Ｊ言語　と　固有値問題　　（その４）

再帰法の解説　及び　チュートリアル な　レビュー

中野嘉弘　（８４才・札幌市）

FAX 専　011-588-3354　　yoshihiro@river.ocn.ne.jp

前稿「Ｊ 言語 と 固有値問題（その３）」（再帰法・英文）への更なる解説と共に加えて固有値問題へのチュートリアル なエビユーを提供したい。
「ダニレフスキー法とは何か」の話題の決着は付けられるか？

は　し　が　き

「Ｊ 言語 と 固有値問題」の前２稿（文献１、２）で用いられた「直接法」の愚直なプログラムは、長過ぎる難があった。　その後、再帰プラグラム化して、圧倒的に短縮出来たので、その英文速報を第３報とした。
これは、不慣れな英文の為、思うような説明が不足した。
その補足説明をすると共に、一般的に「Ｊと固有値問題」について、レビューしたい。
チュートリアルにも意味があれば幸せである。

ミスプリント訂正：第２報「Ｊ言語と固有値問題（その２）」p.3 下から 6行目付近の数字列に脱落あり、正しくは、この下行の全角文字で表示部分の２数値を追加。
12353145 _13176688 6117748 _1613472 ２６４１１０ __２７４４０ 1764 _64 1

１．　直　接　法　の長所と短所

元来、固有値問題の正解は「直接法」で与えられる。　「神々しい名前」の多くの方法は実践の為の「近似法」である。　しかし、そもそも「直接法」とは何か？
この辺を旨く解説した国産の名著がある。
戸川隼人著「数値計算」（文献４）である。　その pp. 139-143 の記事に、
固有値問題の解法に、およそ５通りあり。
２）特性多項式を作って解く。
１）行列式　｜Ａ -λＩ｜ のゼロ点を、とにかく探す（スツルム法、２分法等）。
３）変換を繰り返して、対角行列または三角行列に収束させる。
４）固有ベクトルを先に求める。
５）その他（ホモトピー法など）

原理的に明快なのが、２）特性多項式法で、私はこれを「直接法」と呼びたい。
勿論、我らのＪ言語のように、多項式の求根が簡単に出来る事が前提にある。
しかし、直接法には、別に２つの問題点が指摘される。　上記の戸川の説明では：
（１）特性多項式の係数の算出に手間がかかる。
この点では、能率の良い計算法（たとえばダニレフスキー法）が開発されて、問題はほぼ解決されている。
（２）この高次多項式の求根に誤差が入り、信頼出来難い。

これは致命的で、よほど小規模な問題（目安は ｎ≦５）でなければ、実用に使えぬ。

また、一松　信氏の例（文献５）では：
ｉ）固有（特性）多項式は敏感である（係数に僅かの誤差が入っても大きく影響される）。　元来、対称行列の固有値は、これらの摂動（perturbation）に対して安定な筈である。
ｉｉ）固有多項式に展開すると、もとの行列成分のもつ多くの情報が紛失して、計算や検算がし難くなる。
ｉｉｉ）原行列が正則でない時には、解に誤差が入りやすい。
ｉｖ）従って、行列の大きさが ｎ が ４ 以下で、すべての成分が整数で（さらに固有値が簡単な有理数か平方根で綺麗に求められる）場合以外には、実用的でない。
と、大変、手厳しい！
ただし、評価は １９９１年時点のもので、時代とともに、変わる可能性がある　と断ってはいる。
我らの「直接法」の、先の数編の報告（文献１、２、３等）の実例では、原行列のサイズは ｎ＝１０ に達し、これら上記の悲観論を充分、乗り越えたものと思う。
これらは、我らが「Ｊ言語」の効果であろう。

２．　ダニレフスキー法　とは何か　？

戸川は、上で、能率のよい計算法、それは「ダニレフスキー法」と述べた。
その実態は何か？　これが、未だに判らぬ。
戸川氏の別著「数値計算入門」（文献６）**p.84** の記述では

