JAERI-M
9650

# ダイナミックリンク機能の利用方法

1981年9月

原田　裕夫・木原　和久*・浅井　清

日本原子力研究所
Japan Atomic Energy Research Institute

# ダイナミックリンク機能の利用方法

日本原子力研究所東海研究所計算センター
原田裕夫・木原和久*・浅井　清

（1981年8月12日受理）

計算機プログラムの実行時に，ある入力データに対して実際に使用される部分（ロードモジュール）だけを動的に結合する機能をダイナミックリンクという。

1個のロードモジュールとしてまとめて主記憶に入れておくには過大であるプログラムや，ある入力ケースに対しては不用部分が多く不経済であるような大規模プログラムの開発，保守や実行に対して，ダイナミックリンク機能は有効である。また，この機能を利用することによって，プログラムの標準化，共用化にも効果がある。

ここでは，ダイナミックリンク機能の概要と標準的な使用方法，および使用上の主な注意事項について述べる。また，ダイナミックリンク機能の効果を分析するソフトウェアツール（ダイナリート）の利用方法と実際の原子力コードへの適用例についても述べる。

*　富士通(株)

# Dynamic Link : User's Manual

Hiroo HARADA, Kazuhisa KIHARA* and Kiyoshi ASAI

Computer Center, Tokai Research Establishment, JAERI

(Received August 12, 1981)

The purpose of dynamic link facility is to link a load module dynamically only when it is used in execution time.

The facility is very useful for development, execution and maintenance of a large scale computer program which is too big to be saved as one load module in main memory, or it is poor economy to save it due to many unused subroutines depending on an input. It is also useful for standardization and common utilization of programs.

Standard usage of dynamic link facility of FACOM M-200 computer system, a software tool which analyzes the effect of dynamic link facility and application of dynamic link to nuclear codes are described.

Keywords; Dynamic Link, Modular Coding, Computer Code System, Datapool, Manual

---

* Fujitsu, Ltd.

# 目　　次

Contents

# 1. はじめに

## 1.1 ダイナミックリンク機能の効用

プログラムの実行に必要なモジュールの結合方法には，静的な結合と動的な結合の2種類がある。

リンケージエディタでの処理時（結合編集時）に，プログラムに含まれるすべてのモジュールを結合して，1個のロードモジュールを作り上げる方法を「静的な結合」という。

これに対して，プログラムの実行時に，必要なモジュールだけを必要になった時点で結合する方法を「動的な結合」という。

動的な結合を行えば，実行の段階で使用しないモジュールは結合されないので，必要となる記憶領域の量を減らすことができる。

更に動的な結合を行うことにより，プログラムの変更あるいは，修正作業が能率的に行えるようになる。

モジュールを動的に結合するよう指定する方法としてリンケージエディタで処理する際に，必要なモジュールが組み込まれていないと，動的に結合するように特別な処理を加える方法がある。このような方法で手続部（サブルーチン）を動的に結合することを「ダイナミックリンク」といい，共通ブロック（コモン）を動的に結合することを「ダイナミックコモン」という。以後はこの二つを総称して「ダイナミックリンク」という。

ダイナミックリンクの使用者は，静的な結合（従来の方法による結合）でも，動的な結合（ダイナミックリンクによる結合）でも，FORTRANソースプログラムを変更する必要はない。

ダイナミックリンクを使用するためには，単にコンパイラのオプションで，DYNAMICを指定し，かつリンケージエディタのオプションで，DYNAMIC，NCALを指定するだけでよい。これらの指定によって共通ブロック（コモン）も，プログラム単位（サブルーチン）も，いずれも動的に結合するロードモジュールが作成でき，これを実行すればよいことになる。

ダイナミックリンクを使用したときの実行時間は，静的な結合による従来の実行時間とほとんど同じである。

### 1.1.1 大規模プログラムの実行時における効用

大規模なプログラムを実行する場合の主記憶使用量を考えてみる。

従来の方法（静的な結合）では，ロードモジュールはひとまとまりになっていて，全部が主記憶に入る。従って大きなメモリ量が必要になり，入力データによって実際には使用されないモジュールや，長い実行時間の内で最後の方でしか使用されないモジュールが最初から主記憶に入ることになる。

ダイナミックリンクによる方法では，サブルーチン単位およびコモン（共通ブロック）単位にロードモジュールが作成され，ある入力データに対して，実行時に必要なモジュールだけが必要

になった時点で主記憶に入る。

従って，利用者はオーバレイ構造など意識しなくてもメモリ量を小さくでき，それによってジョブの優先順位も上り，待ち時間も短くなることになる。（図 1.1）

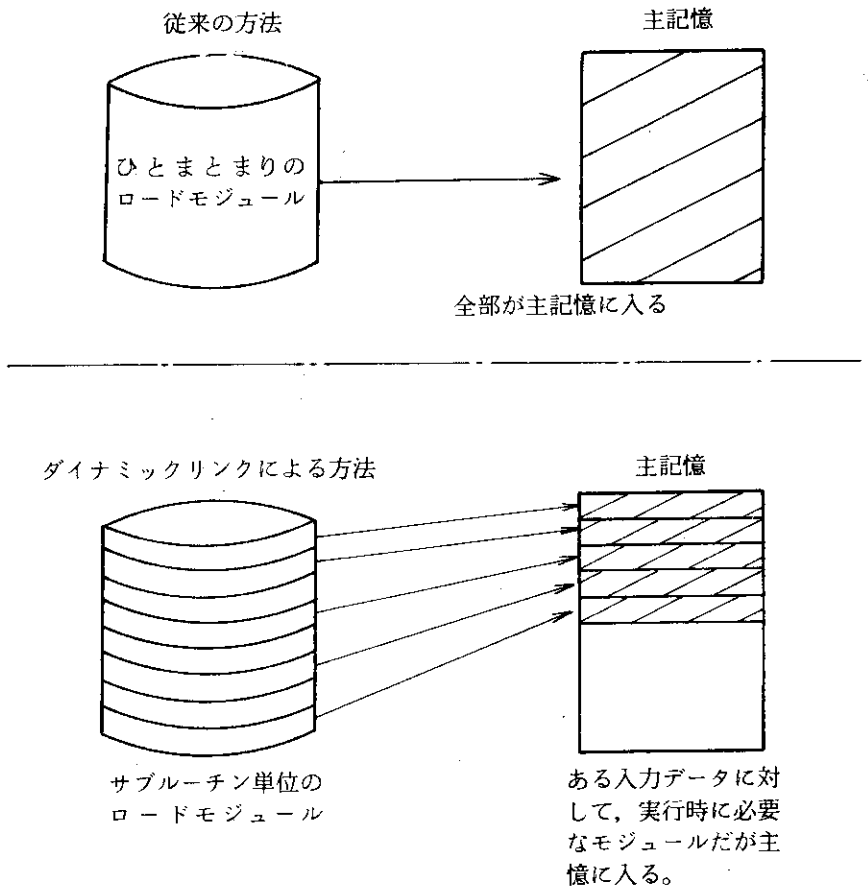



Fig. 1.1 Merits in execution of large scale programs

1.1.2　大規模プログラムの開発，保守における効用

大規模なプログラムの開発，保守の段階で，2,3のサブルーチンだけを修正して実行する場合を考えてみる。

従来の方法では，ロードモジュールを全部更新する必要があり，その処理には修正前のロードモジュールの2倍のサイズのファイルと，かなりの時間を要する。

ダイナミックリンクによる方法では，修正するサブルーチンだけをDYNAMIC指定でロードモジュールを作り，実行時の制御文（STEPLIB）の初めに指定し，修正しないロードモジュールを次に指定するだけでよいことになる。これによって，ロードモジュールのファイルのサイズは小さくて済み，処理時間も短くなる。（図1.2）

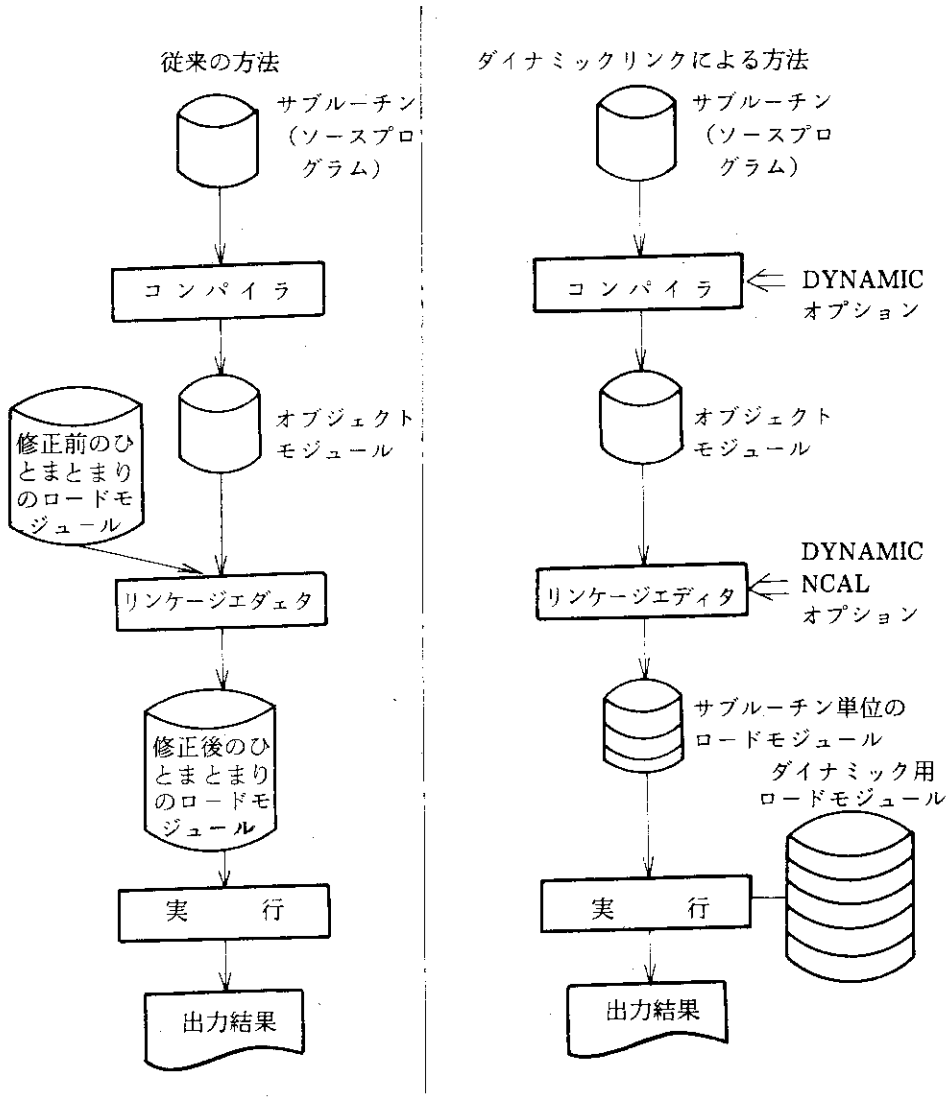



Fig. 1.2　Merits in development and maintenance of large scale programs

1.1.3　プログラムの標準化，共用化における効用

ダイナミックリンクを使用することによって，プログラムの標準化，共用化が促進される。

それは今まで，同種の目的あるいは機能を果すプログラムにおいて，入力データのケース（対称と非対称，２次元と３次元など）によって別々のプログラムになっていたものを，ダイナミックリンクの利用によって，ひとまとまりのプログラムにできるからである。

一方利用者は，標準ライブラリを含めた，ひとまとまりのプログラム全体でなく，各種の利用ケースで必要となる利用者固有のサブルーチンだけを管理すればよいので，プログラムの標準性，共用性を満たすと同時に，個々の利用者の多様性も満たすことができる。（図 1.3）

