GotoBLAS入門

木村欣司 (京都大学大学院情報学研究科)

September 23, 2009

講演のタイトルの読み方

“ゴトー” ブラスにゅうもん

“ゴートゥー” ブラスにゅうもん ではない

後藤さん作のBLASという意味

GO TO文が、ソースコードにたくさん入ったソフトウェアという意味ではない

Q. じゃ，後藤さんが解説をすれば，よいのでは?

A. 後藤さんは，現在，テキサス大学におられます

Internetを使って，講演をしてもらうとか。。。

まぁ，細かいことは言わずに，今日は僕でご勘弁

後藤さんの後輩の私が，“作者も驚きのGotoBLAS活用法” を紹介します

GotoBLASは，なにをするためのソフトウェア？

例えば，浮動小数点数のベクトルの内積ができます

浮動小数点数の密行列とベクトルの積ができます

浮動小数点数の密行列と密行列の積ができます

専門家の中にも，それだけだと誤解している人が多数いますが

実は，GotoBLAS単体で(LAPACKがなくても),

連立一次方程式を，高速に解く機能が入っています

もちろん，LU分解できます

コレスキー分解だって，高速に計算できます

逆行列だって計算できます

素朴な疑問

浮動小数点数のベクトルの内積ができれば,
それを基礎に
浮動小数点数の密行列とベクトルの積ができ,
それを基礎に
浮動小数点数の密行列と密行列の積ができる

どこがすこいソフトウェアなのか数学者には，まったくわからない???

実は，密行列とベクトルの積では，少し，密行列と密行列の積では十分に，データの再利用が行える

メインメモリからロードしてきたデータをCPUのキャッシュというところに閉じ込めて，極力，キャッシュとレジスタ(演算器)とのデータのやり取りで，演算を行うようにして，メインメモリにデータを取りに行く回数が少なくなるような工夫を行列と行列の積では行えるのである

密行列と密行列の積では，CPUの理論演算性能値に近い値まで，CPUの演算性能を引き出すことが可能である

GotoBLASは，限界までCPUがその性能を発揮できるようにプログラミングされたソフトウェアなのである

逆に言うと，CPUの性能を発揮できないソフトウェアは，メインメモリからのデータの到着待ちになっている場合が多い

私は，固有値を計算したいのですが?

BLASの意味

=Basic Linear Algebra Subprograms

このSubprogramsを下位ルーチンとして，上位にLAPACKというソフトウェアがある

LAPACKは，連立一次方程式系の問題だけでなく固有値も計算できる，ベクトルの直交化もできる

要するに，LAPACKとGotoBLASを組み合わせれば，固有値だって計算できるようになるのです

“組み合わせる”ということは，もしかして，後藤さん作でないBLASもあるってこと?

正解，後で紹介します

じゃ，逆に，LAPACKじゃないけど，BLASを下位ルーチンとして固有値などを計算するソフトウェアがあるってこと?

正解，後で紹介します

後藤さん作でないBLAS

1)Intel Math Kernel Library(MKL)
2)AMD Core Math Library(ACML)
3)IBM ESSL/BLAS for Cell
4)Sun Performance Library
5)Fujitsu SSL2
6)ATLAS

これらのライブラリは，BLASの機能だけでなくLAPACKのコードをさまざまなCPUに対してチューニングした実行コードも含んでいる

BLASの計算をCPU以外でも行うことで計算を加速させるもの(ライブラリ名も一部含む)

1)CUBLAS for NVIDIA GPU

2)ACML-GPU for ATI GPU

3)ClearSpeed社のアクセラレータボード

4)K&F Computing Research社の科学技術計算向け加速ボードGRAPE-DR

さらに，NVIDIAには，CULAtoolsというライブラリがあり，それはLAPACKが持つ関数の一部を提供している

LAPACKじゃないけど，BLASを下位ルーチンとして固有値などを計算する上位ルーチン

LAPACKの並列計算向けライブラリとして，Scala-packとPLAPACKがある

その他としては，
京都大学中村佳正研究室の密行列特異値分解のためのI-SVDライブラリがある
http://www-is.amp.i.kyoto-u.ac.jp/lab/isvd/

GotoBLASの使い方

http://www.tacc.utexas.edu/
research-development/tacc-projects/

まずは, ダウンロード の前に,
アカウントの取得, ライセンス条項を読みましょう

次は, いよいよコンパイル, でも, その前によく考えて
ほしいことあり

GotoBLASのデフォルトのコンパイルオプションでは，マルチコアCPU向けにスレッド並列が行われるように設定されている

GotoBLASを利用するユーザーの中には，わざわざ，GotoBLASをシングルスレッドで動かしたくなるユーザーがいる???

世界に一人は，確実にいる ⇒ 私

理由 線形代数の上位層で並列化をする必要あり(???)

