数理解析研究所講究録
359巻 1979年 78-99

78

数学ソフトウェアの品質保証について

――― SSLⅡの開発 ―――

富士通　　三上次郎*, 山下真一郎

═ 目次 ═

頁

§1　はじめに ---------- 2

§2　高品質ライブラリの条件と品質保証 ---------- 3

§3. SSLⅡ (Scientific Subroutine LibraryⅡ)
の開発 ---------- 7

§3.1　必要最少限の開発量(信頼性) ---------- 7

§3.2　徹底したテスト(信頼性) ---------- 13

§3.3　容易なサブルーチンの選択(使い易さ) --- 16

§3.4　統一された仕様(使い易さ) ---------- 17

§4. 数学ソフトウェアライブラリ開発の
今後の課題 ------ 20

§1. はじめに

数学ソフトウェア*の開発は、従来理学，工学の専門家により，単発的に開発が行なわれる傾向にあった。しかし近年、国の内外を問わず、詳細な数値解析理論とソフトウェア開発技術の研究が確立するにつれ、より組織的で体系的な開発が盛んになってきた。
又、これら数学ソフトウェアの利用者の底辺は拡大し、かつ多様化してきている。

それにともない、開発された数学ソフトウェアの図書館ともいえるライブラリに対する評価は、単に個々のソフトウェアが持つ性能（精度、処理速度、使用メモリ）上の評価にとどまらず、ライブラリ全体としての信頼性，使い易さ，保守性等の諸観点に基づく総合的な品質評価へと移行しつつある。

本稿では、まず“高品質ライブラリとは”について論じた後、富士通の科学技術計算ライブラリＳＳＬⅡの開発における品質保証の考え方と、いくつかの具体例について紹介する。

＊ 本稿での“数学ソフトウェア”とは、FORTRAN基本関数以外の、数値計算を行うソフトウェアをいう。

§2. 高品質ライブラリの条件と品質保証

総合的な品質評価に基づく、高品質ライブラリの条件として、下表の諸項目が考えられる.

項	条件	例	環境
a.	性能	• 精度が良い. • 処理速度が速い. • 使用メモリが少ない.	• 利用者の目標は、不特定である.
b.	信頼性	• テストシステムが確立している. • プログラム, マニュアルに誤りがない.	
	①頑健さ	• いかなるデータに対してもプログラムが正常に動作する.	• 処理するデータの性質は不特定である.
		• 計算機を変えてもプログラムが正常に動作する.	
	②ポータビリティ	• プログラムの持ち運びが容易であり、計算機を変えても正常に動作する.	• 計算機の種類が非常に多い.
c.	使い易さ	• 適切で充分な機能が用意されている. • 充分で、かつ理解し易い解説書が用意されている.	• 機能の種類が多い. • 利用者の層が不特定である.

		・仕様が統一されている.	
d.	保守性	・積極的な改良が行なわれる. ・改良にともなう利用者へのサポート体制が確立している.	・計算技術は、日進月歩である.
e.	その他	・リースプログラムが公開される. ・利用者と提供者とのコミュニケーションの手段が確立している.	

これらの諸条件のすべてを満す事が望ましいが、いくつかの条件は互いに相反する要因を含んでいる。
特に、“a. 性能”および、“b. 信頼性”の条件においては、以下のような対立要因を含んでいる。

(1) 精度と処理速度

精度を上げる為に、高次の近似式を利用したり、プログラムの一部を倍精度演算で行う等が考えられるが、これらは一般的に処理速度を低下させる要因である。同様に、処理速度を上げる為に、作業領域等の多くのメモリを利用する事により可能となる場合もある。

(2) 処理速度と頑健さ

異常なデータや、数値的に性質の悪いデータに対しても、プログラムを正常に動作させる為、特別な処理を付加し頑健さを高めるが、これは性質のよいデータに対しては、オーバーヘッドとなって処理速度を低下させる要因となる場合がある。