ダニレフスキー法　**Danilevski's method**：
行列サイズが　３　より大きい場合には、固有代数方程式は簡単には作れない。
行列式の値を計算するサブルーチンがあっても、λと云う文字の入る式の計算が出来ないから使えない（無理にやろうとしても式の割算が出て來るのでやり難い）。
そこで、消去を旨くやって、式の計算をしないでも、代数方程式の形にする方法が色々考案されている。　ダニレフスキー法もその一種で、どんなタイプの問題にも適用出来る。　（この先の解説が無く、効能書き止まりで判らぬ、残念！）

中野の第一報告（文献１）に、その辺の苦労談があるが、要は、中野の旧友・北大名誉教授・小田島　晟博士の御援助を得て、ある程度まで、理解は進んだ。
先ず、岩波数学辞典第３版（文献７ａ）に簡単な記事があるとの情報を得た。
実は、先行する岩波数学辞典第２版（文献７）に既に似た記述があった。　紹介する。
第２版　**p.422 R**
【行列を変形する方法】　与行列 Ａ を、適当な行列 Ｓ による相似変換で、０要素の多い形 Ｂ に移す、 Ａ → Ｂ＝Ｓ’Ｂ　Ｓ　である（中野注：Ｂ は フロベニウス行列とでも通称するか？）。　この　Ｂ が３重対角行列にでもなれば　万歳！である。例えば ギヴンス法、ハウスホルダー法、ランツォス法の如くに。
ダニレフスキー法では、Ｂをもっと簡単な形、即ち「第１行以外では、主対角線の左隣は １で、他は全て ０ と云う形に、直接、変形する。　その為に、Ｓ は第１行以外は対角行列であるような行列の積になるように選ぶ。
第３版　ページ　**130D**
Ｄ．行列を変形する方法：　　下から順に消去法を実行する。　すなわち、上記 Ｓ を順次１行以外は対角行列である行列の積とし、Ｂ を主対角線の左隣が１、他は第１行を除いて ０ と云う形に変形する。しかし数値的安定性を改良するため、最近では行交換が組み合わされている。
第４版（文献７ｂ）では、この記述は消えた。

● 小田島名誉教授手持ちの資料（文献８）： 同巧である。

しかし、いずれにしろ、不敏なる筆者には、これらの説明は何の事か判らぬ。
少なくとも「直接法」の明快さは感じられない。
実は、有名なウエストレイク著・戸川隼人訳の線形計算のハンドブック（文献９）では「直接法」の中に、なんと、大変多くの方法が包含されている。
例えば、 **Cramer** の公式法、**Cayley-Hamilton** の定理法、三角化法（細分して **Gauss** 消去法、**Cholesky**法、越境法）、対角化法（細分して **Gauss-Jordan**消去法、逆行列法、合同変換法）、直交化法、分割法、三重対角化法 等々。
これだから、固有値問題に分類学が必要になるのだ。

筆者の云う「直接法」は、これら分類学を必要としない、最簡のものを意味する。
つまり、特性方程式法である。 従って、幻の「ダニレフスキー法」とは無縁のものだと、自ら納得している事を断言する。
尚、小田島名誉教授の真剣な応援には感謝して居ります。

