
dy.c  540:              iseed = time( NULL );
```

```
dy.c  541:          }
dy.c  542:        }
dy.c  543:        MPI_Bcast(&iseed, 1, MPI_INT, hostid, MPI_COMM_WORLD);
dy.c  544: /*
dy.c  545:        iseed = time( NULL )*(myid+1);
dy.c  546: */
dy.c  547:        srand48( iseed ); /* Set seed */
dy.c  548:        if(myid==hostid){
dy.c  549:          printf("乱数の種に%d を設定しました.¥n",iseed);
dy.c  550:        }
dy.c  551:      }
dy.c  552:      iLHScount = 0;/* dy_hep.c の LHS 内部的に使うカウンター */
dy.c  553:
dy.c  554:      if(myid==hostid){
dy.c  555:        /***** read VALUE data file.                    *****/
dy.c  556:        ecv_read();                                     /* -> dy_mcsrd.c */
dy.c  557:
dy.c  558:        /***** Read & Sort & Print the mcs data *****/
dy.c  559:        mcsdrd();
dy.c  560:
dy.c  561:        /***** SORT MCS                                 *****/
dy.c  562:        sortmcutset();
dy.c  563:
dy.c  564:        /* MCS カテゴリーデータの読み込み */
dy.c  565:        pre_srch_ec_cat();
dy.c  566:
dy.c  567:        rm_old_sdf_file();
dy.c  568:
dy.c  569:        rstfile_remove_eo();
dy.c  570:
dy.c  571:        t_mark2 = time(&timeX);
dy.c  572:        printf("基本データ読み込み終了        CPU 経過時間 = %d¥n",t_mark2-t_mark1);
dy.c  573:      }
dy.c  574:
dy.c  575: #ifdef PARALLEL
dy.c  576:      dist_init_data();
dy.c  577: #endif
dy.c  578:
dy.c  579:      cmt[0] = 0;
dy.c  580:
dy.c  581:      if(myid==hostid){
dy.c  582:        if ( ( fp_re = fopen("output/dym.out","w+") ) == NULL ){
```

```
dy.c  583:          printf("file create error!!! [%s]¥n","output/dym.out");
dy.c  584:          /* mpi_abort */
dy.c  585:          exit(1);
dy.c  586:       };
dy.c  587:       fprintf(fp_re,"ｶｯﾄｾｯﾄ, 従来の発生頻度(使命時間%5.1f 時間), EOP を考慮した発生頻度, NRF¥n",mis_time);
dy.c  588:
dy.c  589:       /* 従来の発生頻度の書き出し */
dy.c  590:       if ( ( fp_sdfx = fopen("output/dy.sdf.x","w") ) == NULL ){
dy.c  591:          printf("file create error!!! [output/dy.sdf.x]¥n");
dy.c  592:          /* mpi_abort */
dy.c  593:          exit(1);
dy.c  594:       };
dy.c  595:
dy.c  596:       fprintf(fp_sdfx,"#MC¥n");
dy.c  597:       fprintf(fp_sdfx,"#1 %c¥n", mcs_set[m_indx[0]].stat[1]);
dy.c  598:    }
dy.c  599:    ro = rn = 0.0;
dy.c  600:
dy.c  601:    /*-----------------------------------*/
dy.c  602:
dy.c  603:    for ( i = 0 ; i < n_of_mcutset ; i++ ){
dy.c  604:
dy.c  605:       n_of_seq = 0;
dy.c  606:
dy.c  607:       if(myid==hostid){
dy.c  608:          sdf_file_open(i);
dy.c  609:       }
dy.c  610:
dy.c  611:       org_value = 1.0;
dy.c  612:       for(j=0;j<mcs_set[m_indx[i]].nUR;j++) org_value *= mis_time;
dy.c  613:       org_value = mcs_set[m_indx[i]].p_U * ie_freqncy * org_value;
dy.c  614:
dy.c  615:       if(myid==hostid){
dy.c  616:          printf("従来の発生頻度(IE*MCS;使命時間%5.1f 時間)=%10.3e¥n", mis_time, org_value);
dy.c  617:          fprintf(fp_sdfx,"#P%15.8e%15.8¥n", org_value,frq_calc_factor);
dy.c  618:
dy.c  619:          fflush(fp_sdfx);
dy.c  620:       }
dy.c  621:
dy.c  622:       exec_sim(i);
dy.c  623:
dy.c  624:       t_m4 = time(&timeX);
```

```
dy.c  625:
dy.c  626: #ifdef PARALLEL
dy.c  627:        MPI_Reduce(&n_CM_occur, &tot_n_CM_occur, 1, MPI_DOUBLE, MPI_SUM, hostid, MPI_COMM_WORLD);
dy.c  628:        n_CM_occur = tot_n_CM_occur;
dy.c  629: #endif
dy.c  630:
dy.c  631:        if(myid==hostid){
dy.c  632:           printf("(frq_calc_factor*n_CM_occur=%9.2e)¥n",frq_calc_factor*n_CM_occur);
dy.c  633:
dy.c  634:           printf("炉心損傷回数(n_CM_occur)            =%9.2e¥n", n_CM_occur);
dy.c  635:
dy.c  636:           if ( mcs_set[m_indx[i]].nUR == 0 ){
dy.c  637:              printf("バイアス値(biasv)                   =無し(UR が 0 なので)¥n");
dy.c  638:           }
dy.c  639:           else{
dy.c  640:              printf("バイアス値(biasv)                   =¥n");
dy.c  641:              for(j=0;j<mcs_set[m_indx[i]].nUR;j++){
dy.c  642:                 printf("%9.2e ",biasv[j]);
dy.c  643:              }
dy.c  644:              printf("¥n");
dy.c  645:           }
dy.c  646:
dy.c  647:           if ( dbias > 0.0 ){
dy.c  648:              printf("(診断実施時刻を求めるための診断実施確率にバイアス値が使われています.)¥n");
dy.c  649:           }
dy.c  650:
dy.c  651:           printf("UR の数(nUR)                          =%9d¥n",     mcs_set[m_indx[i]].nUR);
dy.c  652:
dy.c  653:           printf("試行回数(ntryx)                       =%9d¥n",     g_ntryx);
dy.c  654:
dy.c  655:           printf("起因事象発生発生頻度(ie_freqncy)=%9.2e¥n", ie_freqncy);
dy.c  656:
dy.c  657:           printf("MCS の UA 値(p_UA)                    =%9.2e¥n", mcs_set[m_indx[i]].p_UA);
dy.c  658:
dy.c  659:           r1 = 1.0;
dy.c  660:           for(j=0;j<mcs_set[m_indx[i]].nUR;j++) r1 *= biasv[j];
dy.c  661:           printf("(Πbiasv=%9.2e)¥n",r1);
dy.c  662:
dy.c  663:           x = ( n_CM_occur * r1 ) / (double)g_ntryx;
dy.c  664:
dy.c  665:           printf("発生確率(x=n_CM_occur*(Πbiasv)/ntryx)=%9.2e¥n",x);
dy.c  666:
```

```
dy.c  667:          x *= mcs_set[m_indx[i]].p_UA;
dy.c  668:
dy.c  669:          printf("MCS の UA 値を乗じた値(x=x*p_UA)     =%9.2e¥n",x);
dy.c  670:
dy.c  671:          r3 = x * ie_freqncy;
dy.c  672:          printf("¥n モンテカルロ法で求めた発生頻度(x=x*ie_freqncy)=%10.3e¥n", r3);
dy.c  673:
dy.c  674:          r2 = 1.0;
dy.c  675:          for(j=0;j<mcs_set[m_indx[i]].nUR;j++) r2 *= mis_time;
dy.c  676:          y = mcs_set[m_indx[i]].p_U * ie_freqncy * r2;
dy.c  677:          printf("¥n 従来の発生頻度(OLD=IE*MCS;使命時間%5.1f 時間)=%10.3e¥n", mis_time, y);
dy.c  678:
dy.c  679:          pr_info_Z();
dy.c  680:
dy.c  681:          printf("¥n¥n");
dy.c  682:          printf("従来の発生頻度(使命時間%5.1f 時間) (A) :%10.3e¥n",mis_time, y);
dy.c  683:          printf("EOP を考慮した発生頻度                    (B) :%10.3e¥n",r3);
dy.c  684:          printf("NRF                                    (B/A) :%10.3e¥n",r3/y);
dy.c  685:
dy.c  686:          fprintf(fp_re,"%10.3e %10.3e %10.3e¥n", y, r3, r3/y);
dy.c  687:
dy.c  688:          rn += y;
dy.c  689:          ro += r3;
dy.c  690:
dy.c  691:          fclose(fp_sd);
dy.c  692:
dy.c  693:        }
dy.c  694:      }
dy.c  695:
dy.c  696:      if(myid==hostid){
dy.c  697:
dy.c  698:        fclose(fp_sdfx);
dy.c  699:
dy.c  700:        /*-----------------------------------*/
dy.c  701:
dy.c  702:        fprintf(fp_re,"-----------------------------------------------------------------¥n ");
dy.c  703:        fprintf(fp_re,"合計¥n");
dy.c  704:        fprintf(fp_re,"EOP 評価なし=%10.3e EOP 評価あり=%10.3e NRF(あり/なし)=%10.3e¥n", rn, ro, ro/rn);
dy.c  705:
dy.c  706:        fclose(fp_re);
dy.c  707:
dy.c  708:        t_mark3 = time(&timeX);
```

```
dy.c  709:        printf("¥n 全計算終了                    CPU 経過時間 = %d¥n",t_mark3-t_mark1);
dy.c  710:        printf("¥n= %d, %d , %d¥n",t_mark2-t_mark1, t_m4-t_mark2, t_mark3-t_m4);
dy.c  711:        printf("¥f");
dy.c  712:    }
dy.c  713:
dy.c  714: #ifdef PARALLEL
dy.c  715:    MPI_Finalize();
dy.c  716: #endif
dy.c  717:    return 0;
dy.c  718: }
```

A.2 ソースプログラム dy_dst.c

```
dy_dst.c    1: /*-----------------------------------------------------------------------*/
dy_dst.c    2: /* void dist_init_data()                                                 */
dy_dst.c    3: /*       プロセス 0 で初期化した外部変数を全プロセスに放送                 */
dy_dst.c    4:
dy_dst.c    5: #include "mpi.h"
dy_dst.c    6: #include <stdio.h>
dy_dst.c    7: #include <stdlib.h>
dy_dst.c    8: #include <math.h>
dy_dst.c    9: #include <string.h>
dy_dst.c   10: #include <memory.h>
dy_dst.c   11: #include <ctype.h>
dy_dst.c   12: #include <sys/types.h>
dy_dst.c   13: #include <sys/time.h>
dy_dst.c   14: #include "dy.h"
dy_dst.c   15:
dy_dst.c   16: /*-----------------------------------------------------------------------*/
dy_dst.c   17: void dist_init_data(){
dy_dst.c   18:     int i, j;
dy_dst.c   19:
dy_dst.c   20:     MPI_Bcast(&current_method,              1, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c   21: /*
dy_dst.c   22: EXTERN FILE      *fp_sd;
dy_dst.c   23: EXTERN FILE      *fp_re;
dy_dst.c   24: */
dy_dst.c   25:     MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c   26:     MPI_Bcast(&mis_time,                    1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   27:     MPI_Bcast(&trace_max,                   1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   28:     MPI_Bcast(&frq_calc_factor,             1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   29:     MPI_Bcast(&sim_time_start_point,        1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   30:     MPI_Bcast(&copd_calc_deltaAA,           1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   31:     MPI_Bcast(&copd_calc_deltaFN,           1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   32:     MPI_Bcast(&copd_calc_deltaRF,           1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   33:     MPI_Bcast(&taRVNaO,                     1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   34:     MPI_Bcast(&tbRVNaO,                     1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   35:     MPI_Bcast(&tcRVNaO,                     1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   36:     MPI_Bcast(&taACSNaO,                    1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c   37:     MPI_Bcast(&tbACSNaO,                    1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
```

```
dy_dst.c  38:  MPI_Bcast(&tcACSNaO,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  39:  MPI_Bcast(&faACSNaO,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  40:  MPI_Bcast(&fbACSNaO,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  41:  MPI_Bcast(&fcACSNaO,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  42:  MPI_Bcast(&faPLOOPNa,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  43:  MPI_Bcast(&fbPLOOPNa,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  44:  MPI_Bcast(&fcPLOOPNa,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  45:  MPI_Bcast(&fbPLOOPNa2,              1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  46:  MPI_Bcast(&raPPUMP,                 1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  47:  MPI_Bcast(&rbPPUMP,                 1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  48:  MPI_Bcast(&rcPPUMP,                 1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  49:  MPI_Bcast(&f_CORE,                  1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  50:  MPI_Bcast(&min_Na_level,            1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  51:  MPI_Bcast(&faSLOOPNa,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  52:  MPI_Bcast(&fbSLOOPNa,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  53:  MPI_Bcast(&fcSLOOPNa,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  54:  MPI_Bcast(&hNaLEVEL,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  55:  MPI_Bcast(&hNaLEVELGV,              1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  56:  MPI_Bcast(&fNaLeak,                 1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  57:  MPI_Bcast(&prev_dTdt,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  58:  MPI_Bcast(&dTdt,                    1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  59:  MPI_Bcast(&dTdt_dTdt,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  60:  MPI_Bcast(&prev_taRVNaO,            1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  61:  MPI_Bcast(&copd_mid_sim_width,      1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  62:  MPI_Bcast(&air_rate,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  63:  MPI_Bcast(&acs_nc_na_out_fac,       1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  64:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  65:  MPI_Bcast(&ope_G,                   1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  66:  MPI_Bcast(&t_UR_occur,         400000, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  67:  MPI_Bcast(&n_MCS_not_occur,         1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  68:  MPI_Bcast(&n_MCS_occur,             1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  69:  MPI_Bcast(&n_CM_occur,              1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  70:  MPI_Bcast(&ocr_indexG,         400000, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  71:  MPI_Bcast(&t_UR_occur_v,       100000, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  72:  MPI_Bcast(&current_aprob,           1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  73:  MPI_Bcast(&x_CM_o_list,           100, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  74:  MPI_Bcast(&n_x_CM_o_list,           1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  75:  MPI_Bcast(mcs_top_gate_name,       17, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  76:  MPI_Bcast(mcs_cat_code,             6, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  77:  MPI_Bcast(ope_validation,          21, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  78:  MPI_Bcast(&scenario_ID,             1, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  79:  MPI_Bcast(loop_status,             32, MPI_CHAR   , hostid, MPI_COMM_WORLD);
```

```
dy_dst.c  80:  MPI_Bcast(pv,                        160, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  81:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  82:  MPI_Bcast(pv_hist,          PVHISTMAX*160, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  83:  MPI_Bcast(&pv_hist_stkptr,             1, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  84:  MPI_Bcast(pv_hist_stkptr_hist,       150, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  85:  MPI_Bcast(&n_of_seq,                   1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  86:  MPI_Bcast(case_id,                   101, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  87:  MPI_Bcast(cmt,             MAX_CMT_LENGTH, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  88:  MPI_Bcast(cmt0,            MAX_CMT_LENGTH, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  89:  MPI_Bcast(val_f_name,                 81, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  90:  MPI_Bcast(mcs_f_name,                 81, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  91:  MPI_Bcast(act_f_name,                 81, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  92:  MPI_Bcast(cdf_f_name,                 81, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  93:  MPI_Bcast(cdf_param,                 810, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  94:  MPI_Bcast(current_path,              512, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  95:  MPI_Bcast(path_stack,              25000, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  96:  MPI_Bcast(&path_stack_ctr,             1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  97:  MPI_Bcast(sf_str,                     20, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  98:  MPI_Bcast(copd_rstf11,               320, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  99:  MPI_Bcast(copd_rstf41,               320, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c 100:  MPI_Bcast(ope_code_set,              120, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c 101:  MPI_Bcast(&n_of_ope_code_set,          1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 102:  MPI_Bcast(&copdcntLMT,                 1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 103:  MPI_Bcast(&n_try,                      1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 104:  MPI_Bcast(&ntryx,                      1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 105:  MPI_Bcast(&i__try,                     1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 106:  MPI_Bcast(&i_try_loop_extern,          1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 107:  MPI_Bcast(&ieno,                       1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 108:  MPI_Bcast(&iseed,                      1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 109:  MPI_Bcast(&iLHScount,                  1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 110:  MPI_Bcast(&flg_LHS,                    1, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c 111:  MPI_Bcast(&flg_APX,                    1, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c 112:  MPI_Bcast(&n_of_cls,                   1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 113:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c 114: /*
dy_dst.c 115:  MPI_Bcast(&cpd,              sizeof(cpd), MPI_BYTE   , hostid, MPI_COMM_WORLD);
dy_dst.c 116:  MPI_Bcast(&cls,              sizeof(cls), MPI_BYTE   , hostid, MPI_COMM_WORLD);
dy_dst.c 117:  MPI_Bcast(&omg,              sizeof(omg), MPI_BYTE   , hostid, MPI_COMM_WORLD);
dy_dst.c 118: */
dy_dst.c 119:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c 120:
dy_dst.c 121:  MPI_Bcast(&omgCur,                     1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
```