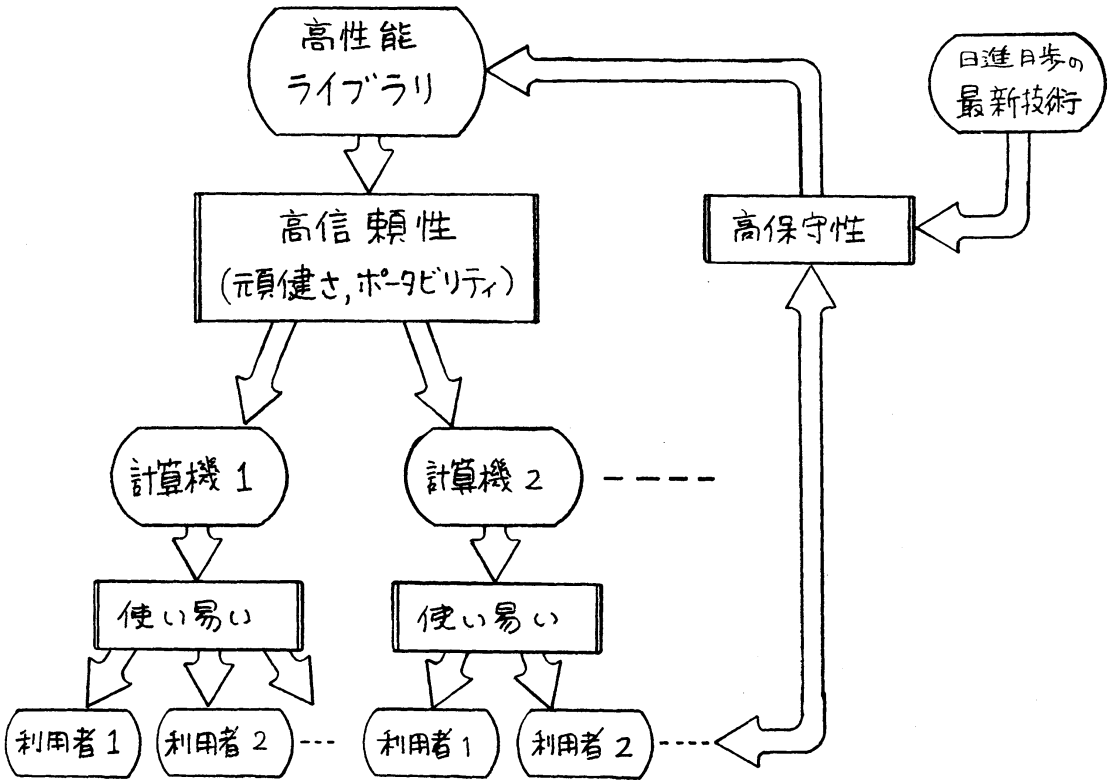
(3) 性能とポータビリティ

異なった計算機間でのプログラムの正常動作を保証し、プログラムの持ち運びを容易にする為、プログラム内の計算機に依存する部分（ex. 演算桁数，進数に関わる定数）を、プログラム内で自動的に決定する等の方法が考えられるが、これは、若干の精度の低下や処理速度の低下の要因となる。

このような対立要因に対しては、ライブラリ開発時にトレイドオフを設定し、各機能の目標の分化（精度優先，処理速度優先等の目標別機能の導入）や、ライブラリ全体としての目標の明確化を図る事が必要であろう。

一方、諸条件のうち、“b.信頼性”，“c.使い易さ”および、“保守性”については、高品質を維持するうえで、互いに助長する条件と考えられる。

即ち、ライブラリは“多くの利用者により、多くの計算機上で利用されることにより、より改善される”という発想の基に、下図に示すようなサイクルで各条件が連係している。



このような高品質ライブラリの条件をふまえた、数学ソフトウェアの品質保証とは、高度に信頼性を維持した上で、性能、使いやすさを適度にバランスさせることと考える。

§3. SSLⅡ (Scientific Subroutine Library Ⅱ) の開発

富士通の数学ソフトウェアライブラリSSLⅡの開発では、多種の計算機と多様な利用者層を前提として、限られた工数で高品質のライブラリを効率よく開発することを目差している。