シングルスレッド動作とマルチスレッド動作を途中で行ったり来たりしたい人もいる

理由 縦長行列のQR分解のためのAllReduceアルゴリズムでは，各コアが独立にQR分解できる場面が現れるから，そこでは，行列サイズがかなり小さくなるため，スレッドレベルの並列化には向かない，しかし，統合部分ではマルチスレッドで動作させたい

線形代数の上位層での並列化

整数行列Aの固有多項式$f(x) = \det(A - xI)$を計算することを考える

$$
\begin{aligned}
f_1(x) &= f(x) \bmod p_1 \\
f_2(x) &= f(x) \bmod p_2 \\
f_3(x) &= f(x) \bmod p_3 \\
&\vdots
\end{aligned}
$$

中国剰余定理により，$f(x)$を復元

$f_i(x)$を計算するときに，GotoBLASが必要になる

OpenMPを使って，OpenMPのスレッド毎にGotoBLASを呼びだす場合，GotoBLASはシングルスレッドモードで動作する

Makefile.rule の中で，USE_OPENMP ＝ 1 とすればその設定となる

AllReduceアルゴリズムでは，8コアを2スレッド4並列のように割り当てる必要があるが，最大スレッド数をgoto_set_num_threads(int)で指定できるため，MPIで4並列にすればよい

GotoBLASのコンパイル

途中で，LAPACKを自動的にダウンロードするため，
export http_proxy=”proxyサーバー名:8080” を設定しておく
(最近のGotoBLASは，LAPACKをコンパイラでコンパイルし，中に取り込む)

Linux，x86_64ならば，./quickbuild.64bit でOK

GotoBLASへのリンク

LU分解したいならば，プログラム中に
ポインタ渡しの引数でdgetrf_ という関数名を書く

自分のソースコードのコンパイル時には，
gcc -o 実行ファイル名 ソースコードファイル名
-L[GotoBLASのディレクトリ] -lgoto2
とすればよい

連立一次方程式が高速に解けることがなんの役に立つのか?

用途がありすぎて，このスライドにはおさまらない

最も**風変りな用途**をこれから紹介する

GotoBLASの数値計算への応用

行列の標準固有値問題

Aを与えられた行列とすると,

$$Av = \lambda v$$

vはベクトル, λは固有値である

この問題を一般化した問題を考える

非線形固有値問題

非線形固有値問題とは,

$$P(\lambda) = \begin{pmatrix} p_{11}(\lambda) & \cdots & p_{1n}(\lambda) \\ \vdots & & \vdots \\ p_{n1}(\lambda) & \cdots & p_{nn}(\lambda) \end{pmatrix}$$

$$P(\lambda)v = 0$$

の形の固有値問題のことをいう, ただし, $p_{ij}(\lambda)$はλの関数である

標準固有値問題の書き換え

$$\begin{pmatrix} a_{1,1}-\lambda & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2}-\lambda & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3,}-\lambda \end{pmatrix} v = 0$$

標準固有値問題は，非線形固有値問題に含まれる

非線形固有値問題の例

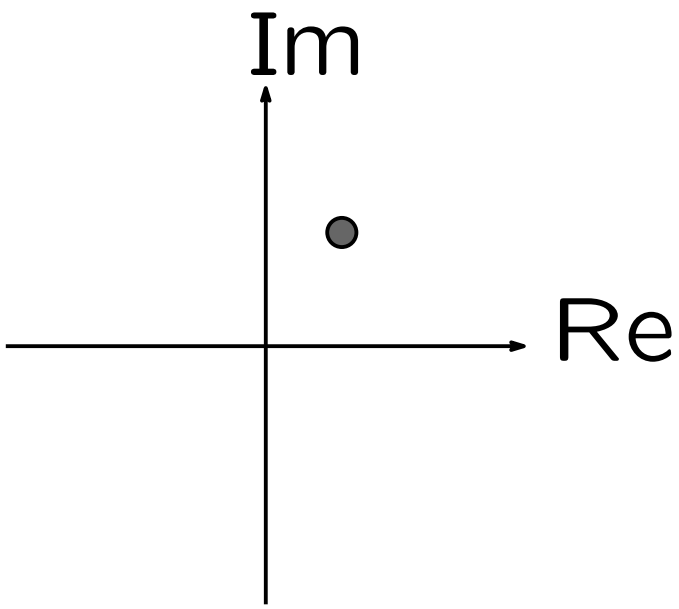
$$\begin{pmatrix} 1 & 0 & \exp(\lambda^2)-3 \\ \exp(\lambda) & -1 & 0 \\ 0 & \exp(\lambda) & -1 \end{pmatrix} v = 0$$

