Pade近似の計算に関するメモ–2003.12.3–

1　一般的なこと

1.1　序

$$f(x) = \sum_{n=0}^{2N+1} c_n x^n \tag{1}$$

$$f(x) \sim \frac{a_0 + a_1 x + a_2 x^2 + ...a_N x^N}{1 + b_1 x + b_2 x^2 + ...b_N x^N} \tag{2}$$

diagonal Pade approximant, つまり $[N|N]$pade 近似の数値計算プログラムに関するメモである。「**注意：**すべての過程を programing すると大変そうだが、その必要はない。SSL2 や Numerical Recipes などのライブラリーを使って組み立てればよい。」　と書いたように、ライブラリーを組み込んで Pade 近似を計算し、その分子、分母の零点を計算する program である。ここでの使用言語は Fortrun であり、Numerical Recipes を subroutine として用いる。ただし、ソースファイルを subroutine としてくっ付けてそれを呼んでいる。program に対し、内容がわかるようにコメントを加えるかたちで説明する。参考書は「Numerical Recipes in C」(手元にあるもの)

1.2　反復法

Pade 近似のなかで使用しているのは、LU 分解と反復法である。LU 分解は別ノートで説明したので、反復法の説明のみする。ベクトル x を連立方程式 $A \cdot x = b$ の厳密解とする。数値的に得られた解を $x + \delta x$ とする。ここで、δx は未知の誤差である。A を乗じると、

$$A \cdot (x + \delta x) = b + \delta b \tag{3}$$

となる。つまり、

$$A \cdot \delta x = \delta b. \tag{4}$$

である。この式に、(3) を代入する。

$$A \cdot \delta x = A \cdot (x + \delta x) - b \tag{5}$$

右辺の $x + \delta x$ は数値的に得ているので、この式から誤差である未知量 δx を求める。そして、はじめの数値解 $x + \delta x$ から差っぴくと解が改良される。もちろん、A はＬＵ分解したものを用いると良い。引き算による桁落ちも減らせる。また、そのためには別ノートで書いたように、大きな値の係数がないようにあらかじめ scale しておくことも重要。

1.3 その他

自分で使ってはいないが、海外ではNAG-Libraryが良く使われている。購入してもよいかもしれない。Berrettiらもこれを使っていると思う??その中にPade近似もあり、すぐに使える。説明も丁寧。たとえば、http://www.nag.com/numeric/fl/manual/pdf/E02/e02raf.pdf

2 Program

リストの中にある「!」で始まるところは、コメント行でる。行の途中からも使われている。

```
!     Last change:  HY   30 Nov 2003    4:18 pm
!*********************************
! pade approximation
!*********************************
!   SUBROUTINE pade(cof,n,resid)
!   USES lubksb,ludcmp,mprove     <==この３つのライブラリーをsubとしてつかう
!     Given cof(1:2*n+1), the leading terms in
!     the powerbseries expansion of a function, solve the
!     linear Pade equations to return the coefficients
!     of a diagonal rational function approximation to
!     the same function, namely(cof(1)+cof(2)x+......+cof(n+1)x^N)/
   (1+cof(n+2)x+.....+cof(2*n+1)x^N).
   <===  2*n+1個の一次元配列 cof(1:2*n+1)  が入力の展開係数、
            出力もこれで返すので中身は失われる。必要なら保存せよ
!     The value resid is the the norn of the residual vector;
!     a small value indicates a well-converged solution.


   implicit REAL*8 ( A-H, O-Z )
   PARAMETER(ndim=6000,n21=2*ndim+1)
   PARAMETER (NMAX=1000,BIG=1.E30,n6=6*(ndim+1))
   REAL*8 cof(n21), dcof(5000)
  INTEGER*4 indx(NMAX)
  REAL*8 q(NMAX,NMAX),qlu(NMAX,NMAX),x(NMAX),y(NMAX),z(NMAX)

  REAL*8  xr(NMAX), yr(NMAX),vw(n6)
  COMPLEX*16 zout(ndim)

  do ii=1,nddd
    READ(1,*) ndum, dcof(ii),bum ! the cof.
  enddo
  READ(1,*) gamma
!  WRITE(*,*) nddd,gamma

nnum2=50     <== 分子、分母で求める係数の数
```

