解析接続の数値計算による Stokes 係数の近似的確かめ

高山信毅

Hankel 関数 $H_\nu^{(2)}(z)$ を上半平面から下半平面へ負の実軸を横切るように解析接続した関数を考える ($\arg z > \pi$ への解析接続). 高野の本 (常微分方程式, 朝倉) によれば, この関数は

$$-(1+e^{2\pi i\nu})T_1(z)+T_2(z)$$

に漸近展開される. $T_1(z)$, $T_2(z)$ の定義は本 p.159 を参照 ($\sqrt{2}$ を右辺にかけておくこと). これらの関数に巾関数があらわれるが, $\arg z > \pi$ であることに注意.

1 準備

数学ソフトを使いこなす第一歩は手計算/自分の知識とソフトの計算が一致する/矛盾しないのを確かめることであろう.

Maple のコマンド `diff`(微分), `dsolve`(微分方程式を解く), `subs`(置き換え) などが自分の知識で想定される答と同一のものを出すかまず確かめる.

```
> diff(cos(2*x),x);
                    -2 sin(2 x)

> diff(cos(2*x),x$2);
                    -4 cos(2 x)

> subs(x=2,x+y);
                     2 + y

> subs({x=2,y=3},x+y);
                     5
```

次に常微分方程式の近似解を求めるコマンドを試してみよう.

```
with(DEtools):
test0x:=proc()
 RETURN( dsolve({diff(y0(x),x)=y1(x), diff(y1(x),x)=-y0(x),y0(0)=1, y1(0)=0},
               numeric, range=0..5) );
end:
```

振動の方程式 $y_0' = y_1, y_1' = -y_0, y_0(0)=1, y_1(0)=0$ ($y_0''+y_0=0, y_0(0)=1, y_0'(0)=0$) を解く. `y0` は $\cos(x)$ に等しいはず.

```
> A:=test0x();
          A := proc(rkf45_x)  ...  end proc

> A(0);
               [x = 0., y0(x) = 1., y1(x) = 0.]

> A(1);
    [x = 1., y0(x) = .540302304342744399, y1(x) = -.841471136026033961]

> evalf(cos(1));
                      .5403023059
```

2　解析接続の数値計算

Hankel 関数 $H_\nu^{(2)}(z)$ が $z=-2+2i$ から $z=-2-2i$ へ解析接続される時の値を調べよう ($\arg z$ が π を越える時の解析接続).

Maple の関数 $x^\alpha = e^{\alpha\log x}$ は, 次のグラフでみるように $-\pi < \arg x \le \pi$ で定義されている (負の実軸を横切るとき不連続点がある).

```
plotsetup(x11):
plot(Im((-2-x*I)^(1/2)),x=-2..2);
```

同じように Maple の関数 `HankelH2` は次のプログラムの出力にみるように $\arg z = \pi$ のところに branch cut がある.

```
with(DEtools):
plotsetup(x11):
h2g:=proc()
  plot(Im(HankelH2(1/3,-2-I*x)),x=-2..2);
end:
```