ブロック櫻井–杉浦法

固有値とは，$\det(P(\lambda)) = 0$の根である

固有値は，一般に複素数(例 $2 + 3i$)である

複素数は，複素平面にプロットできる



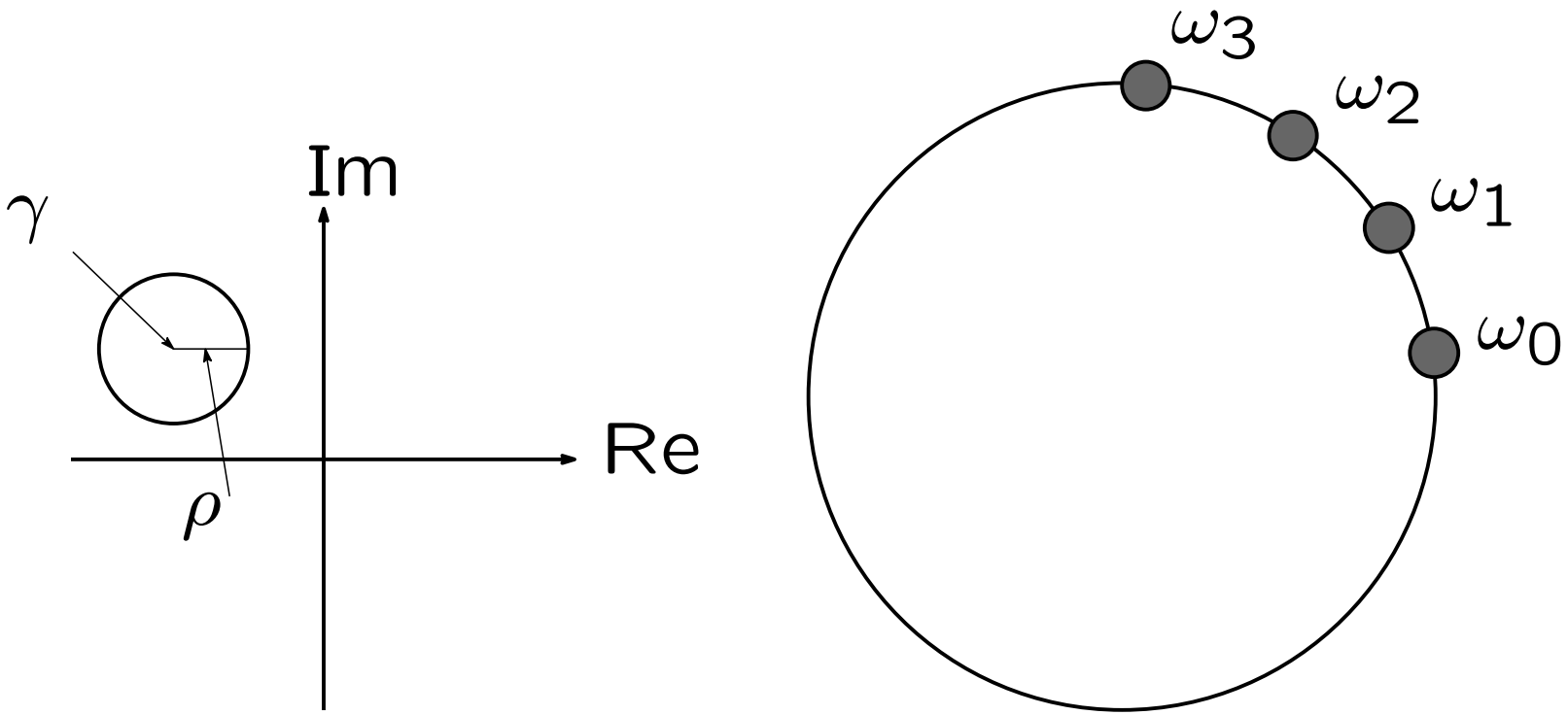
ブロック櫻井–杉浦法とは，複素平面の自分が指定した円の内部のすべての固有値を余すところなく求められるアルゴリズムである

“自分が指定した円の内部の固有値すべてを求める” という機能は，あまりに限定された機能ではないのか?

実問題に，たくさんの応用が考えられる

分子軌道計算(FMO-MO法),

実空間密度汎関数法(RSDFT)など



Input: $U, V \in \mathbb{C}^{n\times L}$(乱数で生成), $N, K, L, \delta, \gamma, \rho$

nは行列のサイズ，Nは計算精度に関係する量である．

Output: $\hat{\lambda}_1, \cdots, \hat{\lambda}_{\hat{m}}, \hat{v}_1, \cdots, \hat{v}_{\hat{m}}$，**$\hat{m}$は，円の内部の固有値の数**，$\hat{m} < KL$

1. Set $\omega_j \leftarrow \gamma + \rho \exp(2\pi i(j+1/2)/N),\ j = 0, \cdots, N-1$
2. Compute $P(\omega_j)^{-1}V,\ j = 0, \cdots, N-1$

異なる問題の連立一次方程式を独立に解くことに対応する

3. Compute $\widehat{S}_k,\ k = 0, \cdots, 2K-1$ by

$$\widehat{S}_k = \frac{1}{N} \sum_{j=0}^{N-1} \left(\frac{\omega_j - \gamma}{\rho} \right)^{k+1} P(\omega_j)^{-1} V$$

4. Form $\widehat{M}_k = U^H \widehat{S}_k,\ k = 0, \cdots, 2K-1$
5. Construct $\widehat{H}_{KL} = [\widehat{M}_{i+j-2}]_{i,j=1}^{K}$ and $\widehat{H}_{KL}^{<} = [\widehat{M}_{i+j-1}]_{i,j=1}^{K} \in \mathbb{C}^{KL \times KL}$
6. Perform singular value decomposition of $\widehat{H}_{KL} = \widehat{U}_{KL} \widehat{\Sigma} \widehat{V}_{KL}^H$

7.Omit small singular value components $s < \delta$ so that $\hat{U}_{\hat{m}} = \hat{U}_{KL}(1 : \hat{m})$, $\hat{V}_{\hat{m}} = \hat{V}_{KL}(1 : \hat{m})$, where $\hat{m} < KL$

8.Compute the eigenpairs $(\zeta_1, w_1), \cdots, (\zeta_{\hat{m}}, w_{\hat{m}})$ of the pencil $\hat{U}_{\hat{m}}^H \hat{H}_{KL}^{<} \hat{V}_{\hat{m}} - \lambda \hat{U}_{\hat{m}}^H \hat{H}_{KL} \hat{V}_{\hat{m}}$

9.Construct $S = [\hat{S}_0, \cdots, \hat{S}_{\hat{m}-1}]$

10.Compute $x_j = Sw_j,\ j = 1, \cdots, \hat{m}$

11.Set $\hat{\lambda}_j \leftarrow \gamma + \rho\zeta_j,\ j = 1, \cdots, \hat{m}$

