Proceedings of the Risa Consortium
1997 pp. 69– pp. 82

数式処理を用いた数値計算アルゴリズム選択

保田 貴史
愛媛大学工学部情報工学科

和田 武
愛媛大学総合情報処理センター

甲斐 博*
愛媛大学工学部情報工学科

野田 松太郎†
愛媛大学工学部情報工学科

(RECEIVED 1997/6/15 REVISED 1997/9/9)

Abstract.

コンピュータの利用分野の拡大するとともに、問題別に膨大な数の計算アルゴリズムが開発され使用されている。しかし、浮動小数計算で数値的にある種の近似解を得ようとする数値計算においては、適切にアルゴリズムを選択しなければ必ずしも正しい結果を求められないことも良く知られている。特に問題が多様化し、解きにくい問題や悪条件問題が出現する場合には不適切なアルゴリズムを選択することによって深刻な問題を引き起こす可能性がある。そこで本論では与えた問題の性質を自動的に判定し、適切な数値計算のアルゴリズムを選択し、かつその計算実行までもを行い得るような利用者支援システムの開発を行う。具体的には解きにくい問題の例として硬い常微分方程式の数値解法を考え、硬さの度合い（スティッフ率）を数式処理システムを利用して計算し、データベースに格納されたアルゴリズム中から適切なものを選択・実行するようにした。ユーザインターフェースを簡便にするため、Netscape Navigator などの Web ブラウザでのシステムの利用を可能とした。実際のシステム動作を広く使用されている数値計算アルゴリズム SSL–II を対象として示した。

1. はじめに

計算の自動化が進むみ、取り扱う問題が多岐にわたるにつれ従来は見逃されていた問題の重要性が増すことがある。その典型的な問題が数値計算のアルゴリズム選択問題である。数値計算では常にある種の近似解を求めるため、浮動小数計算による誤差が重要になったり、

*kai@cs.ehime-u.ac.jp
†noda@cs.ehime-u.ac.jp

Problem	Numeric Integral			
function	G32	DE	Romb	NC
f_a	-0.01622972	-0.02763097	-0.68482819	5.40966656
f_b	0.19994034	24.6736125	2.98225113	3.13759258
f_c	- 580.39410	-24965.007	-5759.10716	230.75544

Table 1. 悪条件関数の数値積分

計算精度との関連でおかしな結果を生んだりすることは一般に知られている。例えば係数がわずかに変化するだけで、結果が大きく変わるような問題においては計算を精密に行う必要がある。このような問題を悪条件問題と呼んでいるが、現実の問題では意識しないうちに悪条件問題を解く必要に迫られている場合も多い。一般には、悪条件問題にはそれに適した数値計算用のアルゴリズムが開発されていて数値計算用のライブラリに登録されているが、問題を悪条件と意識しない場合には、ライブラリ中から不適切な数値計算アルゴリズムを使用する危険がある。わかりやすい例として以下の関数の定積分を考えてみる。

$$f_a = \int_0^1 \frac{dx}{1000x(x-1)-0.001},$$
$$f_b = \int_0^1 \frac{dx}{1000(x-0.5)^2+0.001},$$
$$f_c = \int_0^1 \frac{dx}{x^5-x^4-0.75x^3+x^2-0.25x-10^{-6}}$$

関数 f_a は定積分の範囲のすぐ外に特異点を持ち、範囲内の両端で関数は鋭く変化している。また f_b は逆に定積分範囲の中央部分に鋭いピークを持つ。f_c は f_a, f_b 双方の性質を合わせ持つもので、これらに対して安定した結果を数値積分公式で得ようとすると、積分アルゴリズムを慎重に選択しなければならない。実際に、よく使われる 4 種の数値計算アルゴリズムを適用してみる。それらは、32 点ガウス積分公式 (G32) 、二重指数公式 (DE)、ロンバーグ公式 (Romb)、適応型ニュートンコーツ公式 (NC) である。結果を Table 1 に示す。どの方法を用いるべきかを判断することは、非常に困難であることが分かる。関数の振る舞いが事前にわかり、かつ積分公式について詳しい知識を持ったユーザなら、f_a に対しては二重指数公式を、f_b に対してはニュートンコーツ公式を使用し、f_c に対しては積分範囲を分割するであろうが、すべてのユーザがこのような知識を有しているわけではない。ここに、アルゴリズム選択の重要性がある。