```
dy_dst.c 122:   MPI_Bcast(&cur_omg_trace,          1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 123:   MPI_Bcast(&omgCur,                 1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 124:   MPI_Bcast(&cur_omg_trace,          1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 125:   MPI_Bcast(&dbias,                  1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 126:   MPI_Bcast(&ie_freqncy,             1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 127:   MPI_Bcast(&prblimt,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 128:   MPI_Bcast(&tuplimt,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 129:   MPI_Bcast(&tdnlimt,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 130:   MPI_Bcast(&elpslimt,               1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 131:   MPI_Bcast(&delta_t,                1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 132:   MPI_Bcast(&biasv,                 12, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 133:
dy_dst.c 134: /*
dy_dst.c 135: EXTERN time_t   t_mark1,t_mark2,t_mark3;
dy_dst.c 136: */
dy_dst.c 137:   MPI_Bcast(&n_of_mcutset,          1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 138:   MPI_Bcast(&n_of_valdat,            1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 139:   MPI_Bcast(&m_indx,          MAX_MCUTSET, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 140:   MPI_Bcast(&flgKT,                  1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 141:   MPI_Bcast(&flgKTonly,              1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 142:   MPI_Bcast(&flg1NCon,               1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 143:   MPI_Bcast(&flgVCLOSE,              1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 144:   MPI_Bcast(&flgPMTok,               1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 145:   MPI_Bcast(&flgJstart,              1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 146:   MPI_Bcast(fnECC,                  81, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c 147:   MPI_Bcast(&flgPSOSon,              1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c 148:   MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c 149:
dy_dst.c 150:
dy_dst.c 151: /*
dy_dst.c 152: c  MPI_Bcast(&ec_v,          sizeof(ec_v), MPI_BYTE   , hostid, MPI_COMM_WORLD);
dy_dst.c 153:   for(i=0; i < MAX_EC; i++){
dy_dst.c 154:     MPI_Bcast(&ec_v[i].prob   ,          1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c 155:     MPI_Bcast(ec_v[i].ec      ,         16, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c 156:   }
dy_dst.c 157: */
dy_dst.c 158:   {
dy_dst.c 159:         MPI_Datatype Ec_v_type;
dy_dst.c 160:         MPI_Datatype type[2] = {MPI_DOUBLE, MPI_CHAR};
dy_dst.c 161:         int          block[2] = {1, 16};
dy_dst.c 162:         MPI_Aint     disp[2];
dy_dst.c 163:         MPI_Address(ec_v, disp);
```

```
dy_dst.c  164:          MPI_Address(ec_v[0].ec, disp+1);
dy_dst.c  165:          MPI_Type_struct(2, block, disp, type, &Ec_v_type);
dy_dst.c  166:          MPI_Type_commit(&Ec_v_type);
dy_dst.c  167:
dy_dst.c  168:          MPI_Bcast(MPI_BOTTOM, MAX_EC, Ec_v_type, hostid, MPI_COMM_WORLD);
dy_dst.c  169:  }
dy_dst.c  170:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  171:  /*- Begin Broadcast clscdf_set -*/
dy_dst.c  172:  for(i=0; i < n_of_mcutset; i++){
dy_dst.c  173:    MPI_Bcast(mcs_set[i].probUA,      4, MPI_DOUBLE    , hostid, MPI_COMM_WORLD);
dy_dst.c  174:    MPI_Bcast(mcs_set[i].probUR,      4, MPI_DOUBLE    , hostid, MPI_COMM_WORLD);
dy_dst.c  175:    MPI_Bcast(&mcs_set[i].p_UA,       1, MPI_DOUBLE    , hostid, MPI_COMM_WORLD);
dy_dst.c  176:    MPI_Bcast(&mcs_set[i].p_U,        1, MPI_DOUBLE    , hostid, MPI_COMM_WORLD);
dy_dst.c  177:    MPI_Bcast(&mcs_set[i].nUA,        1, MPI_INT       , hostid, MPI_COMM_WORLD);
dy_dst.c  178:    MPI_Bcast(&mcs_set[i].nUR,        1, MPI_INT       , hostid, MPI_COMM_WORLD);
dy_dst.c  179:    MPI_Bcast(&mcs_set[i].c_stat,     1, MPI_CHAR      , hostid, MPI_COMM_WORLD);
dy_dst.c  180:    MPI_Bcast(mcs_set[i].stat,        5, MPI_CHAR      , hostid, MPI_COMM_WORLD);
dy_dst.c  181:    MPI_Bcast(&mcs_set[i].n_TRY,      1, MPI_INT       , hostid, MPI_COMM_WORLD);
dy_dst.c  182:  }
dy_dst.c  183:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  184:  if(myid != hostid){
dy_dst.c  185:    for(i=0; i < n_of_mcutset; i++){
dy_dst.c  186:      for(j=0; j < mcs_set[i].nUA; j++){
dy_dst.c  187:        mcs_set[i].ec_UA[j] = (char *)malloc(sizeof(char)*14);
dy_dst.c  188:      }
dy_dst.c  189:      for(j=0; j < mcs_set[i].nUR; j++){
dy_dst.c  190:        mcs_set[i].ec_UR[j] = (char *)malloc(sizeof(char)*14);
dy_dst.c  191:      }
dy_dst.c  192:    }
dy_dst.c  193:  }
dy_dst.c  194:
dy_dst.c  195:  MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  196:  for(i=0; i < n_of_mcutset; i++){
dy_dst.c  197:    for(j=0; j < mcs_set[i].nUA; j++){
dy_dst.c  198:      MPI_Bcast(mcs_set[i].ec_UA[j],    14, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  199:    }
dy_dst.c  200:    for(j=0; j < mcs_set[i].nUR; j++){
dy_dst.c  201:      MPI_Bcast(mcs_set[i].ec_UR[j],    14, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  202:    }
dy_dst.c  203:  }
dy_dst.c  204:
dy_dst.c  205:  /*- End of Broadcast clscdf_set -*/
```

```
dy_dst.c  206:   MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  207:
dy_dst.c  208:
dy_dst.c  209:   /* MPI_Bcast(&rootAT,          sizeof(rootAT), MPI_BYTE   , hostid, MPI_COMM_WORLD);*/
dy_dst.c  210:   /* MPI_Bcast(&rf_lst,          sizeof(rf_lst), MPI_BYTE   , hostid, MPI_COMM_WORLD);*/
dy_dst.c  211:   /* MPI_Bcast(&current_mcs,sizeof(current_mcs), MPI_BYTE   , hostid, MPI_COMM_WORLD);*/
dy_dst.c  212:
dy_dst.c  213:   /*- Begin Broadcast clscdf_set -*/
dy_dst.c  214:   for(i=1;i<1000;i++){
dy_dst.c  215:     MPI_Bcast(&clscdf_set[i].n_of_cls,       1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  216:   }
dy_dst.c  217:
dy_dst.c  218:   if(myid != hostid){
dy_dst.c  219:     for(i=1;i<1000;i++){
dy_dst.c  220:       if(clscdf_set[i].n_of_cls > 0){
dy_dst.c  221:         clscdf_set[i].cls = (double *)malloc(sizeof(double)*clscdf_set[i].n_of_cls);
dy_dst.c  222:         clscdf_set[i].cpd = (double *)malloc(sizeof(double)*clscdf_set[i].n_of_cls);
dy_dst.c  223:         clscdf_set[i].omg = (double *)malloc(sizeof(double)*clscdf_set[i].n_of_cls);
dy_dst.c  224:       }else{
dy_dst.c  225:         clscdf_set[i].n_of_cls = 0;
dy_dst.c  226:         clscdf_set[i].cls = NULL;
dy_dst.c  227:         clscdf_set[i].cpd = NULL;
dy_dst.c  228:         clscdf_set[i].omg = NULL;
dy_dst.c  229:       }
dy_dst.c  230:     }
dy_dst.c  231:   }
dy_dst.c  232:   MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  233:   for(i=1;i<1000;i++){
dy_dst.c  234:     if(clscdf_set[i].n_of_cls > 0){
dy_dst.c  235:       MPI_Bcast(clscdf_set[i].cls, clscdf_set[i].n_of_cls, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  236:       MPI_Bcast(clscdf_set[i].cpd, clscdf_set[i].n_of_cls, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  237:       MPI_Bcast(clscdf_set[i].omg, clscdf_set[i].n_of_cls, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  238:     }
dy_dst.c  239:   }
dy_dst.c  240:   MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  241:   /*- End of Broadcast clscdf_set -*/
dy_dst.c  242:
dy_dst.c  243: /*
dy_dst.c  244: EXTERN TYPE_OF_ECV       ec_v[MAX_EC];
dy_dst.c  245: EXTERN TYPE_OF_MCSET     mcs_set[MAX_MCUTSET];
dy_dst.c  246: EXTERN TYPE_OF_NODE      *rootAT;
dy_dst.c  247: EXTERN TYPE_OF_RF_LST    *rf_lst[MAX_RF_LST];
```

```
dy_dst.c  248: EXTERN TYPE_OF_MCSET    *current_mcs;
dy_dst.c  249: EXTERN TYPE_OF_CLSCDF   clscdf_set[1001];
dy_dst.c  250:
dy_dst.c  251: */
dy_dst.c  252:    MPI_Bcast(&mcs_on,                  1, MPI_INT    , hostid, MPI_COMM_WORLD);
dy_dst.c  253:    MPI_Bcast(rst_file_PLOHS,         120, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  254:    MPI_Bcast(rst_file_PLOHSNH,       120, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  255:    MPI_Bcast(rst_file_LORL,          120, MPI_CHAR   , hostid, MPI_COMM_WORLD);
dy_dst.c  256:
dy_dst.c  257:    MPI_Bcast(&time_of_NHSstart,        1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  258:    MPI_Bcast(&time_of_NCstart,         1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  259:    MPI_Bcast(&time_of_MCSstart,        1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  260:    MPI_Bcast(&time_of_PMTstart,        1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  261:    MPI_Bcast(&time_of_FC1start,        1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  262:    MPI_Bcast(&time_of_PSOSstart,       1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  263:    MPI_Bcast(&time_of_KTstart,         1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  264:    MPI_Bcast(&time_of_VCstart,         1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  265:    MPI_Bcast(&time_of_MCSvalid,        1, MPI_DOUBLE, hostid, MPI_COMM_WORLD);
dy_dst.c  266:
dy_dst.c  267:    MPI_Barrier(MPI_COMM_WORLD);
dy_dst.c  268: }
```

A.3 ソースプログラム dy_hep.c

```
dy_hep.c    1:
dy_hep.c    2: /* dy_hep.c        <M>              */
dy_hep.c    3: /* 1999.2                          */
dy_hep.c    4:
dy_hep.c    5: #include "dy.h"
dy_hep.c    6: double rand_lf(void);
dy_hep.c    7:
dy_hep.c    8: /*-----------------------------------------------------------------*/
dy_hep.c    9: void search_omg_value( double d_t )
dy_hep.c   10: {                                        /*診断時間からω(omg)の値を求める*/
dy_hep.c   11:    int i;
dy_hep.c   12:
dy_hep.c   13:    for(i=1;i<n_of_cls;i++){
dy_hep.c   14:       if ( cls[i] == d_t ) goto LOOPEXIT;
dy_hep.c   15:    }
dy_hep.c   16:    LOOPEXIT:;
dy_hep.c   17:    omgCur = omg[i];
dy_hep.c   18:    printf("(i=%3d omg[i]=%9.2e)¥n", i, omg[i] );
dy_hep.c   19: }
dy_hep.c   20:
dy_hep.c   21: /*-----------------------------------------------------------------*/
dy_hep.c   22: double rcdf( double x )
dy_hep.c   23: {
dy_hep.c   24:    int i;
dy_hep.c   25:
dy_hep.c   26:    for(i=1;i<n_of_cls;i++){
dy_hep.c   27:       if ( x < cpd[i] ) goto LOOPEXIT;
dy_hep.c   28:    }
dy_hep.c   29:    LOOPEXIT:;
dy_hep.c   30:    if ( i == n_of_cls ){
dy_hep.c   31:       return ceil( cls[n_of_cls-1] );
dy_hep.c   32: /*
dy_hep.c   33:       return cls[n_of_cls-1];
dy_hep.c   34: */
dy_hep.c   35:    }
dy_hep.c   36:    else{
```

```
dy_hep.c   37:      return ceil( (cls[i]-cls[i-1])*(x-cpd[i-1])/(cpd[i]-cpd[i-1])+cls[i-1] );
dy_hep.c   38: /*
dy_hep.c   39:      return ( (cls[i]-cls[i-1])*(x-cpd[i-1])/(cpd[i]-cpd[i-1])+cls[i-1] );
dy_hep.c   40: */
dy_hep.c   41:    }
dy_hep.c   42:
dy_hep.c   43:    /* 1999.2.7 : tam : ceil を止めてみた */
dy_hep.c   44: }
dy_hep.c   45:
dy_hep.c   46: /*-----------------------------------------------------------------*/
dy_hep.c   47: double rLHS( int *lc, int kkk )
dy_hep.c   48: {
dy_hep.c   49:    const int nofcell = 100;
dy_hep.c   50:    int      i, i0, i1;
dy_hep.c   51:    double xdum, prb, r;
dy_hep.c   52:
dy_hep.c   53:    if( kkk == 1 ) for( i=0;i<nofcell;i++ ) lc[i] = i;
dy_hep.c   54:    xdum = rand_lf();
dy_hep.c   55: /*xdum = drand48(); */
dy_hep.c   56:    i0     = (int)( xdum * (double)(nofcell-kkk+1) );
dy_hep.c   57:    prb    = rand_lf() / (double)(nofcell);
dy_hep.c   58: /*prb    = drand48() / (double)(nofcell); */
dy_hep.c   59:    i1     = i0;
dy_hep.c   60:    if( i1 <= 0 ) i1 = 0;
dy_hep.c   61:    if( i1 >= nofcell - kkk ) i1 = nofcell - kkk;
dy_hep.c   62:    r = (double)lc[i1] / (double)nofcell + prb;
dy_hep.c   63:    lc[i1] = lc[nofcell-kkk];
dy_hep.c   64:    return r;
dy_hep.c   65: }
dy_hep.c   66:
dy_hep.c   67: /*-----------------------------------------------------------------*/
dy_hep.c   68: double rndLHS( void )
dy_hep.c   69: {
dy_hep.c   70:    const int nofcell = 101;
dy_hep.c   71:    static double r_pool[101];
dy_hep.c   72:    static int      latnv[101];
dy_hep.c   73:    int      j;
dy_hep.c   74:
dy_hep.c   75:    iLHScount++;
dy_hep.c   76:    if ( iLHScount >= nofcell ) iLHScount = 1;
dy_hep.c   77:    if ( iLHScount == 1 ){
dy_hep.c   78:      for( j=1;j<nofcell;j++ ){
```

```
dy_hep.c   79:          r_pool[j] = rLHS( latnv , j );
dy_hep.c   80: #ifdef DYDEBUG
dy_hep.c   81:          printf("###乱数を予め求める###%3d %15.6e¥n", j, r_pool[j]);
dy_hep.c   82: #endif
dy_hep.c   83:       }
dy_hep.c   84:    }
dy_hep.c   85: #ifdef DYNORMO
dy_hep.c   86:    printf( "random number=%13.6e¥n", r_pool[iLHScount] );
dy_hep.c   87: #endif
dy_hep.c   88:    return r_pool[iLHScount];
dy_hep.c   89: }
dy_hep.c   90:
dy_hep.c   91: /*-----------------------------------------------------------------*/
dy_hep.c   92: void calc_D_hep(TYPE_OF_ITEM *x, double *hep, int depend)
dy_hep.c   93: {
dy_hep.c   94:    double r,v,t;
dy_hep.c   95:    int      i,j;
dy_hep.c   96:
dy_hep.c   97:    /* 正しい診断が下される時間をもとめる */
dy_hep.c   98:
dy_hep.c   99:    if ( depend == 1 ){
dy_hep.c  100:       j = x->depD_cdf_id;
dy_hep.c  101:    }
dy_hep.c  102:    else{
dy_hep.c  103:       j = x->d_cdf_id;
dy_hep.c  104:    }
dy_hep.c  105:    n_of_cls = clscdf_set[j].n_of_cls;
dy_hep.c  106: #ifdef DYNORMO
dy_hep.c  107:    printf("(診断 CDF/ID=%d, N_of_C=%d)¥n", j, n_of_cls);
dy_hep.c  108: #endif
dy_hep.c  109:    if ( n_of_cls == 0 ){
dy_hep.c  110:       printf("Error : CDF ﾃﾞｰﾀの数が 0 です.(j=%d)¥n",j);
dy_hep.c  111:       if ( clscdf_set[j].cls == NULL ){
dy_hep.c  112:          printf("Error : clscdf_set[j].cls==NULL¥n");
dy_hep.c  113:       }
dy_hep.c  114:       if ( clscdf_set[j].cls == NULL ){
dy_hep.c  115:          printf("Error : clscdf_set[j].cls==NULL¥n");
dy_hep.c  116:       }
dy_hep.c  117:       exit(1);
dy_hep.c  118:    }
dy_hep.c  119:    else{
dy_hep.c  120: #ifdef DYNORMO
```