３． 再帰的 直接法

数式処理の関係では、固有値問題は、およそ、特性方程式法が優先するようだ。
例えば、解説書「数式処理 入門から高度利用まで」（文献１０）には **Macsyma, Maple, Mathematica、 Reduce** 等の例があるが、大抵は ３次 どまりである。
出来ないのではなくて、展開係数その他、出力が膨大な為であろう。
行列式の次数と展開項の個数の関係は、第１報（文献１）の **pp.3-4** に示して置いたが、再録すれば、λの最高次項を左端に、順次、最低の０次の項を右端にして
ｎ＝２ **1 2 2** 総計 **5** 項
左端はλの最高次の項、２番目の項は行列の主対角要素の個数で、その和は **Trace** を与える。 右端は行列式を与える項の個数である。
ｎ＝３ **1 3 6 6** 総計 **16** 項
右端から２番目は、**mdet (minor det** 主対角線に沿う小行列式の和 **)**に対応。
ここまで知れば、固有方程式の主要部は書き下せる。
ｎ＝４ **1 4 12 24 24** 総計 **65** 項
右端から３番目は、先の **mdet** より一段低次の **mdet** に対応する。
以下、同様であるが、実は、それらの逐次計算が、肝心の大問題なのである。
ｎ＝５ **1 5 20 60 120 120** 総計 **326** 項
ｎ＝６ **1 6 30 120 360 720 720** **1957**
ｎ＝７ **1 7 42 210 840 2520 5040 5040** **13700**
ｎ＝８ **1 8 56 336 1680 6720 20160 40320 40320** **109601**
ｎ＝９ **1 9 72 504 3024 15120 60480 181440 362880** 〃 **986410**
そして ｎ＝１０ ともなれば
1 10 90 720 5040 30240 151200 6048-- 1814400 3628800 〃 で
総計は **9854101** 項となるから、ほぼ、１千万項を処理するプログラムを書く必要がある。

これが実際どの程度の作業量か、著者は Ｍａｐｌｅ での印刷例で実体験した。
関数 **charpoly** のＡ４版・印刷結果は、４次の場合で１頁だが、６次になると ３３ 頁になった。 これは、暗算でも判るが、４次の場合の ５ｘ６＝３０倍 に当たる。
それと同等のプログラミングをＪ言語で行った場合を、お察し下さい。
西川の告「数式処理による固有値問題の解」（文献１１）で、処理の上限が５次だったのはうなずかれる。 また、横浜の老友・山下紀幸氏のＦＡＸは「私は７次が限度」だと報じて来た（文献１２）。
著者は今回の報告「その２」（文献２）の作成中では、実行した９次の プログラムの

肝心の部分のみで、Ａ４版の紙１４頁になったので、やむなく８次の印刷例で投稿した（それでも５頁分）。

この上、さらに １０次 ともなれば、プログラム作成の原理は判っているが、書き下す作業量が大変で（９次の場合の１０倍）、うんざりして仕舞う。
何とか工夫せにゃならん！

最近、Ｊ言語の **Recursive** 再帰的プログラムを工夫して、僅々 ２０行 程度に圧縮出来たので報告「その３（英文）」を **BCS (**英国コンピュータ学会**)** の **VECTOR** 誌に送付した。（文献３） 筆者の下手な英文では、説明を はしょり がちであったので、和文説明を以下に補足する。

要は、データ行列式の展開を、ある行（または列）に沿うのでは無く、主対角線沿いに行うものである。　　データの ｎ次 行列の主対角要素は　ｎケ、そのそれぞれの **minor**（余行列式）は ｎケあるが、その各々の次数は （ｎ－１）に低下する。
次数の低下した行列式について、主対角線沿いに同様の余行列式展開を行う。
その為の関数が　　**minorir** 、結果の　**BOX** 内の　小行列式の各々　**each** について、同様なことを反復して、次数を下げる。
最後に、２次の行列式に到達すれば、その先の展開は暗算でも出来る程であろう。

４次から３次への展開の　**BOX** 表示の　例を示して置く。

```
   nt4   与  4次行列
 1  2  3  4
 5  6  7  8
 9 10 11 12
13 14 15 16
   NB.    演算
 4  Neigen  nt4

 show the nevski method !    NB. Nakano method
my    小行列式（minors）主対角線沿い
 ３次へ
┌──────┬──────┬──────┬──────┐
│ 1  2  3│ 1  2  4│ 1  3  4│ 6  7  8│
│ 5  6  7│ 5  6  8│ 9 11 12│10 11 12│
│ 9 10 11│13 14 16│13 15 16│14 15 16│
└──────┴──────┴──────┴──────┘
  ２次へ
┌─────┬─────┬─────┐
│ 1 2│1  3│ 6  7│
│ 5 6│9 11│10 11│
└─────┴─────┴─────┘

┌─────┬─────┬─────┐
│ 1 2│ 1  4│ 6  8│
│ 5 6│13 16│14 16│
└─────┴─────┴─────┘

┌─────┬─────┬─────┐
│ 1  3│ 1  4│11 12│
│ 9 11│13 16│15 16│
└─────┴─────┴─────┘

┌──────┬─────┬──────┐
│  6  7│ 6  8│11 12│
```

```
| 10 11 | 14 16 | 15 16 |
└──────┴──────┴──────┘
```

```
   mdet =
_7.10543e_15
   nans4  =
0 7.10543e_15 _80 _34 1
```
   固有値　４ケ

```
┌──────────────────────────────────┐
| 36.2094 _2.20937 8.88178e_17 0 |
└──────────────────────────────────┘
```

