On Numerical Calculation of the Plasma Dispersion Function

Hua-sheng XIE (谢华生)
huashengxie@gmail.com
Department of Physics, Institute for Fusion Theory and Simulation,
Zhejiang University, Hangzhou 310027, P.R.China

First, 2010-11-18; Update, 2011-10-09.

Abstract

Numerical calculation of the plasma dispersion function (PDF) $Z(\zeta)$ using different methods and the comparison with Fried and Conte's book [Fried1961] is discussed or listed. The application to get the exact solution of dispersion relation is also mentioned. The PDF is well-known in the plasma community. But, it seems that there is no enough comprehensive discussion of it in the literature[1]. **Especially, people are easily make mistake when treat PDF at** $|y| \le 0$ **plane, and then may get imprecise or incorrect answers.** The purpose of this document is to provide an overview of PDF, and also how to calculate and apply it. But, one should note that this is still not an enough comprehensive one[2].

y<=0

[1] [Fried1961] is too old and full of typo errors, which need be pointed out. See also the untraceable erratum by Y. L. L(1963) and Fettis(1972).
[2] If you have any suggestions or find mistakes in this document, please email me, thanks!

1 Introduction[3]

In the theory of linearized waves or oscillations in a hot plasma, with or without a magnetic field, a certain function of complex argument, which we call the ***plasma dispersion function***, occurs repeatedly whenever the unperturbed velocity distribution is taken to be Maxwellian (i.e., Gaussian). The function may be defined as

$$Z(\zeta) = \pi^{-1/2} \int_{-\infty}^{\infty} dx \exp(-x^2) / (x - \zeta), \quad \mathrm{Im}\,\zeta > 0 \tag{0.1}$$

and as the analytic continuation[4] of this for $\mathrm{Im}\,\zeta \le 0$. The alternative representation

$$\begin{aligned} Z(\zeta) &= \pi^{-1/2} \int_C dx \exp(-x^2) / (x - \zeta) \\ &= 2i \exp(-\zeta^2) \int_{-\infty}^{i\zeta} \exp(-t^2) dt \\ &= i\pi^{1/2} \exp(-\zeta^2)[1 + erf(i\zeta)], \end{aligned} \tag{0.2}$$

is valid for either sign of $\mathrm{Im}\,\zeta$ and, in addition, shows that $Z(\zeta)$ is closely related to the

[3] Part of the descriptions here is rewritten directly from [Fried1961].

[4] Actually, the analytic continuation is related to the so called Landau contour [Landau1946] 'C' in the integral of (0.2), which is also called as causality condition. Many books have discussed the Landau contour for analytic continuation, **but most of them are not clearly or even wrong!!!** However, the best reference I found is [Nicholson1983], chapter 6 and appendix C, which is correct and easy to understand. Please note that the analytic continuation is not a trivial problem. That is why Vlasov, as a mathematician, also made mistake here.

error function.

In plasma applications, the variable $\zeta = x + iy$ has the significance of the ratio of phase velocity of the wave to some thermal velocity,

$$\zeta = \omega / ka, \tag{0.3}$$

where ω and k are the frequency and wave number of a wave and a is the thermal velocity of the particles. For waves which are either damped or unstable, ω will be complex. Comparing with Fadeeva function [Fadeeva1954], we get

Faddeeva

$$w(\zeta) = Z(\zeta) / i\pi^{1/2}. \tag{0.4}$$

2 Properties of PDF

2.1 Symmetry properties

$$\begin{aligned} Z(\zeta^*) &= -[Z(-\zeta)]^*, \\ Z(\zeta^*) &= [Z(\zeta)]^* + 2i\pi^{1/2} \exp[-(\zeta^*)^2] \quad (y > 0). \end{aligned} \tag{0.5}$$

The asterisk (*) denotes complex conjugation.