```
dy_hep.c  121:        printf("(CDF ﾃﾞｰﾀの数は%d)¥n",n_of_cls);
dy_hep.c  122: #endif
dy_hep.c  123:     }
dy_hep.c  124:     cls = clscdf_set[j].cls;
dy_hep.c  125:     cpd = clscdf_set[j].cpd;
dy_hep.c  126:     omg = clscdf_set[j].omg;
dy_hep.c  127:     if ( cls == NULL ){
dy_hep.c  128:        printf("Error : cls==NULL¥n");
dy_hep.c  129:        exit(1);
dy_hep.c  130:     }
dy_hep.c  131:     if ( cpd == NULL ){
dy_hep.c  132:        printf("Error : cpd==NULL¥n");
dy_hep.c  133:        exit(1);
dy_hep.c  134:     }
dy_hep.c  135:     /*printf("CLS(%3d)=%12.5e CPD(%3d)=%12.5e¥n",i , cls[i] , i , cpd[i]);*/
dy_hep.c  136:
dy_hep.c  137:     /* rcdf : 一様乱数を受け取り、累積確率分布に従った分布に変換する */
dy_hep.c  138:
dy_hep.c  139:     if ( flg_LHS == 'Y' ){
dy_hep.c  140:        x->vt_diag = rcdf( rndLHS() );
dy_hep.c  141:     }
dy_hep.c  142:     else{
dy_hep.c  143:        r = rand_lf();
dy_hep.c  144: /*   r = drand48(); */
dy_hep.c  145: #ifdef DYNORMO
dy_hep.c  146:        printf( "random number=%13.6e¥n", r );
dy_hep.c  147: #endif
dy_hep.c  148:        x->vt_diag = rcdf( r );
dy_hep.c  149:     }
dy_hep.c  150:
dy_hep.c  151:     /* モンテカルロでは hepD は考慮しない                    */
dy_hep.c  152:     /* 診断には失敗しない。だた時間が多く掛かるだけ */
dy_hep.c  153:
dy_hep.c  154:     if ( dbias > 0.0 ){
dy_hep.c  155:        search_omg_value( x->vt_diag );
dy_hep.c  156: #ifdef DYNORMO
dy_hep.c  157:        printf("診断所要時間(分):%9.2e  ﾊﾞｲｱｽ法調整値:%9.2e¥n",
dy_hep.c  158:                                    x->vt_diag,          omgCur);
dy_hep.c  159: #endif
dy_hep.c  160:     }
dy_hep.c  161:     else{
dy_hep.c  162: #ifdef DYNORMO
```

```
dy_hep.c  163:      printf("診断所要時間(分):%9.2e¥n", x->vt_diag);
dy_hep.c  164: #endif
dy_hep.c  165:    }
dy_hep.c  166: }
dy_hep.c  167:
dy_hep.c  168: /*-----------------------------------------------------------------*/
dy_hep.c  169: void calc_O_hep(TYPE_OF_ITEM *x, double *hep, int tgtsubpath)
dy_hep.c  170: {
dy_hep.c  171:    double v,t,hNaL,psf1,psf2;
dy_hep.c  172:    int      i,j;
dy_hep.c  173:
dy_hep.c  174:    j            = x->eo_cdf_id[tgtsubpath];
dy_hep.c  175:    n_of_cls = clscdf_set[j].n_of_cls;
dy_hep.c  176: #ifdef DYNORMO
dy_hep.c  177:    printf("(操作 CDF/ID=%d, N_of_C=%d)¥n", j, n_of_cls);
dy_hep.c  178: #endif
dy_hep.c  179:    cls = clscdf_set[j].cls;
dy_hep.c  180:    cpd = clscdf_set[j].cpd;
dy_hep.c  181:    omg = clscdf_set[j].omg;
dy_hep.c  182:    /*printf("CLS(%3d)=%12.5e CPD(%3d)=%12.5e¥n",i , cls[i] , i , cpd[i]);*/
dy_hep.c  183:
dy_hep.c  184:    /* rcdf : 一様乱数を受け取り、累積確率分布に従った分布に変換する */
dy_hep.c  185:
dy_hep.c  186:    if ( flg_LHS == 'Y' ){
dy_hep.c  187:       x->vt_ope = rcdf( rndLHS() );
dy_hep.c  188:    }
dy_hep.c  189:    else{
dy_hep.c  190:       v = rand_lf();
dy_hep.c  191: /*   v = drand48(); */
dy_hep.c  192: #ifdef DYNORMO
dy_hep.c  193:       printf( "random number=%13.6e¥n", v );
dy_hep.c  194: #endif
dy_hep.c  195:       x->vt_ope = rcdf( v );
dy_hep.c  196:    }
dy_hep.c  197:
dy_hep.c  198:    *hep = x->p_opeFailure[tgtsubpath];
dy_hep.c  199:
dy_hep.c  200:    hNaL = hNaLEVEL + 18.25;
dy_hep.c  201:
dy_hep.c  202:    if ( current_mcs->stat[1] == 'L' ){
dy_hep.c  203:       if ( hNaL < 28.15 ){
dy_hep.c  204:          psf1 = x->psf_maxL;
```

```
dy_hep.c  205:        }
dy_hep.c  206:        else{
dy_hep.c  207:          if ( hNaL > 33.05 ){
dy_hep.c  208:            psf1 = 1.0;
dy_hep.c  209:          }
dy_hep.c  210:          else{
dy_hep.c  211:            psf1 = x->psf_maxL - (x->psf_maxL-1.0)/(33.05-28.15) * (hNaL - 28.15);
dy_hep.c  212:          }
dy_hep.c  213:        }
dy_hep.c  214:      }
dy_hep.c  215:      else{
dy_hep.c  216:        psf1 = 1.0;
dy_hep.c  217:      }
dy_hep.c  218:
dy_hep.c  219:      if ( taRVNaO > 650.0 ){
dy_hep.c  220:        psf2 = x->psf_maxT;
dy_hep.c  221:      }
dy_hep.c  222:      else{
dy_hep.c  223:        if ( taRVNaO < 529.0 ){
dy_hep.c  224:          psf2 = 1.0;
dy_hep.c  225:        }
dy_hep.c  226:        else{
dy_hep.c  227:          psf2 = (x->psf_maxT-1.0)/(650.0-529.0) * (taRVNaO - 529.0) + 1.0;
dy_hep.c  228:        }
dy_hep.c  229:      }
dy_hep.c  230:
dy_hep.c  231: #ifdef DYNORMO
dy_hep.c  232:      printf("HEP=%9.2e PSF1=%9.2e PSF2=%9.2e¥n",
dy_hep.c  233:                *hep,        psf1,        psf2);
dy_hep.c  234: #endif
dy_hep.c  235:
dy_hep.c  236:      *hep = *hep * psf1 * psf2;
dy_hep.c  237:
dy_hep.c  238:      if ( *hep > 1.0 ) *hep = 1.0;
dy_hep.c  239:
dy_hep.c  240: #ifdef DYNORMO
dy_hep.c  241:      printf("¥n");
dy_hep.c  242:      printf("操作所要時間(分):%9.2e     操作失敗確率:%9.2e¥n", x->vt_ope, *hep);
dy_hep.c  243: #endif
dy_hep.c  244:
dy_hep.c  245: }
dy_hep.c  246:
```

A.4 ソースプログラム dy_rand.c

```
dy_rand.c    1: /* leapflog 法による乱数生成のエミュレーション*/
dy_rand.c    2:
dy_rand.c    3: #include "mpi.h"
dy_rand.c    4: #include "dy.h"
dy_rand.c    5:
dy_rand.c    6: double drand48(void);
dy_rand.c    7:
dy_rand.c    8: double rand_lf(void){
dy_rand.c    9:     int i;
dy_rand.c   10:     static int ii=0;
dy_rand.c   11:     double r, r0;
dy_rand.c   12:     for(i = 0; i < nproc1; i++){
dy_rand.c   13:         r = drand48();
dy_rand.c   14:         if(i == myid) r0 = r;
dy_rand.c   15:     }
dy_rand.c   16:
dy_rand.c   17:     return(r0);
dy_rand.c   18: }
```

付録 B

サンプル計算結果(PLOHS シーケンス)
プロセス数 5

2000年03月10日 17時14分35秒
ファイル名=data/dyana.dat.plohs

NAMELIST : DYDATA.

num.of process = 5

------ DYANA9803 : Dynamic Analysis Program for Event Transition. ------ 計算開始日時 Fri Mar 10 17:14:36 2000

(1999/01/31 17:00)

モンテカルロ法

超簡易版

時刻(秒)=... :ﾘｽﾀｰﾄﾌｧｲﾙに記録されている時刻(秒),起因事象発生時刻を 0 秒とする.
ｼﾐｭﾚｰｼｮﾝ時刻 :事象推移で問題にしている時刻(秒),起因事象発生時刻を 0 秒とする.
PSF1 :漏洩時に炉内 Na 液位から求められた PSF.
PSF2 :R/V 出口 Na 温度から求められた PSF.
LHS :ﾗﾃﾝﾊｲﾊﾟｰｷｭｰﾌﾞｻﾝﾌﾟﾘﾝｸﾞを使用する(Y),使用しない(N).
ISEED :乱数の種. -1 で日付から求められる.

```
COMMENT    = PLOHS
APX        = N
IENO       =  7
IEFREQ     =  7.20000e-02
TMISS      =  2.30000e+01
TTRCMAX    =  2.30000e+01
VALFNAME   = data/val.dat
MCSFNAME   = data/cbe1.mcs
ACTFNAME   = data/act.pnc.0812
CDFFNAME   =
CDFPARAM   = 1:o 15.0 3.00 10
             2:o 10.0 3.33 11
             3:o  5.0 3.33 12
             4:d 1.0 0.001 5.0 300.0 Z 100
             5:d 1.0 0.001 5.0 300.0 L 101
             6:d 1.0 0.001 5.0 300.0 M 102
             7:d 1.0 0.001 5.0 300.0 H 103
             8:d 1.0 0.001 5.0 300.0 C 104
             9:
            10:
DBIAS      =  0.00000e+00 (ﾀｲﾑﾒｯｼｭ法では使われない)
PRBLIMIT   =  1.00000e-15
TUPLIMIT   =  6.50000e+02
TDNLIMIT   =  2.50000e+02
MINNALVL   =  2.77500e+01
ICCOUNT    =            12
NTRY       =         20000 (ﾀｲﾑﾒｯｼｭ法では使われない)
LHS        = N             (ﾀｲﾑﾒｯｼｭ法では使われない)
ISEED      =     846211394 (ﾀｲﾑﾒｯｼｭ法では使われない)
DT         =  1.00000e+01 (ﾓﾝﾃｶﾙﾛ法では使われない)
ECCFN      = data/mcs_attr.dat.pnc.0812
```

パラメータデータから確率分布累積値データを作ります.

操作実施確率 : NDF
ID = 10 MEAN = 1.50e+01 S.D. = 3.00e+00

操作実施確率 : NDF
ID = 11 MEAN = 1.00e+01 S.D. = 3.33e+00

操作実施確率 : NDF
ID = 12 MEAN = 5.00e+00 S.D. = 3.33e+00

診断実施確率
ID:100

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
         |          *
         |           *
         |            *
         |             *
1.00e-03 |              *************
         |
         |
         +-----------------------------
                 5.00e+00   3.00e+02          Time(min)
```

Dependency :Z
coefficient:a =-1.687146e+00 b =1.179264e+00

診断実施確率
ID:101

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
         |          *
         |           *
         |            *
         |             *
1.00e-03 |              *************
         |
         |
         +-----------------------------
                 5.00e+00   3.00e+02          Time(min)
```

Dependency :L
coefficient:a =-1.687146e+00 b =1.179264e+00

診断実施確率
ID:102

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
         |          *
         |           *
         |            *
         |             *
1.00e-03 |              *************
         |
         |
         +-----------------------------
                 5.00e+00   3.00e+02          Time(min)
```

Dependency :M
coefficient:a =-1.687146e+00 b =1.179264e+00

診断実施確率
ID:103

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
```

```
            |          *
            |           *
            |            *
            |             *
1.00e-03 |              *************
            |
            |
            +-----------------------------
                        5.00e+00   3.00e+02          Time(min)
Dependency :H
coefficient:a =-1.687146e+00 b =1.179264e+00
```

診断実施確率

```
ID:104
     Prob.|
1.00e+00 |********
            |         *
            |          *
            |           *
            |            *
            |             *
            |              *
1.00e-03 |              *************
            |
            |
            +-----------------------------
                        5.00e+00   3.00e+02          Time(min)
Dependency :C
coefficient:a =-1.687146e+00 b =1.179264e+00
```

累積確率データの読み込み

```
ID                =10
Number of Class=34
Comment           =MEAN= 1.50e+01, S.D.= 3.00e+00
  CLS(  0)= 0.00000e+00 CPD(  0)= 0.00000e+00 OMG(  0)= 1.00000e+00
  CLS(  1)= 0.00000e+00 CPD(  1)= 2.86652e-07 OMG(  1)= 1.00000e+00
  CLS(  2)= 1.00000e+00 CPD(  2)= 1.53063e-06 OMG(  2)= 1.00000e+00
  CLS(  3)= 2.00000e+00 CPD(  3)= 7.34342e-06 OMG(  3)= 1.00000e+00
  CLS(  4)= 3.00000e+00 CPD(  4)= 3.16712e-05 OMG(  4)= 1.00000e+00
  CLS(  5)= 4.00000e+00 CPD(  5)= 1.22866e-04 OMG(  5)= 1.00000e+00
  CLS(  6)= 5.00000e+00 CPD(  6)= 4.29060e-04 OMG(  6)= 1.00000e+00
  CLS(  7)= 6.00000e+00 CPD(  7)= 1.34990e-03 OMG(  7)= 1.00000e+00
  CLS(  8)= 7.00000e+00 CPD(  8)= 3.83038e-03 OMG(  8)= 1.00000e+00
  CLS(  9)= 8.00000e+00 CPD(  9)= 9.81536e-03 OMG(  9)= 1.00000e+00
  CLS( 10)= 9.00000e+00 CPD( 10)= 2.27501e-02 OMG( 10)= 1.00000e+00
  CLS( 11)= 1.00000e+01 CPD( 11)= 4.77904e-02 OMG( 11)= 1.00000e+00
  CLS( 12)= 1.10000e+01 CPD( 12)= 9.12112e-02 OMG( 12)= 1.00000e+00
  CLS( 13)= 1.20000e+01 CPD( 13)= 1.58655e-01 OMG( 13)= 1.00000e+00
  CLS( 14)= 1.30000e+01 CPD( 14)= 2.52493e-01 OMG( 14)= 1.00000e+00
  CLS( 15)= 1.40000e+01 CPD( 15)= 3.69441e-01 OMG( 15)= 1.00000e+00
  CLS( 16)= 1.50000e+01 CPD( 16)= 5.00000e-01 OMG( 16)= 1.00000e+00
  CLS( 17)= 1.60000e+01 CPD( 17)= 6.30559e-01 OMG( 17)= 1.00000e+00
  CLS( 18)= 1.70000e+01 CPD( 18)= 7.47507e-01 OMG( 18)= 1.00000e+00
  CLS( 19)= 1.80000e+01 CPD( 19)= 8.41345e-01 OMG( 19)= 1.00000e+00
  CLS( 33)= 3.20000e+01 CPD( 33)= 1.00000e+00 OMG( 33)= 1.00000e+00
ID                =11
Number of Class=31
Comment           =MEAN= 1.00e+01, S.D.= 3.33e+00
  CLS(  0)= 0.00000e+00 CPD(  0)= 0.00000e+00 OMG(  0)= 1.00000e+00
  CLS(  1)= 0.00000e+00 CPD(  1)= 1.33665e-03 OMG(  1)= 1.00000e+00
  CLS(  2)= 1.00000e+00 CPD(  2)= 3.43891e-03 OMG(  2)= 1.00000e+00
  CLS(  3)= 2.00000e+00 CPD(  3)= 8.14391e-03 OMG(  3)= 1.00000e+00
  CLS(  4)= 3.00000e+00 CPD(  4)= 1.77722e-02 OMG(  4)= 1.00000e+00
  CLS(  5)= 4.00000e+00 CPD(  5)= 3.57883e-02 OMG(  5)= 1.00000e+00
```

```
 CLS(   6)= 5.00000e+00 CPD(   6)= 6.66129e-02 OMG(   6)= 1.00000e+00
 CLS(   7)= 6.00000e+00 CPD(   7)= 1.14837e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 7.00000e+00 CPD(   8)= 1.83821e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 8.00000e+00 CPD(   9)= 2.74053e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 9.00000e+00 CPD( 10)= 3.81974e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.00000e+01 CPD( 11)= 5.00000e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.10000e+01 CPD( 12)= 6.18026e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.20000e+01 CPD( 13)= 7.25947e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.30000e+01 CPD( 14)= 8.16179e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.40000e+01 CPD( 15)= 8.85163e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.50000e+01 CPD( 16)= 9.33387e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.60000e+01 CPD( 17)= 9.64212e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.70000e+01 CPD( 18)= 9.82228e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.80000e+01 CPD( 19)= 9.91856e-01 OMG( 19)= 1.00000e+00
 CLS( 30)= 2.90000e+01 CPD( 30)= 1.00000e+00 OMG( 30)= 1.00000e+00
ID                =12
Number of Class=26
Comment           =MEAN= 5.00e+00, S.D.= 3.33e+00
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 1.00000e+00
 CLS(   1)= 0.00000e+00 CPD(   1)= 6.66129e-02 OMG(   1)= 1.00000e+00
 CLS(   2)= 1.00000e+00 CPD(   2)= 1.14837e-01 OMG(   2)= 1.00000e+00
 CLS(   3)= 2.00000e+00 CPD(   3)= 1.83821e-01 OMG(   3)= 1.00000e+00
 CLS(   4)= 3.00000e+00 CPD(   4)= 2.74053e-01 OMG(   4)= 1.00000e+00
 CLS(   5)= 4.00000e+00 CPD(   5)= 3.81974e-01 OMG(   5)= 1.00000e+00
 CLS(   6)= 5.00000e+00 CPD(   6)= 5.00000e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 6.00000e+00 CPD(   7)= 6.18026e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 7.00000e+00 CPD(   8)= 7.25947e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 8.00000e+00 CPD(   9)= 8.16179e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 9.00000e+00 CPD( 10)= 8.85163e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.00000e+01 CPD( 11)= 9.33387e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.10000e+01 CPD( 12)= 9.64212e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.20000e+01 CPD( 13)= 9.82228e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.30000e+01 CPD( 14)= 9.91856e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.40000e+01 CPD( 15)= 9.96561e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.50000e+01 CPD( 16)= 9.98663e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.60000e+01 CPD( 17)= 9.99522e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.70000e+01 CPD( 18)= 9.99843e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.80000e+01 CPD( 19)= 9.99953e-01 OMG( 19)= 1.00000e+00
 CLS( 25)= 2.40000e+01 CPD( 25)= 1.00000e+00 OMG( 25)= 1.00000e+00
ID                =100
Number of Class=301
Comment           =DEP=Z, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 2.64793e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 4.33160e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 5.47499e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 6.29048e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 6.89459e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 7.35587e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 7.71688e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 8.00529e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 8.23973e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 8.43315e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 8.59480e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 8.73142e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 8.84804e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 8.94847e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 9.99000e-01 OMG(300)= 1.00000e+00
ID                =101
Number of Class=301
```