従って、SSLⅡの品質保証は、最新技術の導入により高性能を維持することもさることながら、より信頼性の高い、使い易いライブラリの開発に主眼がおかれている。

本節では、信頼性と使い易さの観点から、品質保証の為の具体的な方法について紹介し、その効果、問題点について述べる。

§3.1 必要最少限の開発量 (信頼性)

大量のソフトウェアを開発する際の、人間による誤りを防止する為の素朴な発想として、情報量を最少限にする事が望ましい。

(1) 演算桁数に依存しないプログラムの開発

収束判定値や π の値など、演算桁数に依存する定数や手続きを、プログラム中で固定しない。

Fig 1.は、SSLⅡの単精度サブルーチンのプログラムの1例である。

```
C@ A 01  18 LAX
      SUBROUTINE LAX(A,.........,ICON)
        :
        :
      ONE=1.0
      PAI=4.0*ATAN(ONE)
        :
        :
C@ A 01  11 AMACH,DMACH,QMACH
      EPS=AMACH(EPS)
        :
        :
C@ A 01  13 ALU
      CALL  ALU(A,..........)
        :
        :
      IF(AM.LT.EPS) GOTO 100
C@ B 01    120
        :
        :
      CALL MGSSL(MCODE,ICON)
      RETURN
      END
```

Fig. 1

本例では、πの値をFORTRAN 関数ATAN を利用する事により求めている。又、収束判定値(変数EPS)は、演算桁数や進数等に依存する"丸め誤差の単位*"を与える関数AMACHを利用している。

ちなみに、関数AMACHと同様な関数は、SSLⅡ 全体で、5～10項目程度である。

---

* 丸め誤差の単位： $1.0+\varepsilon \fallingdotseq 1.0$ となる、最小の正の値εを丸め誤差の単位という。

(サブルーチンALUは、他のSSLⅡサブルーチンであり、MGSSLは、コンディションメッセージ出力ルーチンである。)

効果

(1) 倍精度、拡張精度の各プログラムの開発は、自動化される。

即ち、Fig.1のプログラムでは、手続き自体はすでに演算桁数に依存していないので、倍精度、拡張精度の各プログラムの実現は、変数の型宣言及び、サブルーチン名の変更により可能である。SSLⅡでは、変更手続きをユーティリティにより自動化している。Fig.2参照。

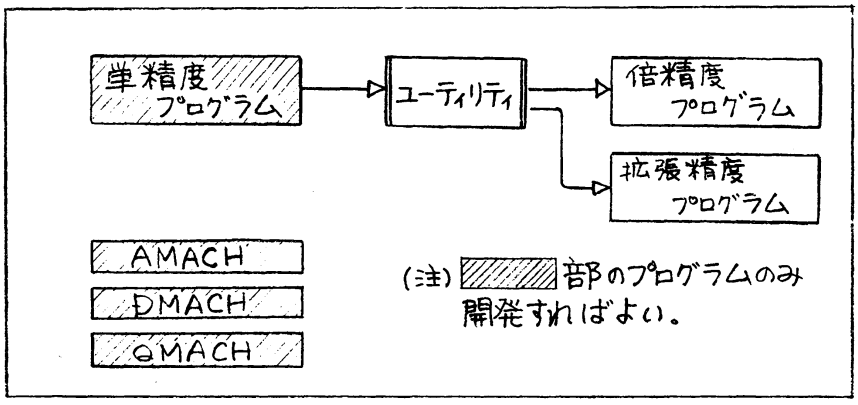


Fig.2

ユーティリティにより生成された、倍精度プログラムの1例を、Fig.3に示す。サブルーチン名の変更は、プログラム中の“C@␣A”で始まるステートメントで指示された内容に従って

行なわれる。

```
C@ A 01  18 LAX
      SUBROUTINE DLAX(A,.....,ICON)
        :
        :
      ONE=1.0
      PAI=4.0*ATAN(ONE)
        :
        :
C@ A 01  11 AMACH,DMACH,QMACH
      EPS=DMACH(EPS)
        :
        :
C@ A 01  13 ALU
      CALL  DALU(A,.....)
        :
        :
      IF(AM.LT.EPS) GOTO 100
C@ B 01    120
        :
        :
      CALL MGSSL(MCODE,ICON)
      RETURN
      END
```