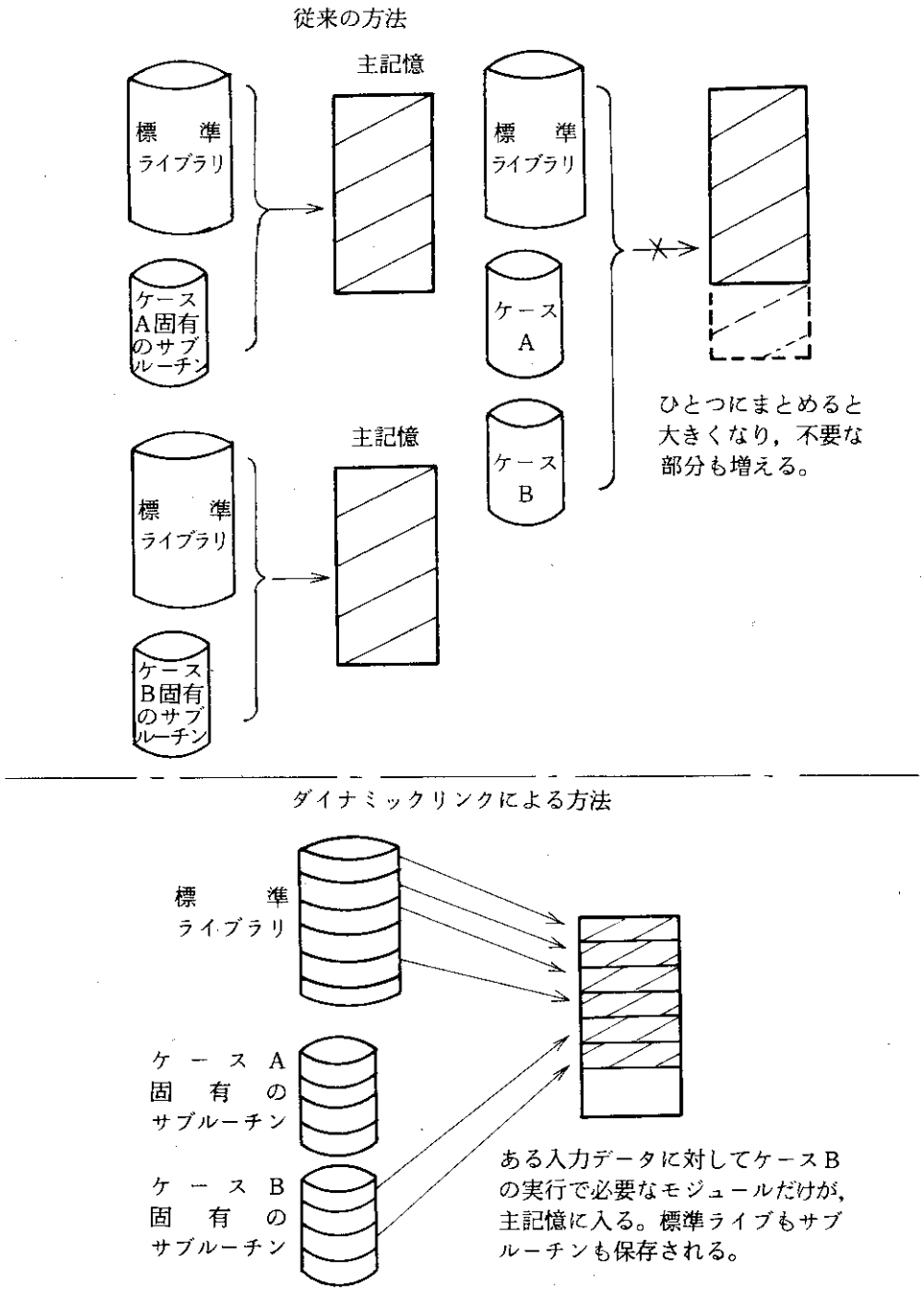



Fig. 1.3　Merits in stanardization and common utilization of programs

## 1.2 ダイナミックリンク効果分析ツール（ダイナリート）

ダイナミックリンク効果分析ツール（ダイナリート）とは，ダイナミックリンク機能を使用した時のメモリ使用状況をグラフに出力するソフトウエアツールである。

このダイナリートに，ソースプログラムと入力データを与えると，ダイナミックリンク機能によって，動的に結合して動作するロードモジュールを作成し実行されると同時に，実行時間の経過につれて，採用される（動的に結合される）共通ブロック名や副プログラム名と動的なメモリ使用状況などの情報が，データプールに書き込まれる。

そして，書き込まれた情報から，データプールのグラフィック機能によって，TSSのグラフィック端末（T 4014）から，次に示すような動的なメモリ使用状況をグラフに出力する。（図1.4，図1.5）

図1.4，図1.5において，斜線で示した部分が，ダイナミックリンク機能を使用することによって減少したメモリ部分である。

ダイナミックリンク効果分析ツール（ダイナリート）についての詳細は，「6. ダイナミックリンク効果分析ツールの利用方法」で述べる。

なお，本書に述べるダイナミックリンクの仕様は，FORTRAN－HE及びリンケージエディタ（LKED Version 3）に適合するものであり，将来，FORTRAN 77及びLKED Version 5では一部（共通ブロックの採用，表3.1，表3.3）を手直しする予定であるが，他の部分は変らない。

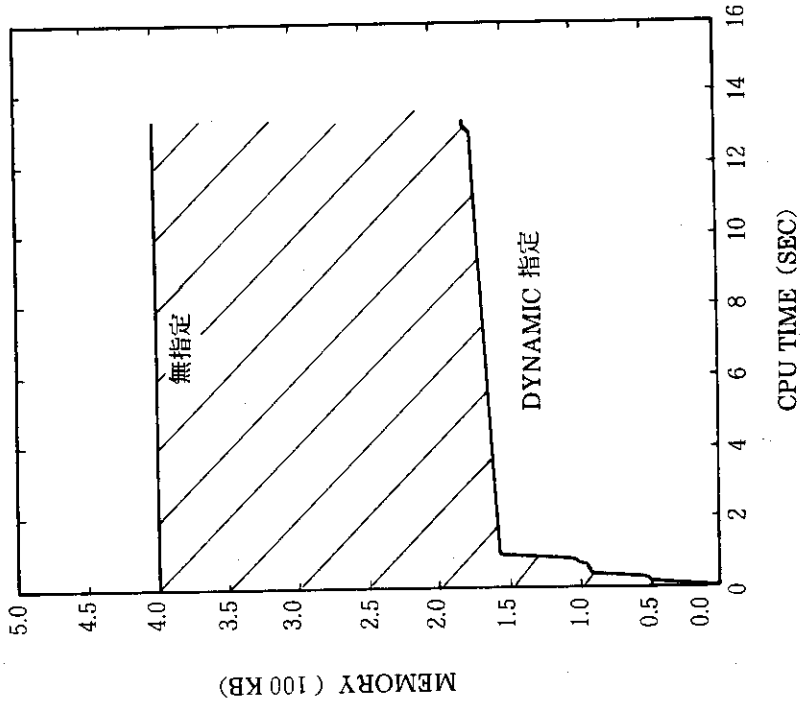



Fig. 1.5 Dynamically used memory (cpu time)

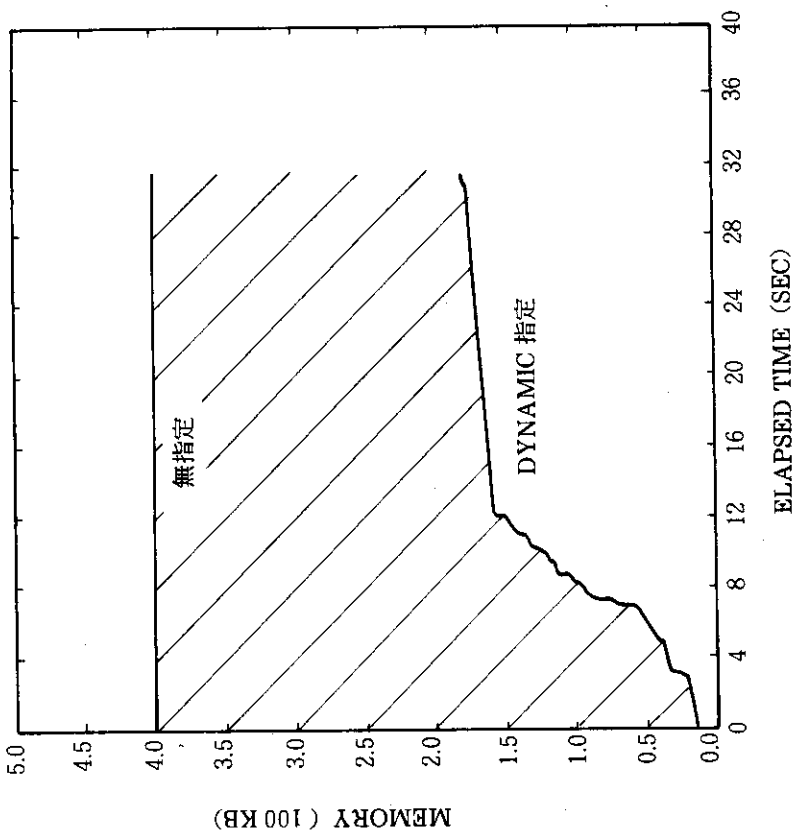



Fig. 1.4 Dynamically used memory (elapsed time)

# 2. ダイナミックリンク使用方法

## 2.1 使用方法の概要

FORTRANプログラムをダイナミックリンク機能を使って，動的な結合で実行するため手順の概略は次に示すとおりである。

(1) 翻　　訳：FORTRANソースプログラムをコンパイラのオプションで，DYNAMICを指定して翻訳する。
TSSで使用する場合には，FORTHEのコマンドのオペランドでDYNAMICを指定する。

(2) 結合編集：このオブジェクトモジュールをリンケージエディタのオプションで，DYNAMIC，NCALを指定して結合編集し，動的結合で動作可能なロードモジュールを作成する。

(3) 実　　行：このロードモジュールデータセットとFORTRAN Ⅳ・ライブラリ（SYS1.FORTLIB）を連結して，ジョブステップライブラリ（STEPLIB）として与えた上で，作成されたロードモジュールを実行する。

ただし，同じ名前の共通ブロックはプログラム中で最大のものが採用される。共通ブロックの大きさが同一の場合には，初期値つきが優先して採用される。詳しくは，「3. 共通ブロックの採用」を参照。

## 2.2 バッチジョブの手続き

バッチジョブとしてダイナミックリンクを使用する場合の例を次に示す。

```
//  EXEC FORTHE,
//    SO='J9999.SAMPLE'
//    A='ELM(*),DYNAMIC'
//  EXEC LKED,
//    A='DYNAMIC,NCAL',
//    MODS='100,20,50' (注1)
//SYSPRINT DD DUMMY
//SYSTERM  DD DIMMY    (注2)
//  EXEC GO,
//    PNM=MAIN (注3)
//STEPLIB DD DSN=&&LM,DISP=(OLD,DELETE)
//         DD DSN=SYS1.FORTLIB,DISP=SHR (注4)
```

（注１）　動的結合で使用する場合は，静的結合で使用する場合に比較して，ロードモジュールのメンバ数が多くなるため，MODSのパラメータで（ロードモジュール領域の初期値，増分値），ディレクトリ・ブロック数を指定する必要がある。ダイナミックリンク使用者はFORTRANプログラムのモジュール数，共通ブロック数に応じて，ディレクトリ・ブロック数を指定しなければならない。

モジュール数をm，共通ブロック（コモン）数をｃとすれば，ディレクトリ・ブロック数ｄは多少，多めに

$$d = (m + c) / 4$$

程度を指定しておけば十分である。

（注２）　ダイナミックリンクを使用する場合，結合編集時の出力リストが極端に増えるためSYSPRINTファイル，SYSTERMファイルをダミーファイルにする。ただし，正常に作られたかどうか確認するときは，モジュールマップ（SYSPRINT）を見る。

（注３）　NAME（主プログラム名）オプションまたはELEMENT文で主プログラム名を変更した場合は，実行のパラメータPNM＝MAINをその名前に変更しなければならない。

（注４）　実行時のジョブステップ・ライブラリとして，プログラム・ロードモジュール・データセットとFORTRAN Ⅳ ライブラリ（SYS１．FORTLIB）を与えて実行する。

SSLやSSLⅡを使用していれば，これらのライブラリもジョブステップライブラリとして与える必要がある。SSLやSSLⅡもダイナミックに結合されるが，再入可能プログラムとしては使用されない。

### 2.3　会話型ジョブの手続き

TSSで，ダイナミックリンクを使用する場合の例を次に示す。

```
FORTHE SAMPLE DYNAMIC OBJ
LINK SAMPLE DYNAMIC NCAL
ALLOC F(SYSPRINT) DUMMY
ALLOC F(SYSTERM) DUMMY
LIB (SAMPLE,'SYS1.FORTLIB')
ALLOC F(FT06F001) DA(*)
CALL SAMPLE(MAIN)
```

# 3. 共通ブロック採用の優先順位

ダイナミックリンク機能のもとで利用される共通ブロックをダイナミックコモンという。

ここでは，ダイナミックコモンを使用する場合の，結合編集時と実行時の共通ブロックの採用のされ方について述べる。ダイナミックコモンにおいて，プログラムが実行時に使用する共通ブロックは，結合編集時および実行時の条件により，以下のように定まる。

## 3.1 結合編集時における動的共通ブロック採用の優先順位

同一の名前の共通ブロックが2個以上存在する場合に，リンケージエディタが採用する共通ブロックは，次のようになる。ここでは，同一名の共通ブロックの大きさが相違する場合と同一の場合のそれぞれについて，オブジェクトモジュールから新しく実行プログラムライブラリ（ロードモジュール）を作成する場合と，すでにある実行プログラムライブラリを置き換える場合の共通ブロックの採用のされ方について，次の表に示す。（表3.1参照）

Table 3.1 Priority of common block linking of linkage editor

<table>
<tr><th colspan="2">動的共通ブロック</th><th>オブジェクトモジュールのみの場合</th><th>既存のロードモジュールを置き換える場合</th></tr>
<tr><td rowspan="2">大きさが相違する場合</td><td>初期値なし共通ブロック</td><td rowspan="2">オブジェクトモジュールの中で，最大の共通ブロックが，実行ロードモジュールの中の共通ブロックとなる。</td><td rowspan="2">左の条件に加えて，既存のロードモジュール（実行プログラムライブラリ）の同一名の共通ブロックとの大きさを比較し，大きいものが採用される。</td></tr>
<tr><td>初期値つき共通ブロック</td></tr>
<tr><td rowspan="2">大きさが同一の場合</td><td>初期値なし共通ブロック</td><td>先に表われた共通ブロックが採用される。</td><td>既存のロードモジュールの中の共通ブロックがそのまま採用される。</td></tr>
<tr><td>初期値つき共通ブロック</td><td colspan="2">初期値つき共通ブロックが優先して採用される。<br>初期値つき共通ブロックが複数個ある場合には，後から表われた共通ブロックが採用される。</td></tr>
</table>