ブロック櫻井–杉浦法は，複素関数論における周回積分，留数などが算法の背景となっている

$P(\omega_j)^{-1}V$は，異なる問題の連立一次方程式を独立に解くことに対応する

連立一次方程式の解法

1)密行列の問題ならば，そのままGotoBLASを適用可能

2)疎行列ならば，次のスライドのようにクリロフ部分空間法の前処理行列の作成にGotoBLASを活用する

$P(\omega_j)$は，ωにω_jを代入したことにより，数値行列になっている

$P(\omega_j)$の(i,j)成分を，$p_{i,j}$と書く

行列Qの(i,j)成分を，次のように定義する．

$$q_{i,j} = \begin{cases} p_{i,j} & \mathrm{abs}(p_{i,j}) > \Delta \times \max(\mathrm{abs}(p_{i,j})) \\ 0 & \mathrm{abs}(p_{i,j}) \leq \Delta \times \max(\mathrm{abs}(p_{i,j})) \end{cases}$$

$\Delta(0 < \Delta < 1)$は，ユーザーが定める閾値

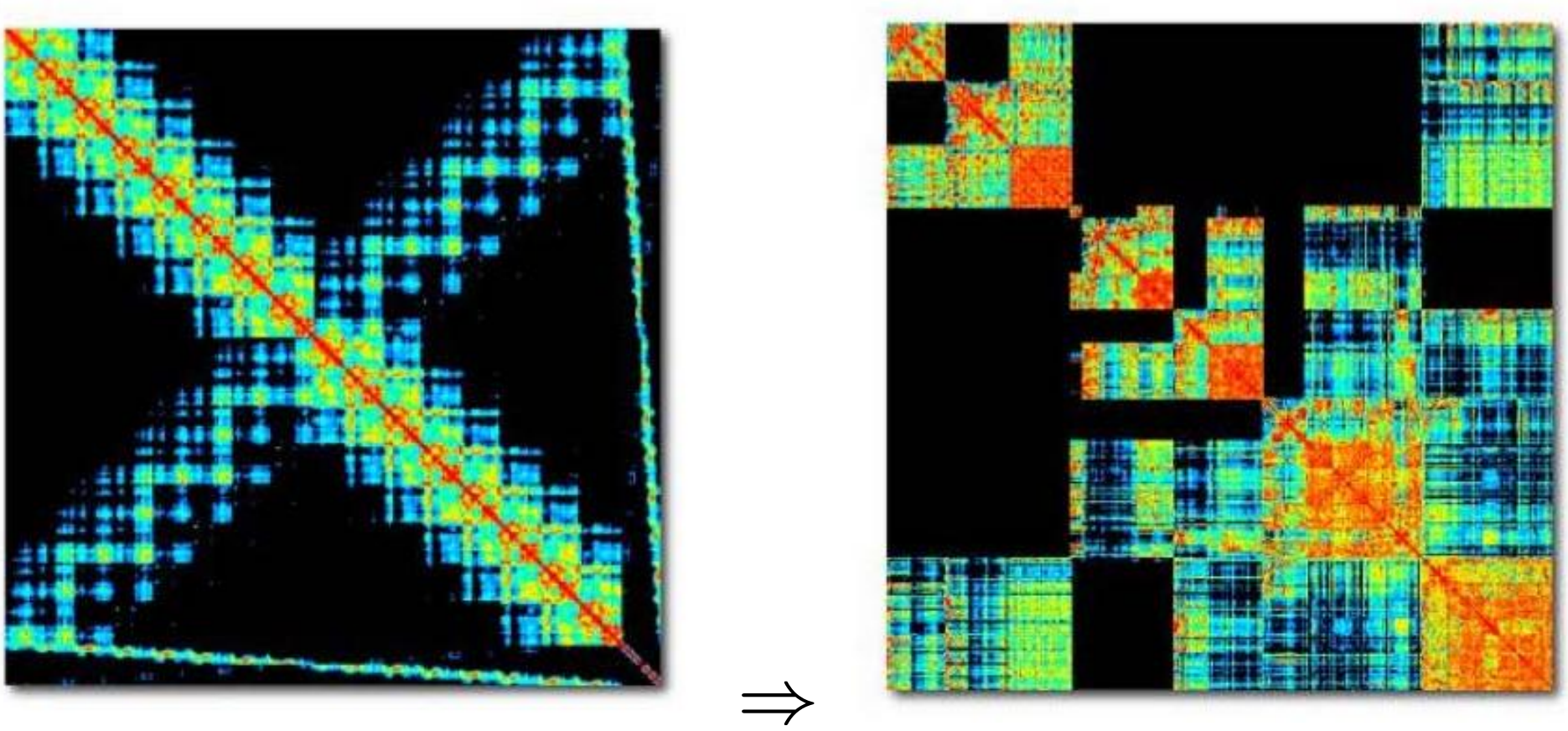
行列Qを，**Nested-Dissection Ordering**を用いて並び替え，行列Rを作る

Nested-Dissection Ordering後の行列Rは，部分密行列のLU分解と行列積を用いて，全体のLU分解が可能である

行列RのLU分解の結果を，クリロフ部分空間法の前処理行列として用いる

“Nested-Dissection Ordering後の行列は，GotoBLASが得意とする行列の姿になっているというのがここでのポイント”

Nested-Dissection Orderingの例



非線形固有値問題と連立代数方程式の関係

$$f(x,y) = x^2 + y^2 - 3, g(x,y) = xy - 1$$

交点を求める，Dixon multipolynomial resultant の理論より

$$\frac{\begin{vmatrix} f(x,y) & g(x,y) \\ f(\alpha,y) & g(\alpha,y) \end{vmatrix}}{x-\alpha} = \begin{pmatrix} \alpha & 1 \end{pmatrix} \begin{pmatrix} y & -1 \\ -1 & -y^3+3y \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} = 0$$