この種の研究としては、数値計算アルゴリズムを木構造のデータベース化した GAMS [1] が知られていたが、ここではアルゴリズム名を羅列したにとどまっていた。近年、数値計算ラ

イブラリ中の多数のアルゴリズムの利用の便をはかる目的で、特に数値計算アルゴリズムが持つ引数の数や型に注意を払うことなく検索・実行するためにユーザインターフェースを改良する試みが提案されている。数値計算ライブラリとしてNAGを考え、引数に対する配慮を数式処理システムAXIOMでの入力のようにして利用の便の向上を図ったエキスパートシステムANNAの提案[4]や、楕円型偏微分方程式に対しての引数選択を容易にしたPYTHIA[3]が考えられている。特徴として、旧来からのアルゴリズム選択システムと異なり、いずれもが知識ベースの処理に何らかの意味で数式処理システムを使用している点がある。しかし、数式処理システムの利用は単に数式の入出力部分にとどまっており、上で議論した数式の特徴を抽出して最適のアルゴリズムを選択するような使用は行われていない。後者はELLPACK[2]の一つの発展的形態と見ることが出来る。また、数値計算で解きにくい問題の代表例の一つである「硬い(stiff)」常微分方程式に対してはパソコンで稼働するエキスパートシステムであるPLOD[5]等が開発されているが、ここではいくつかの数値アルゴリズムに対して関数評価の回数を知らせる程度の働きしかしていない。

そこで本論文では、数値計算ライブラリ中のアルゴリズムを用いて数値計算する場合に、入力した数式の悪条件性の判定等に数式処理を利用することによって、複雑な問題にも安定な数値結果を得ることが可能なシステムの構築を考える。ここでは、数値計算ライブラリとして富士通で開発され、大型計算機を中心として広く利用されているSSL–IIを、数式処理システムとしてMapleを、またデータベースとしてオブジェクト・リレーショナル型のIllustraを用いた。ユーザインターフェースの向上を図るため、上の諸システムをNetscape Navigator等のWebブラウザから自由に操作できるようにした。以下では、主に硬い常微分方程式に関してシステムの構築の意味と結果について述べる。

2. アルゴリズム選択と硬い常微分方程式

アルゴリズム選択の例として、一階連立常微分方程式の数値解を得る場合を考える。このための、アルゴリズムは多様だが、解に対する要求精度や常微分方程式が「硬い」か否かで、対応するアルゴリズムを慎重に選択する必要がある。硬い(stiff:スティッフな)連立常微分方程式については数値解析の教科書等で詳しく述べられているが、以下本論の一貫性のために、簡単に例を挙げながらその性質をみる。次の連立常微分方程式を考える。

$$\begin{cases} x' = -2\mu x + \mu y, & x(0) = 0 \\ y' = x - y, & y(0) = 2 \end{cases} \tag{1}$$

$$\mu = 10^6$$

ベクトル記法を用いれば

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -2\mu & \mu \\ 1 & -1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

となり、行列 A の固有値は、

$$\begin{cases} \lambda_1 = -\frac{1}{2} - \mu + \frac{1}{2}\sqrt{4\mu^2+1} \\ \lambda_2 = -\frac{1}{2} - \mu - \frac{1}{2}\sqrt{4\mu^2+1} \end{cases}$$

となる。よって、(1) の解は $s = \sqrt{4\mu^2+1}$とすると

$$\begin{cases} x(t) = \frac{4\mu}{s}\{e^{\lambda_2 t} - e^{\lambda_1 t}\} \\ y(t) = \frac{1}{s}\{(s+2\mu-1)e^{\lambda_1 t} + (s-2\mu+1)e^{\lambda_2 t}\} \end{cases}$$