ダイナミックリンクを使用する場合のソースプログラムから，ロードモジュール作成までの例を次に図示する。（図3.1）

図3.1において，オブジェクトモジュールのところで(*)で示したモジュールが，リンケージエ

ディタの採用するモジュールである。

COMMON／A／ は，大きさが同じなので初期値つきのBLOCKDATAのものが採用される。

COMMON／B／ は，大きさが異なるので，最大のMAINのものが採用される。

COMMON／C／ は，大きさが異なるので，最大のSUBのものが採用される。

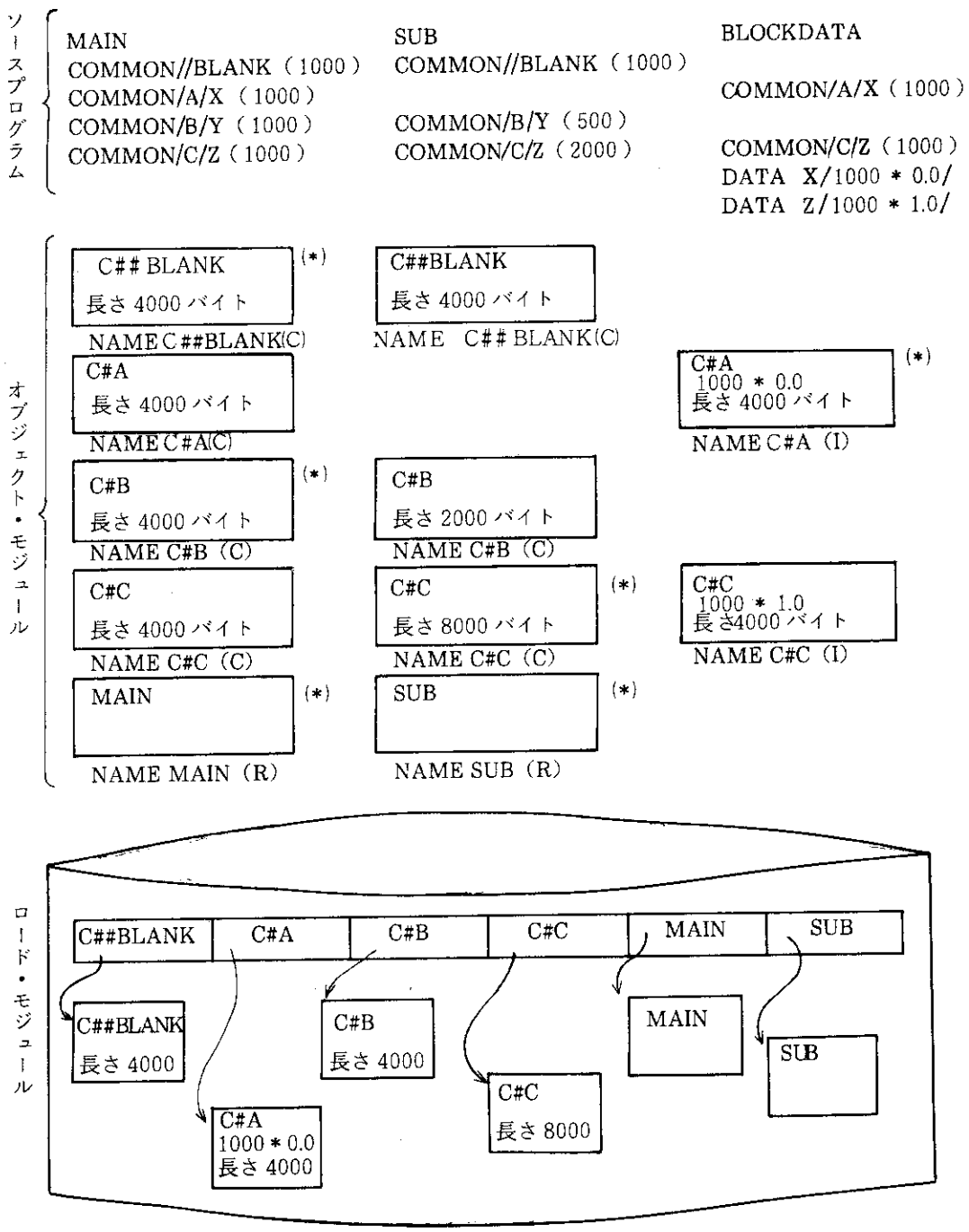



Fig. 3.1 Priority for adaptation of common blocks by dynamic link

### 3.2 静的結合の場合との比較

ダイナミックリンクの場合，結合編集時の共通ブロックの採用のされ方は，前記のとおりであるが，静的結合の場合と多少相違するところがある。以下では，オブジェクトモジュールから新しくロードモジュールを作成する場合の共通ブロックの採用のされ方を比較してみる。

前に入力された共通ブロックの種類と，後から入力された共通ブロックの種類に従って，リンケージエディタは次のように共通ブロックを採用する。（表3.2，表3.3）

Table 3.2 Priority of common block linking in static link

| 後の入力 / 前の入力 | 初期値つき共通ブロック | 初期値なし共通ブロック |
|---|---|---|
| 初期値つき共通ブロック | 前に入った共通ブロックがとられる。 | 初期値つき共通ブロックがとられる。（長さのチェックを行なう。） |
| 初期値なし共通ブロック | 初期値つき共通ブロックがとられる。（長さのチェックを行なう。） | 大きい方の共通ブロックがとられる。 |

Table 3.3 Priority of common block linking in dynamic link

| 後の入力 / 前の入力 | 初期値つき共通ブロック | 初期値なし共通ブロック |
|---|---|---|
| 初期値つき共通ブロック | 大きい方がとられる。（大きさが同じなら後の共通ブロックがとられる。） | 大きい方がとられる。（大きさが同じなら初期値つきがとられる。） |
| 初期値なし共通ブロック | 大きい方がとられる。（大きさが同じなら初期値つきがとられる。） | 大きい方がとられる。（大きさが同じなら前の共通ブロックがとられる。） |

### 3.3 実行時における動的共通ブロック採用の優先順位

同一の名前の共通ブロックが，複数個の実行プログラム・ライブラリにある場合，使用される共通ブロックは，一般のプログラム検索順序（STEPLIBのDD文で上から下への順序）に従って最初に見つけられた共通ブロックが，採用される。複数のプログラム・ライブラリのうち，最大の共通ブロックが，必ずしも採用されるわけではない。

次に例を示す。

```
//  EXEC GO,PNM=MAIN
//STEPLIB DD DSN=J9999.AAA.LOAD,DISP=SHR
//        DD DSN=J8888.BBB.LOAD,DISP=SHR
//        DD DSN=J7777.CCC.LOAD,DISP=SHR
//        DD DSN=SYS1.FORTLIB,DISP=SHR
```

| | コモン名 | 長　さ |
|---|---|---|
| J9999.AAA.LOAD(MAINを含む) | /COM/ | 4000 byte |
| J8888.BBB.LOAD | /COM/ | 2000 byte |
| J7777.CCC.LOAD | /COM/ | 8000 byte |

このとき，COMという名前の共通ブロックのうち最初に見つけられるのは，J9999．AAA．LOADの中の長さ4000バイトの共通ブロックである。従って，この4000バイトの共通ブロックが採用されて，J7777．CCC．LOADの中にある長さ8000バイトの共通ブロックCOMは採用されない。

このように同じ名前の共通ブロックが，複数のプログラム・ライブラリにある時には，注意を要する。特に，無名共通ブロック（ブランク・コモン）がある場合には，検索順序に注意して，STEPLIBのDD文を並べる。

従って，ダイナミックリンクを利用して，動的結合で実行する場合には，結合編集で一つの実行プログラムライブラリを作成する。すなわち，ロードモジュールのファイルは1つにすることが望ましい。

# 4. 翻訳時と結合編集時のオプション

## 4.1 翻訳時のオプション

{DYNAMIC / NODYNAMIC}　{NAME / NONAME}　{LIL / NOLIL}

### 4.1.1 DYNAMICオプション

共通ブロックを含む動的な結合を行なうオブジェクトを出力するかどうかを指定するオプションである。

このオプションを指定して翻訳したオブジェクトモジュールを，リンケージエディタでDYNAMIC，NCALを指定して結合編集し，ロードモジュールを作成すれば，共通ブロックを含めて動的に結合するロードモジュールを得ることができる。

### 4.1.2 NAMEオプション

各プログラム単位のオブジェクトモジュールの最後に，リンケージエディタの制御文（ALIAS文とNAME文のみ）を出力するかどうかを指定するオプションである。

DYNAMICオプションを指定した場合には，NAMEオプションを指定しなくてもよい。DYNAMICを指定した場合には，コンパイラが自動的にNAMEオプションを指定したものとみなす。

ただし，DYNAMIC，NAMEオプションが指定された場合と，NODYNAMIC，NAMEオプションが指定された場合とでは，共通ブロックを別のロードモジュールとするか，プログラムと同一のロードモジュールに含めるかの相違があるので，共通ブロックの扱いには特に注意を要する。

### 4.1.3 LILオプション

言語間結合用のオブジェクトモジュールにするかどうかを指定するオプションである。

DYNAMICオプションを指定した場合は，LILオプションを同時に指定してはならない。すなわち，言語間結合で作成されたプログラムを，共通ブロックを含む動的な結合の元で動作させることはできない。

ただし，アセンブラとの結合は，FORTRANのオブジェクトの構造を知った上で使用は可能であるが，十分に注意を要する。

アセンブラソースとFORTRANソースから，ダイナミックリンクによって動的な結合で実行する場合の例を次に示す。

```
//  EXEC FORTHE,
//    SO='J9999.SAMPLE',
//    A='ELM(*),DYNAMIC'
//  EXEC ASM,
//    SO='J9999.SPASM',
//    A='NAME,OBJ,BATCH,NOLIST',
//    DISP=MOD
//  EXEC LKED,
//    A='DYNAMIC,NCAL',MODS='100,20,50'
//SYSPRINT DD DUMMY
//SYSTERM  DD DUMMY
//  EXEC GO,
//    PNM=MAIN
//STEPLIB DD DSN=&&LM,DISP=(OLD,DELETE)
//         DD DSN=SYS1.FORTLIB,DISP=SHR
```

## 4.2 結合編集時のオプション

DYNAMIC，NCAL，OVLY

### 4.2.1 DYNAMICオプション

動的な結合で実行するロードモジュールを作成することを指定するオプションである。

DYNAMICが指定されると出力ロードモジュール中の未解決な外部参照名（ただし，他のプログラムに分岐するために使用されている外部記号に対するV型アドレス定数のみ）に対して，DALTAB（ダイナミックリンクテーブル）項目が作成され，実行時に，対応する記号をメンバ名または別名としてもつロードモジュールと動的に結合できる。

翻訳時にDYNAMICオプションを指定して作成されたオブジェクトモジュールをリンケージエディタのDYNAMICオプションを指定して結合編集すれば，共通ブロックを含めて動的に結合するロードモジュールを作成することができる。

翻訳時にNODYNAMICオプション指定で作成されたオブジェクトモジュールをリンケージエディタのDYNAMICオプションを指定して結合編集した場合，共通ブロックを動的に結合するロードモジュールを作成することはできない。

その他，翻訳時のオプション指定と結合編集時のオプション指定の組合せについては，「4.3 翻訳時と結合編集時のオプションの組合せ」で述べる。

### 4.2.2 NCALオプション

未解決の外部参照名に対して作るべきDALTAB（Dynamic Link Table）記入項目の作成を保証するオプションである。

NCALオプションが指定されると，未解決の外部参照名がある場合でも作成されたロードモジュールは実行可能となる。従って，動的に結合するロードモジュールを作成するためには，リン

ケージエディタのオプションで，DYNAMICとNCALを指定しなければならない。

このNCALオプションを指定するとLIBRARY制御文を用いることができないので，実行時のジョブステップ・ライブラリとして，FORTRAN Ⅳ ライブラリ（SYS1．FORTLIB）を付け加えるかまたは，ジョブ・ライブラリとして与えなければならない。

また，SSL，SSL ⅡやJSSLなどを使用していれば，これらのライブラリもジョブ・ライブラリまたは，ジョブステップ・ライブラリとして与えなければならない。

4.2.3　OVLYオプション

OVERLAY制御文及びINSERT制御文によってオーバレイ構造のプラグラムを作成して，オーバレイ属性をロードモジュールに与えるよう指示するオプションである。

オーバレイ構造では，ダイナミックリンクは使用できない。すなわち，リンケージエディタのオプションとして，OVLYとDYNAMICは排他関係である。

現在，エラーチェックは行なっていないが，ダイナミックリンクを使用する場合は，OVLYオプションを指定しないで使用するようにする。

## 4.3　翻訳時と結合編集時のオプションの組合せ

4.3.1　翻訳時DYNAMIC－結合編集時NODYNAMIC

翻訳時にDYNAMICオプションを指定して作成されたオブジェクトモジュールを，結合編集時に，無指定で（DYNAMICオプションを指定せず），静的に結合したロードモジュールを作成することもできる。従って，翻訳時にDYNAMICを指定して作成されたオブジェクトモジュールと翻訳時にNODYNAMICオプションで作成されたオブジェクトモジュールとを混合して，結合編集時にリンケージエディタのDYNAMICオプションを指定しないで，静的に結合したロードモジュールを得ることができる。

ただし，翻訳時にDYNAMICオプションを指定して作成されたオブジェクトモジュールには，NAME文が含まれているので注意を要する。

従って，静的に結合したロードモジュールを作成する場合には，もし，翻訳時にDYNAMICオプションを指定して作成されたオブジェクトモジュールがある時は，いったん，結合編集時にDYNAMICオプションを指定して，ダイナミックリンク用のロードモジュールを作成した上で，再びリンケージエディタのDYNAMICオプションを指定しないで結合編集して，静的に結合したロードモジュールを作成するようにする。

4.3.2　翻訳時DYNAMIC－結合編集時DYNAMIC

翻訳時にDYNAMICオプションを指定して作成したオブジェクトモジュールのうち，2個以上のいくつかのプログラム単位を静的に結合して1個のロードモジュールとし，他のプログラム単位はそのまま動的に結合するロードモジュールを作成して，動的結合で動作させる場合には，静的結合で作られたロードモジュールに含まれる外部手続き名と共通ブロック名をメンバ名または別名として与えておかなければならない。

ただし，それらの外部手続き名と共通ブロックが，その他のロードモジュールから参照されなければ，別名を与える必要はない。

4.3.3　翻訳時NODYNAMIC－結合編集時DYNAMIC

翻訳時にNODYNAMICオプション指定で，作成されたオブジェクトモジュールは，結合編集時にDYNAMICオプションを指定しても，動的に結合するロードモジュールは作成できない。