αに関係なく成立する

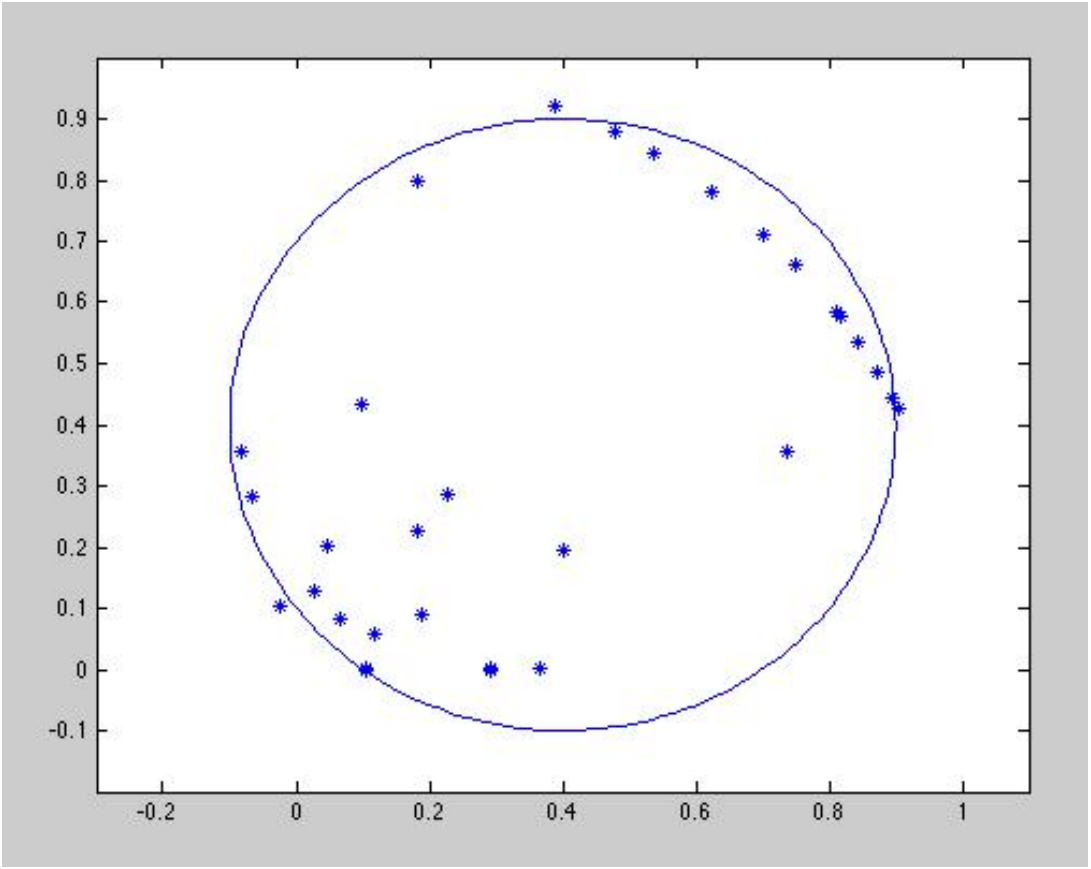
$$y = \lambda, v = \begin{pmatrix} x \\ 1 \end{pmatrix}$$