(注) 倍精度プログラム名は、通常先頭が"D"で始まる。
_____部が変更されたプログラム名である。

Fig.3

(ロ) プログラムの持ち運びが容易である。

計算機を変えてプログラムを実行する場合、関数AMACHに相当する部分だけを変更すればよい。Fig.4参照。

(ハ) 改良・修正が集中的にできる。

最新技術に基づき、プログラムを改良したり、障害が発生し修正する場合に、単精度プログラムのみ実施すればよい。

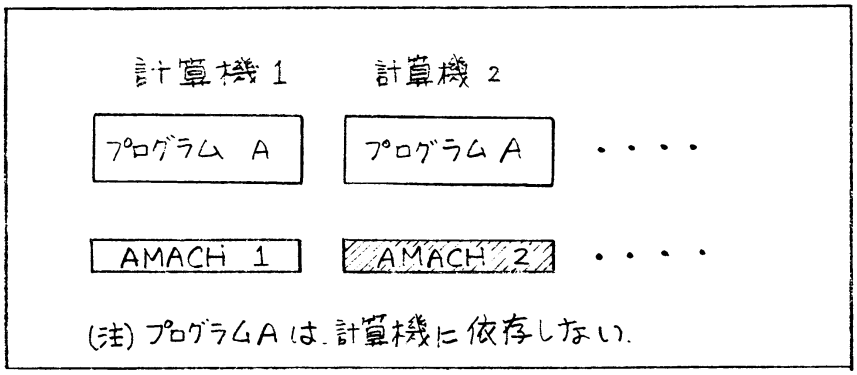


Fig. 4

問題点

(イ) 10の値などを固定せず、プログラム内で自動的に決定しているため、若干の精度低下や、処理速度の低下がみられる。

(2) 機能の細分化と標準パスの設定

線型方程式、固有値問題、フーリエ変換などのプログラムでは、機能の細分化を図りコンポーネントルーチンを用意すると同時に、それらを組合せた標準パス（標準ルーチン）を用意する。Fig. 5 参照。