以上をまとめて図に表わすと次のようになる。

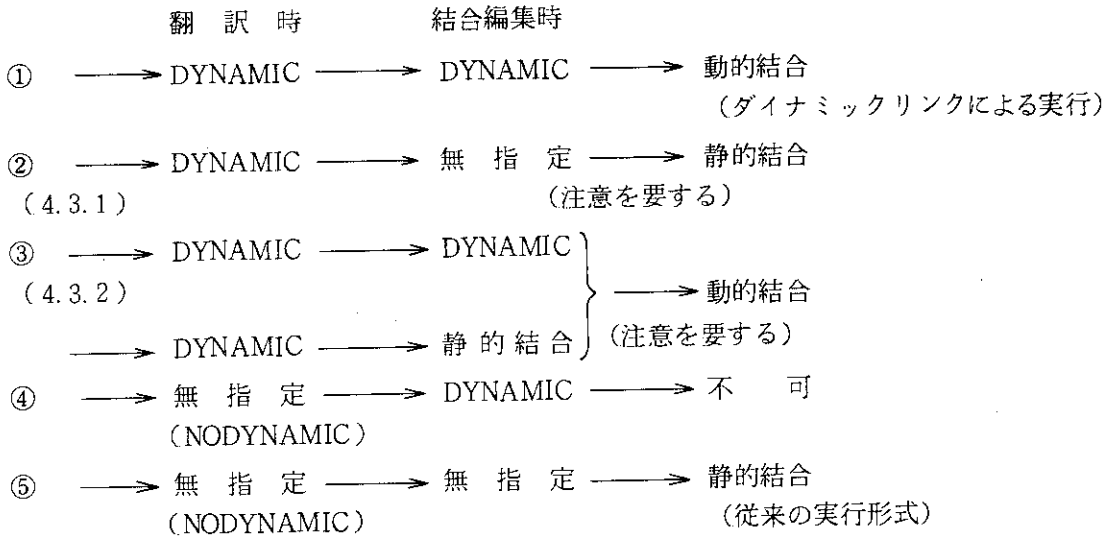



Fig. 4.1　Combinations of compiler and linkage editor options

# 5. その他の注意事項

## 5.1 言語仕様の互換性

FACOM OS Ⅳ FORTRAN 文法書で記述された言語仕様がすべて完全に適用される。

DYNAMIC／NODYNAMIC間では，完全に互換性がある。ただし，共通ブロックの採用のされ方には，注意を要するので，「3. 共通ブロックの採用」を参照すること。

## 5.2 副プログラムの数

一つのプログラム単位内で使用する共通ブロックの数と，そのプログラム単位内で呼び出す副プログラムの数および使用する入出力の種類の数（READとWRITEで2種類，その他に特別な入出力があれば，その種類数）の合計が255個以下でなければならない。

各プログラム単位で255個以下であれば，全プログラム単位では，255個以上でも問題はない。ただし，「6. ダイナミックリンク効果分析ツールの利用方法」で述べる，メモリの動的な使用状況を調べるツールを使用するためには，全プログラム単位で1000個以下であることが望ましい。

## 5.3 共通ブロック名

共通ブロック名と副プログラム名は一致してはならない。

## 5.4 共通ブロックの削除

一度，動的に結合された共通ブロックおよび副プログラムは削除されない。

ダイナミックリンクで作成されたロードモジュール以外のロードモジュールを同時に実行する場合には，実行時に採用される共通ブロックの大きさと初期値は，実行時に最初にその共通ブロックが採用された時に決まる。すでに結合された同じ名前の共通ブロックを削除した後に，再びその同じ名前で別の大きさと初期値をもった共通ブロックをローディングして，動的に結合することはできない。詳しくは，「3. 共通ブロックの採用（実行時の共通ブロックの採用）」を参照すること。

ただし，これは実行時の問題で，新しくオブジェクトモジュールから，動的に結合したロードモジュールを作成する場合には問題ない。

### 5.5　再入可能プログラム

再入可能プログラムでダイナミックリンクを使用する場合には，翻訳時と結合編集時のオプションで，RENTを追加指定する。

### 5.6　配列要素の参照

FORTRANソースプログラムにおいて，配列の要素を参照する時に，相対的な番地で参照してはならない。ダイナミックリンクを使用する時は，各配列ごとの絶対的な番地で参照するようにする。

# 6. ダイナミックリンク効果分析ツール(ダイナリート)の利用方法

## 6.1 ダイナリート実行の準備

ダイナミックリンク効果分析ツールを使って，動的結合による実行時のメモリ使用状況を調べるために，ユーザのソースプログラムに対して，次に述べる事項について，必要ならば若干の修正および確認を要する。

ダイナミックリンク効果分析ツール（Dynamic Link Effect Analysis Tool )を以後，ダイナリート（DYNALEAT）と略称する。

### 6.1.1 FORTRANソースプログラム

(1) データセット（ファイル）識別番号89

FORTRANプログラムの中で，ファイル識別番号89を使用している場合は，その番号を89以外の適当な番号に変更する必要がある。

FT89F001で定義されたファイルは，ダイナリートで，ダイナミックリンク情報（動的結合による実行時のCPU TIME，ELAPSED TIME，各プログラム単位のメモリ量などの情報のことを「ダイナミックリンク情報」という）をデータプールに書き込むのに使用している。

(2) 変数名等について

ユーザのFORTRANソースプログラムにおいて，変数名，配列名，コモン名などの名前で，XXXX，YYYY，ZZZZZZ，QQQQQQのいずれかを使用している場合には，これら以外の適当な名前に変更する必要がある。これらの名前はダイナリートで使用している。

(3) STOP文について

ユーザ・プログラムの実行の終了は，STOP文で終わるようにする。

正常終了の場合は，必らずSTOP文で終了するようにする。エラー処理ルーチン等で終了する場合も，できるかぎりSTOP文で終わるようにする。その他，EXITルーチンやアセンブラルーチン終了等も，できればSTOP文で終わるようにする。

ダイナリートでは，プリプロセッサでSTOP文をCALL文に自動的に変更し，実行が終了するときにダイナミックリンク情報を書き込むのに利用している。

(4) 順編成ファイルへの変更

ユーザのFORTRANソースプログラムが，すでに順編成ファイル（PSファイル）である場合は，変更の必要はない。そうでない場合は，ユーザのプログラムを一つの順編成ファイルに変更する必要がある。

GEMファイルを順編成ファイルに変更する方法を，TSSの場合とBATCHの場合について，次に示す。

GEMファイルのPSファイルへの変更

(i) TSSの場合

```
ALLOC DA(FORTPS.FORT) F(FPS) NEW SP(50 10) T UNIT(TSSWK)
GEM FORTGEM.FORT OLD
GROUP A
PUT A OUT(FPS)
```

(ii) BATCHの場合

```
//   EXEC GEMTOPS,
//    DSN='J9999.FORTGEM',
//    ODSN='J9999.FORTPS',
//    UNIT=TSSWK
```

6.1.2 アセンブラルーチンについて

ユーザ・プログラムにおいて，アセンブラルーチンを使用する場合は，FORTRANプログラムからの参照関係に十分注意を要する。

FORTRANソースプログラムとアセンブラルーチンを一緒にダイナリートで実行するときは，カタログ・プロシジャとして，PROCESSAを使用する。その例は，後ほど「6.2.1 ダイナリート実行の制御文」で示す。

また別な方法として，アセンブラルーチンだけを先にロードモジュールを作成しておいて，実行時にSTEPLIBでFORTRANのロードモジュールに付け加えて実行する方法がある。ただし，この場合アセンブラルーチンの翻訳時のオプションでNAME，結合編集時のオプションでALIASを指定する。

従って，アセンブラルーチンを一緒に使用する場合は，その構造，参照関係をよく知った上で使用することが望ましい。

6.1.3 その他の準備

(1) 静的結合時のメモリ使用量

ダイナミックリンクを使用せず，従来の方法で静的に結合して実行した時に，RUN STEP(実行時）で使用されたメモリ量を調べておく。

そのメモリ使用量は，図6.1に示すように，SYSTEM MESSAGES LISTのRUN STEPのメモリ量（単位はキロバイト）を参照すればよい。（図6.1）

この静的結合による実行時のメモリ使用量は，動的結合による実行時のメモリ使用量と比較するために，ダイナリートの入力データとして与える。与え方は「6.2.1 ダイナリート実行の制御文」を参照する。

```
                                                    <<<  SYSTEM MESSAGES LIST  >>>

   27      JDJ686I DDNAME REFERRED TO ON DDNAME KEYWORD IN PRIOR STEP WAS NOT RESOLVED
KDS70001I J3472     LAST ACCESS AT 13:09:11 ON 81.119
JDJ142I C3472054 FORT FORTHE - STEP WAS EXECUTED - COND CODE 0000
JDJ373I STEP /FORT     / START 81120.1620
JDJ374I STEP /FORT     / STOP  81120.1620 CPU    0MIN 00.45SEC SRB    0MIN 00.02SEC VIRT   320K
ACT0611                                   EXCP            2TIMES
JDJ142I C3472054 LINK LKED - STEP WAS EXECUTED - COND CODE 0000
JDJ373I STEP /LINK     / START 81120.1620
JDJ374I STEP /LINK     / STOP  81120.1621 CPU    0MIN 00.18SEC SRB    0MIN 00.03SEC VIRT   256K
ACT0611                                   EXCP          112TIMES
JDJ142I C3472054 RUN RUN - STEP WAS EXECUTED - COND CODE 0000
JDJ373I STEP /RUN      / START 81120.1621
JDJ374I STEP /RUN      / STOP  81120.1621 CPU    0MIN 00.06SEC SRB    0MIN 00.00SEC VIRT   164K
ACT0611                                   EXCP             0TIMES
JDJ375I  JOB /C3472054/ START 81120.1620
JDJ376I  JOB /C3472054/ STOP  81120.1621 CPU    0MIN 00.69SEC SRB    0MIN 00.05SEC

                    ************************************************************************
                    ****
                    ****          JAERI   COMPUTING   CENTER        OSIV/F4   E40 (V12/L01)
                    ****
                    ****      (( JOB  ACCOUNTING  INFORMATION  ))  SYSTEM-ID  ((  SYSC  ))
                    ****
                    ************************************************************************
```

**Fig.6.1 Used memory size in run step by static link**

(2) データプール・ノード名

ダイナミックリンクによる動的結合時の情報をデータプールを書き込む時のデータ名として，ノード名を適当な英数字８文字以内で決めておく。（原子力コード名の連想できるものが望ましい。）

ダイナリートへのノード名の入力の仕方は，「6.2.1　ダイナリート実行の制御文」を参照する。

なお，データプールについては，「6.2.2　データプールのファイルについて」で詳しく述べる。

(3) プログラム単位の個数

ダイナミックリンク機能を使用する上での制限は，一つのプログラム単位から参照される副プログラムやコモンなどの数が，255個以内であることであり，ユーザのプログラム全体では個数に制限はない。しかし，ダイナリートを利用する上では，現在，ユーザのプログラム全体で，主プログラム，副プログラムの数およびコモンの個数の合計が，1000個以内であるように制限されているので，このことを確認しておく。

(4) その他

共通ブロックの採用のされ方など，ダイナミックリンクに関する注意事項は，ダイナリートにおいても同様である。

## 6.2 ダイナリートの実行方法

### 6.2.1 ダイナリート実行の制御文

ダイナリートのジョブ制御文の例を示す。始めにFORTRANソースプログラムのみでダイナリートを利用する場合の例を示し，後でFORTRANソースプログラムとアセンブラルーチンで一緒にダイナリートを利用する場合について簡単に述べる。

(1) FORTRAN・ソースのみ

ここでは例として，FORTRANソースプログラムの順編成ファイルをJ9999．FORTPS．FORTとし，入力データのファイルをJ9999．SPLDATA．DATAとする。他に作業用ファイルは，使用しないものとする。データプールのノード名は，SAMPLEとし，ダイナミックリンクを使用せず静的結合による実行時のメモリ使用量は925キロバイトとする。（図6.2）

```
//PROCESS EXEC PROCESS,DSN='J9999.FORTPS'

//RUN     EXEC PGM=MAIN
//STEPLIB DD DSN=&&LM,DISP=(OLD,DELETE)
//         DD DSN=SYS1.FORTLIB,DISP=SHR

// EXPAND DISKTO,DDN=FT05F001,DSN='J9999.SPLDATA'
//FT06F001 DD SYSOUT=*

//FT89F001 DD DSN=&&ST3,DISP=(NEW,PASS),UNIT=WK10,
//            SPACE=TRK,(10,10)),DCB=(RECFM=FB,LRECL=80,
//            BLKSIZE=3200)

//SORTPOOL EXEC SORTPL

//SORT.SYSIN DD *
  925                          静的結合による実行時のメモリ使用量
/*
//POOL.SYSIN DD *
 'J9999','SAMPLE'              ユーザ番号とノード名
/*
```

Fig. 6.2 Example of the Dynaleat job control cards

(2) FORTRANとアセンブラ

FORTRANソースプログラムとアセンブラルーチンで一緒にダイナリートを利用する場合は，カタログプロシジャとしてPROCESSの代りに，PROCESSAを使用する。

従って，図6.2において，第1行目の制御文を次のように変更する。

```
//PROCESS EXEC PROCESSA,DSN='J9999.FORTPS',ASM='J9999.ASSEMBL'
```

ただし，DSN＝のパラメータは，FORTRANソースプログラムのファイルであり，ASM＝のパラメータは，アセンブラソースプログラムのファイルである。

その他の制御文は，図6.2と同じでよい。

(3) ダイナリートの実行時間

ダイナミックリンクを使用した場合の実行（RUN)に要する時間は，静的結合による実行時間とほぼ同じである。だが，ダイナリート全体では，翻訳，結合編集の時間の他に，ダイナミックリンク情報を採取および処理するために4つのジョブステップが加わるので，その時間が加算される。