となる。ここで$\lambda_1 \simeq -1/2, \lambda_2 \simeq -2\mu, s \simeq 2\mu$とすると、上述の式は近似的に

$$\begin{cases} x(t) \simeq e^{-(1/2)t} - e^{-2\mu t} \\ y(t) \simeq 2e^{-(1/2)t} \end{cases}$$

と表される。μの値が 10^6と極端に大きいため、$x(t)$ の第 2 項は、$t = 0$ のごく近傍でのみ意味を持つにすぎず、t の値が 0 から少しでも離れれば次のようにみなし得る。

$$x(t) \simeq \frac{1}{2}y(t) = e^{-(1/2)t}.$$

この方程式を良く知られたオイラー法によって数値的に解く。数値解は

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1-2\mu h & \mu h \\ h & 1-h \end{bmatrix}\begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{3}$$

を満たす。もとの微分方程式の真の解は、$t \to \infty$ で $x(t) \to 0, y(t) \to 0$ となるから、数値解も当然、$n \to \infty$ で 0 に収束する。そのためには (3) の右辺の行列

$$B = \begin{bmatrix} 1-2\mu h & \mu h \\ h & 1-h \end{bmatrix}$$

の固有値の絶対値がいずれも 1 以下でなければならない。この条件から刻み幅 h に関し

$$h < \frac{4}{2\mu+1} \simeq 2 \times 10^{-6}$$

という制約がつく。$t = 1$ までの数値解を得ようとすると、10^6 ステップもの計算が必要になること。この状況はオイラー法より少し複雑なルンゲ-クッタ法やアダムス法(Adams-Bashforth

t	x (RKG)	x (ODGE)	x(真の解)
0.00E+00	0.00000000E+00	0.00000000E+00	0.0000000000000
0.10E-05	-0.55133384E+04	0.99999511E+00	0.9999949979513
0.30E-04	-0.16767962E+12	0.99998522E+00	0.9999850001124
0.50E-04	-0.50987906E+19	0.99997509E+00	0.9999750003124
0.80E-04	-0.85496337E+30	0.99996012E+00	0.9999600007999
0.90E-04	overflow	0.99995500E+00	0.9999550010124
0.10E-03	overflow	0.99994993E+00	0.9999500012499

Table 2. 硬い常微分方程式の数値解と真の解の比較

法など) を適用しても同様である。これは、行列 A の固有値の大きさが極端に異なることに起因している。このような微分方程式を硬い微分方程式と呼び、電気回路網、化学反応、原子炉の問題、制御系、偏微分方程式に対する「 線の方法」などによくあらわれる。

また、必ずしも線形の問題とは限らず、一般には連立常微分方程式

$$y' = f(x, y)$$

の右辺のヤコビ行列

$$J = \left(\frac{\partial f(x,y)}{\partial y} \right)$$

の固有値を$\lambda_i, i = 1, \ldots, N$ とすると

$$\begin{aligned} &\mathrm{Re}(\lambda_i(x)) < 0, \quad i = 1, 2, \ldots, N, \quad x \in I \\ &\frac{\max(|\ \mathrm{Re}(\lambda_i(x))\ |)}{\min(|\ \mathrm{Re}(\lambda_i(x))\ |)} \gg 1, \quad x \in I \end{aligned} \tag{4}$$

を満たせばその方程式はスティッフであるといい、(4) の左辺をスティッフ率 (stiff ratio) という。

(1) の連立常微分方程式を、ルンゲ-クッタ-ギル法 (RKG) 及び硬い連立常微分方程式に適したギア法 (ODGE) で解いた結果を Table 2 に示す。overflow は数値結果が浮動小数の表現範囲を越えたために演算が不可能になったことを意味する。なお、計算には富士通が開発した数値計算ライブラリ、SSL-II に含まれるアルゴリズムを用いた。

このように硬い常微分方程式を解く際にはアルゴリズム選択が極めて重要になる。しかし、常微分方程式がスティッフであるか否かは一般に計算前にはわからないので、誤って不安定な結果を生じるアルゴリズムによる結果を盲信する危険がある。スティッフ率の大きさを事前に求めれれば、正しいアルゴリズムを選択できることは明らかである。しかし、数値計算