2.2 Differential equations

$$\begin{aligned} Z' &= -2(1 + \zeta Z), \\ Z(0) &= i\pi^{1/2}. \end{aligned} \tag{0.6}$$

2.3 Values for special arguments

Real argument

$$Z(x) = \exp(-x^2)[i\pi^{1/2} - 2\int_0^x \exp(t^2)dt]. \tag{0.7}$$

Imaginary argument

$$Z(iy) = i\pi^{1/2} \exp(y^2)[1 - erf(y)]. \tag{0.8}$$

Modulus 45°

$$Z[\rho \exp(-\pi i / 4)] = i\pi^{1/2} \exp(i\rho^2)\{1 + (2i)^{1/2}[C(\rho^2) - iS(\rho^2)]\}, \tag{0.9}$$

Where C and S are the Fresnel functions

$$C(x) + iS(x) = \int_0^x \exp(\pi i t^2 / 2)dt. \tag{0.10}$$

2.4 Power series

$$\begin{aligned} Z(\zeta) &= i\pi^{1/2} \exp(-\zeta^2) - 2\zeta[1 - 2\zeta^2 / 3 + 4\zeta^4 / 15 - 8\zeta^6 / 105 + \cdots] \\ &= i\pi^{1/2} \exp(-\zeta^2) - \zeta \sum_{n=0}^{\infty} (-\zeta^2)^n \pi^{1/2} / (n + 1/2)!. \end{aligned} \tag{0.11}$$

2.5 Asymptotic expansion

$$\begin{aligned} Z(\zeta) &\simeq i\pi^{1/2}\sigma\exp(-\zeta^2) - \zeta^{-1}[1+1/2\zeta^2+3/4\zeta^4+\cdots] \\ &= i\pi^{1/2}\sigma\exp(-\zeta^2) - \sum_{n=0}^{\infty}\zeta^{-(2n+1)}(n-1/2)!/\pi^{1/2}, \end{aligned} \tag{0.12}$$

Where [Huba2009],

$$\sigma = \begin{cases} 0 & y > |x|^{-1} \\ 1 & |y| < |x|^{-1} \\ 2 & y < -|x|^{-1} \end{cases}, \tag{0.13}$$

Or, for simplification [Fried1961],

$$\sigma = \begin{cases} 0 & y > 0 \\ 1 & y = 0 \\ 2 & y < 0 \end{cases}. \tag{0.14}$$

2.6 Two-pole approximations

Good for ζ in upper half plane except when $y < \pi^{1/2}x^2\exp(-x^2),\ x \gg 1$ [Fried1968]:

$$\begin{aligned} Z(\zeta) &\approx \frac{0.50+0.81i}{a-\zeta} - \frac{0.50-0.81i}{a^*-\zeta}, \quad a = 0.51-0.81i; \\ Z'(\zeta) &\approx \frac{0.50+0.96i}{(b-\zeta)^2} - \frac{0.50-0.96i}{(b^*-\zeta)^2}, \quad b = 0.48-0.91i. \end{aligned} \tag{0.15}$$

3 Table Generation and Accuracy[5]

3.1 The continued fraction

Several methods have been proposed for computing the error function in various regions of the complex ζ plane [Salzer1951]. For small values of y, numerical integration of the differential equation (0.6) is both accurate and convenient. For large values of y, and especially along the positive imaginary axis, this method is unsatisfactory because of the accumulation of truncation and round-off errors. In this range, a continued fraction based on the asymptotic expansion of $Z(\zeta)$ was derived and proved to be completely satisfactory. The continued fraction is an analytic continuation of the asymptotic expansion for $Z(\zeta)$ and is most easily derived using the quotient difference algorithm [Henrici]. The continued fraction has the form (the typo have been corrected here and after)

[5] From [Fried1961]. One can also refer [Cuyt2008] for the detailed derivation of the continued fraction.

$$Z(\zeta)=\cfrac{\zeta}{-\zeta^2+\cfrac{1}{2}+\cfrac{(-1)(1/2)}{-\zeta^2+\cfrac{5}{2}+\cfrac{(-2)(3/2)}{-\zeta^2+\cfrac{9}{2}+\cfrac{-a_{n+1}}{b_{n+1}+\cfrac{-a_{n+2}}{\cdots}}}}}, \tag{0.16}$$