　次数の低下・展開のプログラムが約１０行、最後の２次の行列式のに展開に約１０行、合計２０行ほどで事は終わる。　次数低下手法は **recursive** 又は　**iterative**、その名前はなんとでも云え、要するにそう云う事だ。
　演算例は　英文報告（文献３）に示す。
実は、私の現用の　Ｊ６０１　システム　（**Sharp PC-AL70F Win XP Home Ed.**、**CPU AMD　2000+, Athlon XP-M　266～　333MHz, 768MB, HD 60GB**）では、１０次がＲＵＮの上限であった。　即ち、データ行列が　１１次以上では、演算途中で（スタート約１０分後）**Windows** 画面は真っ白になり動作はフリーズする。　後は強制終了のみ。
　Ｊ　が演算的には非能率なインタープリータである宿命であろう。　かくて、やっと到達した現在の、この再帰プログラムを更に短縮する小細工には興味は無い。
全く別な対策ならば、意味はあろう。
そこで、次節以降をチュートリアル的に付言したい。

４．　　ＬＡＰＡＣＫ　のこと（導入編）

　この「Ｊと固有値問題」の仕事にのめり込み始めたころ、**JAPLA** の畏敬すべき会友・志村正人氏から、**LAPACK** の事を聞かされた。　これは「固有値問題」に関して、よろずのコンピュータ言語の区別を超えた、共通なプログラム・パッケージであるとの事だ。
志村氏の御指示（下に再録）に従って、私の **J601** 版から **ADDON**　で **LAPACK** を呼び出せるようにした。　一度に進まず、数回のメール交換（文献１３－１７）を行って、やっと何とかなった？　手順を再録する。

　　志村　より：（文献１３、１４）
http://www.jsoftware.com/jai/j601/addons/ これはトロント、にアクセスせよ。
math LAPACK1.1.3.win.zip を **DL**（ダウンロード）する。
ADDON/LAPACK にでも解凍し、２本のスクリプト　**japack.ijs** と **dgeev.ijs**　を読み込む。　**dgeev**　は普通の固有値問題用（データが実数の行列の場合の意味）。
　Lab用のチュートリアル　・ファイル **LAPACK.ijt** もあるので、これを **system/extra/labs/math** にでも **copy** し、**LAB** で始動せよ。

　　中野　返：（文献１５）
load'c:j601\system\extra\labs\lapack.ijt'　　の直後で、残念ながら **spelling error**発生、しかも　エラー箇所　^　印の処はブランク、対応するもの何も見えず。（送信側でゴミが混入かな？）

　　志村　より：　（文献１６）
Ｊ　の **studio | lab** にはいると、チュートリアルの一覧が出て來る。　うまく登録

されていれば、ここに　LAPACK のそれがあると思う。　それを RUN すると
Jlapack.ijs と　dgeev.ijs　の２本のファイルを読んで、いろいろやってくれます。
進行（advance）は　CTRL と J のキーで。