だけでこれを行おうとすると、微分方程式の各項の係数などの変化とともに毎回判定を行う必要があり、計算時間・費用の大幅な損失になる。以下では、この部分に数式処理の機能を採用し、アルゴリズム選択を自動的に行えるシステムを設計することを考える。

3. アルゴリズム選択システム

3.1. アルゴリズム選択システムの形態

すでに上でも述べたが、アルゴリズム選択システムには基本的に次の方法がある。

1. ユーザ支援システムとして、アルゴリズム名を出力し、利用はユーザの知識に委ねる。場合によってはアルゴリズムの引数や性質等の解説を付ける。
2. 数値計算中のアルゴリズムの引数の数や型等をユーザが意識しなくても良いようにする。アルゴリズム名だけでなく、引数情報などをデータベースに格納し、これを知識ベース的に活用するある種のエキスパートシステム。
3. エキスパートシステムをさらに進め、アルゴリズムの特性を知り、入力した問題に最適なアルゴリズムを自動的に選択する。

第1の者は各種の大規模な数値計算ライブラリをユーザに開放している計算センターで開発されていたが、一般にはシステム依存性が強かった。汎用化を目指しネットワークを経由して利用可能にしたこの種のシステムとしては GAMS [1] が著名である。また、第2のものとしては楕円型偏微分方程式の数値解法に対して開発された ELLPAC [2] や、その発展形である PYTHIA [3] や、NAG ライブラリとの結合をはかった ANNA [4] 等がある。本研究での意図は、後者をさらに発展させ悪条件問題等の数値的に解きにくい問題に対して、単にアルゴリズム名を羅列するのみでなく、最適な数値解法を自動的に選択し、その実行までもを行うシステムの開発にある。さらに、初心者でも利用可能なように簡単な入力でアルゴリズム選択が可能なように心がけた。このためユーザーインターフェースの向上を目指して Netscape Navigator などの Web ブラウザを用いてのシステムの利用を可能とした。これを前節の硬い常微分方程式に関して述べる。硬い微分方程式を安定に解くためには、以下の手続きが必要になる。

1. 微分方程式の入力（Web ブラウザを用いる）
2. スティッフ率の計算（ヤコビ行列の計算等を記号的に計算する）
3. スティッフ率がある値より大なら、硬い常微分方程式に適するギア法を選択
4. スティッフ率が大きくない場合は係数行列の形状が特殊な場合は対応する解法を選択

5. 以上以外の場合は要求精度によって、ルンゲクッタ法その他を選択
6. 初期値等のパラメータをメニュー的に入力し計算実行
7. 結果の出力

ここで、数値計算ライブラリの情報はオブジェクトリレーショナル型データベース Illustra に格納され、記号的にスティッフ率を求める部分は数式処理システム Maple によっている。

3.2. 数式処理システムの利用

記号的にスティッフ率を求める必要性は、今取り扱っている問題では

- Web ブラウザによって入力された常微分方程式の係数部分の取り出しの数式操作
- ヤコビ行列を求めるための数式微分

にある。係数部分の取り出しやヤコビ行列の計算を行う場合は、各々の操作を数式処理システム Maple で行うため、Maple で読み込み可能な形式のファイルを作成したり、Maple からの出力を数値計算に引き渡すためのファイルを用いる必要がある。特に、アルゴリズム選択後に入力常微分方程式の実行を行うためには、入力数式や初期値その他の情報を持つファイルを作成保持する必要がある。SSL-II は FORTRAN プログラムの集合であるので、このファイルは FORTRAN 形式で書かれていて、選択した最適アルゴリズムの正しい位置に組み込まれなければならない。したがって、実行ファイルには、アルゴリズム選択システムで選択した最適アルゴリズムと入力数式及び初期値等の関連するデータが含まれることになる。