```
Comment          =DEP=L, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 2.51553e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 4.11502e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 5.20124e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 5.97595e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 6.54986e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 6.98807e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 7.33104e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 7.60503e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 7.82774e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 8.01149e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 8.16506e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 8.29485e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 8.40564e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 8.50105e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 9.49050e-01 OMG(300)= 1.00000e+00
ID                =102
Number of Class=301
Comment          =DEP=M, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 2.26965e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 3.71280e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 4.69285e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 5.39184e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 5.90965e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 6.30503e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 6.61447e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 6.86168e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 7.06263e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 7.22842e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 7.36697e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 7.48407e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 7.58404e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 7.67012e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 8.56286e-01 OMG(300)= 1.00000e+00
ID                =103
Number of Class=301
Comment          =DEP=H, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 1.32396e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 2.16580e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 2.73749e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 3.14524e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 3.44730e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 3.67793e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 3.85844e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 4.00265e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 4.11986e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 4.21658e-01 OMG( 15)= 1.00000e+00
```

```
 CLS( 16)= 1.60000e+01 CPD( 16)= 4.29740e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 4.36571e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 4.42402e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 4.47424e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 4.99500e-01 OMG(300)= 1.00000e+00
ID                =104
Number of Class=301
Comment           =DEP=C, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 0.00000e+00 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 0.00000e+00 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 0.00000e+00 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 0.00000e+00 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 0.00000e+00 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 0.00000e+00 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 0.00000e+00 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 0.00000e+00 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 0.00000e+00 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 0.00000e+00 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 0.00000e+00 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 0.00000e+00 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 0.00000e+00 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 0.00000e+00 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 0.00000e+00 OMG(300)= 1.00000e+00
```

乱数の種に 846211394 を設定しました.

Value Data 一覧(最初の 10 個のみ).

```
5.00e-07 AA005RE01RS01
3.60e-10 AA043PP01CL14
3.60e-08 AA043PP01CP14
2.80e-09 AA044PP01CL14
2.80e-07 AA044PP01CP14
3.20e-09 AA045PP01CL14
3.20e-07 AA045PP01CP14
9.20e-10 AA046PP01CL12
9.20e-08 AA046PP01CP12
4.50e-10 AA047PP01CL12
```

Minimal Cut Set 一覧(最初の 100 個のみ. 全リストは output/mcslist.readin に書き込まれます)

```
Equation ID.=CBE1

    1: AZ401MV01DD33[A]


Minimal Cut Set 総数 =      1
```

ソートされた MCS 一覧(最初の 100 個のみ.全リストは output/mcslist.sorted に書き込まれます.)

```
No.      発生確率(p_U),UA 部分のみの発生確率(p_UA),....
   1 :  2.40e-05   2.40e-05 [UA] AZ401MV01DD33 ---------- ---------- ----------
                                 [UR] ---------- ---------- ---------- ----------
```

操作コードの読み込み.

01NCc　自然循環手動切替(中央制御室)
02NCf　自然循環手動切替(現場)
03PSc　PSOS 手動起動(中央制御室)
04KT　原子炉容器 Na 温度維持操作

05PS　PSOS 故障復旧操作
06MCS　MCS の起動(中央制御室)
07PMTc　M-G ｾｯﾄ一括 CS「停止」位置
08PMTf　M-G ｾｯﾄ界磁遮断器「切り」
09VC1　MV30 と MV31 を閉(中央制御室)
10VC2　MV26 と MV100 を閉(中央制御室)
11VCf　MV30 を手動閉(現場)

基本データ読み込み終了　　CPU 経過時間 ＝24
従来の発生頻度(IE*MCS;使命時間 23.0 時間)= 1.728e-06

対象 MCS==========
番号　　発生頻度(IE*UA*UR)　Event Code[UA],[UR]
　1 ::　1.73e-06 AZ401MV01DD33 ---------- ---------- ----------
　　---------- ---------- ---------- ----------

(上記発生頻度は UR が使命時間1時間の値を使用.)

(buff=CBE　OO---O-----AZ401MV01DD33 +)
(buff=CBE1 OO---O-----AZ401MV01DD33 +)
(ope_validation=OO---O-----)
(mcs_cat_code=CBE1)
(mcs_top_gate_name=CBE1)

UR はありません.

無発生回数　0.0000e+00
発生回数　　2.0000e+04

ｶﾃｺﾞﾘｰｺｰﾄﾞ CBE1
(frq_calc_factor= 8.64e-11)

目的のｱｸｼｮﾝﾂﾘｰの ID:P

**** ｱｸｼｮﾝﾂﾘｰﾃﾞｰﾀの読み込み ****
SN=　1
　直前操作 SN　　=　0
　直前操作状況　=----
　操作 1　　=有効
　ｺﾒﾝﾄ　　=ACS NC 手動切り替え(中制)
　次診断移行時間=　6.00e+01
　従属性診断 ID　=　　0
　診断確率 ID　　=　100
　移動 1 時間　　=　0.00e+00
　操作 1 ｺｰﾄﾞ　　=　NCc
　操作 1 確率 ID　　=　　10
　操作 1 失敗確率 =　1.00e-02
　液位 PSF 最大値 =　5.00e+00
　温度 PSF 最大値 =　5.00e+00
　待ち時間　　=　1.00e+01
SN=　2
　直前操作 SN　　=　1
　直前操作状況　=失敗
　操作 1　　=有効
　ｺﾒﾝﾄ　　=ACS NC 手動切り替え(現場)
　次診断移行時間=　6.00e+01
　従属性診断 ID　=　101
　診断確率 ID　　=　100
　移動 1 時間　　=　1.00e+01
　操作 1 ｺｰﾄﾞ　　=　NCf
　操作 1 確率 ID　　=　　11
　操作 1 失敗確率 =　1.00e-02
　液位 PSF 最大値 =　5.00e+00

```
    温度 PSF 最大値 =   5.00e+00
    待ち時間        =   1.00e+01
SN=  3
    直前操作 SN     =  2
    直前操作状況   =失敗
    操作 1          =有効
    ｺﾒﾝﾄ            =MCS 起動
    次診断移行時間=  0.00e+00
    従属性診断 ID   =  102
    診断確率 ID     =  100
    移動 1 時間     =  0.00e+00
    操作 1 ｺｰﾄﾞ     =  MCS
    操作 1 確率 ID  =   10
    操作 1 失敗確率 =  1.00e-02
    液位 PSF 最大値 =  5.00e+00
    温度 PSF 最大値 =  5.00e+00
    待ち時間        =  1.50e+01
SN=  4
    直前操作 SN     =  1
    直前操作状況   =成功
SN=  5
    直前操作 SN     =  2
    直前操作状況   =成功
SN=  6
    直前操作 SN     =  3
    直前操作状況   =成功
SN=  7
    直前操作 SN     =  3
    直前操作状況   =失敗

アクションツリーデータ読み込み終了

 loop 部 = 158.156250
(frq_calc_factor*n_CM_occur= 1.97e-09)
炉心損傷回数(n_CM_occur)          = 4.57e+00
バイアス値(biasv)                 =無し(UR が 0 なので)
UR の数(nUR)                      =          0
試行回数(ntryx)                   =      20000
起因事象発生発生頻度(ie_freqncy)= 7.20e-02
MCS の UA 値(p_UA)                = 2.40e-05
(Πbiasv= 1.00e+00)
発生確率(x=n_CM_occur*(Πbiasv)/ntryx)= 2.28e-04
MCS の UA 値を乗じた値(x=x*p_UA)   = 5.48e-09

モンテカルロ法で求めた発生頻度(x=x*ie_freqncy)= 3.944e-10

従来の発生頻度(OLD=IE*MCS;使命時間  23.0 時間)= 1.728e-06


従来の発生頻度(使命時間  23.0 時間) (A) : 1.728e-06
EOP を考慮した発生頻度              (B) : 3.944e-10
NRF                                (B/A) : 2.283e-04

全計算終了                  CPU 経過時間 = 185
```

並列部 ＝159.464844,　シリアル部 ＝25.347656

****

DYANA：結果のまとめ

ｶｯﾄｾｯﾄ, 従来の発生頻度(使命時間 23.0 時間), EOP を考慮した発生頻度, NRF
AZ401MV01DD33　　1.728e-06　3.944e-10　2.283e-04

合計
EOP 評価なし＝1.728e-06 EOP 評価あり＝3.944e-10 NRF(あり/なし)＝2.283e-04

EOP 考慮なしの従来の発生頻度で並び換えた結果
　1：AZ401MV01DD33　　1.73e-06　3.94e-10　2.28e-04

EOP を考慮した発生頻度で並び換えた結果
　1：AZ401MV01DD33　　1.73e-06　3.94e-10　2.28e-04
****

OSF1 jjvedu00.jnc.go.jp V4.0 1229 alpha

****
2000 年 03 月 10 日 17 時 17 分 41 秒

付録 C

サンプル計算結果(LORL/G シーケンス)
プロセス数 5

2000年03月10日 17時11分49秒
ファイル名=data/dyana.dat.g

NAMELIST : DYDATA.

num.of process = 5

------ DYANA9803 : Dynamic Analysis Program for Event Transition. ------　　計算開始日時 Fri Mar 10 17:11:51 2000

(1999/01/31 17:00)

モンテカルロ法

超簡易版

時刻(秒)=...　　:ﾘｽﾀｰﾄﾌｧｲﾙに記録されている時刻(秒).起因事象発生時刻を0秒とする.
ｼﾐｭﾚｰｼｮﾝ時刻　　:事象推移で問題にしている時刻(秒).起因事象発生時刻を0秒とする.
PSF1　　:漏洩時に炉内Na液位から求められたPSF.
PSF2　　:R/V出口Na温度から求められたPSF.
LHS　　:ﾗﾃﾝﾊｲﾊﾟｰｷｭｰﾌﾞｻﾝﾌﾟﾘﾝｸﾞを使用する(Y),使用しない(N).
ISEED　　:乱数の種. -1で日付から求められる.

```
COMMENT     = G Test Case
APX         = N
IENO        =  3
IEFREQ      =  1.00000e-02
TMISS       =  2.30000e+01
TTRCMAX     =  2.30000e+01
VALFNAME    = data/val.dat
MCSFNAME    = data/g.mcs
ACTFNAME    = data/act.pnc.0812
CDFFNAME    =
CDFPARAM    = 1:o 15.0 3.00 10
              2:o 10.0 3.33 11
              3:o   5.0 3.33 12
              4:d 1.0 0.001 5.0 300.0 Z 100
              5:d 1.0 0.001 5.0 300.0 L 101
              6:d 1.0 0.001 5.0 300.0 M 102
              7:d 1.0 0.001 5.0 300.0 H 103
              8:d 1.0 0.001 5.0 300.0 C 104
              9:
             10:
DBIAS       =  0.00000e+00 (ﾀｲﾑﾒｯｼｭ法では使われない)
PRBLIMIT    =  1.00000e-18
TUPLIMIT    =  6.50000e+02
TDNLIMIT    =  2.50000e+02
MINNALVL    =  2.88500e+01
ICCOUNT     =              3
NTRY        =          20000 (ﾀｲﾑﾒｯｼｭ法では使われない)
LHS         = N              (ﾀｲﾑﾒｯｼｭ法では使われない)
ISEED       =      900892802 (ﾀｲﾑﾒｯｼｭ法では使われない)
DT          =  2.00000e+00 (ﾓﾝﾃｶﾙﾛ法では使われない)
ECCFN       = data/mcs_attr.dat.pnc.0812
```

パラメータデータから確率分布累積値データを作ります.

操作実施確率 : NDF
ID = 10　　MEAN = 1.50e+01　　S.D. = 3.00e+00

操作実施確率 : NDF
ID = 11　　MEAN = 1.00e+01　　S.D. = 3.33e+00

操作実施確率 : NDF
ID = 12 MEAN = 5.00e+00 S.D. = 3.33e+00

診断実施確率
ID:100

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
         |          *
         |           *
         |            *
         |             *
1.00e-03 |              *************
         |
         |
         +-----------------------------
                 5.00e+00   3.00e+02          Time(min)
```

Dependency :Z
coefficient:a =-1.687146e+00 b =1.179264e+00

診断実施確率
ID:101

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
         |          *
         |           *
         |            *
         |             *
1.00e-03 |              *************
         |
         |
         +-----------------------------
                 5.00e+00   3.00e+02          Time(min)
```

Dependency :L
coefficient:a =-1.687146e+00 b =1.179264e+00

診断実施確率
ID:102

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
         |          *
         |           *
         |            *
         |             *
1.00e-03 |              *************
         |
         |
         +-----------------------------
                 5.00e+00   3.00e+02          Time(min)
```

Dependency :M
coefficient:a =-1.687146e+00 b =1.179264e+00

診断実施確率
ID:103

```
      Prob.|
1.00e+00 |********
         |        *
         |         *
```

```
            |          *
            |           *
            |            *
            |             *
1.00e-03 |              *************
            |
            |
            +----------------------------
                        5.00e+00   3.00e+02          Time(min)
Dependency :H
coefficient:a =-1.687146e+00 b =1.179264e+00
```

診断実施確率

```
ID:104
      Prob.|
1.00e+00 |********
            |          *
            |           *
            |            *
            |             *
            |              *
            |               *
1.00e-03 |              *************
            |
            |
            +----------------------------
                        5.00e+00   3.00e+02          Time(min)
Dependency :C
coefficient:a =-1.687146e+00 b =1.179264e+00
```