　中野　返：（文献１７）
LAPACK の件、おかげさまで「時々動くようになりました。」　この「時々」が曲者！

実は、小生のうっかりミスか？後遺症があった。　チュートリアルの意味で、メモを残して置く。　どなたか、旨い使用説明をまとめて下さいな。　初級会員の為に。
それは、第２報の文献　11) の項目の末尾「試行錯誤」の条であるが、に、感謝と悔しさとがごっちゃになっている。　しかし、お世話になりました。　　最近、やっと復元出来ました。
尚、後に、英国ＡＰＬ協会の VECTOR誌の旧号（1999）に、 J/LAPACK Interface の記事を再発見した。（文献１８）　しかし、導入などチュートリアルな話は見えぬ。

しかし、LAPACK は　要は　FORTRAN や　Ｃ言語で書かれている。 従って、Ｊ 言語の如き、インタープリータではなく、高速なコンパイルー仕様である。
これと比較したら、Ｊ言語による固有値問題解法は、能率では太刀打ち出来ぬ。
しかし、固有値問題「直接法」のアイデアの実現では、Ｊのプログラムの方が優ろう！

５．　ＬＡＰＡＣＫ　のこと（例題編）

固有値問題の求解には、インタープリータ方式のＪプログラムでは、１０ｘ１０行列までは出来るが、それ以上は無理であった。
昔、「行列計算ソフトウエア」なる「フロッピーディスク付き」の大著があった。（文献１9）　Fortran 77 を供えたスーパーコン用であった。
Ｊのユーザーは、そのＦＤなどを利用せずとも、ＬＡＰＡＣＫで簡便にやれる。
そのトライ例を示しておこう。

例１）　１１次実対称帯行列　（文献１8 p.136）
　昔話、ＡＰＬだけで解く時は、 Pentium -II PC (200Mhz) で　83 min.　だったが
　J/LAPACK の利用では、僅かに　2 min. で、４０倍の高速となったと云う。

```
f1111=:  , 0 ". ] ;._2 (0 : 0)   NB.  Vector '99 July p.136
 1 _2  1  0  0  0  0  0  0  0  0
_2  5 _4  1  0  0  0  0  0  0  0
 1 _4  6 _4  1  0  0  0  0  0  0
 0  1 _4  6 _4  1  0  0  0  0  0
 0  0  1 _4  6 _4  1  0  0  0  0
 0  0  0  1 _4  6 _4  1  0  0  0
 0  0  0  0  1 _4  6 _4  1  0  0
 0  0  0  0  0  1 _4  6 _4  1  0
 0  0  0  0  0  0  1 _4  6 _4  1
 0  0  0  0  0  0  0  1 _4  5 _2
 0  0  0  0  0  0  0  0  1 _2  1
)
 V11=: F1111=: 11 11 $ f1111
```

演算は　J studio | lab　で
```
>1{dgeev_jlapack_ V11          NB. just the eigenvalues
```

15.2692 10.3706 13.2436 7.24053 4.412 2.26237 0.913734 0.0344625 0.253526 4.09459e_16 _9.23228e_17

V11n=. x: 15.2692 10.3706 13.2436 7.24053 4.412 2.26237 0.913734 0.0344625 0.253526 4.09459e_16 _9.23228e_17

```
11 1 $ V11n
  15.2692
  10.3706
  13.2436
   7.24053
   4.412
   2.26237
   0.913734
   0.0344625
   0.253526
   4.09459e_16
  _9.23228e_17
```