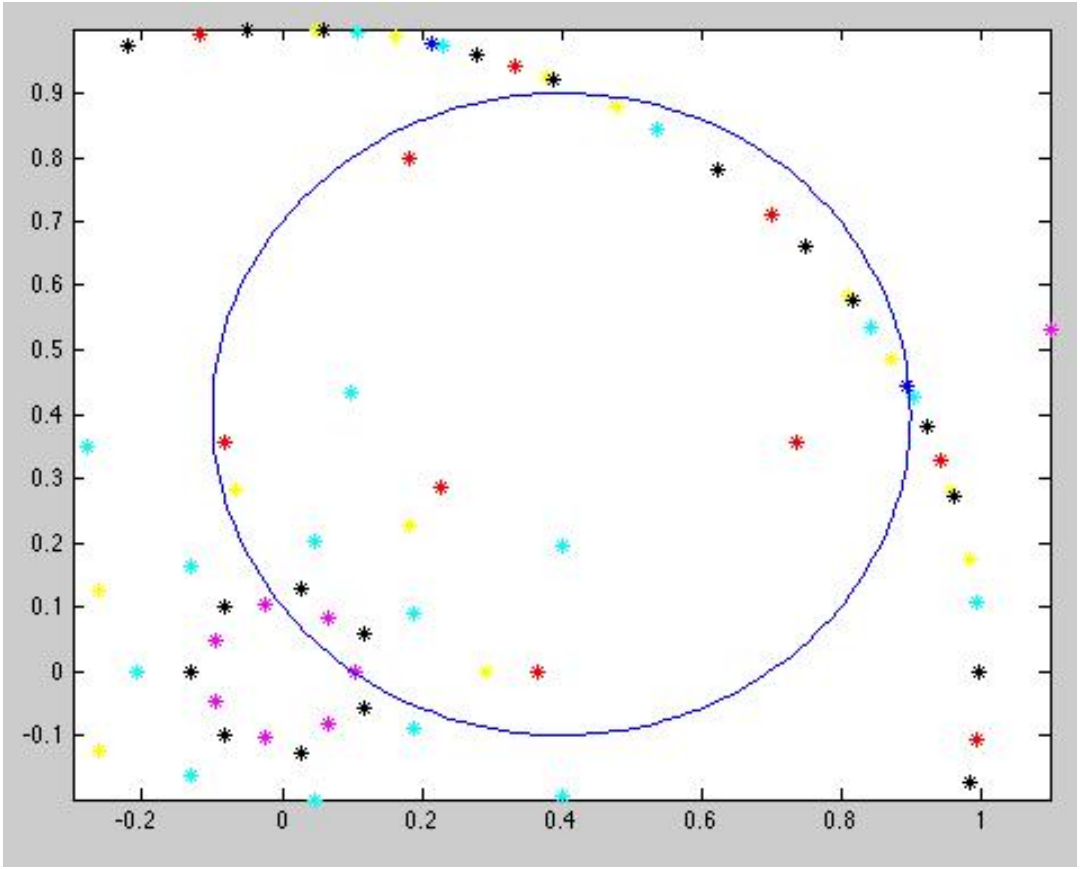
とおけば,

$$\begin{pmatrix} \lambda & -1 \\ -1 & -\lambda^3 + 3\lambda \end{pmatrix} v = 0$$

非線形固有値問題が得られる，この方法は，３変数以上でも適用可能である

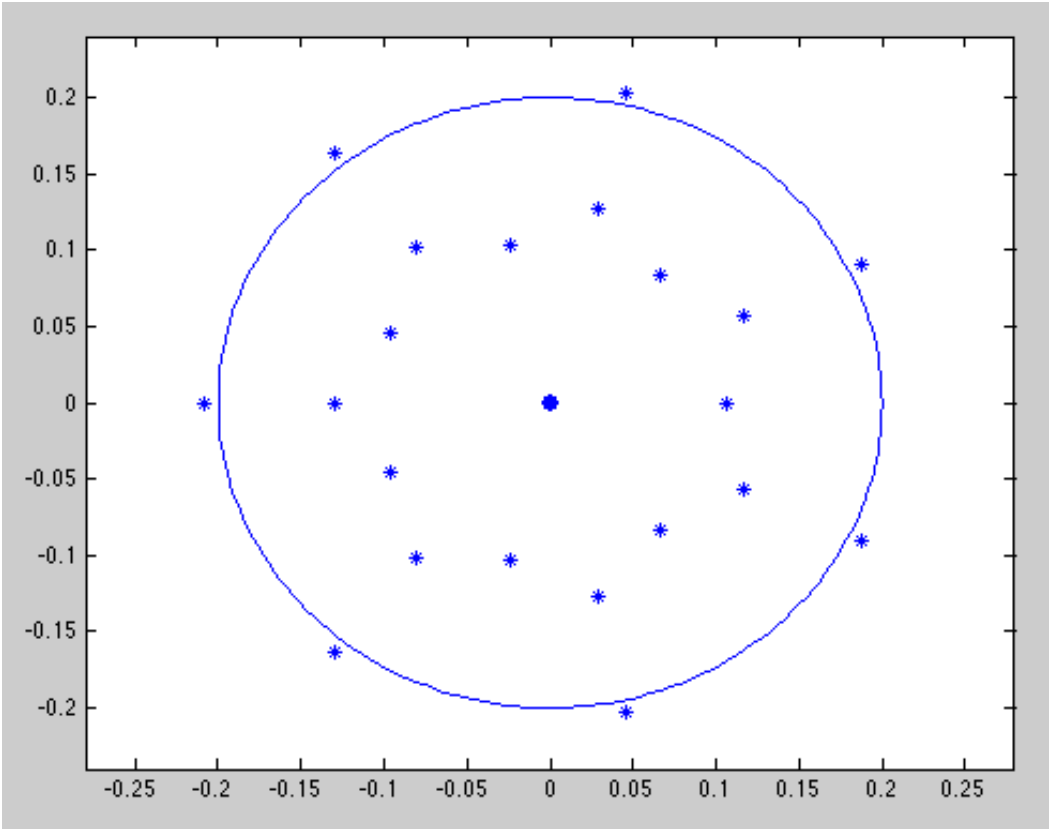
例題:Cyclic-7への応用(その1)