累積確率データの読み込み

```
ID                =10
Number of Class=34
Comment           =MEAN= 1.50e+01, S.D.= 3.00e+00
 CLS(  0)= 0.00000e+00 CPD(  0)= 0.00000e+00 OMG(  0)= 1.00000e+00
 CLS(  1)= 0.00000e+00 CPD(  1)= 2.86652e-07 OMG(  1)= 1.00000e+00
 CLS(  2)= 1.00000e+00 CPD(  2)= 1.53063e-06 OMG(  2)= 1.00000e+00
 CLS(  3)= 2.00000e+00 CPD(  3)= 7.34342e-06 OMG(  3)= 1.00000e+00
 CLS(  4)= 3.00000e+00 CPD(  4)= 3.16712e-05 OMG(  4)= 1.00000e+00
 CLS(  5)= 4.00000e+00 CPD(  5)= 1.22866e-04 OMG(  5)= 1.00000e+00
 CLS(  6)= 5.00000e+00 CPD(  6)= 4.29060e-04 OMG(  6)= 1.00000e+00
 CLS(  7)= 6.00000e+00 CPD(  7)= 1.34990e-03 OMG(  7)= 1.00000e+00
 CLS(  8)= 7.00000e+00 CPD(  8)= 3.83038e-03 OMG(  8)= 1.00000e+00
 CLS(  9)= 8.00000e+00 CPD(  9)= 9.81536e-03 OMG(  9)= 1.00000e+00
 CLS( 10)= 9.00000e+00 CPD( 10)= 2.27501e-02 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.00000e+01 CPD( 11)= 4.77904e-02 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.10000e+01 CPD( 12)= 9.12112e-02 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.20000e+01 CPD( 13)= 1.58655e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.30000e+01 CPD( 14)= 2.52493e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.40000e+01 CPD( 15)= 3.69441e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.50000e+01 CPD( 16)= 5.00000e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.60000e+01 CPD( 17)= 6.30559e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.70000e+01 CPD( 18)= 7.47507e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.80000e+01 CPD( 19)= 8.41345e-01 OMG( 19)= 1.00000e+00
 CLS( 33)= 3.20000e+01 CPD( 33)= 1.00000e+00 OMG( 33)= 1.00000e+00
ID                =11
Number of Class=31
Comment           =MEAN= 1.00e+01, S.D.= 3.33e+00
 CLS(  0)= 0.00000e+00 CPD(  0)= 0.00000e+00 OMG(  0)= 1.00000e+00
 CLS(  1)= 0.00000e+00 CPD(  1)= 1.33665e-03 OMG(  1)= 1.00000e+00
 CLS(  2)= 1.00000e+00 CPD(  2)= 3.43891e-03 OMG(  2)= 1.00000e+00
 CLS(  3)= 2.00000e+00 CPD(  3)= 8.14391e-03 OMG(  3)= 1.00000e+00
 CLS(  4)= 3.00000e+00 CPD(  4)= 1.77722e-02 OMG(  4)= 1.00000e+00
 CLS(  5)= 4.00000e+00 CPD(  5)= 3.57883e-02 OMG(  5)= 1.00000e+00
```

```
 CLS(  6)= 5.00000e+00 CPD(  6)= 6.66129e-02 OMG(  6)= 1.00000e+00
 CLS(  7)= 6.00000e+00 CPD(  7)= 1.14837e-01 OMG(  7)= 1.00000e+00
 CLS(  8)= 7.00000e+00 CPD(  8)= 1.83821e-01 OMG(  8)= 1.00000e+00
 CLS(  9)= 8.00000e+00 CPD(  9)= 2.74053e-01 OMG(  9)= 1.00000e+00
 CLS( 10)= 9.00000e+00 CPD( 10)= 3.81974e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.00000e+01 CPD( 11)= 5.00000e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.10000e+01 CPD( 12)= 6.18026e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.20000e+01 CPD( 13)= 7.25947e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.30000e+01 CPD( 14)= 8.16179e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.40000e+01 CPD( 15)= 8.85163e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.50000e+01 CPD( 16)= 9.33387e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.60000e+01 CPD( 17)= 9.64212e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.70000e+01 CPD( 18)= 9.82228e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.80000e+01 CPD( 19)= 9.91856e-01 OMG( 19)= 1.00000e+00
 CLS( 30)= 2.90000e+01 CPD( 30)= 1.00000e+00 OMG( 30)= 1.00000e+00
ID                =12
Number of Class=26
Comment           =MEAN= 5.00e+00, S.D.= 3.33e+00
 CLS(  0)= 0.00000e+00 CPD(  0)= 0.00000e+00 OMG(  0)= 1.00000e+00
 CLS(  1)= 0.00000e+00 CPD(  1)= 6.66129e-02 OMG(  1)= 1.00000e+00
 CLS(  2)= 1.00000e+00 CPD(  2)= 1.14837e-01 OMG(  2)= 1.00000e+00
 CLS(  3)= 2.00000e+00 CPD(  3)= 1.83821e-01 OMG(  3)= 1.00000e+00
 CLS(  4)= 3.00000e+00 CPD(  4)= 2.74053e-01 OMG(  4)= 1.00000e+00
 CLS(  5)= 4.00000e+00 CPD(  5)= 3.81974e-01 OMG(  5)= 1.00000e+00
 CLS(  6)= 5.00000e+00 CPD(  6)= 5.00000e-01 OMG(  6)= 1.00000e+00
 CLS(  7)= 6.00000e+00 CPD(  7)= 6.18026e-01 OMG(  7)= 1.00000e+00
 CLS(  8)= 7.00000e+00 CPD(  8)= 7.25947e-01 OMG(  8)= 1.00000e+00
 CLS(  9)= 8.00000e+00 CPD(  9)= 8.16179e-01 OMG(  9)= 1.00000e+00
 CLS( 10)= 9.00000e+00 CPD( 10)= 8.85163e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.00000e+01 CPD( 11)= 9.33387e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.10000e+01 CPD( 12)= 9.64212e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.20000e+01 CPD( 13)= 9.82228e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.30000e+01 CPD( 14)= 9.91856e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.40000e+01 CPD( 15)= 9.96561e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.50000e+01 CPD( 16)= 9.98663e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.60000e+01 CPD( 17)= 9.99522e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.70000e+01 CPD( 18)= 9.99843e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.80000e+01 CPD( 19)= 9.99953e-01 OMG( 19)= 1.00000e+00
 CLS( 25)= 2.40000e+01 CPD( 25)= 1.00000e+00 OMG( 25)= 1.00000e+00
ID                =100
Number of Class=301
Comment           =DEP=Z, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(  0)= 0.00000e+00 CPD(  0)= 0.00000e+00 OMG(  0)= 0.00000e+00
 CLS(  1)= 1.00000e+00 CPD(  1)= 0.00000e+00 OMG(  1)= 0.00000e+00
 CLS(  2)= 2.00000e+00 CPD(  2)= 0.00000e+00 OMG(  2)= 0.00000e+00
 CLS(  3)= 3.00000e+00 CPD(  3)= 0.00000e+00 OMG(  3)= 0.00000e+00
 CLS(  4)= 4.00000e+00 CPD(  4)= 0.00000e+00 OMG(  4)= 0.00000e+00
 CLS(  5)= 5.00000e+00 CPD(  5)= 0.00000e+00 OMG(  5)= 0.00000e+00
 CLS(  6)= 6.00000e+00 CPD(  6)= 2.64793e-01 OMG(  6)= 1.00000e+00
 CLS(  7)= 7.00000e+00 CPD(  7)= 4.33160e-01 OMG(  7)= 1.00000e+00
 CLS(  8)= 8.00000e+00 CPD(  8)= 5.47499e-01 OMG(  8)= 1.00000e+00
 CLS(  9)= 9.00000e+00 CPD(  9)= 6.29048e-01 OMG(  9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 6.89459e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 7.35587e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 7.71688e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 8.00529e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 8.23973e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 8.43315e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 8.59480e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 8.73142e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 8.84804e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 8.94847e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 9.99000e-01 OMG(300)= 1.00000e+00
ID                =101
Number of Class=301
```

```
Comment          =DEP=L, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 2.51553e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 4.11502e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 5.20124e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 5.97595e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 6.54986e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 6.98807e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 7.33104e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 7.60503e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 7.82774e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 8.01149e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 8.16506e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 8.29485e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 8.40564e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 8.50105e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 9.49050e-01 OMG(300)= 1.00000e+00
ID               =102
Number of Class=301
Comment          =DEP=M, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 2.26965e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 3.71280e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 4.69285e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 5.39184e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 5.90965e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 6.30503e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 6.61447e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 6.86168e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 7.06263e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 7.22842e-01 OMG( 15)= 1.00000e+00
 CLS( 16)= 1.60000e+01 CPD( 16)= 7.36697e-01 OMG( 16)= 1.00000e+00
 CLS( 17)= 1.70000e+01 CPD( 17)= 7.48407e-01 OMG( 17)= 1.00000e+00
 CLS( 18)= 1.80000e+01 CPD( 18)= 7.58404e-01 OMG( 18)= 1.00000e+00
 CLS( 19)= 1.90000e+01 CPD( 19)= 7.67012e-01 OMG( 19)= 1.00000e+00
 CLS(300)= 3.00000e+02 CPD(300)= 8.56286e-01 OMG(300)= 1.00000e+00
ID               =103
Number of Class=301
Comment          =DEP=H, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
 CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
 CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
 CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
 CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
 CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
 CLS(   5)= 5.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
 CLS(   6)= 6.00000e+00 CPD(   6)= 1.32396e-01 OMG(   6)= 1.00000e+00
 CLS(   7)= 7.00000e+00 CPD(   7)= 2.16580e-01 OMG(   7)= 1.00000e+00
 CLS(   8)= 8.00000e+00 CPD(   8)= 2.73749e-01 OMG(   8)= 1.00000e+00
 CLS(   9)= 9.00000e+00 CPD(   9)= 3.14524e-01 OMG(   9)= 1.00000e+00
 CLS( 10)= 1.00000e+01 CPD( 10)= 3.44730e-01 OMG( 10)= 1.00000e+00
 CLS( 11)= 1.10000e+01 CPD( 11)= 3.67793e-01 OMG( 11)= 1.00000e+00
 CLS( 12)= 1.20000e+01 CPD( 12)= 3.85844e-01 OMG( 12)= 1.00000e+00
 CLS( 13)= 1.30000e+01 CPD( 13)= 4.00265e-01 OMG( 13)= 1.00000e+00
 CLS( 14)= 1.40000e+01 CPD( 14)= 4.11986e-01 OMG( 14)= 1.00000e+00
 CLS( 15)= 1.50000e+01 CPD( 15)= 4.21658e-01 OMG( 15)= 1.00000e+00
```

```
  CLS( 16)= 1.60000e+01 CPD( 16)= 4.29740e-01 OMG( 16)= 1.00000e+00
  CLS( 17)= 1.70000e+01 CPD( 17)= 4.36571e-01 OMG( 17)= 1.00000e+00
  CLS( 18)= 1.80000e+01 CPD( 18)= 4.42402e-01 OMG( 18)= 1.00000e+00
  CLS( 19)= 1.90000e+01 CPD( 19)= 4.47424e-01 OMG( 19)= 1.00000e+00
  CLS(300)= 3.00000e+02 CPD(300)= 4.99500e-01 OMG(300)= 1.00000e+00
ID                    =104
Number of Class=301
Comment               =DEP=C, Prob-Hi= 1.00e+00, Prob-Low= 1.00e-03, time_1= 5.00e+00, time_2= 3.00e+02
  CLS(   0)= 0.00000e+00 CPD(   0)= 0.00000e+00 OMG(   0)= 0.00000e+00
  CLS(   1)= 1.00000e+00 CPD(   1)= 0.00000e+00 OMG(   1)= 0.00000e+00
  CLS(   2)= 2.00000e+00 CPD(   2)= 0.00000e+00 OMG(   2)= 0.00000e+00
  CLS(   3)= 3.00000e+00 CPD(   3)= 0.00000e+00 OMG(   3)= 0.00000e+00
  CLS(   4)= 4.00000e+00 CPD(   4)= 0.00000e+00 OMG(   4)= 0.00000e+00
  CLS(   4)= 4.00000e+00 CPD(   5)= 0.00000e+00 OMG(   5)= 0.00000e+00
  CLS(   6)= 6.00000e+00 CPD(   6)= 0.00000e+00 OMG(   6)= 1.00000e+00
  CLS(   7)= 7.00000e+00 CPD(   7)= 0.00000e+00 OMG(   7)= 1.00000e+00
  CLS(   8)= 8.00000e+00 CPD(   8)= 0.00000e+00 OMG(   8)= 1.00000e+00
  CLS(   9)= 9.00000e+00 CPD(   9)= 0.00000e+00 OMG(   9)= 1.00000e+00
  CLS( 10)= 1.00000e+01 CPD( 10)= 0.00000e+00 OMG( 10)= 1.00000e+00
  CLS( 11)= 1.10000e+01 CPD( 11)= 0.00000e+00 OMG( 11)= 1.00000e+00
  CLS( 12)= 1.20000e+01 CPD( 12)= 0.00000e+00 OMG( 12)= 1.00000e+00
  CLS( 13)= 1.30000e+01 CPD( 13)= 0.00000e+00 OMG( 13)= 1.00000e+00
  CLS( 14)= 1.40000e+01 CPD( 14)= 0.00000e+00 OMG( 14)= 1.00000e+00
  CLS( 15)= 1.50000e+01 CPD( 15)= 0.00000e+00 OMG( 15)= 1.00000e+00
  CLS( 16)= 1.60000e+01 CPD( 16)= 0.00000e+00 OMG( 16)= 1.00000e+00
  CLS( 17)= 1.70000e+01 CPD( 17)= 0.00000e+00 OMG( 17)= 1.00000e+00
  CLS( 18)= 1.80000e+01 CPD( 18)= 0.00000e+00 OMG( 18)= 1.00000e+00
  CLS( 19)= 1.90000e+01 CPD( 19)= 0.00000e+00 OMG( 19)= 1.00000e+00
  CLS(300)= 3.00000e+02 CPD(300)= 0.00000e+00 OMG(300)= 1.00000e+00
```

乱数の種に 900892802 を設定しました.

Value Data 一覧(最初の 10 個のみ).

```
5.00e-07 AA005RE01RS01
3.60e-10 AA043PP01CL14
3.60e-08 AA043PP01CP14
2.80e-09 AA044PP01CL14
2.80e-07 AA044PP01CP14
3.20e-09 AA045PP01CL14
3.20e-07 AA045PP01CP14
9.20e-10 AA046PP01CL12
9.20e-08 AA046PP01CP12
4.50e-10 AA047PP01CL12
```

Minimal Cut Set 一覧(最初の 100 個のみ. 全リストは output/mcslist.readin に書き込まれます)

```
Equation ID.=G

    1: PZB01CB01HEB3[A]   PB011RE01AC63[A]


Minimal Cut Set 総数 =      1
```

ソートされた MCS 一覧(最初の 100 個のみ.全リストは output/mcslist.sorted に書き込まれます.)

```
No.      発生確率(p_U),UA 部分のみの発生確率(p_UA),....
   1 :   3.50e-08   3.50e-08 [UA] PZB01CB01HEB3 PB011RE01AC63 ---------- ----------
                                  [UR] ---------- ---------- ---------- ----------
```

操作コード'の読み込み.

```
01NCc    自然循環手動切替(中央制御室)
02NCf    自然循環手動切替(現場)
03PSc    PSOS 手動起動(中央制御室)
04KT     原子炉容器 Na 温度維持操作
```

05PS　PSOS 故障復旧操作
06MCS　MCS の起動(中央制御室)
07PMTc　M-G セット一括 CS「停止」位置
08PMTf　M-G セット界磁遮断器「切り」
09VC1　MV30 と MV31 を閉(中央制御室)
10VC2　MV26 と MV100 を閉(中央制御室)
11VCf　MV30 を手動閉(現場)

基本データ読み込み終了　　CPU 経過時間 =24
従来の発生頻度(IE*MCS;使命時間 23.0 時間)= 3.500e-10

対象 MCS==========
番号　　発生頻度(IE*UA*UR)　Event Code[UA],[UR]
　1 ::　3.50e-10 PZB01CB01HEB3 PB011RE01AC63 ---------- ----------
　　---------- ---------- ---------- ----------

(上記発生頻度は UR が使命時間1時間の値を使用.)

(buff=G　　------XO---PZB01CB01HEB3 * PB011RE01AC63 +)
(ope_validation=------XO---)
(mcs_cat_code=G)
(mcs_top_gate_name=G)

UR はありません.

無発生回数　0.0000e+00
発生回数　　4.0000e+03

ｶﾃｺﾞﾘｰｺｰﾄﾞ G
(frq_calc_factor= 8.75e-14)

目的のｱｸｼｮﾝﾂﾘｰの ID:G

**** ｱｸｼｮﾝﾂﾘｰﾃﾞｰﾀの読み込み ****
SN=　1
　直前操作 SN　　=　0
　直前操作状況　=----
　操作 1　　=有効
　ｺﾒﾝﾄ　　=M-G セット発電機磁界遮断機を手動で「切り」とする
　次診断移行時間=　0.00e+00
　従属性診断 ID　=　　0
　診断確率 ID　　=　100
　移動 1 時間　　=　1.00e+01
　操作 1 ｺｰﾄﾞ　　=PMTf
　操作 1 確率 ID　　=　　11
　操作 1 失敗確率 =　1.00e-02
　液位 PSF 最大値 =　5.00e+00
　温度 PSF 最大値 =　5.00e+00
　待ち時間　　=　1.00e+01
SN=　2
　直前操作 SN　　=　1
　直前操作状況　=成功
SN=　3
　直前操作 SN　　=　1
　直前操作状況　=失敗

アクションツリーデータ読み込み終了

loop 部 =136.906250
(frq_calc_factor*n_CM_occur= 1.40e-10)
炉心損傷回数(n_CM_occur)　　=1.60e+03
バイアス値(biasv)　　=無し(UR が 0 なので)

UR の数(nUR) = 0
試行回数(ntryx) = 20000
起因事象発生発生頻度(ie_freqncy)= 1.00e-02
MCS の UA 値(p_UA) = 3.50e-08
(Πbiasv= 1.00e+00)
発生確率(x=n_CM_occur*(Πbiasv)/ntryx)= 7.98e-02
MCS の UA 値を乗じた値(x=x*p_UA) = 2.79e-09

モンテカルロ法で求めた発生頻度(x=x*ie_freqncy)= 2.794e-11

従来の発生頻度(OLD=IE*MCS;使命時間 23.0 時間)= 3.500e-10

従来の発生頻度(使命時間 23.0 時間) (A) : 3.500e-10
EOP を考慮した発生頻度 (B) : 2.794e-11
NRF (B/A) : 7.982e-02

全計算終了 CPU 経過時間 = 164

並列部 = 138.101562, シリアル部 = 25.449219

****

DYANA : 結果のまとめ

ｶｯﾄｾｯﾄ, 従来の発生頻度(使命時間 23.0 時間), EOP を考慮した発生頻度, NRF
PZB01CB01HEB3 PB011RE01AC63 3.500e-10 2.794e-11 7.982e-02

合計
EOP 評価なし= 3.500e-10 EOP 評価あり= 2.794e-11 NRF(あり/なし)= 7.982e-02

EOP 考慮なしの従来の発生頻度で並び換えた結果
1 : PZB01CB01HEB3 PB011RE01AC63 3.50e-10 2.79e-11 7.98e-02

EOP を考慮した発生頻度で並び換えた結果
1 : PZB01CB01HEB3 PB011RE01AC63 3.50e-10 2.79e-11 7.98e-02
****