効果

(イ) 細分化された機能が共用できる。

(ロ) 改良・修正が容易である。

最新技術に基づき、プログラムを改良したり、障害が発生し修正する場合は、該当のコンポー

ネントルーチンのみ実施すればよい。共用のコンポーネントルーチンである場合は、重複して実施する必要がない。

(ハ)利用者は、コンポーネントルーチンの機能も利用できるので、ライブラリとしての有用性が高まる。

問題点

(イ)作業領域や計算の手間が若干増加したり、プログラム呼出しのオーバヘッド等の為に、性能が低下する場合がある。

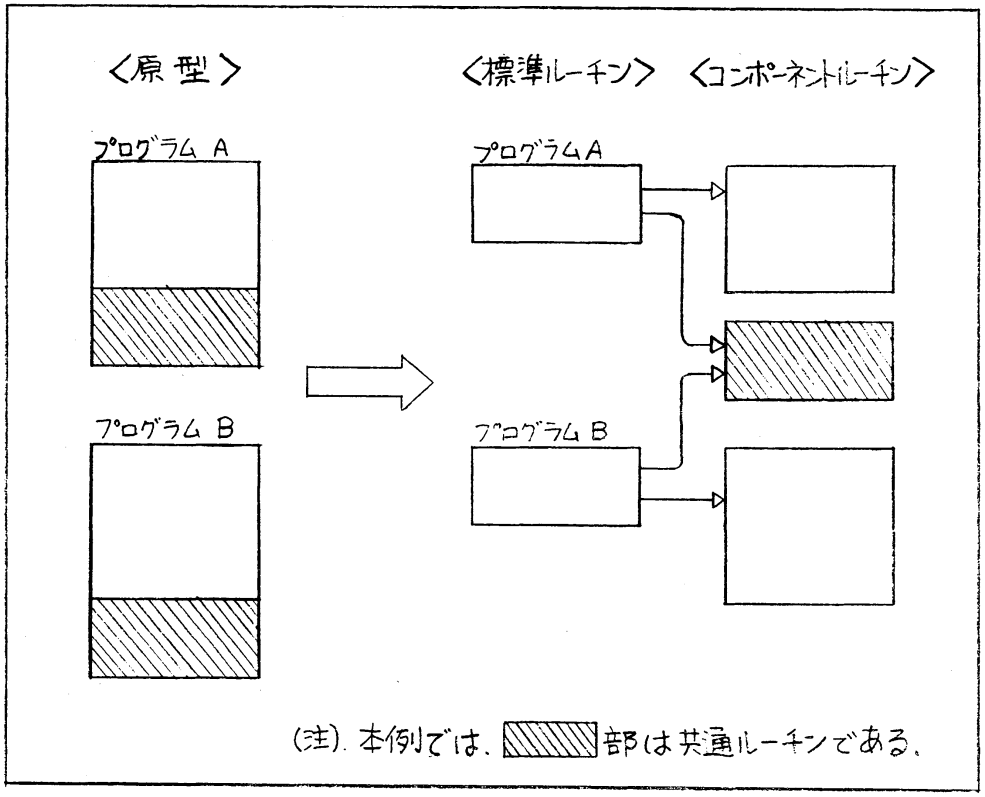


Fig. 5

§3.2 徹底したテスト（信頼性）

開発における、設計・作成・テストの各工程の内容を標準化し、プログラムの仕様，マニュアル，テスト結果等に対する徹底したレビューを実施する事が望ましい。

(1) 開発テスト内容の標準化

個々のプログラムに対するテスト内容を、下表のように標準化する。

記	呼称	テスト内容
P	パラメタチェック	制限外の異常な値（入力データ）を検出する機能の動作確認.
F	フローチェック	アルゴリズムが正しくコード化されているか、プログラム内のすべてのルートを実行し確認.
A	アルゴリズムチェック	速度，精度等の性能や、頑健さの検証.

(2) 開発テスト手順の標準化

標準化されたテスト内容に従って、Fig. 6 に示される手順で実施する。

まず、開発した単精度プログラムで、P, Fチェックを行ない、

後にFig.2で示したユーティリティにより倍精度プログラムを生成し、同様なP,F チェックを行なう。デバッグツールとして、ユーティリティが持つ補助機能（デバッグパケットの設定）や、FORTRANのデバッグ機能を使用する。