nnum1=2*nnum2+1　＜＝＝Pade に使う係数の数

以上で適当な入力がなされたとしてここからが中身
簡単なコメントが行の後ろに与えてある。
短いので、subroutine を使いながら自分で作ったほうがわかりやすい。

```
  do j=1,nnum2                    ! Set up matrix for solving
          x(j)=cof(nnum2+j+1)
          y(j)=x(j)
          do k=1,nnum2
                  q(j,k)=cof(j-k+nnum2+1)
                  qlu(j,k)=q(j,k)
         enddo
  enddo

  call ludcmp(qlu,nnum2,NMAX,indx,d)    !Solve by LU decomposition
                                                ! and backsubstitution.
  call lubksb(qlu,nnum2,NMAX,indx,x)

  rr=BIG
1    continue
      !Important to use iterative improvement, since the the
      !Pede equation tend to be illconditioned.

    rrold=rr
    do j=1,nnum2
          z(j)=x(j)
    enddo

  call mprove(q,qlu,nnum2,NMAX,indx,y,x)  ＜＝これを何度もよび収束するまで improve

    rr=0.

    do j=1,nnum2                ! Calculate residual.
         rr=rr+(z(j)-x(j))**2
    enddo

  if(rr.lt.rrold) goto 1   ! If it is no longer improving, call it quits.
  resid=sqrt(rr)

  do k=1,nnum2                  ! Calculate the remaining coefficiates.
       sumd=cof(k+1)
       do  j=1,k
          sumd=sumd-x(j)*cof(k-j+1)
       enddo
       y(k)=sumd
```

```
  enddo                    ! Copy answers to output.
  do  j=1,nnum2
       cof(j+1)=y(j)        ! cof(1)=dcof(1) remained
       cof(j+nnum2+1)=-x(j)
  enddo
```
ここで program の中心部終了
計算結果は、y と$ｘ$の中に入っていることになる。

```
!**********
! output
!**********
 do ii=1,nnum2
!     WRITE(*,*) ii,y(ii),-x(ii)
     WRITE(3,*) ii,y(ii),-x(ii)   <==ｙが分子、－ｘが分母の多項式の係数.
                    cof の並びと同じく１次ー>N次の順。ただし、a_0=c_0, b_0=1.0
 end do
```

次は、求まった係数を使い、分子、分母の零点計算。
何を用いても良いが、たまたま SSL2 の drjetr という
ライブラリーを呼んで使っている。
多項式の次数と係数を入力すると、複素根を返してくれる。
係数の並べ方にのみ注意が必要。

```
!*********************************
! calculate zeros of the numerator
!*********************************
! reverse order
 do kk=1,nnum2
   yr(nnum2-kk+1)=y(kk)
 end do
!*****
   nm=nnum2
   yr(nm+1)=cof(1)
   CALL drjetr(yr,nm,zout,vw,icon)  ! ssL2
WRITE(*,*)'icon=',icon,'# of roots nm=',nm

WRITE(*,*) 'zeros of the numerator'
WRITE(3,*) 'zeros of the numerator'
do i=1,nnum2
   WRITE(*,*) i, DBLE(zout(i)), aimag(zout(i))
   WRITE(3,*) i, DBLE(zout(i)), aimag(zout(i))
enddo

!*********************************
! calculate poles of the denominator (bunnbo)
!*********************************
```

```
! reverse order
 do kk=1,nnum2
   xr(nnum2-kk+1)=-x(kk)
 end do
!*****
   nm=nnum2
   xr(nm+1)=1.d0
   CALL drjetr(xr,nm,zout,vw,icon)  ! ssL2
WRITE(*,*)'icon2=',icon,'# of roots nm=',nm

WRITE(*,*) 'poles of the denominator'
WRITE(3,*) 'poles of the denominator'
do i=1,nnum2
  WRITE(*,*) i, DBLE(zout(i)), aimag(zout(i))
   WRITE(3,*) i, DBLE(zout(i)), aimag(zout(i))
enddo

112  stop
  END
```