OSF1 jjvedu00.jnc.go.jp V4.0 1229 alpha

****
2000 年 03 月 10 日 17 時 14 分 35 秒

付録 D

近似関数

D.1 近似関数の仕様

D.2 近似関数を用いた際の
PLOHS,LOLR/G シーケンスの振る舞い

D. 1　近似関数の仕様

表 D-1　apxFC1 の仕様

関数名	apxFC1(T0)		
内 容			
	変数名	内容	単位
入力変数	T0	原子炉トリップ時刻	minute
出力変数	TRVA	A ループ原子炉容器出口 Na 温度	℃
	PSPA	時刻 T2 における A ループ1次主冷却系循環ポンプ回転数	%
	FACA	時刻 T2 における A ループ補助冷却設備 AC 出口 Na 流量	t/h

処　理

```
PSPA=10.0
FACA=235.0

規格化
  t=T0/20.0/60.0

x11=1/(1+exp(4.421001e+00-8.767041e+00*t))
x12=1/(1+exp(4.511197e-01-1.078845e+01*t))
x13=1/(1+exp(2.299923e+00-8.931274e+00*t))
x14=1/(1+exp(4.990791e-01-4.292014e+01*t))
x15=1/(1+exp(4.002737e+00-3.162502e+00*t))
x16=1/(1+exp(3.443740e+00-2.832908e+00*t))

x21=1/(1+exp(-(-
        3.458991e+00*x11+1.451473e+00*x12+1.667578e+00*x13+8.947553e-01*x14
        -1.301340e+00*x15-1.097485e+00*x16-4.361833e-01)))
x22=1/(1+exp(-(-2.870234e+00*x11-2.408293e+00*x12-3.727829e+00*x13-
        2.987290e+01*x14-2.354833e+00*x15-2.259765e+00*x16+1.025357e+01)))
x23=1/(1+exp(-(+2.690030e-01*x11-8.254828e-01*x12+9.816641e-01*x13
        +4.808160e-01*x14-1.974065e-01*x15+2.254191e-01*x16-1.351354e+00)))
x24=1/(1+exp(-(-1.641891e+00*x11+3.204430e+00*x12-1.225954e+00*x13-
        1.172115e+01*x14-1.291983e+00*x15-1.130905e+00*x16+5.409658e+00)))
x25=1/(1+exp(-(-1.390364e+00*x11-2.143618e-01*x12-
        1.817202e+00*x13+8.031175e+00*x14-1.102176e+00*x15-7.964458e-01*x16-
        2.366518e+00)))
x26=1/(1+exp(-(+5.898928e+00*x11-4.472900e+00*x12+1.194782e+00*x13-
        1.318769e+00*x14+4.839294e+00*x15+3.007770e+00*x16-4.733753e+00)))

z=1/(1+exp(-(1.969470e+00*x21-1.607705e+01*x22+2.168268e+00*x23-
        2.385624e+00*x24-4.649648e+00*x25+8.451699e+00*x26+2.246485e+00)))

規格化された関数値の復元
  TRVA=459-193.0*((z-0.1)/0.8)
```

表 D-2　apxFC2 の仕様

関数名	apxFC2 (T1)

内　容

起因事象:

1次主冷却系コールドレグ配管の原子炉容器入口部から漏洩が発生したと想定。漏洩箇所の破損口は 1/4 Dt と想定。

安全系の応答:

原子炉スクラムと同時に、1次主冷却系循環ポンプ主モータはトリップしたと想定。

原子炉スクラムと同時に1次アルゴンガス隔離弁は全部閉状態に移行と想定。

1次ナトリウムオーバフロー系は正常に作動すると想定。

NsL は 33.5m の高さであるが、漏洩発生後、スクラム信号が発生するまで漏洩が継続しているため、原子炉スクラム時の原子炉容器 Na 液位は、33.0m と想定する。

本関数は、原子炉スクラム後の経過時間 T1 に対する原子炉容器 Na 液位 LEVRV を与える。

	変数名	内容	単位
入力変数	T1	原子炉スクラム後の経過時間(分)。但し、適用範囲は、0 分～263.86 分である。	minute
出力変数	LEVRV	原子炉容器 Na 液位	m

処　理

規格化:

t = T1 / 263.86

t が 1.0 を超える場合、t=1.0 と設定する。

t が 0.0 を下回る場合は、t=0.0 と設定する。

```
x11=1/(1+exp(-(+1.205173e+01*t-9.543965e-02)))
x12=1/(1+exp(-(-6.690185e+00*t+2.443592e-01)))
x13=1/(1+exp(-(+1.235305e+01*t-1.268841e-01)))
x14=1/(1+exp(-(-8.104923e+00*t+6.053997e-01)))
x15=1/(1+exp(-(+2.020766e+00*t-1.040170e+00)))
x16=1/(1+exp(-(-4.496569e-01*t-3.982979e-01)))

x21=1/(1+exp(-(-9.533077e-01*x11-1.986557e-01*x12-2.892826e-01*x13
        +1.640532e-01*x14+3.779352e-01*x15-1.664090e-01*x16-4.923231e-01)))
x22=1/(1+exp(-(-1.107560e+01*x11+5.838037e+00*x12-
        1.135234e+01*x13+7.585524e+00*x14-1.379908e+00*x15+5.706685e-01*x16
        +5.339205e-01)))
x23=1/(1+exp(-(+4.889897e-01*x11-7.899769e-01*x12+3.537307e-01*x13-
        5.503137e-02*x14-7.422080e-02*x15-7.820398e-01*x16-1.527858e+00)))
x24=1/(1+exp(-(+1.224535e+00*x11-3.185593e+00*x12+1.558303e+00*x13-
        4.226202e+00*x14+3.755077e-01*x15-1.197534e+00*x16-1.773550e+00)))
x25=1/(1+exp(-(+2.506158e+00*x11-1.695953e+00*x12+3.026502e+00*x13-
        2.121820e+00*x14+6.008421e-02*x15-7.077899e-01*x16-1.541790e+00)))
x26=1/(1+exp(-(-4.471312e-01*x11-4.204714e-01*x12-5.617207e-01*x13-
        4.007823e-01*x14+1.006415e-01*x15-4.045504e-01*x16-6.542070e-01)))

z=1/(1+exp(-(1.919493e+00*x21+9.382326e+00*x22+1.887888e-01*x23-
        2.634659e+00*x24-1.598259e+00*x25+1.520232e+00*x26+6.353302e-01)))
```

規格化された関数値の復元

```
LEVRV = ( (z-0.1)/0.8 -1.0 ) * ( 33.0 - 28.836 ) + 33.0
```

表 D-3　apxFC2MF の仕様

関数名	apxFC2MF(T1 , T2)

内　容

起因事象:

1次主冷却系コールドレグ配管の原子炉容器入口部から漏洩が発生したと想定。漏洩箇所の破損口は 1/4 Dt と想定。

安全系の応答:

原子炉スクラムと同時に、A ループと C ループの1次主冷却系循環ポンプ主モータはトリップにしたが、B ループはトリップ失敗と想定。

原子炉スクラムと同時に1次アルゴンガス隔離弁は全部閉状態に移行と想定。

1次ナトリウムオーバフロー系は正常に作動すると想定。

NsL は 33.5m の高さであるが、漏洩発生後、スクラム信号が発生するまで漏洩が継続しているため、原子炉スクラム時の原子炉容器 Na 液位は、33.0m と想定する。

本関数は、原子炉スクラム後の経過時間 T2 に対する原子炉容器 Na 液位 LEVRV を与える。

	変数名	内容	単位
入力変数	T1	ミニマルカットセット成立時刻＝原子炉スクラム時刻　すなわち、0.0	minute
	T2	ミニマルカットセット成立後の経過時間(分)但し、適用範囲は、0 分～65 分である。	minute
出力変数	LEVRV	原子炉容器 Na 液位	m

処　理

規格化

```
  t = T2 / 65.0
```

t が 1.0 を超える場合は、t=1.0 と設定する。

t が 0.0 を下回る場合は、t=0.0 と設定する。

```
x11=1/(1+exp(-(+2.112782e+00*t-1.797732e+00)))
x12=1/(1+exp(-(-7.882815e-01*t-3.633161e-01)))
x13=1/(1+exp(-(-1.465100e+01*t-1.494354e+00)))
x14=1/(1+exp(-(+2.397162e+00*t-1.879184e+00)))
x15=1/(1+exp(-(-9.021974e-01*t-9.675532e-01)))
x16=1/(1+exp(-(+1.988112e+00*t-5.094538e-01)))

x21=1/(1+exp(-(+7.088121e-01*x11-9.521746e-02*x12-3.607934e-02*x13
        +8.328576e-01*x14 -2.825282e-01*x15 -1.954762e-01*x16 -1.381623e+00)))
x22=1/(1+exp(-(+1.379080e+00*x11-1.006649e+00*x12+3.147364e-01*x13
        +1.279773e+00*x14 -6.688289e-01*x15 +1.699326e-01*x16 -1.515953e+00)))
x23=1/(1+exp(-(-2.030470e+00*x11-9.437979e-01*x12+7.225703e+00*x13-
        3.345965e+00*x14 -1.002043e-01*x15 -2.103020e+00*x16 -9.984059e-01)))
x24=1/(1+exp(-(+1.493128e+00*x11-1.921635e+00*x12+2.552843e-01*x13
        +2.079425e+00*x14 -1.046913e+00*x15 -4.013739e-01*x16 -2.058276e+00)))
x25=1/(1+exp(-(-1.058557e+00*x11+8.628029e-01*x12-3.522375e-01*x13-
        1.420262e+00*x14 +7.022323e-01*x15 -3.777142e-01*x16 -5.837735e-01)))
x26=1/(1+exp(-(-1.086505e+00*x11+2.467623e-01*x12+1.190721e+00*x13-
        2.890162e+00*x14 +5.374138e-01*x15 -1.651642e+00*x16 -4.005960e-01)))

z=1/(1+exp(-(-5.373734e-01*x21-2.256544e+00*x22+6.614285e+00*x23-
        3.775786e+00*x24 +2.553060e+00*x25 +4.020109e+00*x26 -1.290484e-01)))
```

規格化された関数値の復元

```
  LEVRV = ( ( z - 0.1 ) / 0.8 - 1.0 ) * ( 33.0 - 28.0 ) + 33.0
```

表 D-4　apxFC2MS の仕様

関数名	apxFC2MS (T1 , T2 , T3)

内　容

起因事象:

1次主冷却系コールドレグ配管の原子炉容器入口部から漏洩が発生したと想定。漏洩箇所の破損口は 1/4 Dt と想定。

安全系の応答:

原子炉スクラムと同時に、A ループと C ループの1次主冷却系循環ポンプ主モータはトリップにしたが、B ループはトリップ失敗と想定。

原子炉スクラムと同時に1次アルゴンガス隔離弁は全部閉状態に移行と想定。

1次ナトリウムオーバフロー系は正常に作動すると想定。

緊急時運転手順に従った運転員による安全機能の回復:

原子炉スクラム後、T2 分経過した時点で、B ループポンプ主モータ停止操作が完了する。

NsL は 33.5m の高さであるが、漏洩発生後、スクラム信号が発生するまで漏洩が継続しているため、原子炉スクラム時の原子炉容器 Na 液位は、33.0m と想定する。

本関数は、原子炉スクラム後の経過時間 T3 に対する原子炉容器 Na 液位 LEVRV を与える。

	変数名	内容	単位
入力変数	T1	ミニマルカットセット成立時刻(分)＝原子炉スクラム時刻　すなわち、0.0	minute
	T2	原子炉スクラム後、B ループポンプ主モータ停止操作が完了するまでの時間	minute
	T3	原子炉スクラム後の経過時間	minute
出力変数	LEVRV	原子炉容器 Na 液位	m

処　理

B ループポンプ主モータ停止操作完了時刻からの経過時間(分)を求め、これを T0 とする。

```
T0 = T3 - T2
```

まず、B ループポンプ主モータ停止操作完了時刻における原子炉容器 Na 液位を求め、これを L1 とする。

```
L1 = apxFC2MF( T1 , T2 )
```

(1) L1 < 28.150 m の場合、原子炉容器出口ノズル上端を下回っているため、サイフォンブレークとなり、原子炉容器 Na 液位は回復しないものと想定する。

```
LEVRV = L1
```

(2) L1 =< 28.835 m の場合、B ループポンプ主モータトリップ後、3 分経過してから、L1 の液位を起点として PSOS により連続運転モードによる汲み上げが開始する。

この 3 分とは、ポンプトリップ後のコーストダウンによる液位のアンダーシュートを仮定したものである。

T0 < 3.0 の場合は、LEVRV = L1 と設定する。

T0 >= 3.0 の場合は以下の通りとする。

```
t1 = apxInvPTRP( L1 )
LEVRV = apxPTRP( t1 + T0 - 3.0 )
```

(3) L1 > 28.835 m の場合、原子炉容器液位は、PSOS 連続運転開始液位である EL 28.835mに向かって低下し続ける。このとき、PSOS は断続運転モードである。

```
t1 = apxInvFC2( L1 )
LEVRV = apxFC2( t1 + T0 )
```

表 D-5　apxInvFC2 の仕様

関数名	apxInvFC2 (LEVRV)

内　容

起因事象:

1次主冷却系コールドレグ配管の原子炉容器入口部から漏洩が発生したと想定。漏洩箇所の破損口は 1/4 Dt と想定。

安全系の応答:

原子炉スクラムと同時に、1次主冷却系循環ポンプ主モータはトリップしたと想定。

原子炉スクラムと同時に1次アルゴンガス隔離弁は全部閉状態に移行と想定。

1次ナトリウムオーバフロー系は正常に作動すると想定。

NsL は 33.5m の高さであるが、漏洩発生後、スクラム信号が発生するまで漏洩が継続しているため、原子炉スクラム時の原子炉容器 Na 液位は、33.0m と想定する。

本関数は、T1<258.414 分における、近似関数 apxFC2(T1)の逆関数である。apxFC2(T1)は、原子炉スクラム後の経過時間 T1 に対する原子炉容器 Na 液位 LEVRV を与える。

	変数名	内容	単位
入力変数	LEVRV	原子炉容器 Na 液位(m)。但し、適用範囲は、28.845m～32.997m である。	m
出力変数	T1	原子炉スクラム後の経過時間(分)	minute

処　理

規格化:

```
t = ( LEVRV - 28.845 ) / ( 32.997 - 28.845 )
```

t が 1.0 を超える場合、t=1.0 と設定する。

t が 0.0 を下回る場合は、t=0.0 と設定する。

```
x11=1/(1+exp(-(-2.709369e+01*t+1.087007e-01)))
x12=1/(1+exp(-(+5.122542e+00*t-1.366823e+00)))
x13=1/(1+exp(-(+2.081356e+01*t-3.363206e-01)))
x14=1/(1+exp(-(+2.760770e+01*t-1.204122e-01)))
x15=1/(1+exp(-(+4.311289e+00*t-2.351627e+00)))
x16=1/(1+exp(-(+1.052327e+01*t-1.024534e+00)))

x21=1/(1+exp(-(+7.795029e-01*x11-8.495786e-01*x12-1.536629e+00*x13-
        2.039524e+00*x14 -4.795885e-01*x15 -1.533625e+00*x16-8.202619e-01)))
x22=1/(1+exp(-(-2.232245e+00*x11+3.809649e-01*x12-5.684256e-01*x13
        +4.660414e-01*x14 +1.076474e+00*x15 +4.770192e-01*x16-
        1.913476e+00)))
x23=1/(1+exp(-(+4.957343e+00*x11-1.837060e-01*x12-3.082430e+00*x13-
        4.030600e+00*x14 -7.442489e-01*x15 -1.459038e+00*x16-3.918947e-01)))
x24=1/(1+exp(-(-3.370470e+00*x11-1.924171e-01*x12-9.319721e-01*x13
        +4.769065e-01*x14+2.135245e+00*x15+1.508286e+00*x16-2.258268e+00)))
x25=1/(1+exp(-(+2.370528e+01*x11-3.092899e+00*x12-1.437927e+01*x13-
        1.935016e+01*x14 -2.780917e+00*x15 -7.430574e+00*x16+2.983395e+00)))
x26=1/(1+exp(-(+1.964495e+00*x11-8.586797e-01*x12-2.120417e+00*x13-
        2.653087e+00*x14 -6.556661e-01*x15 -1.675576e+00*x16-7.351381e-01)))

z=1/(1+exp(-(2.394569e+00*x21-1.155192e+00*x22+3.930080e+00*x23-
        2.729094e+00*x24+1.460056e+01*x25+2.812152e+00*x26-1.769622e-02)))
```

規格化された関数値の復元

```
T1 = ( (z-0.1)/0.8 -1.0 ) * 260.0
```

表 D-6(1/2)　apxInvPTRP の仕様 (1)

関数名	apxInvPTRP (LEVRV)

内　容

起因事象:

1次主冷却系コールドレグ配管の原子炉容器入口部から漏洩が発生したと想定。漏洩箇所の破損口は 1/4 Dt と想定。

安全系の応答:

原子炉スクラムと同時に、A ループと C ループの1次主冷却系循環ポンプ主モータはトリップにしたが、B ループはトリップ失敗と想定。

原子炉容器 Na 液位が、EL 28.835m(EsL+65mm)以下になって B ループのポンプトリップ操作に成功したと想定。

この液位では1次ナトリウムオーバフロー系(PSOS)が連続運転モードで運転に入る。PSOS は EL 28.920m に液位が回復するまで停止しない。

原子炉スクラムと同時に1次アルゴンガス隔離弁は全部閉状態に移行と想定。

1次ナトリウムオーバフロー系は正常に作動すると想定。

NsL は 33.5m の高さであるが、漏洩発生後、スクラム信号が発生するまで漏洩が継続しているため、原子炉スクラム時の原子炉容器 Na 液位は、33.0m と想定する。