翻訳，結合編集，実行を通して，CPU時間T．xかかるジョブであれば，T．（x＋1）程度でダイナリートは実行できると思われる。

6.2.2 データプールのファイルについて

(1) データプールへの書き込み

ダイナリートが実行されると，ユーザのFORTRANソースプログラムがダイナミックリンク機能を使って実行されると同時に，CPU時間，経過時間（elapsed time） およびメモリ使用量などのダイナミックリンク情報が，データプールのファイル（J3472．DATAPOOL．DATA）に書き込まれる。

ダイナミックリンク情報は，

J3472． HPOOL． J9999． NODENAME
（不変） （不変） （ユーザ番号）（ノード名）

というデータ名で，データプールに書き込まれる。

このユーザ番号とノード名は，ダイナリートのジョブ制御文で示したように，POOL．SYSINの入力で与える。

(2) データプールからの読み出し

ダイナミックリンク情報の書き込まれたデータプールのファイル（J3472．DATAPOOL．DATA）から，データを読み出して動的なメモリ使用状況を調べるためには，利用するユーザ（USER ID）に対して，J3472．DATAPOOL．DATAのファイルのUPDATE権が与えられていなければならない。

従って，始めてデータプールのファイルを参照するときは，J3472（計算センタ・原田・TEL 5975）に連絡下されば，PERMITをかけてUPDATE権を与えます。

データプールについての詳細は，「JAERI－M 8715 データプールの概念と機能」に述べられている。

## 6.3 動的メモリ使用状況のグラフ表示方法

### 6.3.1 ダイナリートの実行結果

ダイナリートを使って得られるダイナミックリンク情報から，データプールのグラフィック機能を利用して，動的結合による実行時のメモリ使用状況をグラフに出力する。

実行時間の経過と共に，必要になったモジュールを動的に結合していくことによって，メモリ（記憶領域）が使用されていく様子を二次元のグラフに表わす。

ダイナミックリンク情報として，CPU時間（CPU TIME），経過時間（ELAPSED TIME），メモリ使用量，それに全プログラム単位数（主プログラム，副プログラム，コモンの総数），動的結合による実行で実際に使用されたプログラム単位数（主プログラムと使用された副プログラムの合計数），および静的結合による実行時のメモリ使用量が，次に示す配列に入っている。（表 6.1 参照）

Table 6.1 Data obtained through dynamic link

| 配列名 | 内　　容 | 単位 | 型 |
|---|---|---|---|
| DTE | elapsed time | ( sec ) | REAL |
| DTC | cpu time | ( sec ) | REAL |
| MY | memory size | (100KB) | REAL |
| NMY | 静的結合による実行時の memory size | (100KB) | REAL |
| MM(1) | total subroutine 数 | ( 個 ) | INTEGER |
| MM(2) | Run subroutine 数 | ( 個 ) | INTEGER |
| MM(3) | 静的結合による実行時の memory size | ( KB ) | INTEGER |

（注）DTE，DTC，MY，NMYは，すべて 1 次元配列で，要素は 1000 個用意してある。
MM は，1 次元配列で要素は 3 個あり，各要素には上に示した内容が入っている。

主プログラム（MAIN）から実行が始まり，副プログラムが次々に呼び出されて結合される。例えば，5 番目に副プログラムSUBVが呼び出されたとする。このとき，実行開始から，副プログラムSUBVが呼び出された時刻までのCPU時間が，DTC（5）に入り，経過時間がDTE（5）に入る。

CPU時間は，FORTRANの基本サブルーチンCLOCKMで測定したミリ秒単位の値を秒単位に直して表わしている。

経過時間は，FORTRANの基本サブルーチンTIMEで測定したミリ秒単位の時刻をもとにして経過時間を秒単位で表わしている。

また，副プログラムSUBVが呼び出された時，主プログラムを含めて，5 個のプログラムモジュールのサイズと，このときまでに参照された共通ブロック（コモン）のサイズの合計が，MY（5）に入る。

メモリ使用量は，結合編集時のマップから取った各プログラムモジュールのキロバイト単位のサイズをもとにして，実行時に使用された順序で累積したものを100キロバイト単位に直したものである。

これらのダイナミックリンク情報から，データプールのグラフィック機能を使ってグラフを出力する。

データプールのグラフィック機能は，会話型で利用できるため，TSSのグラフィック端末（ソニーテクトロニクスT 4014）を使って，グラフを見ながらパラメータを変更することによって，より適切なグラフを得ることができる。グラフは，T 4014のハードコピーで取ることができる。

TSS端末からのグラフィック機能の使い方については，次の「6.3.2　データプールのグラフィック機能」で述べる。

### 6.3.2　データプールのグラフィック機能

次にデータプール・システムのグラフィック機能を使って，データプールに書き込まれたダイナミックリンク情報を2次元のグラフに表わす手順の例を示す。

各コマンドの説明は，手順の例の後で述べる。

データプール・システムのコマンドの詳細については，JAERI－M 8175 データプールの概念と機能「3.　TSS用データプール処理システム」に述べられている。

使用例を次に示す。

```
 READY
ALLOC DA('J3472.DATAPOOL.DATA') F(FT01F001) SHR
 READY
CALL 'J9131.DPS.LOAD'
 TEMPNAME ASSUMED AS MEMBERNAME
 08200?A F01=J3472.HPOOL,LRECL=133/U,MAXRCD=1000
 08200?SYN SPL=J3472.HPOOL.J9999.SAMPLE

 08200?LD SPL/MM
 * * * I/O LIST * * *
 DTE,DTC,MY,NMY,MM
 * * * CONTROL VARIABLES * * *
 * * * SYMBOLS * * *
 MM
 * * * DATA * * *
```

全プログラム単位数（主，副プログラム総数）：83

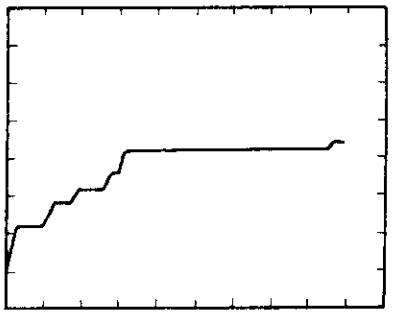
実行時に使用された主，副プログラム数：52

925：静的結合による実行時のメモリ使用量（KB）

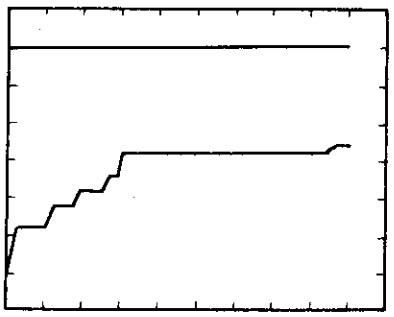
```
08200?LD SPL/MY,1,52,1
* * * I/O LIST * * *
DTE,DTC,MY,NMY,MM
* * * CONTROL VARIABLES * * *
* * * SYMBOLS * * *
MY
* * * DATA * * *
 0.1400000E+00 0.1900000E+00 0.2000000E+00 0.2100000E+00
     ...           ...           ...           ...
 0.5719999E+01 0.5840000E+01 0.6240000E+01 0.6759999E+01
08200?
```

```
08200?GS SIZE
 TYPE IN (XSIZE,YSIZE) NUMBER OF GRIDS FOR EACH AXIS
14400? 10,10

08200?GS LIM
 TYPE IN (XMIN,XMAX,YMIN,YMAX)
13600? 0,0,0,10

08200?GS LX
 24 CHARACTERS AVAIRABLE FOR X-AXIS LABEL
09900? ELAPSED TIME(SEC)

08200? GS LY
 24 CHARACTERS AVAIRABLE FOR Y-AXIS LABEL
10900? MEMORY (100KB)

08200? GS TITLE
 72 CHARACTERS AVAIRABLE FOR THE TITLE
12700? SAMPLE (DYNAMIC LINK)
08200?
```

```
08200? GX SPL/DTE,1,52,1
08200? GY SPL/MY,1,52,1


08200? G OPEN
    . . . . . . .
    . . . . . . .
    . . . . . . .
08200? G 2D
```



```
 return
08200? GY SPL/NMY,1,52,1
08200? G ADD
```



```
 return
08200? G CLOSE
08200? STOP
  . . .
 READY
```

ALLOC

データプールのファイルを定義する。

CALL 'J9131. DPS. LOAD'

データプール・システムを呼び出す。

A

ATTACHコマンドは，既に作成されているデータプールを割付ける。ATTACHコマンドはAと省略形で使える。

SYN

SYNONYMコマンドは，データプール内のノードを表現する同義語を定義する。同義語は8文字以内の英数字名である。SYNONYMコマンドは，SYNと省略形で使える。

LD データ名／配列名，初期値，終値，増分値

LISTDコマンドは，データの持つ数値データを出力する。初期値，終値，増分値を省略した場合は，すべてのデータを出力する。LISTDコマンドは，LDと省略形で使える。

GS SIZE

X軸とY軸の分割数を決める。このコマンドで分割数を設定しなかった時は，自動的にX軸分割数10，Y軸分割数7に設定される。設定できる最大値は，X軸分割数10，Y軸分割数12である。

適切な分割数の選び方については，「6.3.3 グラフのスケールについて」で述べる。

GS LIM

X軸とY軸の目盛りの最大値と最小値を設定する。このコマンドで目盛りを設定しなかった時または，最大値と最小値の両方を0に設定した時は，システム側で適切な値を自動的に設定する。例に示したように，X軸は，最小値0，最大値0で自動設定の値がとられ，Y軸は，最小値0，最大値10でユーザが与えた値を設定するようにもできる。

今回は，動的結合による実行時のメモリ使用状況のグラフに，静的結合による実行時のメモリ使用量を重ね書きするため，Y軸の最大値の設定には工夫がいる。このことについては，「6.3.3 グラフのスケールについて」で述べる。

GS LX

このコマンドは，X軸に描くラベルを指定する。X軸に書けるラベルの長さは，24文字以内である。

GS LY

このコマンドは，Y軸に描くラベルを指定する。Y軸に書けるラベルの長さは，24文字以内である。

GS TITLE

このコマンドは，グラフのタイトルを指定する。このタイトルは，グラフの上部で枠の外側に書かれる。タイトルで指定できる文字数は，72文字以内である。

GX データ名／配列名，初期値，終値，増分値

このコマンドは，X軸のグラフィック・データを設定する。初期値，終値，増分値を省略すると配列のすべてのデータが設定されるが，今回は，動的結合による実行時に実際に使用された副

プログラム数（Run Subroutine数で，配列要素MM(2)にその値が入っている。従って，始めにLDコマンドで，その数を調べておく必要がある。）を終値に設定し，初期値と増分値は，1に設定する必要がある。そうしないと，X軸のデータの入っている配列DTEやDTCは，1000個要素が用意されているので適切なグラフが得られない。

GY データ名／配列名，初期値，終値，増分値

このコマンドは，Y軸のグラフィック・データを設定する。コマンドGXと同様に，終値については，実行時に実際に使用された副プログラム数を設定する。

G OPEN

このコマンドは，グラフィック処理の開設を行う。このコマンドを入力すると，いくつかのメッセージと数行のエラーメッセージが出るが，気にする必要はない。

G 2D

このコマンドを入力することによって，2次元のグラフが出力される。ただし，このコマンドを入力する前に，X軸とY軸のデータをGX，GYのコマンドで設定しておかなければならない。

このコマンドで出力された，動的結合による実行時のメモリ使用状況のグラフに，後で静的結合による実行時のメモリ使用量を重ね書きするため，このコマンドを入力後は重ね書きするまでは，ROBOUT，PAGE等の画面をクリアするキーは打ってはいけない。もし何かの都合で消えてしまった時は，もう一度，G 2Dでグラフを描いてから次のステップに進む。

このコマンドで出力されたグラフが，その枠やスケールから考えて，適切なものでない場合はGS SIZEやGS LIMのコマンドでパラメータの設定を変更して描き直さなければならない。ただし，後で重ね書きするため，その点を考慮に入れなければならない。これらの点について，適切なグラフを書くためのパラメータの設定については，「6.3.3　グラフのスケールについて」で述べる。

G 2Dのコマンドでグラフが出力されている時，復改（RETURN）のキーを押すと，グラフの右上部に 08200 ？の文字列が出て入力可能となる。

次に，GYのコマンドでY軸のデータとして，静的結合による実行時のメモリ使用量の入った配列NMYを設定する。X軸のデータは同じでよいので変更しない。

G ADD

このコマンドは，現在描かれているグラフに重ねて，2次元のグラフを描く。

作業が終ったら，復改（RETURN）キーを押すと 08200 ？が出て，入力可能となる。

G CLOSE

このコマンドは，グラフィック処理を終了するコマンドである。

STOP

このコマンドは，データプール・システムの処理を終了するコマンドである。終了時に数行のエラーメッセージが出る場合があるが，気にする必要はない。データプール・システム終了後はREADY状態になる。

6.3.3　グラフのスケールについて

このデータプールのグラフィック機能は，会話型処理で実際にグラフを見ながら，パラメータを変更して，適切なスケール，目盛，配置でグラフを出力できる点に特徴がある。
（ただし，目盛の数字の大きさ等，改良を要すると思われる点もある。）

従って，一度描いてみたグラフが適切なグラフでない場合，コマンドGS SIZEおよびGS LIMによって，適切な値を設定しなおして，グラフを描くことができる。このとき，パラメータは，変更しなければ，前にセットした値が使われる。

特に注意しなければならないのは，GS LIMで設定するY軸の目盛の最大値$Y_{max}$とGS SIZEで設定するY軸の分割数$Y_{size}$である。

今回は，ダイナミックリンクによる効果を見るため，最初に動的結合による実行時のメモリ使用状況をグラフに出力し，次に静的結合による実行時のメモリ使用量をグラフに重ね書きする。