Where in general,

$$\begin{aligned} a_{n+1}&=n(2n-1)/2, \quad n=1,2,\cdots \\ b_{n+1}&=-\zeta^2+1/2+2n, \quad n=0,1,\cdots \\ a_1&=-\zeta. \end{aligned} \tag{0.17}$$

The continued fraction is evaluated by the recursion relations

$$\begin{aligned} A_{n+1}&=b_{n+1}A_n-a_{n+1}A_{n-1}, \\ B_{n+1}&=b_{n+1}B_n-a_{n+1}B_{n-1}, \\ A_{-1}&=1, \quad A_0=0, \quad B_{-1}=0, \quad B_0=1, \end{aligned} \tag{0.18}$$

And

$$Z(\zeta)=\lim_{n\to\infty}A_n/B_n, \quad y>0. \tag{0.19}$$

3.2 Some comments

This continued fraction representation (0.16) converges in the entire complex plane except for points on the real axis. Near the real axis, the number of terms required for convergence increases, and maintenance of accuracy was difficult, owing to underflow and overflow. Accordingly, in the region $|y|\le 1,\ 0\le x\le 10$, the entries are better found by numerical integration of the differential equation (0.6).

4 Numerical Calculations

4.1 A glimpse

Using (0.1), we can write a MATLAB code like this[6]

```
function out=Z(zeta)
syms x
format long
out=double(sqrt(1.0/pi)*int(exp(-x^2)/(x-zeta),-inf,inf));
```

Typing >>Z(1+0.1*i), we get

```
>>  Z(1+0.1*i)
Warning: Explicit integral could not be found.
ans =
  -0.954563543114130 + 0.661426866417288i
```

In [Fried1961] the value is Z(1+0.1*i) = -0.954564 + 0.661427i, they consist with each other.

[6] Note: only suit for $y>0$.

4.2 The continued fraction

In 1961, to get the value of PDF is not easy. Fried and Conte were famous by using an IBM 709 computer calculating a PDF table based on (0.6) and (0.16). The output of that machine formed the majority of their book, which is a very useful book for plasma physicists for decades and even now.

Using (0.19), we write a MATLAB code as below

```
% Erratum: Math. Comp. v. 26 (1972), no. 119, p. 814.
clear;clc;
format long
zeta=1+0.1*i;
N_max=100;
a(1)=-zeta;b(1)=-zeta*zeta+0.5;
a(2)=0.5;b(2)=-zeta*zeta+0.5+2;
A_1=1.0;A0=0.0;B_1=0.0;B0=1.0;
A(1)=b(1)*A0-a(1)*A_1;B(1)=b(1)*B0-a(1)*B_1;
A(2)=b(2)*A(1)-a(2)*A0;B(2)=b(2)*B(1)-a(2)*B0;
for n=2:N_max
    a(n+1)=n*(2*n-1)/2;
    b(n+1)=-zeta*zeta+0.5+2*n;
    A(n+1)=b(n+1)*A(n)-a(n+1)*A(n-1);
    B(n+1)=b(n+1)*B(n)-a(n+1)*B(n-1);
end
Zn=A(n)/B(n)
```

For N_max=100, zeta=1.0+0.1*i ($|y|=0.1<1.0$),

```
Zn =
   -0.933741777135722 + 0.674209328325076i
```

The difference between this and [Fried1961] (-0.954564 + 0.661427i) is about 2%.

N_max=150,

```
Zn =
   -0.964276252648030 + 0.660158921452315i
```

N_max=200,

```
Zn =
                NaN +                NaNi
```

These results mean that the continued fraction (0.16) form is not suitable for $|y|=0.1<1.0$.

For N_max=100, zeta=9.8+10.0*i ($|y|=10.0>1.0$),

```
Zn =
   -0.049856227146091 + 0.051133797423976i
```

This is the same as [Fried1961] (-0.0498562+0.0511338i).