本関数は、T1<21 分における近似関数 apxPTRP(T1)の逆関数である。液位 LEVRV の適用範囲は、28.152334m～28.917709m である。

なお、apxPTRP は、ポンプ主モータトリップ操作完了後の PSOS による Na 汲み上げによる原子炉容器 Na 液位 LEVRV の上昇曲線及びその後の PSOS 停止後の Na 体積収縮による液位低下曲線
及びその後の PSOS 起動後の汲み上げ曲線を与える。最初の汲み上げ開始は、EL 28.150m(原子炉容器出口ノズル上端)からと想定する。

	変数名	内容	単位
入力変数	LEVRV	B ループポンプ主モータを停止した時刻の原子炉容器 Na 液位(m)。ただし、適用範囲は、28.152334m～28.917709m である。	m
出力変数	T1	近似関数 apxPTRP において、最初の汲み上げ過程における液位に対応する経過時間	minute

表 D-6(2/2)　apxInvPTRP の仕様(2)

処　理

規格化

L = (LEVRV - 28.0) * 1000.0

t = (L - 152.334) / (917.709 - 152.334)

t が 1.0 を超える場合は、t=1.0 と設定する。

t が 0.0 を下回る場合は、t=0.0 と設定する。

```
x11=1/(1+exp(-(+5.526083e+00*t-5.586772e+00)))
x12=1/(1+exp(-(+1.760415e+00*t-1.640032e+00)))
x13=1/(1+exp(-(+7.126275e+00*t+1.918398e+00)))
x14=1/(1+exp(-(-2.776851e+00*t+1.910810e+00)))
x15=1/(1+exp(-(-1.918025e+00*t+1.238465e+00)))
x16=1/(1+exp(-(+8.972321e+00*t-9.148437e+00)))

x21=1/(1+exp(-(-3.140567e-01*x11+3.231335e-01*x12-5.372836e-01*x13-
        1.077667e-01*x14-9.003672e-02*x15-3.852633e-01*x16-1.175426e+00)))
x22=1/(1+exp(-(+3.531991e-01*x11+1.721179e-01*x12+4.381591e-01*x13-
        1.091054e+00*x14-1.077654e+00*x15+1.228498e+00*x16-8.458439e-01)))
x23=1/(1+exp(-(+8.885482e+00*x11+5.779901e-01*x12-6.458398e+00*x13-
        6.601417e+00*x14-5.355325e+00*x15+1.512719e+01*x16-3.691043e+00)))
x24=1/(1+exp(-(+1.183824e-01*x11+9.878323e-01*x12+4.498961e-01*x13-
        7.944473e-01*x14-1.322147e+00*x15+1.347037e+00*x16-9.535680e-01)))
x25=1/(1+exp(-(+8.001222e-02*x11+3.117007e-01*x12-1.929927e-01*x13-
        1.346619e+00*x14-2.836246e-01*x15+8.297018e-01*x16-1.560920e+00)))
x26=1/(1+exp(-(-6.298319e-01*x11-1.101835e+00*x12-3.586861e+00*x13+
        2.224726e+00*x14+1.536502e+00*x15-1.531900e+00*x16-4.494832e-01)))

z=1/(1+exp(-(7.329382e-02*x21+1.658159e+00*x22+8.519245e+00*x23
        +1.760818e+00*x24+1.912508e+00*x25-6.435525e+00*x26-3.863622e-01)))
```

規格化された関数値の復元

T1 = ((z - 0.1) / 0.8) * 21.0

表 D-7(1/2)　apxPTRP の仕様(1)

関数名	apxPTRP(T1)

内　容

起因事象:

1次主冷却系コールドレグ配管の原子炉容器入口部から漏洩が発生したと想定。漏洩箇所の破損口は 1/4 Dt と想定。

安全系の応答:

原子炉スクラムと同時に、A ループと C ループの1次主冷却系循環ポンプ主モータはトリップにしたが、B ループはトリップ失敗と想定。

原子炉容器 Na 液位が、EL 28.835m(EsL+65mm)以下になって B ループのポンプトリップ操作に成功したと想定。

この液位では1次ナトリウムオーバフロー系(PSOS)が連続運転モードで運転に入る。PSOS は EL 28.920m に液位が回復するまで停止しない。

原子炉スクラムと同時に1次アルゴンガス隔離弁は全部閉状態に移行と想定。

1次ナトリウムオーバフロー系は正常に作動すると想定。

NsL は 33.5m の高さであるが、漏洩発生後、スクラム信号が発生するまで漏洩が継続しているため、原子炉スクラム時の原子炉容器 Na 液位は、33.0m と想定する。

本関数は、ポンプ主モータトリップ操作完了後の PSOS による Na 汲み上げによる原子炉容器 Na 液位 LEVRV の上昇曲線及びその後の PSOS 停止後の Na 体積収縮による液位低下曲線

及びその後の PSOS 起動後の汲み上げ曲線を与える。最初の汲み上げ開始は、EL 28.150m(原子炉容器出口ノズル上端)からと想定する。

	変数名	内容	単位
入力変数	T1	最初の汲み上げ開始からの経過時間(分)。但し、適用範囲は、0 分～181 分である。	minute
出力変数	LEVRV	B ループポンプ主モータを停止した時刻の原子炉容器 Na 液位(m)。ただし、適用範囲は、28.15m～28.92m である。	m

表 D-7(2/2)　apxPTRP の仕様 (2)

処　理

規格化

```
  t = T1 / 181.0
  t が 1.0 を超える場合は、t=1.0 と設定する。
  t が 0.0 を下回る場合は、t=0.0 と設定する。

x11=1/(1+exp(-(+1.282293e+01*t-1.365735e+00)))
x12=1/(1+exp(-(+4.697395e+00*t-5.833435e+00)))
x13=1/(1+exp(-(+1.981559e+01*t-1.084302e+00)))
x14=1/(1+exp(-(+4.571568e+00*t-4.625324e+00)))
x15=1/(1+exp(-(-6.152137e+00*t+1.612678e+00)))
x16=1/(1+exp(-(+9.755004e+00*t-8.071936e+00)))

x21=1/(1+exp(-(-3.436282e+00*x11 -5.329353e-01*x12 -5.313615e+00*x13 -
        5.061333e-01*x14 +4.684129e+00*x15 -1.017134e+00*x16- 3.223910e-03)))
x22=1/(1+exp(-(+9.336571e+00*x11 +3.200838e-01*x12 +1.264521e+01*x13
        +1.307126e+00*x14 -2.520025e+00*x15 +3.889266e+00*x16- 1.258227e+00)))
x23=1/(1+exp(-(+9.007861e+00*x11 +1.049399e-02*x12 +9.056454e+00*x13
        +6.824857e-01*x14 -8.249439e+00*x15 +3.833154e+00*x16- 4.305806e+00)))
x24=1/(1+exp(-(-3.561786e+00*x11 -7.790456e+00*x12 +6.406161e+00*x13 -
        5.794801e+00*x14 -2.747907e+00*x15 +3.910618e+00*x16- 1.245669e+00)))
x25=1/(1+exp(-(-2.136330e+00*x11 -1.785471e-01*x12 -2.834367e+00*x13 -
        6.990119e-01*x14 -1.062516e-01*x15 -8.442676e-01*x16- 5.030249e-01)))
x26=1/(1+exp(-(-8.943253e-01*x11 -3.623315e-01*x12 -1.530742e+00*x13
        +1.096953e-02*x14 -3.556686e-01*x15 +2.920868e-01*x16- 8.185241e-01)))

z=1/(1+exp(-(-4.001854e+00*x21+7.553135e+00*x22+2.004313e+00*x23-
        5.596368e+00*x24-2.323561e+00*x25-1.431938e+00*x26-4.232147e+00)))
```

規格化された関数値の復元

```
  L = ( ( z - 0.1 ) / 0.8 ) * ( 920.0 - 150.0 ) + 150.0
  LEVRV = L / 1000.0 + 28.0
```

表 D-8(1/2)　apxMCS の仕様(1)

関数名	apxMCS (T1 , T2 , T3)

内　容

制限:この関数は、T3 が T2 より7時間(420 分)経過するまでの間しか対応していない。T1 の最大値は 1200 分である。

仮定:時刻 T2 における温度を基準とした温度差の時間変化は T2 には依存しないと仮定して作成した。
また、Super-COPD による幾つかの解析結果を参考に作成した。

メンテナンス冷却系(MCS)強制循環への移行操作完了時(時刻 T2)における A ループ原子炉容器出口 Na 温度を TRVA0 と表す。

	変数名	内容	単位
入力変数	T1	ミニマルカットセット成立時刻	minute
	T2	メンテナンス冷却系(MCS)強制循環への移行操作完了時刻	minute
	T3	待ち終了時刻	minute
出力変数	TRVA	A ループ原子炉容器出口 Na 温度	℃

表 D-8(2/2)　apxMCS の仕様(2)

処　理

T1 が 30 分以下の場合 → T1=30.0 と設定する。

```
t1=(T1-30.0)/(1200.0-30.0)

T0=(T3-T2)

t2=T0/420.0
```

t2 が 1.0 を超えたばあいには、t2=1.0 と設定する。

```
x11=1/(1+exp(-(9.728163e+00*t1+7.833869e+00*t2-3.166135e+00)))
x12=1/(1+exp(-(4.854910e+00*t1+5.288530e+00*t2-1.916380e+00)))
x13=1/(1+exp(-(2.843252e+00*t1-1.267569e+01*t2-4.892264e-01)))
x14=1/(1+exp(-(3.582781e+00*t1+6.429009e-01*t2-1.287336e+00)))
x15=1/(1+exp(-(1.431983e+01*t1+4.243388e-03*t2-5.501584e+00)))
x16=1/(1+exp(-(7.342735e+00*t1-8.325485e+00*t2+1.413891e+00)))

x21=1/(1+exp(-(-2.699289e+00*x11-3.479677e-01*x12+9.455372e+00*x13-
        1.064569e+00*x14+3.559633e-01*x15-5.100830e+00*x16-4.760084e-01)))
x22=1/(1+exp(-(-1.352913e-01*x11-2.606613e+00*x12+2.892165e+00*x13-
        1.599086e+00*x14+1.013269e+00*x15+9.658986e-01*x16-5.387667e-01)))
x23=1/(1+exp(-(-1.479420e-01*x11-3.875314e-01*x12+2.324970e-01*x13-
        5.202787e-01*x14+6.478893e-02*x15-7.735113e-01*x16-1.638653e+00)))
x24=1/(1+exp(-(-7.368628e-01*x11-8.065697e-01*x12+5.911086e-02*x13-
        1.364065e+00*x14-5.440199e-01*x15-1.676050e-01*x16-1.293607e+00)))
x25=1/(1+exp(-(-4.525024e+00*x11+9.270578e-03*x12+2.192973e+00*x13-
        7.907390e-01*x14+4.945306e+00*x15+5.725550e+00*x16-3.210385e+00)))
x26=1/(1+exp(-(+1.209128e+00*x11+2.522631e+00*x12-3.440818e+00*x13-
        1.118094e+00*x14+4.468885e+00*x15-3.463093e+00*x16-4.206250e+00)))

z=1/(1+exp(-(6.574458e+00*x21-2.338703e+00*x22-2.598978e-01*x23-
        7.204125e-01*x24+2.354214e+00*x25-7.449322e+00*x26+4.425646e-01)))

temp=((z-0.1)/0.8-1.0)*128.0
```

(1)　TRVA0 が 350℃以上のとき

時刻 T2 における温度を基準とした温度差の時間変化は TRVA0 には依存しないと仮定した。

```
TRVA = TRVA0 + temp
```

(2)　TRVA0 が 350℃以下のとき

時刻 T2 における温度を基準とした温度差の時間変化は、基本的に TRVA0 には依存しないと仮定したが、TRVA が 200.0℃を下回らないように曲線を加工した。

```
TRVA = TRVA0 + temp * ( TRVA0 - 200.0 ) / ( 350.0 - 200.0 )
```

表 D-9　apxNC の仕様

関数名	apxNC (T1 , T2 , T3)

内　容

制限:この関数は、T3 が T2 より5時間経過するまでの間しか対応していない。

仮定:時刻 T2 における温度を基準とした温度差の時間変化は T1 と T2 には依存しないと仮定して作成した。
　また、Super-COPD による、幾つかの解析結果を参考に作成した代表的な曲線である。

自然循環への移行操作完了時(時刻 T2)における A ループ原子炉容器出口 Na 温度を TRVA0 と表す。

	変数名	内容	単位
入力変数	T1	ミニマルカットセット成立時刻	minute
	T2	メンテナンス冷却系(MCS)強制循環への移行操作完了時刻	minute
	T3	計算打ち切り時刻	minute
出力変数	TRVA	A ループの原子炉容器出口 Na 温度	℃

処　理

```
TRVA0 = apxNHS( T1 ,  T2 )

T0=(T3-T2)
t=T0/300.0
```

t が 1.0 を超えた場合には、t=1.0 と設定する。

```
x11=1/(1+exp(-(-1.026013e+01*t-1.200152e+00)))
x12=1/(1+exp(-(-1.209057e+01*t-3.551927e-01)))
x13=1/(1+exp(-(-7.420247e+00*t+3.030572e+00)))
x14=1/(1+exp(-(-1.464181e+01*t+2.362660e+00)))
x15=1/(1+exp(-(-2.247785e+01*t+1.014795e+01)))
x16=1/(1+exp(-(+6.985100e+00*t-8.879392e+00)))

x21=1/(1+exp(-(-1.994316e+00*x11-3.866307e+00*x12+1.189251e+00*x13
        +2.696641e+00*x14-1.247627e+00*x15-2.744211e+00*x16-1.544247e+00)))
x22=1/(1+exp(-(-2.841004e+00*x11-3.321377e+00*x12+1.421469e+00*x13
        +5.222928e+00*x14-5.768899e+00*x15-3.358946e+00*x16-2.959584e+00)))
x23=1/(1+exp(-(-1.944456e+00*x11-2.516330e+00*x12+3.649885e-01*x13+
        1.750830e+00*x14-1.214059e+00*x15-2.540713e+00*x16-1.079185e+00)))
x24=1/(1+exp(-(-2.135523e+00*x11-4.018760e+00*x12+1.131944e+00*x13
        +2.852307e+00*x14-1.418846e+00*x15-2.643751e+00*x16-7.910089e-01)))
x25=1/(1+exp(-(-4.005708e+00*x11-2.140641e+00*x12+6.929271e+00*x13
        +1.087655e+01*x14-1.313244e+01*x15-3.622318e+00*x16-2.544326e+00)))
x26=1/(1+exp(-(-3.898857e+00*x11-5.521417e+00*x12-1.131800e+00*x13
        +4.678428e-01*x14+4.911190e+00*x15-8.029663e+00*x16+1.297457e+00)))

z=1/(1+exp(-(2.237514e+00*x21+2.018240e+00*x22+1.697754e+00*x23
        +2.014742e+00*x24+6.747930e+00*x25+6.248063e+00*x26-7.229041e+00)))

temp=(z-0.1)/0.8*22.2-21.2


TRVA = TRVA0 + temp
```

表 D-10(1/2)　apxNHS の仕様 (1)

関数名	apxNHS (T1, T2)

内　容

制限:この関数は、T3 が T2 より5時間経過するまでの間しか対応していない。

仮定:時刻 T2 における温度を基準とした温度差の時間変化は T1 と T2 には依存しないと仮定して作成した。
　また、Super-COPD による、幾つかの解析結果を参考に作成した代表的な曲線である。

自然循環への移行操作完了時(時刻 T2)における A ループ原子炉容器出口 Na 温度を TRVA0 と表す。

	変数名	内容	単位
入力変数	T1	カットセット成立時刻	minute
	T2	カットセット成立後の時刻	minute
出力変数	TRVA	時刻 T2 における A ループ原子炉容器出口 Na 温度	℃
	PSPA	時刻 T2 における A ループ1次主冷却系循環ポンプ回転数	
	FACA	時刻 T2 における A ループ補助冷却設備 AC 出口 Na 流量	

処　理

```
PSPA=0.0
FACA=0.0

z0=apxFC1( T1 )  カットセット成立時の A ループ原子炉容器出口 Na 温度
t0=apxTG( T1 )   カットセット成立時の猶予時間(A ループ原子炉容器出口 Na 温度が 650℃になるまでの時間)
t2=(T2-T1)/t0
```

(1)　t2 が 1.0 を超えている場合
　T2 は 650℃を超えていることを意味する。
　ここでは便宜的に、次式によって T2 を計算するものとする。

```
TRVA=(0.3+0.7*t2)*(650-z0)+z0
```

表 D-10(2/2)　apxNHS の仕様 (2)