このため，$Y_{max}$ は，予め静的結合による実行時のメモリ使用量NMY（MM（3）の値を100で割って，100キロバイト単位に直した値がNMYの配列に入っている。）を考慮して設定しなければならない。また，その$Y_{max}$ に応じて，$Y_{size}$ も決めなければならない。

次に，$Y_{max}$ と$Y_{size}$ の決め方を示す。

まず，$Y_{max}$ の仮の値を定める。

$$Y_{max} = NMY \times 1.25$$

これは，NMYの値が

$$0.7 \times Y_{max} \leqq NMY \leqq 0.9 \times Y_{max}$$

となるようにするためである。
こうして定められた$Y_{max}$ の値に対して

$$\frac{Y_{max}}{0.4}, \quad \frac{Y_{max}}{0.5}, \quad \frac{Y_{max}}{1}, \quad \frac{Y_{max}}{2}$$

これらの4つの値のうち，12以下で12に最も近い整数を$Y_{size}$ とする。
そして，求められた$Y_{size}$ の値に対して，

$$Y_{max} = 1 \times 10^{0} \times Y_{size}$$
$$\text{or} \quad Y_{max} = 2 \times 10^{0} \times Y_{size}$$
$$\text{or} \quad Y_{max} = 4 \times 10^{-1} \times Y_{size}$$
$$\text{or} \quad Y_{max} = 5 \times 10^{-1} \times Y_{size}$$

これらのうち，$Y_{size}$ を求めた時の分母の値によって，1つ式を選び，$Y_{max}$ の値を求め直す。
例えば，

$$NMY = 9.25 \qquad (100\,KB\text{単位})$$

とすると，

$$Y_{max} = NMY \times 1.25 \fallingdotseq 11.6$$

$$\frac{Y_{max}}{0.4} = 29 \quad , \quad \frac{Y_{max}}{0.5} = 23.2 \quad , \quad \frac{Y_{max}}{1} = 11.6 \quad , \quad \frac{Y_{max}}{2} = 5.8$$

従って

$$\frac{Y_{max}}{1}$$

の値をとって，

$$Y_{size} = 12$$

これから $Y_{max}$ を求めると

$$Y_{max} = 1 \times 10^{0} \times Y_{size} = 12$$

故に，NMY＝9.25 のときに，

$$Y_{size} = 12 \quad , \quad Y_{max} = 12$$

の値が最適であると思われる。

以上のようにして定められたパラメータの値を設定するときは，まず，GS SIZEで $X_{size}$ と $Y_{size}$ を設定し，次にGS LIMで $X_{min}$ ，$X_{max}$ ，$Y_{min}$ ，$Y_{max}$ を設定する。

NMY＝9.25 のときの例を示すと，

GS SIZE

10 , 12

GS LIM

0 , 0 , 0 , 12

### 6.3.4 グラフの見方

グラフによって，動的結合による実行時のメモリ使用状況，及びダイナミックリンクによるメモリ減少の様子を把握する。

グラフのX軸（横軸）は，経過時間（ELAPSED TIME）またはCPU時間を秒単位で表わしY軸（縦軸）は，メモリ使用量を100キロバイト単位で表わす。（図6.3，図6.4参照）

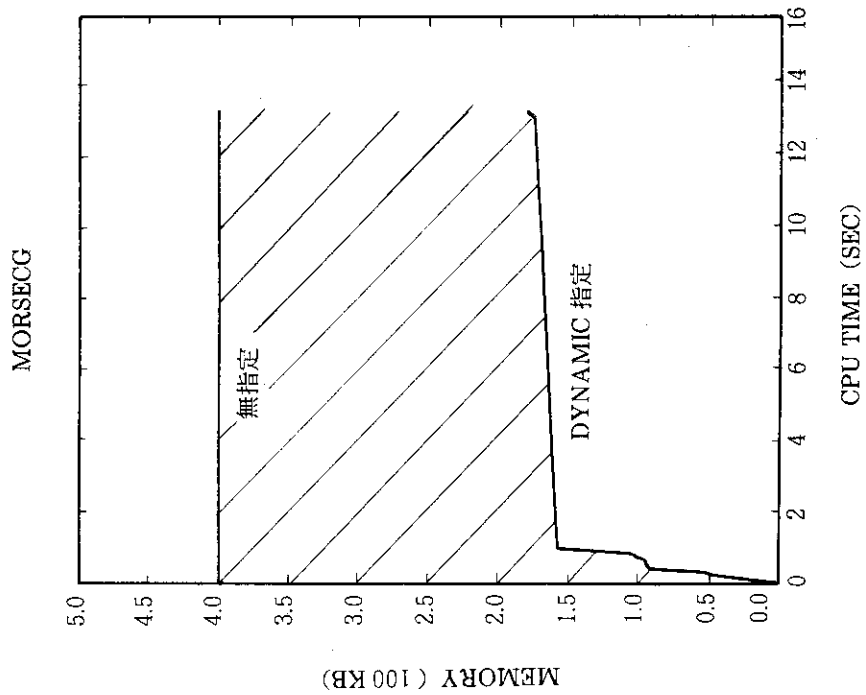



Fig. 6.4 Dynamically used memory (MORSE-CG)

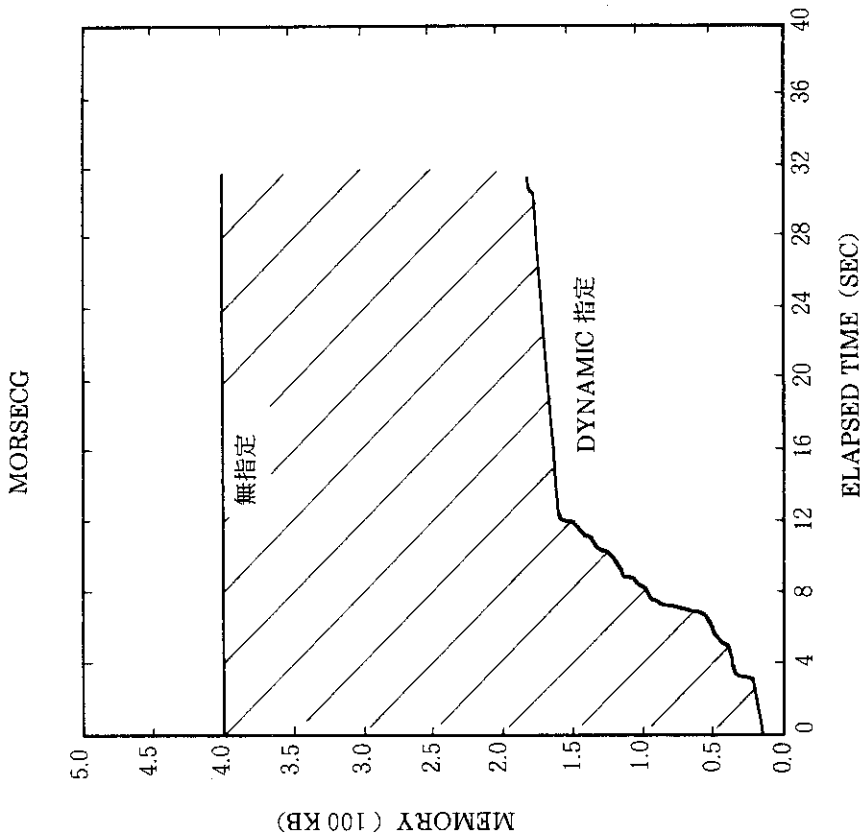



Fig. 6.3 Dynamically used memory (MORSE-CG)

グラフ上方の水平な線は，ダイナミックリンクを使用せず，静的結合によって実行した時のメモリ使用量を表わしている。

グラフ下方の階段状の線は，ダイナミックリンクを使用して，実行時に必要になった副プログラムや共通ブロックを次々に動的に結合して実行した時のメモリ使用状況を表わしている。

図6.3や図6.4で斜線で示された水平な線と階段状の線の間の部分が，メモリが減少した分（ダイナミックリンク機能を使うことによって節約された分）である。

多くのモジュールを含んだプログラムがあり，実行時には，条件によって（入力パラメータによって），その一部のモジュールしか実行されない場合には，ダイナミックリンクを利用すると効果がある。

原子力コードなどにおいて，いろいろな場合に適用できるように多くのモジュールを持っている場合，または，今まで各ケースによって別々のプログラムで実行していた一連の原子力コードをひとまとまりにして大きなコードにした場合など，ダイナミックリンクを使用すれば，1つのケースで必要なモジュールのみが動的に結合され，メモリは1ケースに必要な量だけでよい。

また，実行時間の経過と共に徐々にメモリが使用されていくプログラムにも効果がある。

## 6.4 ダイナリートの構造

ダイナミックリンク効果分析ツール（ダイナリート）の構造の概略を図6.5に示す。

各ステップの処理の概要は，次に述べるとおりである。

処理1

プログラム単位の全てに1行のCOMMON文を挿入する。

COMMON／XXXX／XXXX

STOP文をCALL文に変換する。

例えば，STOP 99 → CALL　YYYY

プログラム単位に含まれる共通ブロック名とプログラム名の表を作成する。

処理2

DYNAMIC指定で作成したロードモジュールを入力として，JSGLISTのLISTPDSを実行する。

その出力ファイルから，各プログラム単位及び共通ブロックのメモリサイズの表を作成する。

処理3

処理1，処理2及び実行から作成されたデータを入力として，実行されたプログラム単位にデータをソートする。

プログラム単位によるメモリ使用状況の図（グラフ1）を作成する。

処理4

処理3でソートしたデータをデータ・プールに書き込む。

処理5

TSSのグラフィック端末（T 4014）を用いて，データ・プールのグラフィック・ユーティリティにより，グラフ（グラフ2）を作成する。

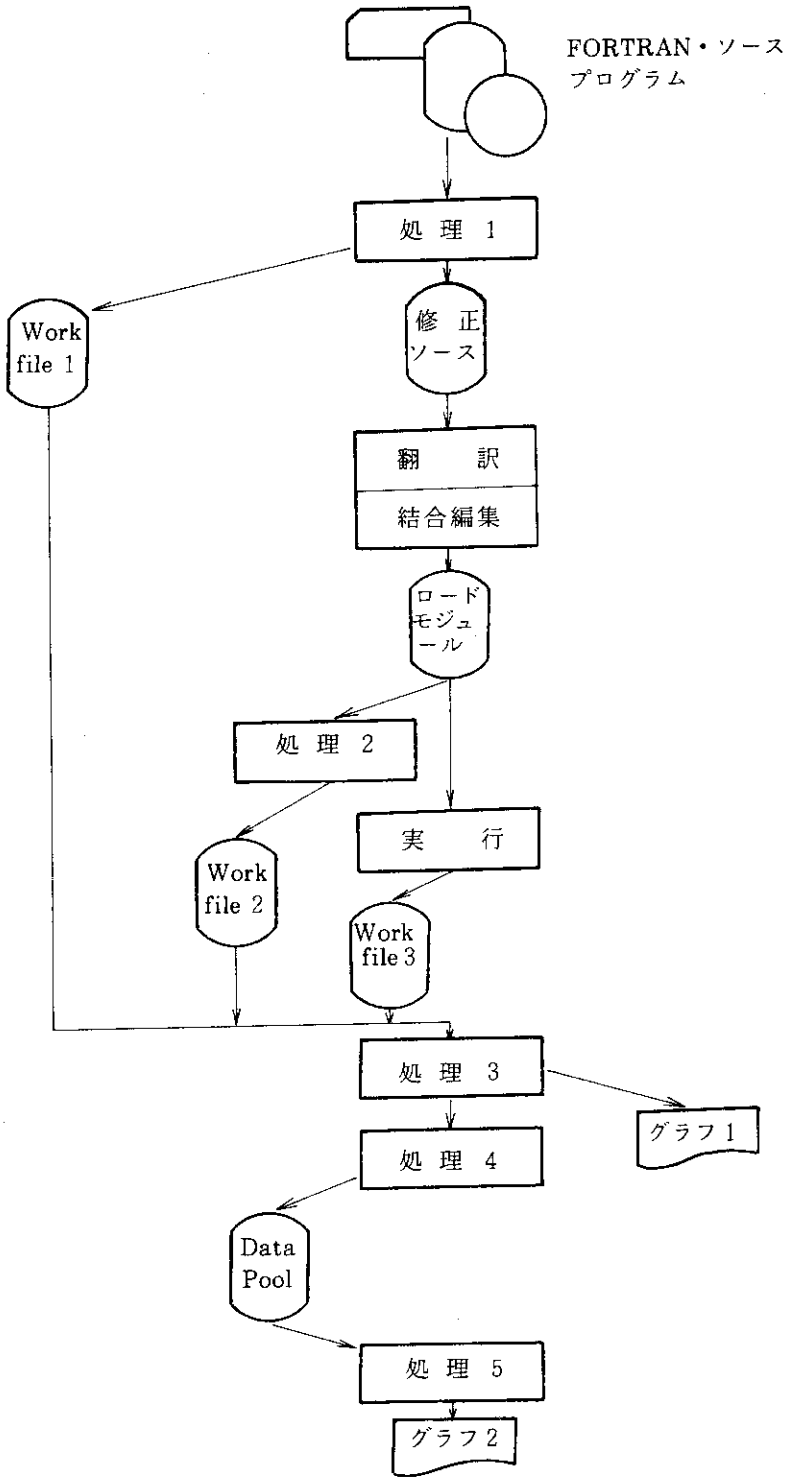



Fig. 6.5 Structure of the Dynaleat

# 7. 原子力コードへのダイナミックリンクの適用例

すでに図6.3，図6.4で示したMORSECGを含めて，7つの原子力コードにダイナミックリンクを適用し，ダイナリートで調べた動的なメモリの使用状況を次に示す。

ただし，表7.2の数値は，サンプル入力データによるものである。

Table 7.1 Names and purposes of nuclear codes which are used for dynamic link test

| コード名 | 種類，計算内容 | グラフ |
|---|---|---|
| ALARM－P2 | 加圧水型原子炉・大破断用ループ・コード<br>1次元ノード・ジャンクション・モデル採用。ループ各部の圧力，流量，エンタルピなどを計算する。 | 図7.1，図7.2 |
| MORSE－I | 3次元モンテカルロ法輸送計算コード<br>トカマク型核融合炉の核設計，遮蔽設計を行うためのトーラス形状の扱えるモンテカルロ法放射線輸送計算コード | 図7.3，図7.4 |
| MORSE－CG | 多目的モンテカルロ法中性子，ガンマ線輸送計算コード<br>CGは，球，直方体，円柱等のような簡単な形の組み合せによって，一般的な3次元の具体的な形状が扱える。 | 図6.3，図6.4 |
| ANISN | 1次元Sn輸送計算コード<br>原子炉，核融合炉の核設計，遮蔽設計のために，球，平板，無限円柱など1次元形状で中性子束，ガンマ線束を計算し，それらによる各種反応率，線量率を計算する。 | 図7.5，図7.6 |
| CITATION | 3次元中性子拡散コード<br>臨界安全性評価コードシステムの1つとして，定形燃料体系用の配列解析を行う3次元拡散コードである。 | 図7.7，図7.8 |
| VENTURE | 3次元中性子拡散コード<br>中性子輸送に対して，有限階差拡散理論近似を適用する多群中性子問題を解く。 | 図7.9，図7.10 |
| EQUCIR | 自由境界トロイダルプラズマのMHD平衡計算コード<br>プラズマ平衡配位の計算。プラズマ電流，平衡配位，外部導体電流の時間変化。平衡配位として得たいプラズマ形状のための外部導体の電流値の計算。 | 図7.11，図7.12 |