4. システムの構成

Fig. 1 に作成したシステムの構成を簡単に示した。点線の左側はユーザののマシン（クライアント）、右側はサーバマシン（ホスト）のシステムである。ユーザは Netscape Navigator 等の Web ブラウザを起動しサーバマシン上の Illustra データベースにアクセスしアルゴリズム検索を行う。データベースには、入力用のメニュー作成を行う HTML 形式の言語と Table 3 の形のアルゴリズムデータ (分類、名称、特徴他) が格納されている。

5. システムの動作

アルゴリズム選択システムの具体的な動作例を示す。システムの動作環境は Windows95 上の Web サーバ、WindowsNT4.0 上のデータベースシステム Illustra を中核とし、UNIX 環境下の数式処理システム Maple と結合する。これらの環境や各種システムを変化させても、本論で構築しているアルゴリズム選択システムは大きな変更なく動作可能である。

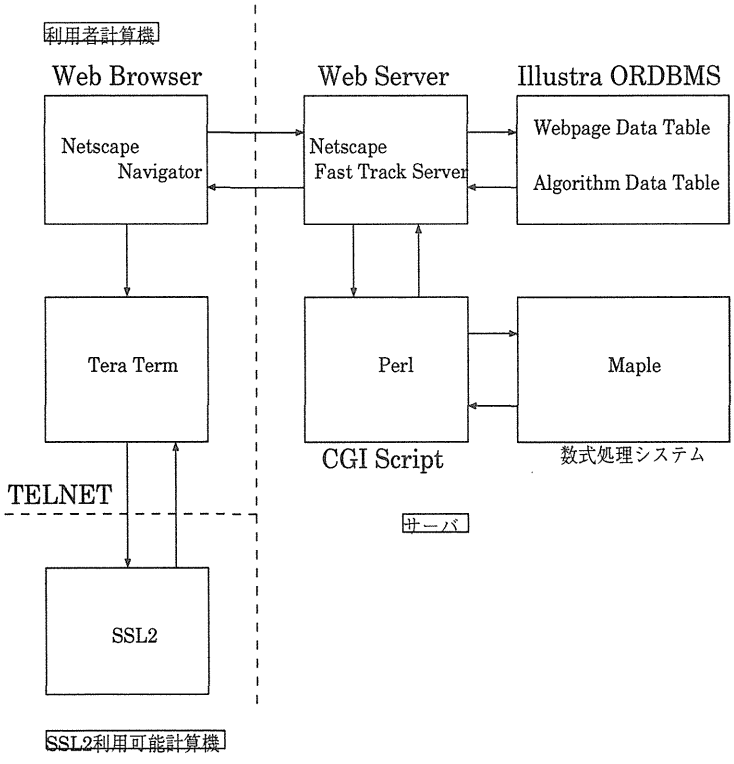

Fig. 1. システム構成

top	class	method	algorithm
⋮	⋮	⋮	⋮
線型計算	連立一次方程式	実バンド行列の連立一次方程式 (ガウス消去法)	LBX1
線型計算	連立一次方程式	正値対称 3 項行列の連立一次方程式 (変形コレスキー法)	LSTX
線型計算	連立一次方程式	実 3 項行列の連立一次方程式 (ガウス消去法)	LTX
⋮	⋮	⋮	⋮
微分方程式	常微分方程式	連立 1 階常微分方程式 (ルンゲ・クッタ・ギル法)	RKG
微分方程式	常微分方程式	連立 1 階常微分方程式 (ハミング法)	HAMNG
微分方程式	常微分方程式	連立 1 階常微分方程式 (ルンゲ・クッタ・バーナー法)	ODRK1
微分方程式	常微分方程式	連立 1 階常微分方程式 (アダムス法)	ODAM
微分方程式	常微分方程式	スティフ連立 1 階常微分方程式 (ギア法)	ODGE
⋮	⋮	⋮	⋮

Table 3. SSL–II のアルゴリズムデータ (線形計算と微分方程式の一部)