例２）　１０次（複素）一般行列　Ｉ１０１０　（本稿、前節２．）

この問題は、ＬＡＰＡＣＫでは、直には解けない。
関数　zgeev_jlapack_　を探し出して用いる準備が要る。
見かけ上、普通の実行列用の求解関数 dgeev を利用の如く見えても、内部で、
複素行列だと判定されれば、それに必要な関数 zgeev を 自動的にロードする
設計がされている。　しかし、その際、パス指定などが的確でないと、未完であり
エラーとなるかも知れぬ。　そう云う事は案外多い筈だ。
　その点、小生らの直接法ならが、如何なる一般行列でも、そのまま解ける。
ただし、Ｊがインタープリーターである為の拘束（CPU や　メモリ関係）はあるが
それは仕方無い。　数学的論理の話では無い！

```
   NB. 10x10   Wilkinson Togawa p.155
a34=: 2j3 3j1 0 0 0 0 0 0 0 0  3j2 _2j_1 1j2 0 0 0 0 0 0 0
a34=:a34, 5j_3 1j2 2j1 _1j4 0 0 0 0 0 0   2j6 _2j3 3j_1 _4j2 5j5 0 0 0 0 0
a34=:a34, 1j4 2j2 _3j7 1j5 2j_3 1j6 0 0 0 0
a34=:a34, 5j_1 0j4 1j5 _8j_1 4j7 7j1 4j_2 0 0 0
a34=:a34, 5j2 1j4 6j_5 8j4 4j_4 _1j5 3j0 _4j6 0 0
a34=:a34, _4j_3 7j3 1j6 2j_4 3j1 1j2 1j4 6j3 7j_1  0
a34=:a34, 5j0 2j2 1j3 1j1 _4j_2 1j6 1j2 2j5 0j1 3j2
a34=:a34, 5j2 2j6 1j_3 7j4 4j1 _7j0 3j_3 5j_4 6j3 2j5
  I1010=:A34=: 10 10 $ a34
  I10n=. >1{zgeev I1010
    10 1 $ I10n
 10.7977j8.62338
 1.03206j9.29413
 8.81131j1.54938
 2.38989j7.26807
 4.16175j3.13751
 5.43645j_3.97143
_4.96687j_8.08712
_2.44755j0.437126
_5.27951j_2.27596
 _1.9352j_3.97509
```

この解は、前の英文報告（その３）　（文献３）の　Ｊ言語解と同じである。

例３）　１６次（複素）一般行列　Ｉ１６１６

（文献９　pp. 160-161　　Wilkinson の例題１６次複素行列）

```
   i1616=:       3j2   4j_1   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   i1616=: i1616, 4j_1 _1j_1 2j4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   i1616=: i1616, 0   2j4 3j_4 3j1 0 0 0 0 0 0 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 3j1  2j3 3j_2  0 0 0 0 0 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 3j_2 _5j1 2j_2 0 0 0 0 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 0 2j_2  1j2  2j3 0 0 0 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 0 0 2j3  5j2  1j3  0 0 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 0 0 0 1j3 _2j1 _2j2  0 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 0 0 0 0 _2j2 1j_2 3j_3 0 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 0 0 0 0 0 3j3 _1j_4 _1j5 0 0 0 0 0 0
   i1616=: i1616, 0 0 0 0 0 0 0 0 0  _1j5 2j1 4j3  0 0 0 0
   i1616=: i1616, 0 0 0 0 0 0 0 0 0 0 4j3 1j_5 1j_6  0 0 0
   i1616=: i1616, 0 0 0 0 0 0 0 0 0 0 0  1j_6 3j1 2j1  0 0
   i1616=: i1616, 0 0 0 0 0 0 0 0 0 0 0 0  2j1 2j4 5j_1  0
   i1616=: i1616, 0 0 0 0 0 0 0 0 0 0 0 0 0 5j_1 _4j3 _3j_4
   i1616=: i1616, 0 0 0 0 0 0 0 0 0 0 0 0 0 0   _3j_4 1j_5

   I1616=: 16 16 $ i1616
```

上記の必要な関数 zgeev のロードを確認した上で

```
I16n =. > 1 { dgeev_jlapack_   I1616
  16 1 $ I16n
```

から　結果の固有値は

```
  _6.10339j3.39653
   6.33009j1.48798
   5.19682j5.1407
   3.62656j5.16801
   5.47875j3.73098
 _0.431815j_8.60507
   4.73331j_4.82187
  _5.27916j3.6122
   3.47333j0.655612
   _3.0159j0.883875
   _1.5385j1.62429
_0.0678014j1.8517
  0.609356j_0.939372
 _0.578741j_5.28493
 _0.178878j_3.7624
  _1.25404j_5.13822
```