Table 7.2 Number of subroutines and memory spaces actually linked in execution

| コ　ー　ド　名 | 全サブルーチン数 | 実行サブルーチン数 | CPU TIME (SEC) | ELAPSED TIME (SEC) | NODYNAMIC (無指定)(KB) | DYNAMIC指定 (KB) |
|---|---|---|---|---|---|---|
| ALARM－P2 | 125 | 100 | 1.7 | 19 | 1152 | 960 |
| MORSE－I | 110 | 68 | 60 | 80 | 640 | 300 |
| MORSE－CG | 83 | 52 | 14 | 28 | 400 | 175 |
| ANISN | 48 | 29 | 450 | 1400 | 1536 | 1390 |
| CITATION | 207 | 42 | 95 | 165 | 1792 | 910 |
| VENTURE | 301 | 150 | 24 | 280 | 2088 | 1549 |
| EQUCIR | 109 | 37 | 15 | 24 | 1012 | 682 |

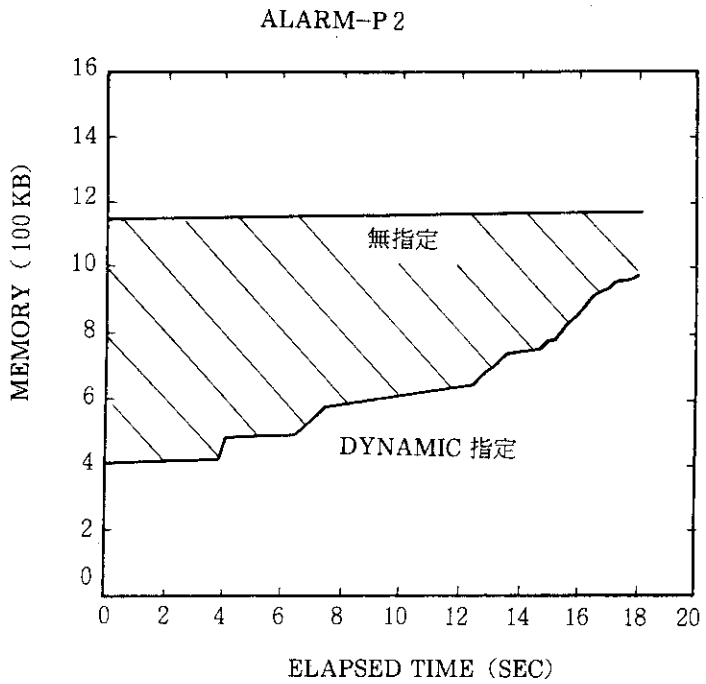



Fig. 7.1 Dynamically used memory (ALARM-P2)

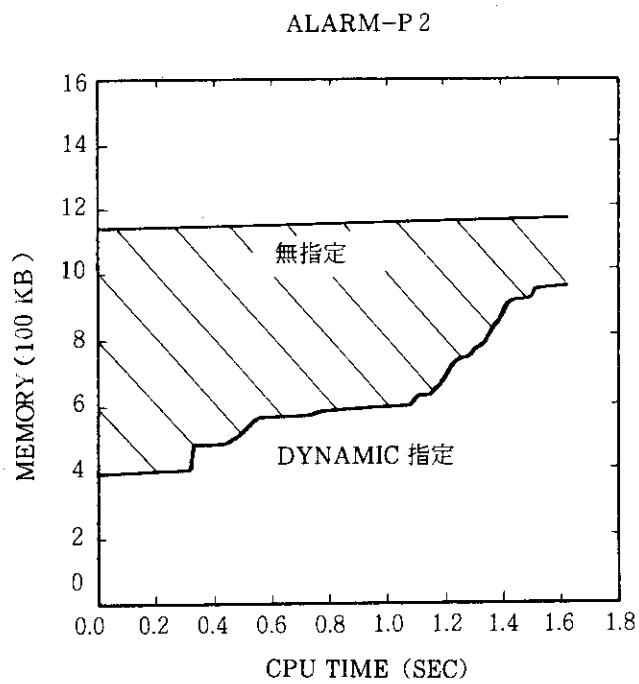



Fig. 7.2 Dynamically used memory (ALARM-P2)

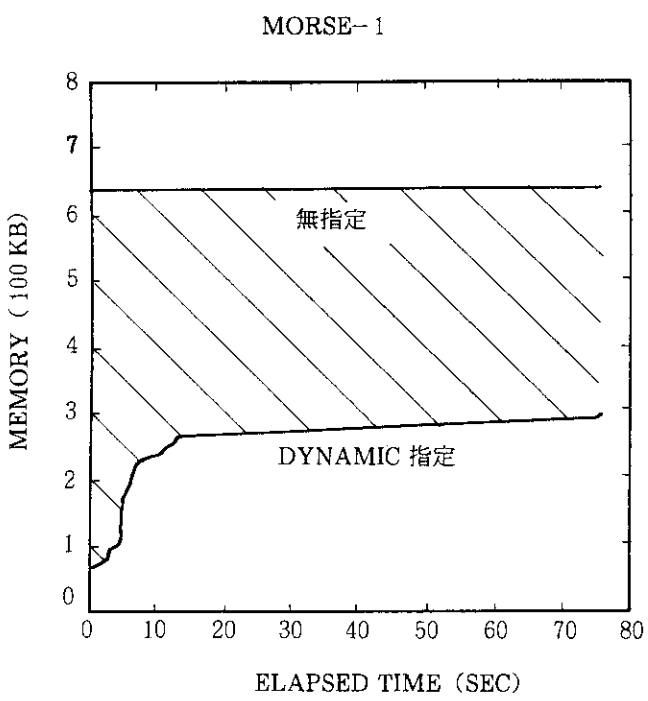



Fig. 7.3 Dynamically used memory (MORSE-I)

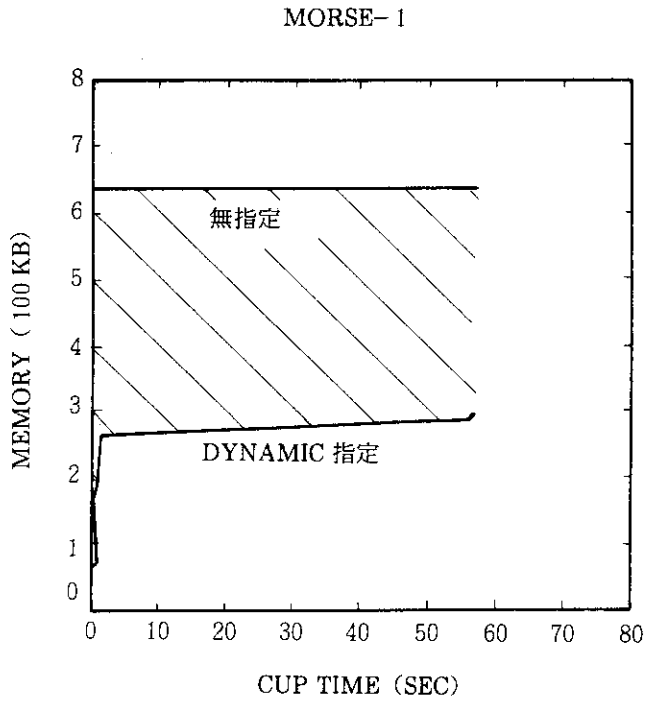



Fig. 7.4 Dynamically used memory (MORSE-I)

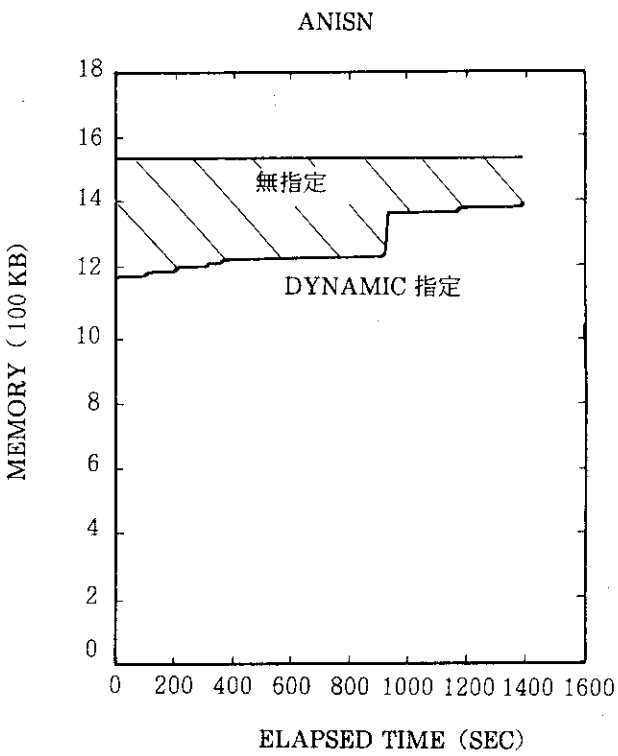



Fig. 7.5 Dynamically used memory (ANISN)

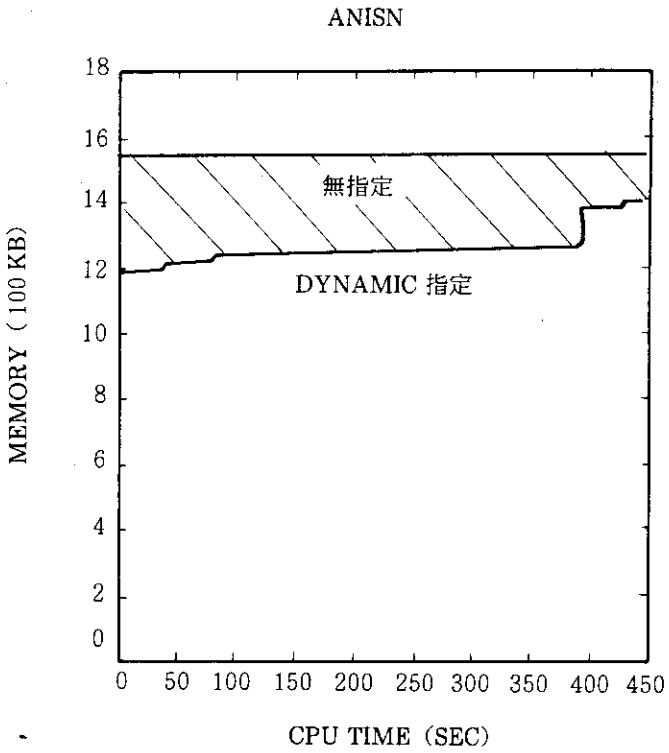



Fig. 7.6 Dynamically used memory (ANISN)

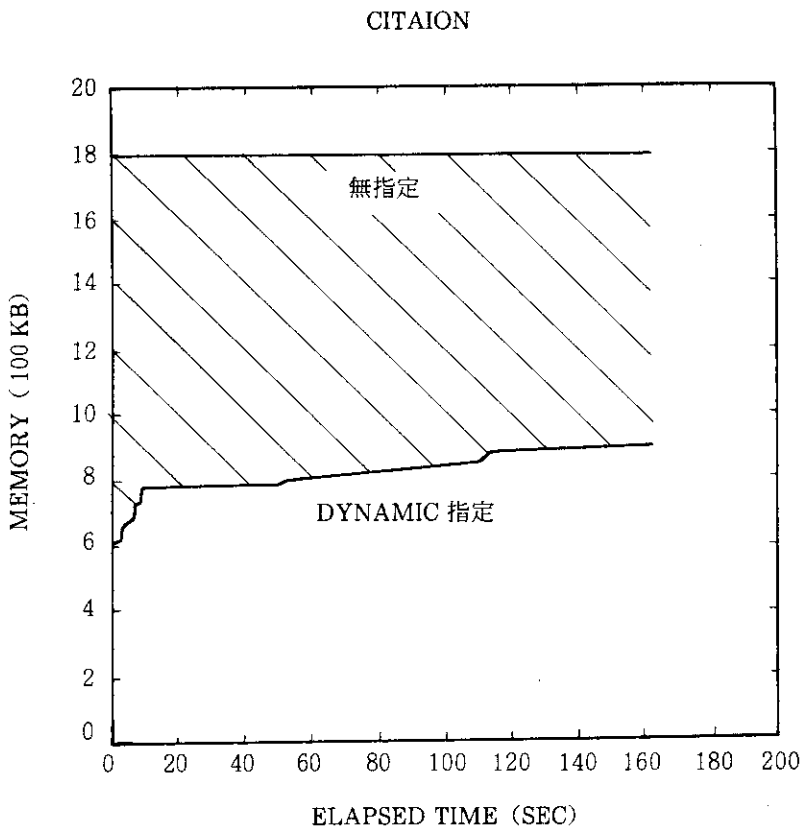



Fig. 7.7 Dynamically used memory (CITATION)