5.1. 木構造を用いたパッケージ別、計算分野別の検索

パッケージ別に数学の問題に対して木構造化された分類を定義しデータベースに格納する。検索可能なパッケージとしてここではSSL–IIを考え、これをFig.2に示す。次に、パッケージ選択後、Fig.3のように、最も広い問題の種類（大分類）を与える。このようにして、目的とするアルゴリズムの種類に至るまで順次検索を行っていく。

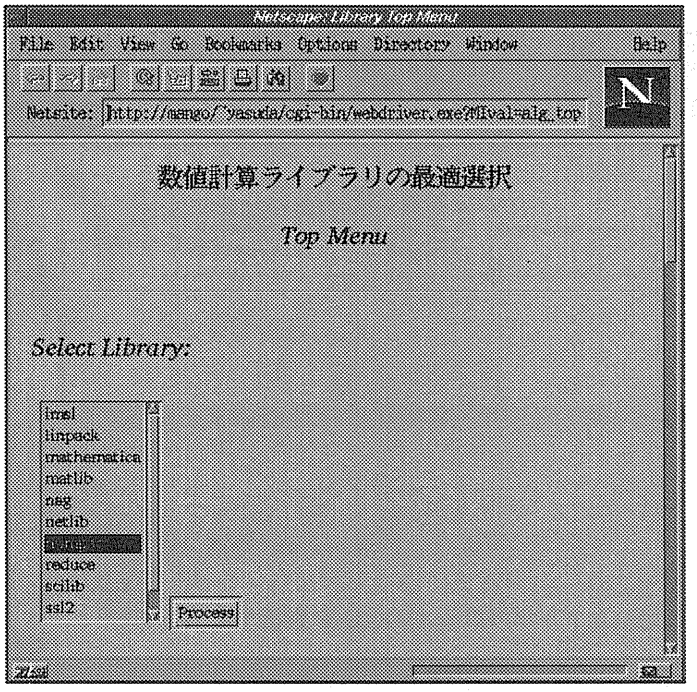

Fig. 2. 第1メニュー

5.2. 連立一階常微分方程式のスティッフ率の演算

常微分方程式の数値解を安定に求めるための最適アルゴリズムの選択に重要なスティッフ率の演算を行う。2. 節で述べた硬い常微分方程式の場合を例としてシステムの動作を示す。

連立一階常微分方程式の入力

木構造的な検索によりSSL–IIの常微分方程式の入力画面を呼び入力を行う(Fig.4)。

スティッフ率を求める

入力された問題をMapleで使える形に変換し、そのヤコビ行列を求め、スティッフ率を求める。例として入力した常微分方程式のスティフ率は4000002と非常に大きいため、スティッ

Fig. 3. 第 2 メニュー

フな常微分方程式に適したアルゴリズム ODGE を選択する (Fig.5)。

5.3. 連立一階常微分方程式の数値解の計算

実際に SSL–II を用いて以下の順に一階連立常微分方程式を数値的に計算する。

- 連立一階常微分方程式の入力
  木構造的な検索により SSL–II の常微分方程式入力画面を呼び入力を行う (Fig.4)。

- 選択されたアルゴリズムを呼び出し、実行可能なプログラムの生成
  入力常微分方程式を Fortran で扱える形式に変換し、選択されたアルゴリズムに方程式や初期値等の入力データを組み込み、実行可能な Fortran プログラムを自動的に作成する。

- プログラムの転送及び実行
  Fortran プログラム及び Telnet 端末用プログラム (ここでは、TeraTerm マクロプログラム) をユーザシステムに転送する。ユーザシステムは自動的に Telnet Window を開