(2)　T1 が 30 分以下の場合

規格化

```
t1=T1/60.0/0.5

x11=1/(1+exp(-(+2.887587e-01*t1+3.983559e+01*t2-2.085756e-01)))
x12=1/(1+exp(-(+5.381341e+00*t1+9.401270e+00*t2-2.560837e-01)))
x13=1/(1+exp(-(-4.205819e-01*t1-1.143044e+00*t2-1.571908e-01)))
x14=1/(1+exp(-(-2.976184e-01*t1+6.945714e+00*t2-4.553192e+00)))
x15=1/(1+exp(-(-2.343340e+00*t1+3.667575e+01*t2+7.794107e-01)))
x16=1/(1+exp(-(-2.081276e+00*t1-1.851370e+01*t2+1.907864e+00)))

x21=1/(1+exp(-(+2.845655e+01*x11+4.347761e+00*x12-3.488792e+00*x13
        +3.879547e+00*x14+2.609345e+01*x15-2.622311e+01*x16-5.859692e+00)))
x22=1/(1+exp(-(+6.529250e-01*x11-1.241865e+00*x12+1.158760e+00*x13-
        8.226559e-01*x14-2.469460e+00*x15+1.607500e+00*x16+5.221853e-01)))
x23=1/(1+exp(-(-4.979097e+00*x11-2.682945e+00*x12+1.015120e+00*x13-
        2.553675e+00*x14-4.999224e+00*x15+3.830828e+00*x16+7.795311e-01)))
x24=1/(1+exp(-(-2.301404e+00*x11+3.311186e+00*x12+7.487102e-01*x13-
        5.971152e-01*x14-3.657240e+00*x15+1.243852e+00*x16+1.536110e+00)))
x25=1/(1+exp(-(+4.103061e+00*x11-2.705248e+00*x12-1.143549e+00*x13
        +1.245181e-01*x14+2.933874e+00*x15-3.090962e+00*x16-1.875021e+00)))
x26=1/(1+exp(-(-1.063910e+00*x11+1.848187e+00*x12-2.098533e+00*x13
        +4.575478e+00*x14+2.072422e+00*x15-9.808780e+00*x16-4.420513e+00)))
z=1/(1+exp(-(1.110971e+01*x21-2.932007e+00*x22-5.802320e+00*x23-
        3.097753e+00*x24-2.386755e+00*x25+7.357673e+00*x26-7.532059e+00)))
```

規格化された関数値の復元

```
TRVA= (z-0.1)/0.8*(650-z0)+z0
```

(3)　T1 が 30 分を超える場合

さらに、T1 が 15 時間を超える場合、T1=15.0*60.0 と再設定した上で以下の計算を行う。T1 が 15 時間以下の場合は、そのまま以下の計算を行う。

規格化

```
t1=(T1-30.0)/60.0/(15.0-0.5)
x11=1/(1+exp(-(+1.127928e-01*t1+6.933182e+00*t2-4.868006e+00)))
x12=1/(1+exp(-(-1.397209e-01*t1+1.322107e+01*t2-6.017367e-01)))
x13=1/(1+exp(-(+1.085464e+01*t1+7.752250e+00*t2-2.178130e+00)))
x14=1/(1+exp(-(-4.731861e+00*t1+1.185464e+01*t2-8.448752e-01)))
x15=1/(1+exp(-(+1.524607e+00*t1+5.794001e+01*t2+1.158385e-01)))
x16=1/(1+exp(-(-1.484263e+00*t1-1.726679e+01*t2+2.627851e+00)))

x21=1/(1+exp(-(+6.850719e+00*x11+9.460124e+00*x12-5.033476e+00*x13
        +9.061568e+00*x14+4.242463e+01*x15-2.059008e+01*x16-6.500953e+00)))
x22=1/(1+exp(-(-6.195830e-01*x11+1.709403e+00*x12-3.748226e-01*x13
        +1.236794e+00*x14-1.688481e+00*x15-5.193373e+00*x16-1.050377e+00)))
x23=1/(1+exp(-(-3.958346e+00*x11-2.493610e+00*x12+2.414843e+00*x13-
        3.243750e+00*x14-1.728902e+00*x15-5.502632e-02*x16-6.977233e-01)))
x24=1/(1+exp(-(-2.292149e+00*x11+3.404123e+00*x12-6.997374e+00*x13
        +1.408453e+00*x14-3.043310e+00*x15+4.408321e+00*x16+1.434293e-01)))
x25=1/(1+exp(-(+5.661978e+00*x11+5.874339e-01*x12-6.570880e+00*x13
        +3.079344e+00*x14+4.074370e+00*x15-1.289418e+00*x16-3.305305e+00)))
x26=1/(1+exp(-(+4.723044e+00*x11+4.037601e-01*x12+1.733040e+00*x13
        +5.204413e-01*x14-4.179425e-01*x15-8.453977e+00*x16-3.889041e+00)))
z=1/(1+exp(-(9.514765e+00*x2-1.581856e+00*x22-6.664881e+00*x23-
        1.565715e+00*x24+1.815393e+00*x25+6.590398e+00*x26-9.262861e+00)))
```

規格化された関数値の復元

```
TRVA= (z-0.1)/0.8*(650-z0)+z0
```

表 A-11　apxTG の仕様

関数名	apxTG (T0)

内　容

Aループの原子炉容器出口 Na 温度が 650℃になるまでの猶予時間

	変数名	内容	単位
入力変数	T0	ミニマルカットセット成立時刻	minute
出力変数	TG	Aループの原子炉容器出口 Na 温度が 650℃になるまでの猶予時間	minute

処　理

規格化

```
t0=T0/60.0/20.0

x11=1/(1+exp(-(+2.334296e+00*t0-1.418644e+00)))
x12=1/(1+exp(-(+2.418986e+00*t0-1.478752e+00)))
x13=1/(1+exp(-(+4.796692e+00*t0-5.684862e+00)))
x14=1/(1+exp(-(-1.015183e+01*t0-1.133184e-01)))
x15=1/(1+exp(-(+4.270255e+00*t0-4.717387e-01)))
x16=1/(1+exp(-(+6.897561e+00*t0-1.965085e-01)))


x21=1/(1+exp(-(+7.466282e-01*x11+1.268503e+00*x12+2.295062e-01*x13-
        1.468022e+00*x14+5.167867e-01*x15-1.498651e-01*x16-8.856861e-01)))
x22=1/(1+exp(-(-2.069151e+00*x11-2.052805e+00*x12-1.110791e+00*x13
        +9.230043e+00*x14-4.640336e+00*x15-6.544996e+00*x16-5.379745e-01)))
x23=1/(1+exp(-(-4.847166e-01*x11-1.668687e-01*x12-5.739419e-01*x13-
        1.631550e-01*x14-4.481453e-01*x15-7.564854e-01*x16-5.475610e-01)))
x24=1/(1+exp(-(+6.853226e-01*x11+5.085521e-01*x12+4.253956e+00*x13-
        1.429684e+00*x14-2.350282e-01*x15-8.809513e-01*x16-2.354660e+00)))
x25=1/(1+exp(-(+7.076930e-01*x11+9.322681e-01*x12+6.128309e+00*x13-
        1.808295e+00*x14-1.973872e-01*x15-1.773478e+00*x16-2.617577e+00)))
x26=1/(1+exp(-(+8.180525e-01*x11+9.079099e-01*x12+2.317118e+00*x13-
        1.492017e+00*x14+4.874983e-01*x15-8.152231e-01*x16-2.023246e+00)))

z=1/(1+exp(-(1.821698e+00*x21-6.498327e+00*x22-5.441958e-01*x23
        +3.939870e+00*x24+5.258412e+00*x25+3.178281e+00*x26-2.008200e+00)))
```

規格化された関数値の復元

```
TG=((z-0.1)/0.8*30.2+8.0)*60.0
```

D.2 近似関数を用いた際の PLOHS,LOLR/G シーケンスの振る舞い

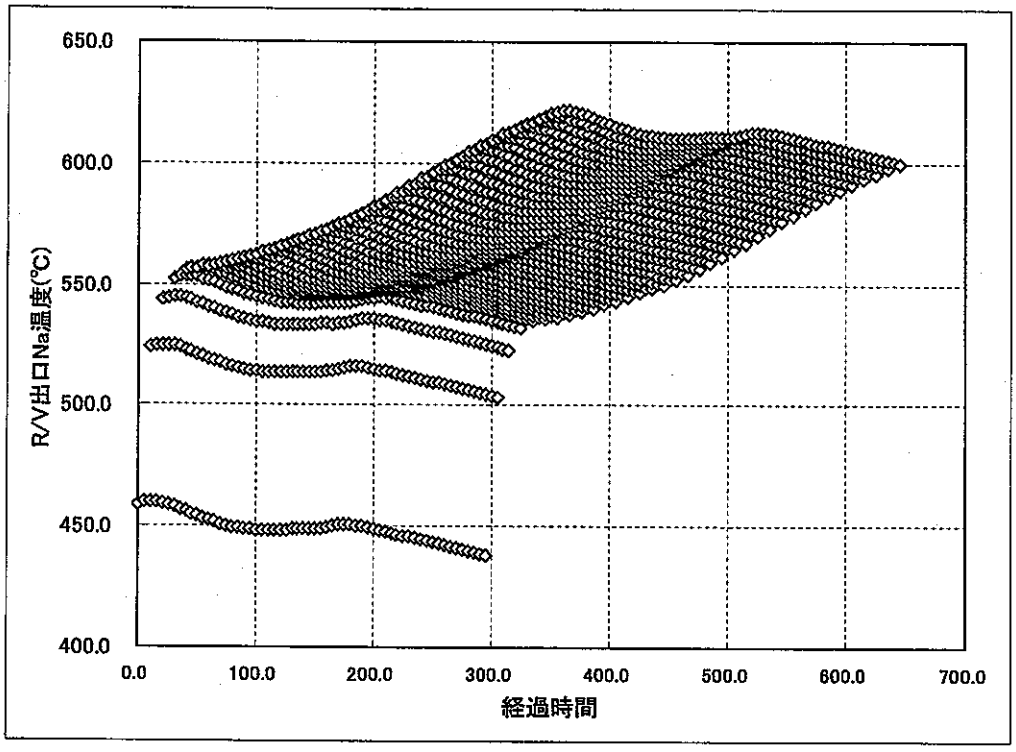

図 D-1　PLOHS シーケンス:除熱無しの状態から自然循環除熱に移行後の温度挙動

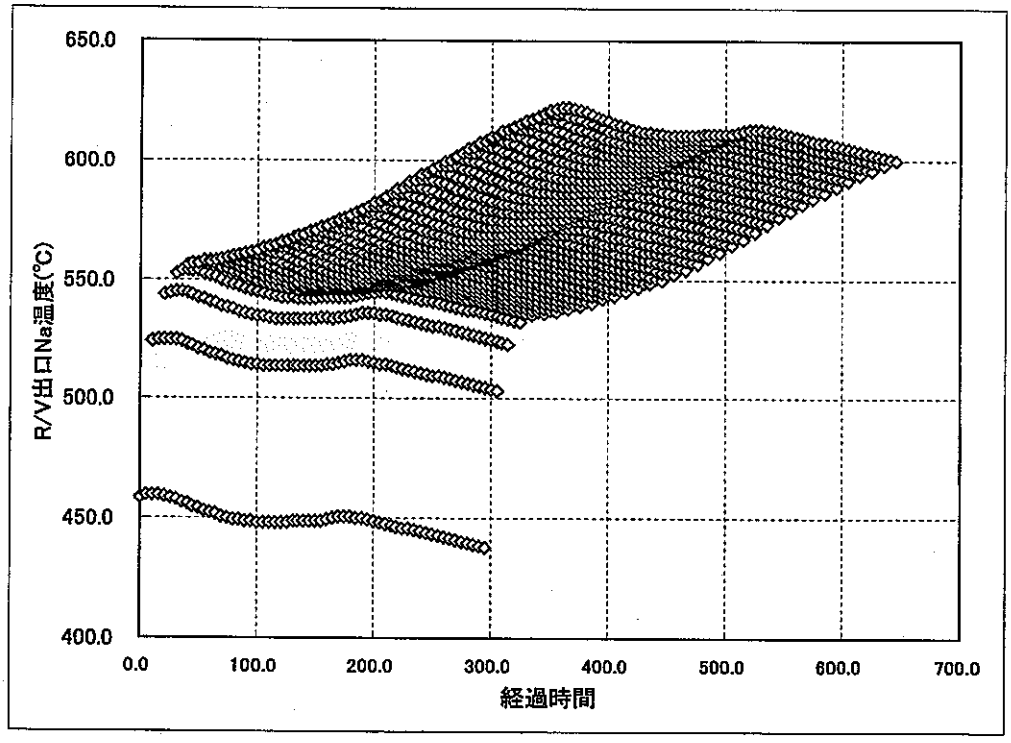

図 D-2 PLOHS シーケンス:除熱無しの状態から MCS 冷却が働き始めたときの温度挙動

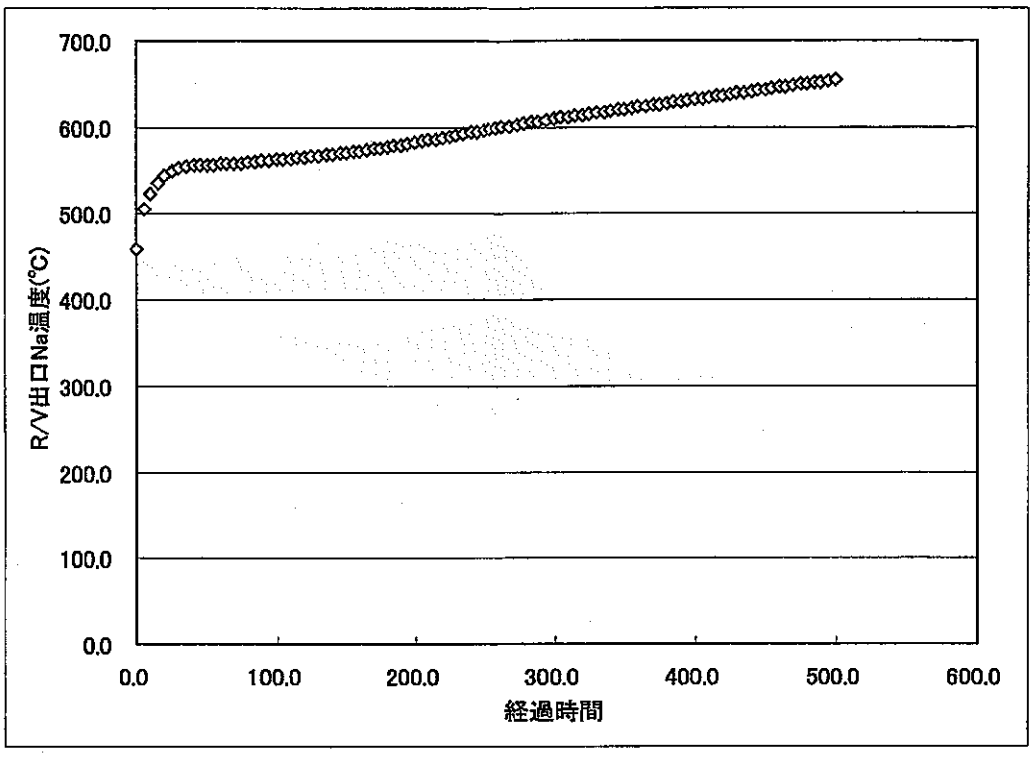

図 D-3　PLOHS シーケンス:除熱が無い場合の温度上昇(近似曲線より)

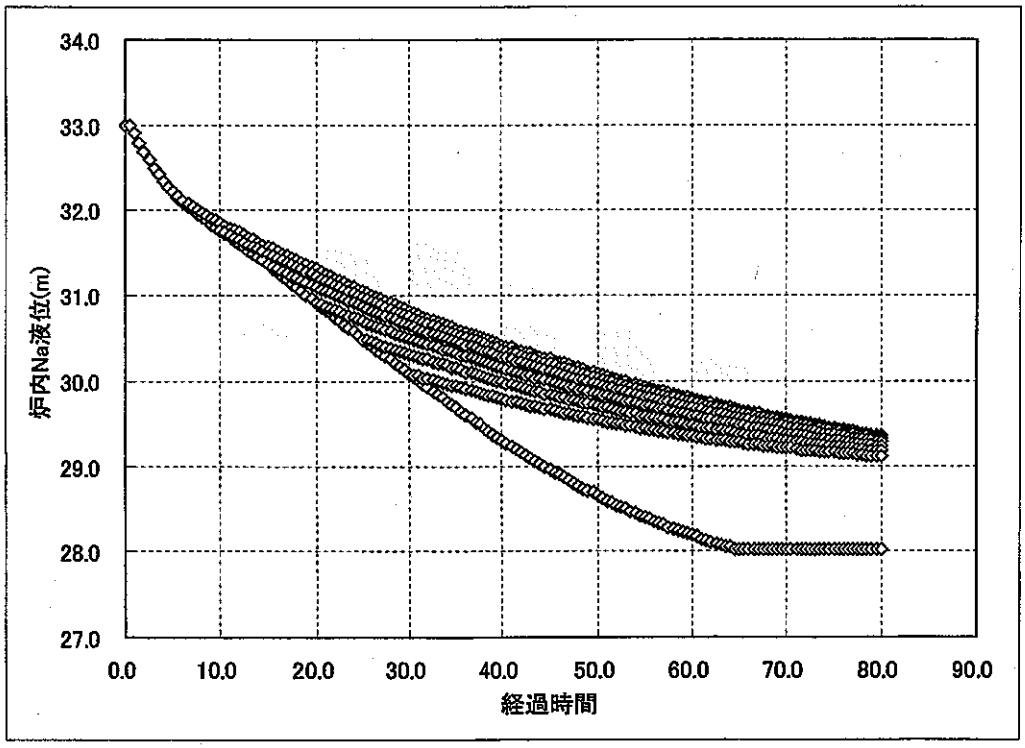

図 D-4 LOLR/G シーケンス:LR 炉内 Na 液位(m)変化
ポンプモータートリップ時刻を 0 秒から 30 秒まで 5 秒刻みで変化させたもの