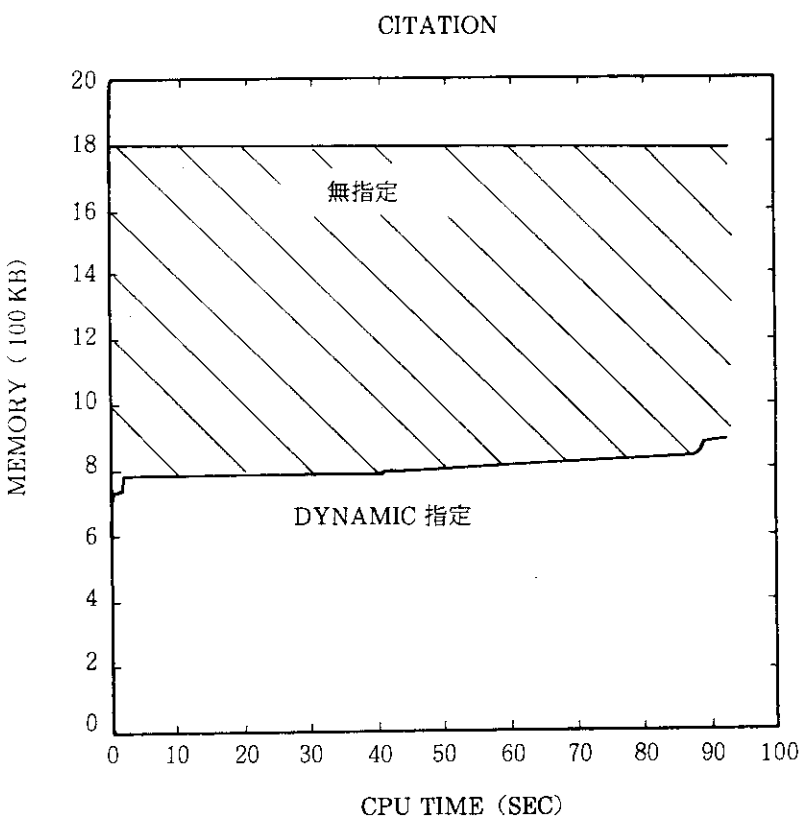



Fig. 7.8 Dynamically used memory (CITATION)

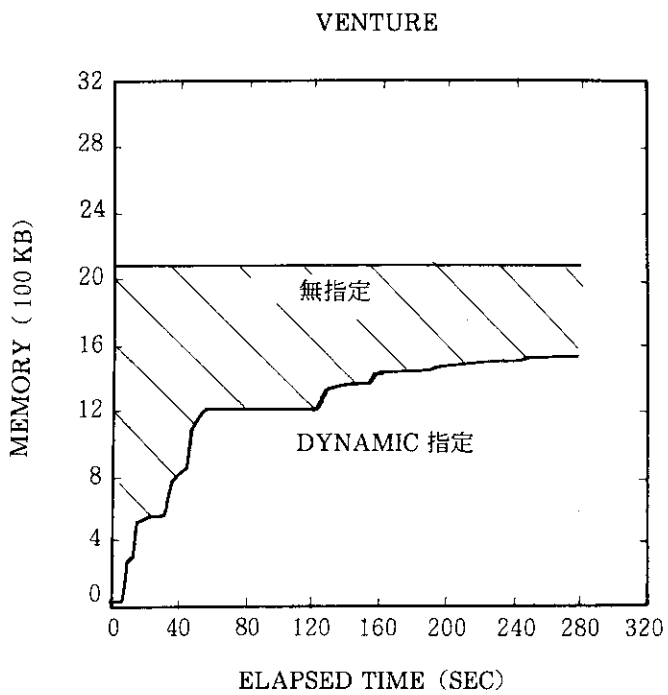



Fig. 7.9 Dynamically used memory (VENTURE)

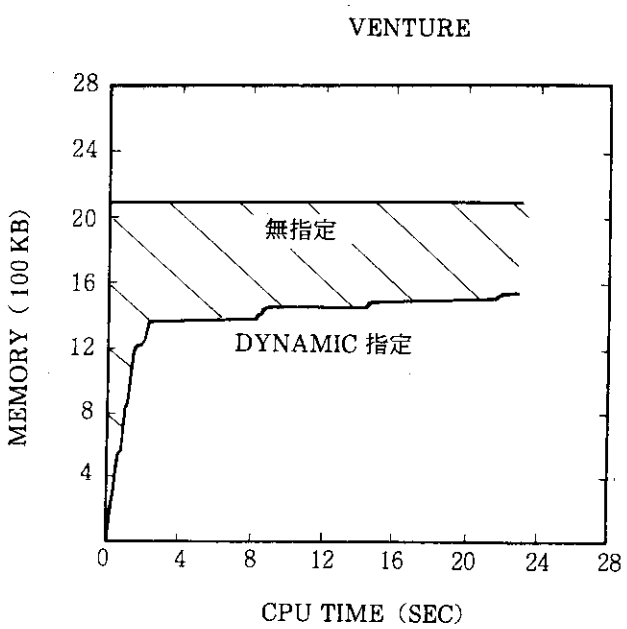



Fig. 7.10 Dynamically used memory (VENTURE)

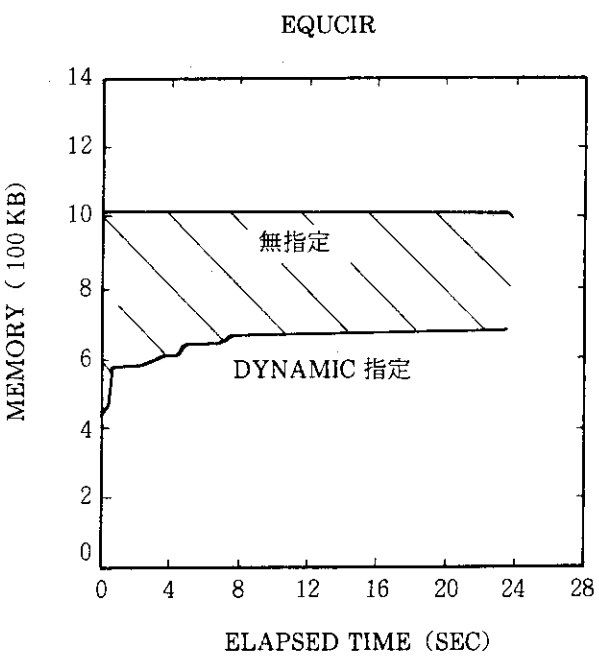



Fig. 7.11 Dynamically used memory (EQUCIR)

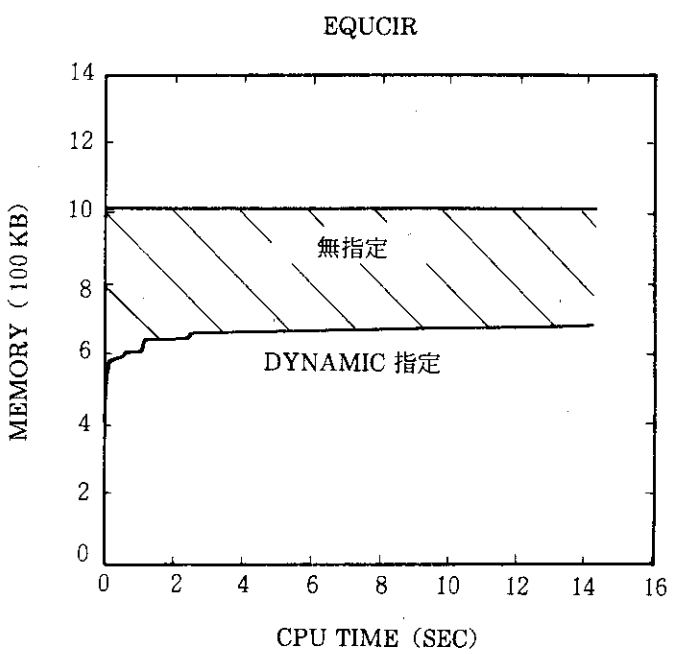



Fig. 7.12 Dynamically used memory (EQUCIR)

# 8. おわりに

現在，原研で使用されている原子力コードのかなりの部分が，ほとんどそのままの形でダイナミックリンク機能を利用でき，かなりのメモリ節約がはかられる。

また，メモリの問題からひとまとまりのコードにできなかった原子力コードもダイナミックリンク機能によって使用できるようになり，原子力コード・システムのモジュール化も促進されるようになる。

ダイナミックリンク効果分析ツール（ダイナリート）については，データプールのグラフィック機能等にまだ改良の余地があるので，利用者の意見も参考にして改良していきたい。

# 謝　　辞

原研の要求に応え，ダイナミックリンク機能の実現に努力していただきました富士通(株)の皆様，特にLP（ランケージ・プロセッサ）部　田口尚三部長，野崎邦明第2LP課長，また，これら機能のテスト等でお世話になりました科学エネルギー・システム部第1原子力課　西常義氏に感謝致します。

ダイナミックリンクの使用に当っては，外来研究員（富士通(株)）滝川好夫氏に数々の適切な助言をいただきましたことに感謝致します。

また，これは，原研が推進している原子力コードの総合化のために必要な計算機の機能のひとつとして，実現されたものである。

原子力コードの総合化に関し，日頃指導，激励していただきます桂木学安全工学部長（総合化専門部会長），平川隆計算センタ室長，総合化専門部会専門委員（次表のメンバ）の方々に感謝致します。

# 8. おわりに

現在，原研で使用されている原子力コードのかなりの部分が，ほとんどそのままの形でダイナミックリンク機能を利用でき，かなりのメモリ節約がはかられる。

また，メモリの問題からひとまとまりのコードにできなかった原子力コードもダイナミックリンク機能によって使用できるようになり，原子力コード・システムのモジュール化も促進されるようになる。

ダイナミックリンク効果分析ツール（ダイナリート）については，データプールのグラフィック機能等にまだ改良の余地があるので，利用者の意見も参考にして改良していきたい。

# 謝　　辞

原研の要求に応え，ダイナミックリンク機能の実現に努力していただきました富士通(株)の皆様，特にLP（ランケージ・プロセッサ）部　田口尚三部長，野崎邦明第2LP課長，また，これら機能のテスト等でお世話になりました科学エネルギー・システム部第1原子力課　西常義氏に感謝致します。

ダイナミックリンクの使用に当っては，外来研究員（富士通(株)）滝川好夫氏に数々の適切な助言をいただきましたことに感謝致します。

また，これは，原研が推進している原子力コードの総合化のために必要な計算機の機能のひとつとして，実現されたものである。

原子力コードの総合化に関し，日頃指導，激励していただきます桂木学安全工学部長（総合化専門部会長），平川隆計算センタ室長，総合化専門部会専門委員（次表のメンバ）の方々に感謝致します。

| 部　会　名 | 氏　名 | 機　関　名 |
|---|---|---|
| 原子力コード総合化専門部会 | ○桂　木　　学 | 原研・安全工学部 |
| 〃 | 下　桶　敦　則 | (財)原子力工学試験センター 原子力安全解析所 |
| 〃 | 鴻　坂　厚　夫 | 〃 |
| 〃 | 荒　井　長　利 | 原研・多目的炉設計研究室 |
| 〃 | 常　松　俊　秀 | 〃　理論解析研究室 |
| 〃 | 浅　井　　清 | 〃　計算センター |
| 〃 | 竹　田　辰　興 | 〃　理論解析研究室 |
| 〃 | 小　山　謹　二 | 〃　高速炉物理研究室 |
| 〃 | 大　西　信　秋 | 〃　反応度安全研究室 |
| 〃 | 内　藤　淑　孝 | 〃　原子炉データ解析室 |
| 〃 | 茅　野　政　道 | 〃　環境第１研究室 |

○印は専門部会長

# 参　考　文　献


[ 1 ]　富士通マニュアル：リンケージエディタ／ローダ使用手引書，富士通（1976）

[ 2 ]　富山峯秀．他：データプールの概念と機能，JAERI－M 8715，日本原子力研究所（1980）

| 部　会　名 | 氏　名 | 機　関　名 |
|---|---|---|
| 原子力コード総合化専門部会 | ○桂　木　　学 | 原研・安全工学部 |
| 〃 | 下　桶　敦　則 | (財)原子力工学試験センター 原子力安全解析所 |
| 〃 | 鴻　坂　厚　夫 | 〃 |
| 〃 | 荒　井　長　利 | 原研・多目的炉設計研究室 |
| 〃 | 常　松　俊　秀 | 〃　理論解析研究室 |
| 〃 | 浅　井　　清 | 〃　計算センター |
| 〃 | 竹　田　辰　興 | 〃　理論解析研究室 |
| 〃 | 小　山　謹　二 | 〃　高速炉物理研究室 |
| 〃 | 大　西　信　秋 | 〃　反応度安全研究室 |
| 〃 | 内　藤　淑　孝 | 〃　原子炉データ解析室 |
| 〃 | 茅　野　政　道 | 〃　環境第１研究室 |

○印は専門部会長

# 参　考　文　献


[ 1 ]　富士通マニュアル：リンケージエディタ／ローダ使用手引書，富士通（1976）

[ 2 ]　富山峯秀，他：データプールの概念と機能，JAERI－M 8715，日本原子力研究所（1980）