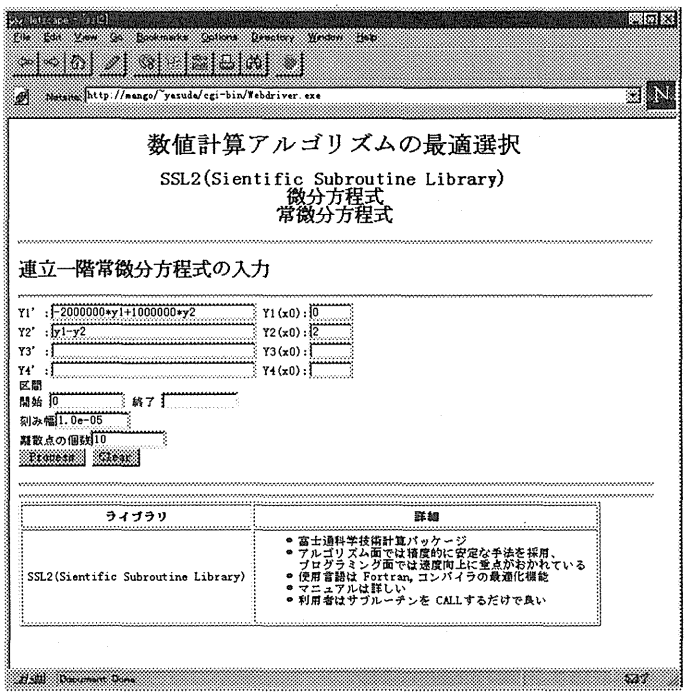

Fig. 4. 常微分方程式の入力

き、SSL-II 利用可能マシンへのログイン、プログラム転送、コンパイル、実行を行う (Fig.7)。

6. 結論

本論では数式処理の機能を利用した数値計算ライブラリ中のアルゴリズム選択を行い、かつその実行までもを行うシステムを作成した。システムの特性は次のようにまとめることが出来る。

1. 数式として入力された常微分方程式の特徴抽出に数式処理システム Maple を活用し、アルゴリズム選択を行う。
2. ユーザインターフェースの向上を目指し、Netscape Navigator などの Web ブラウザでのシステムの利用を可能とした。このことにより、ネットワークを通じて、ユーザの場所、計算機の種類にとらわれない利用を可能とした。
3. 膨大なアルゴリズムの情報や検索システムを格納するため、オブジェクトリレーショナルデータベースシステム Illustra を活用した。

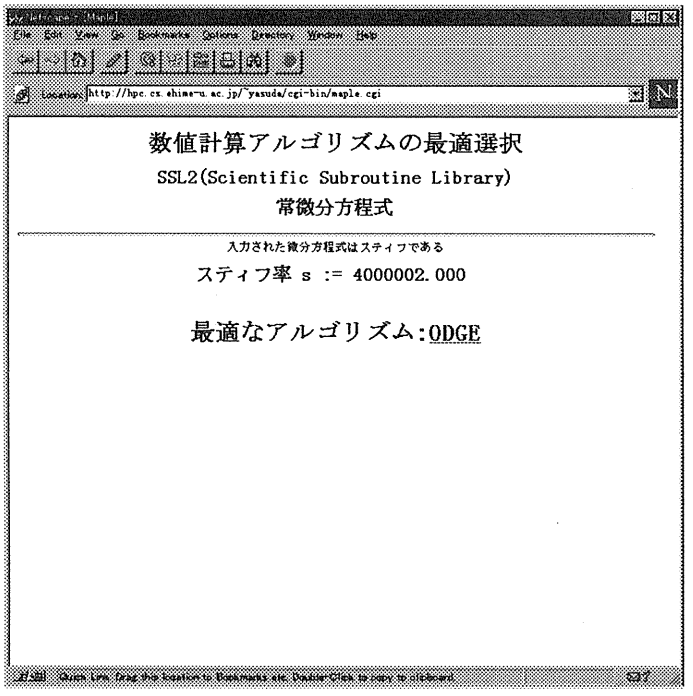

Fig. 5. アルゴリズム表示

4. アルゴリズム名の選択のみでなく、実際の数値計算を行い結果を得ることを可能とした。

本論では主に硬い常微分方程式のアルゴリズム選択を述べ、スティッフ率を数式処理で求めることによる入力数式の特徴抽出を考えた。同様のことは定積分において関数の振る舞いの激しさを抽出して利用する積分公式を選択する等の問題への拡張が容易である。また、本論では数式処理システムとして既存の商用システムである Maple を、数値計算ライブラリとして SSL–II を、データベースシステムとして Illustra を用いたが、これらを他のシステムに置き換えることは困難ではない。