以下は、用いたsubroutineのソースである。
Numerical Recipes in Cに中身の詳しい
説明があるので、ここでは省略する。

```
!***********************************
!     ludcmp.for (numerical recipes)
!***********************************
      SUBROUTINE ludcmp(a,n,np,indx,d)
      implicit REAL*8 ( A-H, O-Z )
      INTEGER*4 indx(n)
      REAL*8 a(np,np)
      PARAMETER (NMAX=500,TINY=1.0e-20)
!      INTEGER i,imax,j,k
      REAL*8 vv(NMAX)
      d=1.
      do 12 i=1,n
        aamax=0.
        do 11 j=1,n
          if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11      continue
!        if (aamax.eq.0.) pause 'singular matrix in ludcmp'
        if (aamax.eq.0.) goto 12
        vv(i)=1./aamax
12    continue
      do 19 j=1,n
```

```
        do 14 i=1,j-1
          sum=a(i,j)
          do 13 k=1,i-1
            sum=sum-a(i,k)*a(k,j)
13        continue
          a(i,j)=sum
14      continue
        aamax=0.
!
        do 16 i=j,n
          sum=a(i,j)
          do 15 k=1,j-1
            sum=sum-a(i,k)*a(k,j)
15        continue
          a(i,j)=sum
          dum=vv(i)*abs(sum)
          if (dum.ge.aamax) then
            imax=i
            aamax=dum
          endif
16      continue
        if (j.ne.imax)then
          do 17 k=1,n
            dum=a(imax,k)
            a(imax,k)=a(j,k)
            a(j,k)=dum
17        continue
          d=-d
          vv(imax)=vv(j)
        endif
        indx(j)=imax
        if(a(j,j).eq.0.)a(j,j)=TINY
        if(j.ne.n)then
!
          dum=1./a(j,j)
          do 18 i=j+1,n
            a(i,j)=a(i,j)*dum
18        continue
        endif
19    continue
      return
      END


!****************************
```

```
! lubksb.for (numerical recipes)
!***************************
      SUBROUTINE lubksb(a,n,np,indx,b)
      implicit REAL*8 ( A-H, O-Z )

      INTEGER*4 indx(n)
      REAL*8 a(np,np),b(n)
!      INTEGER i,ii,j,ll
!      REAL sum
      ii=0
      do 12 i=1,n
        ll=indx(i)
        sum=b(ll)
        b(ll)=b(i)
        if (ii.ne.0)then
          do 11 j=ii,i-1
            sum=sum-a(i,j)*b(j)
11        continue
        else if (sum.ne.0.) then
          ii=i
        endif
        b(i)=sum
12    continue
      do 14 i=n,1,-1
        sum=b(i)
        do 13 j=i+1,n
          sum=sum-a(i,j)*b(j)
13      continue
        b(i)=sum/a(i,i)
14    continue

      return
      END

!*********************
! mprove.for (numerical recipes)
!*********************
      SUBROUTINE mprove(a,alud,n,np,indx,b,x)
      implicit REAL*8 ( A-H, O-Z )

      INTEGER*4 indx(n)
      REAL*8 a(np,np),alud(np,np),b(n),x(n)
      PARAMETER (NMAX=500)
!CU    USES lubksb
!      INTEGER i,j
```

```
      REAL*8 r(NMAX)
      DOUBLE PRECISION sdp
      do 12 i=1,n
        sdp=-b(i)
        do 11 j=1,n
          sdp=sdp+dble(a(i,j))*dble(x(j))
11      continue
        r(i)=sdp
12    continue
      call lubksb(alud,n,np,indx,r)
      do 13 i=1,n
        x(i)=x(i)-r(i)
13    continue
      return
      END
```

References

[1] W. H. Press, S. A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C* (Cambridge University Press, 1988). この手のものは最新版がいいかもしれない。