以上で、チュートリアルズ　を終える。

む　　す　　び

古典的な固有値問題でも、結構やるべき事を発見出来、楽しめた。
行列処理言語　Ｊ　の長所を体感出来た。　実はまだまだ、やりたい事が出来そうだ。
ＪＡＰＬＡの諸賢の応援に感謝致します。

文　献　・　資　料

１）中野嘉弘：「Ｊ言語と高等数学　－　固有値問題（直接法）を主に」
JAPLA 2007 Apr 28　pp.9　６次行列まで

２）中野嘉弘：「Ｊ言語と高等数学（その２）　直接法の発展」
JAPLA 2007 May 26　pp.13　９次行列まで

３）**Y. Nakano: " J and Eigenvakue Problems" 2007 June 6**
10x10 general matrices (to be published by VECTOR)

４）戸川隼人：「情報処理入門コース　７　数値計算」岩波書店、**1991 pp.245**

５）一松　信：「代数学入門　第二課」近代科学社、**1992 pp.260**

６）戸川隼人：「数値計算入門」オーム社、**1970 pp.230**

７）「岩波数学辞典　第２版」　岩波書店、**1954** 第１刷、**1968** 第２版、**p.422 R**
－ａ「岩波数学辞典　第３版」　岩波書店、**1985** 第３版、**1997** 第１２刷 **130D**
－ｂ「岩波数学辞典　第４版 **CD-ROM**付」　岩波書店、**2007**　第４版第１刷

８）小田島名誉教授資料：
Ｒ．ツルミュール原著、瀬川富士・高市成方 共訳「マトリックスの理論と応用」
ブレイン図書出版、 **1972** 初版
Rudolf Zurmühl (Ordentlich Professor an der Technischen Universität Berlin):" Matrizen und ihre techinischen Anwendungen" 1964
Vierte Neubearbeitete Auflage, Springer-Verlag
－ａ）ダニレフスキー原著（チェツコスロバキア語らしい）、名前だけ
DANILEVSKII A.: O čislennom rešenil vekovogo uravneniya.
Mat. Sbornik Bd.2 (1937). S.169-171

９）Ｊ．Ｒ．ウエストレイク原著、戸川隼人 訳：「コンピュータのための線形計算ハンドブック」　培風館、**1972** 昭和**47.10.5** 初版
Joan R. Westlake: " A Handbook of Numerical Matrix and Solution of Linear Equations " 1968

１０）雑誌 インターフェース増刊「数式処理　入門から高度利用まで」アーカイブ
No.１２　ＣＱ出版社　1990

１１）西川利男：「記号処理により行列式を直接展開して固有値を求める」
JAPLA研究会資料 2007/5/26 pp.24

１２）山下ＦＡＸ（**'07.5.29 11:29**）「固有値　**eigen7y**　７次まではやりました」

１３）志村 **e-mail (2007.5.10. 12:44)**「**J**　から **ADDON** で　**LAPACK** 呼出」

１４）志村 **e-mail (2007.5.11. 9:47)**「**LAPACK** は **FORTRAN** のパッケージで・・・」

１５）中野 **e-mail (2007.5.11. 17:50)**「場所不明のスペリング・エラー発生！」

１６）志村 **e-mail (2007.5.14. 10:11)**「**studio | lab** で　**RUN** させます。」

１７）中野 **e-mail (2007.5.22. 13:33)**「時々動くようになった。

１８）　**R.J.Procter and R.L.W.Brown : " J/LAPACK Interface Makes Economic Analysis Fly! " Vector Vol.16 No.1 July 1999 pp.133-141**

１９）小國　力編著　村田健郎、三好俊郎、**J.J.** ドンガラ、長谷川非で秀彦著：
「行列計算ソフトウエア　ＷＳ、スーパーコン、並列計算機
フロッピーディスク付き」　丸善、**1991**、平成３年１１月、￥９，２７０

※　ＬＡＰＡＣＫ　が利用出来れば、このＦＤを今更、使う理由はなかろう。