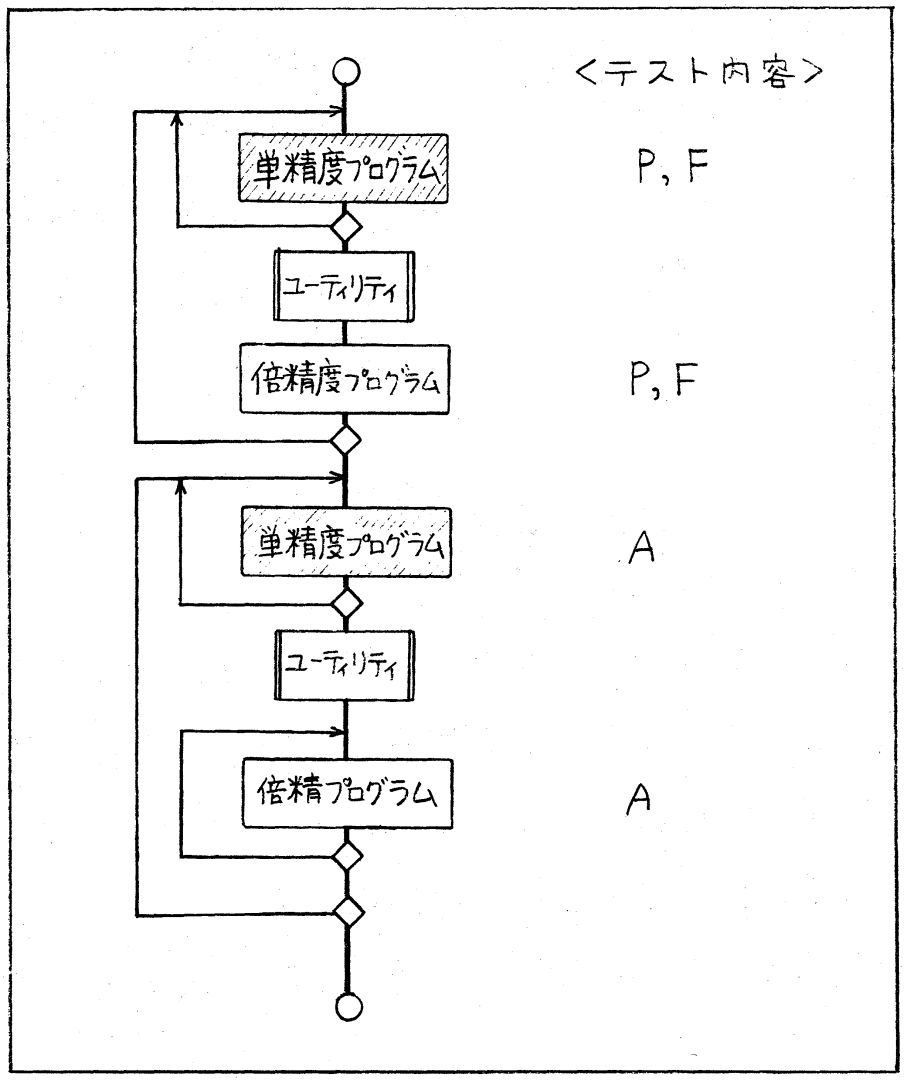


Fig.6

倍精度プログラムのP,Fチェックが正常に終了後は、再び単精度プログラムに戻り、Aチェックを行う。

検証の方法は、プログラムの機能により異なるが、

(イ)性質の異なる各種データによる実行

(ロ)異なったアルゴリズムによる実行

の結果に対する、数値的なふるまいの検討等による。

更に、Aチェックにおいては、演算精数に依存しないプログラムとしての安定性も検証する(即ち、後述の"開発テストシステム"自体も持ち運びが容易であり、Byte及びWord machine上での性能, 頑健さを検証する。)

テストツールとして、FORTUNE[*1], ECHO[*2] 等を必要に応じて利用する。

(3) 充実したテストシステム

● 開発テストシステム

多様なテストデータを保存したデータバンクを活用し、多様な評価に基づくテストを効率よく行う。Fig.7 参照。

● 移行テストシステム

開発テストシステムにより検証されたライブラリを、他の計算機へ移行する場合に、最終的な動作確認を行う。

Fig.7 参照。

---

*1 FORTUNE : FORTRAN TUNER, FORTRANプログラムの実行時の振舞いを解析するFORTRANのプレコンパイラ.