多くの文献では $H_\nu^{(2)}(z)$ は $-2\pi < \arg z < \pi$ で定義された関数となっているが, Maple の `HankelH2` は $-\pi < \arg z < \pi$ で定義されている. Maple は $-2\pi < \arg z < -\pi$ での値, 別の言い方をすれば Maple はこの領域での解析接続を計算してくれない. 解析接続の近似計算のために Hankel 関数のみたす微分方程式 $z^2 f'' + zf' + (z^2-\nu^2)f = 0$ を $z=-2-xi$ と変数変換した微分方程式を差分法で数値計算する. Maple の差分法の関数 `dsolv, numeric` は独立, 従属変数ともに実数値しか認めないので, f を実部と虚部に分けて, 連立微分方程式を解く. これが関数 `test1()` である. (長いプログラムは テキストエディタで書いて, `read ‘ファイル名‘;` で読み込む. (シングルクオートでなくその逆向きの記号であることに注意.)

```
test0:=proc(Nu,RR)
 local F,F1,F2,G1,G2;
 F:=-(-r-I*x)^2*(p2+I*q2)+(-r-I*x)*I*(p1+I*q1)+((-r-I*x)^2-Nu^2)*(p+I*q);
 F:=subs(r=RR,F);
 F:=expand(F);
 F1:=coeff(F,I,0);
 F2:=coeff(F,I,1);
# print(expand(F-F1-F2*I));
 G1:=solve({F1=0,F2=0},{p2,q2});
 return(G1);
```

```
end:

vv:=proc(Nu,RR,X)
 local Rule,P;
 P:=-RR-I*X;
 Rule:={x=X,
        p=Re(evalf(HankelH2(Nu,P))),
        p1=Re(evalf(subs(x=X,diff(subs(z=-RR-I*x,HankelH2(Nu,z)),x)))),
        p2=Re(evalf(subs(x=X,diff(subs(z=-RR-I*x,HankelH2(Nu,z)),x$2)))),
        q=Im(evalf(HankelH2(Nu,P))),
        q1=Im(evalf(subs(x=X,diff(subs(z=-RR-I*x,HankelH2(Nu,z)),x)))),
        q2=Im(evalf(subs(x=X,diff(subs(z=-RR-I*x,HankelH2(Nu,z)),x$2))))
       };
 return(Rule);
end:

test1:=proc()
 local Eqs,G1,Ans,P,Nu,RR,PP,Rule;
 Nu:=1/3;
 RR:=2;
 PP:=-2;
 P:=-RR-PP*I;
 Rule:=vv(Nu,RR,PP);
 G1:=test0(Nu,RR);
 G1:=subs(p=f0(x),p1=f1(x),q=g0(x),q1=g1(x),G1);
 print(G1);
 Eqs:={diff(f0(x),x)=f1(x),diff(f1(x),x)=p2,
       diff(g0(x),x)=g1(x),diff(g1(x),x)=q2,
       f0(PP)=subs(Rule,p),f1(PP)=subs(Rule,p1),
       g0(PP)=subs(Rule,q),g1(PP)=subs(Rule,q1)
      };
 Eqs:=subs(G1,Eqs);
 print(Eqs);
 Ans:=dsolve(Eqs,numeric,range=-2..2);
 return(Ans);
end:
```

関数 `test1()` の戻り値の `g0` が $H_\nu^{(2)}(-2-xi)$ の虚部. ちなみに `g1` は, これの x についての微分.

```
A:=test1();
A(-2);
   3.03
A(-1);
   1.28
A(0);
   0.21
```

ここまでは, 組み込みの値と等しい. たとえば

```
evalf(Im(HankelH2(1/3,-2+2*I)));
                                3.039689313
```

として確かめることができる. ここから先の値が解析接続の値である.

```
A(0.1);
   0.11
A(0.2);
   0.009
A(0.5);
   -0.31
A(2);
   -3.0473
```

補足. プログラムを正しく動かすまでには, bug とりに苦労した. たとえば, 最初は方程式の形, 初期値の取り方で間違いをした. 間違い発見に使ったプログラムは以下のとおり.

```
## test0 で導出した方程式が正しいか数値的に調べる. debug 方法 1.
test0a:=proc()
  local A,R,RR,Nu;
  Nu:=1/3;
  RR:=2;
  A:=test0(Nu,RR);  # p,q,p1,q1,p2,q2 の関係式を求める.
  R:=vv(Nu,RR,-2);  # p,q,p1,q1,p2,q2 の -RR+2*I での値を求める.
  print(A);
  print(R);
  return(subs(R,A));
end:
## 11.11  9:46  右=左 を戻すようになった. OK.
```