今後に残された課題として、以下の事項を早急に実現する必要がある。

1. 最近の関連する論文 [3, 4] に見られるようなアルゴリズム選択システムの知識ベース化により、引数の数や型を意識せずに数値計算ライブラリを使用できるシステムの構成。[1)]

2. 数値計算ライブラリだけでなく、同様のシステムを統計計算ライブラリの使用に拡張し、

[1)] この事項に関しては、東京大学大型計算機センター村尾裕一博士に最初にご指摘いただいた。

Fig. 6. Fortranプログラムの表示

多くのパッケージ間の引数や命令の差異を意識せずに計算可能な知的システムの作成。

なお、本研究遂行に当たっては、株式会社富士通研究所からの奨学寄付金によったことを付記し、感謝する。

参 考 文 献

[1] R.F. Boisvert and J.L. Springman, Internal Structure of the Guide to Available Mathematical Software, NISTIR 89–4042, Gaitersburg, MD 20899, 1993.

[2] J.R. Rice and R.F. Boisvert, Solving Elliptic Problems Using ELLPACK, Sprinber–Verlag, 1985.

[3] S. Weerawarana, E.N. Houstis, J.H. Rice, A. Joshi and C.E. Houstis, PYTHIA : A Knowledge–Based System to Select Scientific Algorithms, ACM Tran. Math. Soft., Vol.22, pp.447–468, 1996.

[4] B.J. Dupée and J.H. Davenport, An Intelligent Interface to Numerical Routines, In *Design and Implementation of Symbolic Computation Systems* Eds. J. Calmet and C. Limongelli, Lecture Notes in Computer Science 1128, pp.252–262, Springer, 1996.

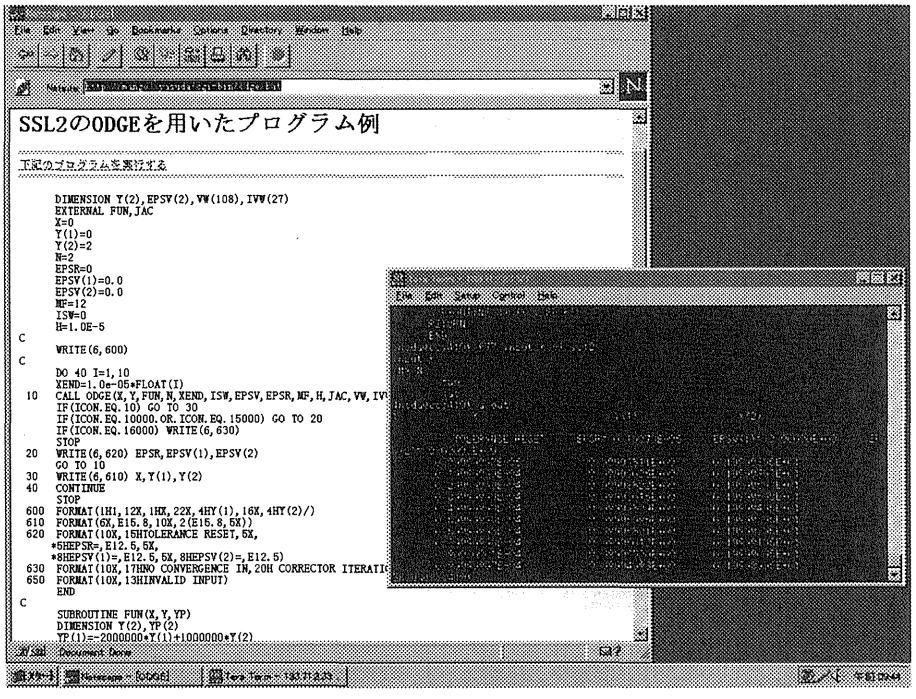

Fig. 7. SSL–II の実行

[5] D. Barnett and D. Kahaner, Experience with an Expert System for ODE, in *Intelligent Mathematical Software System* Eds. E.N. Houstis et al., Elsevier Science Pub., pp.5–13, 1990.