*2 ECHO : 数値的なセンシィビリティを検出する、FORTRANのプレコンパイラ。

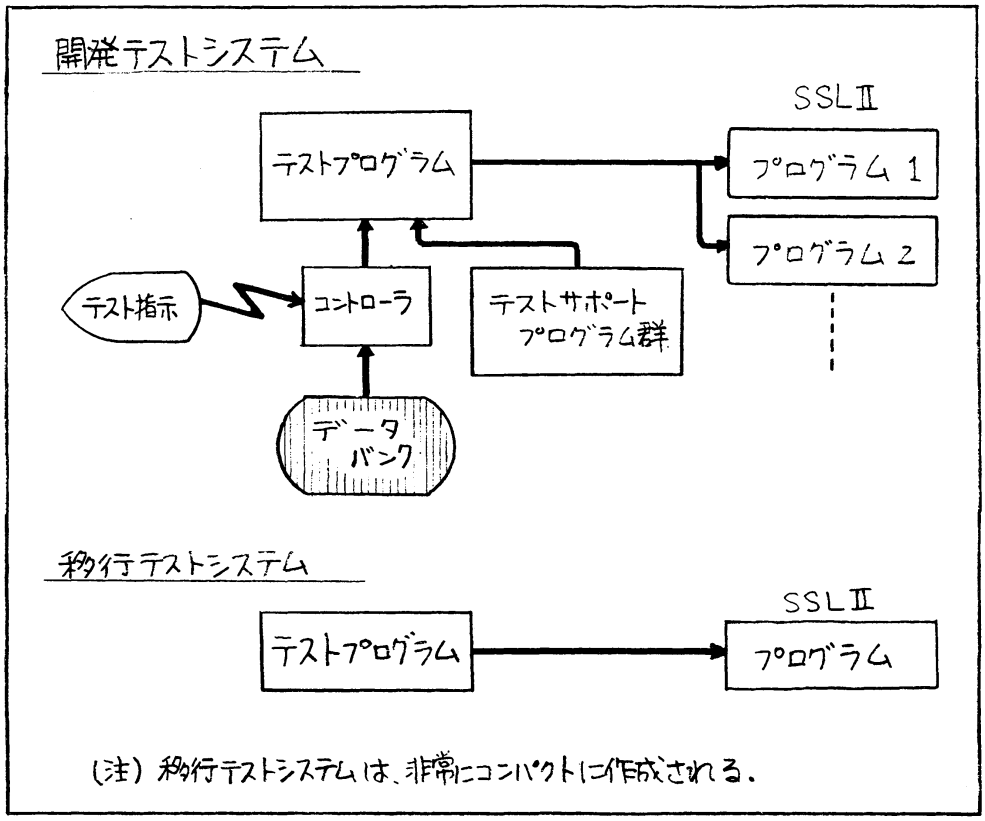


Fig. 7

以上で、ライブラリの品質保証における、"信頼性"の観点からの具体的な方法を述べた。
次に、"使い易さ"の観点からの具体的な方法を述べる。

§3.3 容易なサブルーチンの選択（使い易さ）

ライブラリに登録される機能は非常に多く、利用者の目的に叶った機能が、的確・迅速に利用できる事が望ましい。

(1) 単一解法の精選と問題別解法の導入

線型方程式や固有値問題など、一つの機能に対して解法がしぼれるものは、一貫して単一の解法を精選している。
例えば、"ガウス・ジョルダン, ガウス消去法"を"クラウト法"で一貫している。

一方、数値積分など関数を扱う機能では、関数の持つ性質に応じた問題別の解法を導入している。
例えば、数値積分では、"適応型ニュートンコーツ9点則, クレンショーカーチス型積分法, 二重指数関数型積分法"等被積分関数に応じた使い分けが出来るように複数の解法を導入している。

(2) マニュアルの充実

マニュアルには、利用者の目的に応じたサブルーチンの"使い分け"に関する解説を充実しており、選択が容易である。

§3.4 統一された仕様(使い易さ)

ライブラリに登録される機能は非常に多く、利用者の理解を助長し、使用誤りを防ぐ為には、個々のプログラムの仕様を一貫する事が望ましい。

(1) パラメタの統一

データの受け渡し方法は、唯一“パラメタ”とし（COMMONは使用しない）、その並びと意味づけ等の統一を図る。

(イ)パラメタ並び

一般則として、“入出力パラメタ”,“入力パラメタ”,“出力パラメタ”,“作業領域パラメタ”,“コンディションコードパラメタ”の順とする。

(ロ)パラメタの意味づけ

同一の内容をもつパラメタは、同一の記号を用いる。

Fig.8 に、連立1次方程式に関するサブルーチンのパラメタ並びの例を示す。

```
  LAX(A,K,N,B,EPSZ,ISW,IS,VW,IP,ICON)
  ALU(A,K,N,EPSZ,IP,IS,VW,ICON)
  LUX(B,FA,K,N,ISW,IP,ICON)
LAXR(X,A,K,N,FA,B,IP,VW,ICON)
```

Fig.8

(ハ)パラメタの型

FORTRANの暗黙の型に従う記号を用いる。

即ち、例えば“A”は実数型であり、“K”は整数型である。

特例として、先頭が“Z”で始まる記号は、複素数型とする。

(2) コンディションコードの統一

すべてのサブルーチンには、実行後の状態を示すパラメタ"ICON"を用意する。その値はコード化し、"結果が保証されるか否か"に応じて、大きく3段階にレベル分け(正常, 警告, 異常)する。Fig.9 参照。
これにより、利用者がICON の値に基づき継続処理を制御する事を容易にしている。

```
SUBROUTINE LAX(.....,ICON)

   ICON={    0～ 9999    正常
         |10000～19999   警告
         |20000～29999   異常
         |30000   異常(パラメタエラー)
```