3　漸近展開と Stokes 係数の公式を数値的に確かめたい

```
hc:=proc(Nu,z)
  local V;
  V:=evalf((1+exp(2*Pi*I*Nu))*HankelH1(Nu,z)-HankelH2(Nu,z));
  return(V);
end:

# 漸近展開と Stokes 係数の公式を使って arg z > pi での値を計算 .    Maple では
# -pi < arg z <= pi なので, argz >= pi の公式のために補正 exp(-2 pi i /2)=-1 を
入れる.
hc0:=proc(Nu,z)
  local V;
  V:=evalf((1+exp(2*Pi*I*Nu))*hc1(Nu,z,10)-hc2(Nu,z,10));
  return(V);
end:

## HankelH1  |Z|>=10, N=10
hc1:=proc(Nu,Z,N)
  local V,V0,i;
  V0:=evalf((2/(Pi*Z))^(1/2)*exp(I*(Z-Pi*Nu/2-Pi/4)));
  V:=1;
  for i from 1 to N do
    V:=V+evalf(pochhammer(1/2-Nu,i)*pochhammer(1/2+Nu,i)/
        ((i!)*(2*I*Z)^i));
  od:
  return (V0*V);
end:

## HankelH2  |Z|>=10, N=10
hc2:=proc(Nu,Z,N)
  local V,V0,i;
  V0:=evalf((2/(Pi*Z))^(1/2)*exp(-I*(Z-Pi*Nu/2-Pi/4)));
  V:=1;
  for i from 1 to N do
    V:=V+evalf((-1)^i*pochhammer(1/2-Nu,i)*pochhammer(1/2+Nu,i)/
        ((i!)*(2*I*Z)^i));
  od:
  return (V0*V);
end:
```

```
>read `han.ml`;
>A:=test1();
> A(2);
[x = 2., f0(x) = -1.51328441042763684, f1(x) = -1.48735192378470416,

    g0(x) = -1.11336029731131414, g1(x) = -1.01637269155876275]

> hc0(1/3,-10-2*I);
                                  -1.513285354 - 1.113360632 I
```

一致． Stokes 係数の公式は正しそうだ．

しかし

```
> hc(1/3,-10-2*I);
                                  -1.546315208 - 1.120338558 I
```

と Hankel 関数を使う方は一致せず．謎... (解決編を参照).

[この section の残りの部分は失敗の記録]. ちなみに

```
> evalf(HankelH2(1/3,-10-I*2));
                         .03302985750 + .006977927300 I

> evalf(HankelH1(1/3,-10-I*2));
                        -1.720841265 + .7538632410 I
> evalf((1+exp(2*Pi*I*1/3))*HankelH1(1/3,-10-I*2));
                        -1.513285350 - 1.113360631 I
```

$(1+e^{2\pi i\nu})H_\nu^{(1)}(z)$ で見ると値がなぜかほぼ一致.

hc1 と hc2 の値と HankelH1 と HankelH2 を比べるとさらに謎が...

下半平面では不一致． -100-2*I で比べても 10^(-2)..10^(-3) の誤差あり．

```
> hc1(1/3,-10-2*I,10);
                                 -1.698283277 + .7287475124 I


> evalf(HankelH1(1/3,-10-2*I));
                                 -1.720841265 + .7538632410 I
```

(Matlab でもつぎのように計算できる (Maple を疑った...).

```
>> besselh(1/3,1,-10-2*i)
ans =
  -1.7208 + 0.7539i
```

)

上半平面では近似がよい．

```
> hc1(1/3,-10+2*I,10);
                                .03302985745 - .006977927375 I


> evalf(HankelH1(1/3,-10+2*I));
                                .03302985750 - .006977927300 I
```

下半平面でも実軸から離れていれば近似はよさそう.

```
> evalf(HankelH1(1/3,-10-10*I));
              -3579.298446 + 3029.928829 I

> hc1(1/3,-10-10*I,10);
              -3579.298438 + 3029.928822 I

すこし怪しくなってくる??
> evalf(HankelH1(1/3,-10-5*I));
                -30.46468777 + 18.24056288 I

> hc1(1/3,-10-5*I,10);
                -30.46378268 + 18.23923963 I
```

4 解決編

Stokes 現象の公式の $H_\nu^{(1)}(z)$ は $\arg z > \pi$ での値なので, Maple の `HankelH1` は直接使ってはいけない. (`testh1()` のソースは `han.ml` を参照).

```
> read `han.ml`;
> A:=test1h1();    # test1h1() は負の実軸を越える解析接続を計算. f0+g0*I=H1
> A(2);
[x = 2., f0(x) = 1.69828092354717474, f1(x) = 1.64786670065829410,

    g0(x) = -.728745563793356200, g1(x) = -.804190807042612920]

> -hc1(1/3,-10-2*I,10);
-hc1(1/3,-10-2*I,10);
          1.698283277 - .7287475124 I

> evalf(HankelH1(1/3,-10-2*I)); # 符号を変えても一致せず.
evalf(HankelH1(1/3,-10-2*I));
          -1.720841265 + .7538632410 I
```

なお $H_\nu^{(1)}(z)$ の漸近展開は $-\pi < \arg z < \pi$ で `hc1`, $\pi < \arg z < 2\pi$ で `-hc1` であるが, $-\pi < \arg z < 0$ ではあまり近似がよくないようだ.