For N_max=100, zeta=9.8-10.0*i

```
Zn =
```

```
-0.049856227146091 - 0.051133797423976i
```

This means the continued fraction (0.16) form has not considered the $i\pi^{1/2}\sigma\exp(-\zeta^2)$ term for $y<0$ plane. So, one should cautious about this and add this extra.

4.3 Another recurrence relation

We define

$$Z^{(n+1)}=\cfrac{\zeta}{-\zeta^2+\cfrac{1}{2}+\cfrac{(-1)(1/2)}{-\zeta^2+\cfrac{5}{2}+\cfrac{(-2)(3/2)}{-\zeta^2+\cfrac{9}{2}+\cfrac{-a_{n+1}}{b_{n+1}}}}}=\cfrac{a_1}{b_1-\cfrac{a_2}{b_2-\cfrac{a_3}{b_3-\cdots\cfrac{a_{n+1}}{b_{n+1}}}}} \tag{0.20}$$

Then[7]

$$\begin{aligned} Z^{(1)}&=a_1/b_1,\\ Z^{(2)}&=\frac{a_1}{b_1-a_2/b_2},\\ &\cdots \end{aligned} \tag{0.21}$$

We can also get another recursive relation from (0.16) based on (0.20)

$$Z^{(n+1)}=\frac{X_1^{(n+1)}}{Y_1^{(n+1)}}=\cfrac{a_1}{b_1-\cfrac{X_2^{(n+1)}}{Y_2^{(n+1)}}}=\cdots=\cfrac{\cdots}{b_{k-1}-\cfrac{a_k}{b_k-\cfrac{X_{k+1}^{(n+1)}}{Y_{k+1}^{(n+1)}}}}, \tag{0.22}$$

Gives,

$$X_k^{(n+1)}=a_kY_{k+1}^{(n+1)},\quad Y_k^{(n+1)}=b_kY_{k+1}^{(n+1)}-X_{k+1}^{(n+1)}, \tag{0.23}$$

And

$$X_{n+1}^{(n+1)}=a_{n+1},\quad Y_{n+1}^{(n+1)}=b_{n+1},\quad k=n,n-1,\cdots,1. \tag{0.24}$$

The MATLAB code is

```
clear;clc;
zeta=1.0+0.1*i;
n_max=160; % too small then inaccuracy, too large then NaN
for k=n_max:-1:0
    a(k+1)=k*(2*k-1)/2;
    b(k+1)=-zeta*zeta+0.5+2*k;
end
a(1)=zeta;
x(n_max+1)=a(n_max+1);
```

[7] Here, $a_1=\zeta$.

```
y(n_max+1)=b(n_max+1);
for k=n_max:-1:1
    x(k)=a(k)*y(k+1);
    y(k)=b(k)*y(k+1)-x(k+1);
end
z=x(1)/y(1)
% result, z = -0.952047313017375 + 0.653637452593423i
% exact,  z = -0.954563543114130 + 0.661426866417288i
% not exactly
```

For n_max=160, zeta=1.0+0.1*i ($|y| = 0.1 < 1.0$),

```
z =
  -0.952047313017375 + 0.653637452593423i
```

this result is not accurate, the relative error is about $10^{-3} - 10^{-2}$.

For n_max=100, zeta=9.8+10.0*i ($|y| = 10.0 > 1.0$),

```
z =
  -0.049856227146091 + 0.051133797423976i
```

This is the same as [Fried1961] (-0.0498562+0.0511338i).

This means the recursion relations (0.22) - (0.24) are indeed valid.

5 Pade Approximation[8]

The Pade method was first used to approximate the plasma dispersion function by Martin and Gonzales [Martin1979], and their results were generalized by [Martin1980] and [Nemeth1981]. The basic theory of Pade approximants can be found in the book [Baker1975].