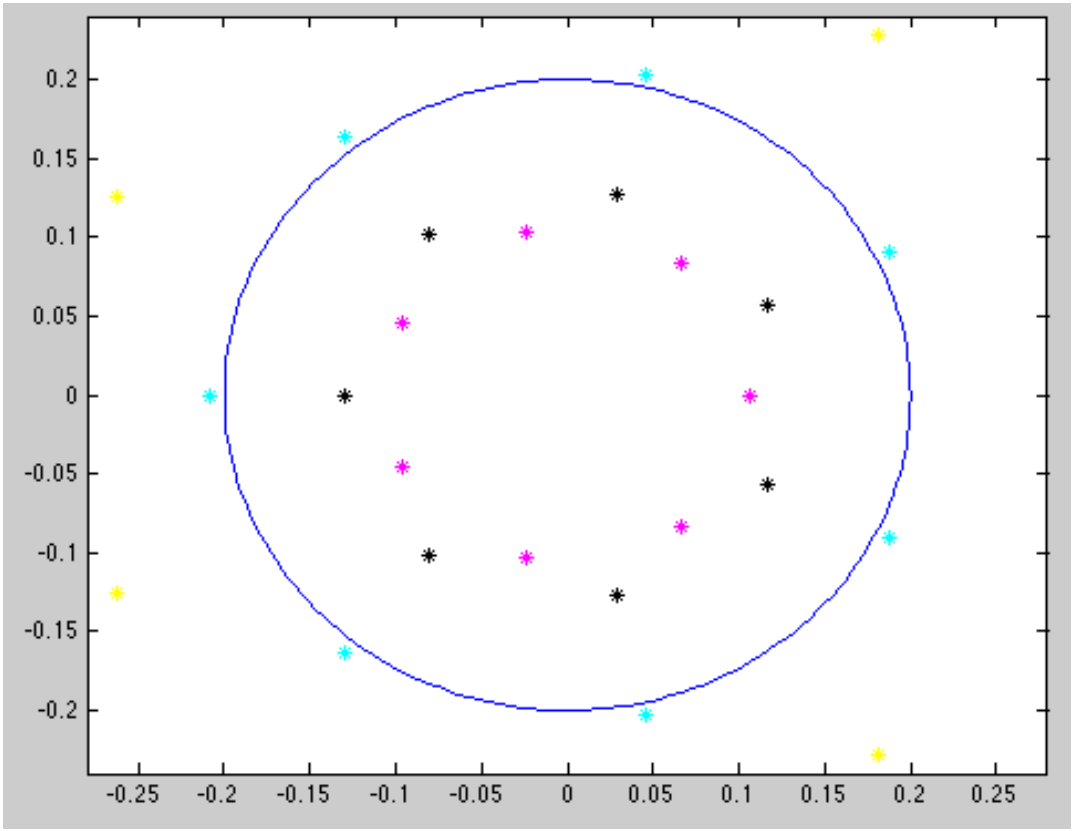




左は, Dixon multipolynomial resultantと櫻井・杉浦法の結果, 右は, ホモトピー法の結果

例題:Cyclic-7への応用(その2)





左は, Dixon multipolynomial resultantと櫻井・杉浦法の結果, 右は, ホモトピー法の結果

GotoBLASの数式処理への応用

整数行列Aの固有多項式$f(x) = \det(A - xI)$を計算することを考える

例

$$A = \begin{pmatrix} 2 & 3 & 7 \\ 4 & 11 & 13 \\ 9 & 1 & 8 \end{pmatrix}, f(x) = \begin{vmatrix} 2-x & 3 & 7 \\ 4 & 11-x & 13 \\ 9 & 1 & 8-x \end{vmatrix}$$

数式処理ソフトは，誤差無し計算の最たるもの
なぜ，浮動小数点数を扱うGotoBLASを活用できるのか?

浮動小数点数IEEE754の規格:$\pm 1.zzzzzzzzzz \times 2^{xxx}$
仮数部52bitあるので，economized formのため，
$2^{53} - 1$までは，正確に整数を表現できる，
さらに，1.0×2^{53}であるから，0から2^{53}までのすべての整数を正確に表現できる

いま，行列サイズをNとするとき，$0 < p < \sqrt{\frac{2^{53}}{N}}$を満たす素数$p$を法とする有限体$\mathbb{Z}/p\mathbb{Z}$を考える

GotoBLASが内積，行列ベクトル積，行列行列積を計算した後，pで余算を行うことで有限体$\mathbb{Z}/p\mathbb{Z}$を実現できる

中国剰余定理のおかげで，有限体$\mathbb{Z}/p\mathbb{Z}$が扱えれば，整数$\mathbb{Z}$における値を復元できる

中国剰余定理

未知の整数をXとする

$$X \bmod 2 = 1$$

$$X \bmod 3 = 2$$

この情報から，**中国剰余定理**により

$$X = \cdots, -13, -7, -1, 5, 11, 17, \cdots$$

が，Xの候補となる，解は一意に定まらない

もし，Xは1桁の整数であるという付加情報が手に入ったとすると，$X = -7, -1, 5$，まだ，3つも候補が残っている

もうすこし情報が手に入ったとする

$$X \bmod \underline{2} = 1$$
$$X \bmod \underline{3} = 2$$
$$X \bmod \underline{5} = 3$$

よって，答えは，中国剰余定理と付加情報により$X = -7$と一意的に定まる，**下線の部分には，素数を使う**

(付加情報)行列式に付随した2つの上界公式

2つのノルムを定義する: qは, 多変数多項式

$$\|q\|_1 = \sum_{\alpha_1,\cdots,\alpha_s} |c(\alpha_1,\cdots,\alpha_s)|,$$

$$\|q\|_2 = \sqrt{\sum_{\alpha_1,\cdots,\alpha_s} c(\alpha_1,\cdots,\alpha_s)^2},$$

ここで,

$q = \Sigma_{\alpha_1,\cdots,\alpha_s} c(\alpha_1,\cdots,\alpha_s) x_1^{\alpha_1}\cdots x_s^{\alpha_s} \in \mathbb{Z}[x_1,\cdots,x_s]$

$A = (a_{i,j}) \in \mathbb{Z}^{N\times N}$ についてのHadamardの上界は,

$$|\det(A)| \leq \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} |a_{i,j}|^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} |a_{i,j}|^2} \right\}$$

$A = (a_{i,j}) \in \mathbb{Z}[x_1, \cdots, x_s]^{N\times N}$ についてのGoldsteinとGrahamの上界は,

$$\|\det(A)\|_2 \leq \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} \|a_{i,j}\|_1^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} \|a_{i,j}\|_1^2} \right\}$$