Fig.9

又、SSLⅡでは利用者の指示により、コンディションメッセージの出力をする事も可能であり、利用者プログラムのデバッグ等に利用する事ができる。Fig.10 の流れにより、出力が制御される。

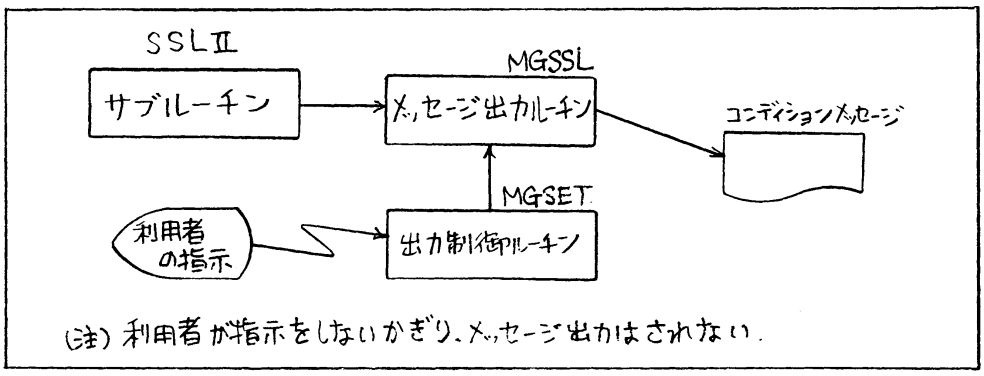


Fig.10

§4. 数学ソフトウェアライブラリ開発の今後の課題

数学ソフトウェアライブラリSSLⅡの開発の現場から、品質保証のあり方と幾つかの具体例について述べた。

最後に、具体的には言及しなかった幾つかの問題にふれたい。

まず、数学ソフトウェアの品質を保証する第一歩は、米国のIMSLや英国のNAGの様に、それぞれの分野の専門家が一致協力して、計算理論の最先端の研究成果をフォローするという事であると思う。数学ソフトウェアライブラリは、高度に学術的な面と、計算道具としての実際的な面を有し、メーカーだけで開発するのは、計算理論の最先端の研究成果をフォロー出来ない恐れがあるし、大学や研究機関の専門家だけでは、実際的な面の為に、流通が困難であるように思われる。数学ソフトウェアは、利用されて初めてその存在の意味を有するものであり、早晩、誰かが何処かで、それぞれの専門家の協力を得て、数学ソフトウェアの集大成を計る組織を確立せねばならないだろうと考える。
さもなければ、数学ソフトウェアを使う全分野の研究が外国に遅れを取りかねない。

我々は、この様な観点に立って数学ソフトウェアの品質保証を積極的に進めようと考えている。

次に、§2で示した高品質ライブラリの1条件である利用者とのコミュニケーションの問題である。
ライブラリは、開発が終れば即終結するものではなく、利用者の要望と最新技術の動向に沿って、逐次改善されていくべきものであると考えている。即ち、ソフトウェアとしてのライブラリという見方だけでなく、むしろライブラリを通じて利用者・開発者・提供者が協力し、円滑にコミュニケートしていく中で、ライブラリがより改善されていく関係を組織する事が重要な問題であると考えている。

さらに、数学ソフトウェアの標準化が問題である。
例えば、技術的・理論的に安定した分野(例えば、線型方程式)に関する呼出し型式は標準化し、FORTRAN 基本関数と同様な発想で利用出来るようにすべきではなかろうか。

最後に、数学ソフトウェアが果す教育的役割についてふれたい。我々が受けている感触では、利用者の中にはライブラリを現実的な教材として利用されている人があるように思う。

これは、直接品質保証には関係しないが、重大な意味を持っている。強力な流布の手段は有するが学術的な面にとぼしいメーカーだけで、ライブラリを開発・提供する事は、教育的役割を果す上で責任が重過ぎるように思う。

本稿が今後の数学ソフトウェアライブラリ開発の参考になれば幸いである。