固有多項式の係数の上界もこの式から得られる

$$B = \begin{vmatrix} a & 2b \\ 3c & 4d + f \end{vmatrix} = \underline{4}ad + \underline{1}af - \underline{6}bc$$

この例の場合，GoldsteinとGrahamの上界は，

$\min(\sqrt{1+4}\sqrt{9+25}, \sqrt{1+9}\sqrt{4+25}) < 13.1$

有限体上のＬＵ分解の実装(その１)

有限体上のLU分解の実装(その2)

パネル分解は，メモリバンド幅を要求するため，倍精度浮動小数点数をintegerに型キャストして，内積形式LU分解で行う，64bitの倍精度浮動小数点数のロードよりも，32bitのintegerのロードのほうが速い，integerとintegerの積の結果はlongに格納するが，64bitのlongをストアする手間を考えると，外積形式LU分解よりも内積形式LU分解のほうがお得である

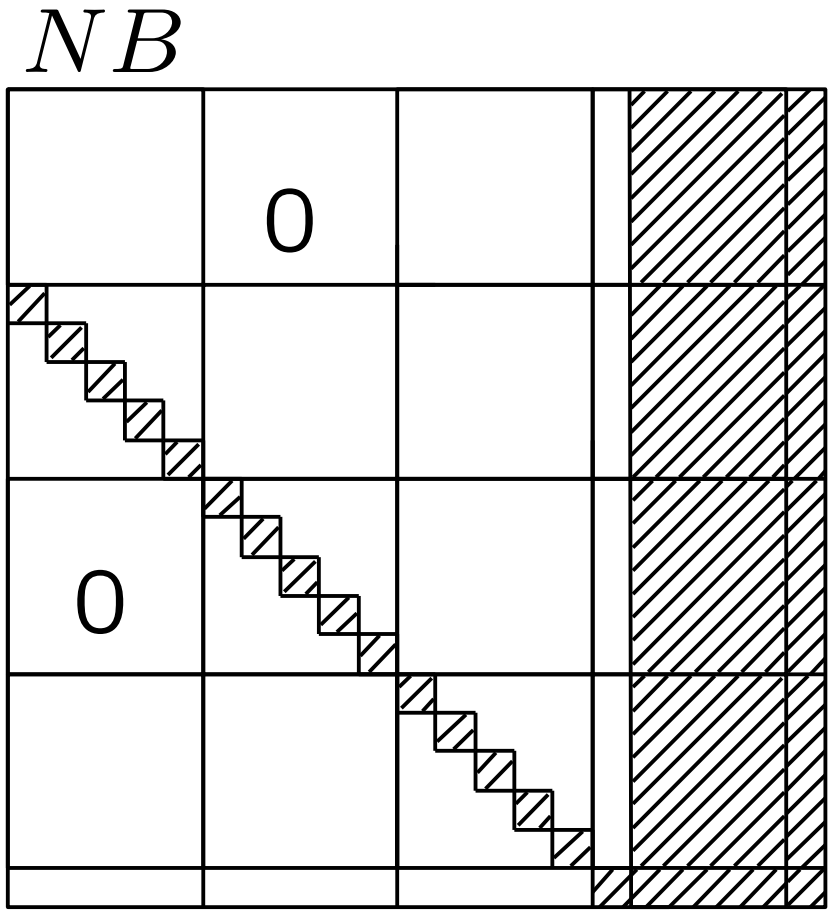
有限体上のＬＵ分解の実装(その３)

dtrsmの部分は，パネル分解の結果として得られるブロッキングサイズＮＢ×ＮＢの行列の下三角行列の逆行列を計算し，その結果と長方形行列をdtrmmで掛け合わせることによって計算する

行列積を利用した固有多項式の計算法

$N \times NB(NB < N)$の縦長の行列Vを用意する,

Cを帯コンパニオン行列とする



Nと$NB \times l$が等しくない場合には，微調整が必要であるが，$N = NB \times l$の場合には，

$$\left(V \ AV \ \cdots \ A^{l-1}V \right) C = \left(AV \ A^2V \ \cdots \ A^lV \right)$$

$$= A \left(V \ AV \ \cdots \ A^{l-1}V \right)$$

この形の連立一次方程式を解くと，Aと同じ固有値を持つ帯コンパニオン行列Cが手に入る

Aの最小多項式の次数＝Aの固有多項式の次数のときでない場合には，この方法は使えないことを注意する

Cの固有多項式の計算法

Cは，疎行列である，疎行列の固有多項式は，Wiedemann algorithmにおける最小多項式の計算法を利用すればよい

クリロフ部分空間を生成する際の余算の削減のため，Cでなく$C^\top$を考えることがポイントである

1)行列ベクトル積を利用した固有多項式計算 V.S.2)行列積を利用した固有多項式計算

実験環境:Intel Core i7 920の1コアのみを使用

行列ベクトル積を利用した固有多項式計算は，C言語のソースコードをgcc 4.4.1でコンパイル

5000次元の行列の有限体($p = 1342177$)上の固有多項式の計算時間

1)83.919sec 2)38.016sec(9.4786sec)

宣伝

最近, 15次の判別式の計算に成功しました
どのように計算するかも含め, 2,3月号の数学セミナーに掲載の予定

そこに掲載予定の手法を使うと

$$f(x) = \begin{vmatrix} a+b-x & a^2+c & abc \\ ab+bc & c^2-x & a^2+c \\ a+c^2 & c & a^3+b^3+c^3-x \end{vmatrix}$$

もGotoBLASで計算可能