Following [Martin1980], we consider approximations $Z_A(s)$ for the plasma dispersion function in the form

$$Z_A(s) = \frac{P^{L-1}(s)}{Q^L(s)} = \sum_{l=1}^{L} \frac{b_l}{s - c_l}, \tag{0.25}$$

Where

$$P^{L-1}(s) = \sum_{l=0}^{L-1} p_l s^l, \quad Q^L(s) = 1 + \sum_{l=1}^{L} q_l s^l. \tag{0.26}$$

Inserting the convergent power series,

$$Z(s) = i\sqrt{\pi} - 2s - i\sqrt{\pi} s^2 + \frac{4}{3} s^3 + \frac{i}{2}\sqrt{\pi} s^4 - \frac{8}{15} s^5 + \cdots \tag{0.27}$$

In the equation

[8] From [Ronnmark1982]. I keep his notations and change little here. Equation (0.15) is another kind of approximation using poles.

$$Z(s)Q^L(s) = P^{L-1}(s), \tag{0.28}$$

And identifying coefficients of equal powers of s we obtain a set of equations

$$\begin{cases} i\sqrt{\pi} = p_0, \\ -2 + i\sqrt{\pi}q_1 = p_1, \\ -i\sqrt{\pi} - 2q_1 + i\sqrt{\pi}q_2 = p_2, \\ 4/3 - i\sqrt{\pi}q_1 - 2q_2 + i\sqrt{\pi}q_3 = p_3, \\ \cdots. \end{cases} \tag{0.29}$$

Here, we take $p_l = 0$ if $l > L-1$ and $q_l = 0$ if $l > L$.

An alternative set of equations is obtained by inserting the asymptotic series

$$Z(s) \simeq -s^{-1} - \frac{1}{2}s^{-3} - \frac{3}{4}s^{-5} - \frac{15}{8}s^{-7} - \cdots, \tag{0.30}$$

In (0.28):

$$\begin{cases} -q_L = p_{L-1}, \\ -q_{L-1} = p_{L-2}, \\ -q_{L-2} - 1/2q_L = p_{L-3}, \\ -q_{L-3} - 1/2q_{L-1} = p_{L-4}, \\ \cdots. \end{cases} \tag{0.31}$$

Since we need $2L$ equations to determine all the $p:s$ and $q:s$, we let $J+K=2L$ and choose J equations from (0.29) and K equations from (0.31). The resulting approximant will satisfy

$$\left|Z_A(s) - Z(s)\right| = \begin{cases} O(s^J), s \to 0, \\ O(s^{-K}), s \to = \infty. \end{cases} \tag{0.32}$$

Alternatively, we could have started from the second form of (0.25) and expanded

$$Z_A(s) = \sum_{l=1}^{L} b_l \begin{cases} -\frac{1}{a_l} - \frac{s}{a_l^2} - \frac{s^2}{a_l^3} - \frac{s^3}{a_l^4} + \cdots, & s \to 0, \\ s^{-1} + c_l s^{-2} + c_l^2 s^{-3} + c_l^3 s^{-4} + \cdots, & s \to \infty. \end{cases} \tag{0.33}$$

Comparison with (0.27) and (0.30) leads to the equations

$$\begin{cases} \sum_{l=1}^{L} \frac{b_l}{c_l} = -i\sqrt{\pi}, \\ \sum_{l=1}^{L} \frac{b_l}{c_l^2} = 2, \\ \sum_{l=1}^{L} \frac{b_l}{c_l^3} = i\sqrt{\pi}, \\ \cdots. \end{cases} \tag{0.34}$$

And

$$\begin{cases} \sum_{l=1}^{L} b_l = -1, \\ \sum_{l=1}^{L} b_l c_l = 0, \\ \sum_{l=1}^{L} b_l c_l^2 = -1/2, \\ \sum_{l=1}^{L} b_l c_l^3 = 0, \\ \cdots. \end{cases} \tag{0.35}$$

In practice, the most convenient way to derive the partial fraction expansion of the approximant is to eliminate the $p:s$ from L of the Equations (0.29) and (0.31) and determine the $q:s$ from these. The equation $Q^L(s)=0$ is then solved for the chosen from (0.34) and (0.35) to determine the $b:s$.

Following this procedure, an eight-pole approximant was derived using ten equation form (0.29) and six equations from (0.31). The values of the coefficients are listed in `ZetaPade.m`, where also,

$$\begin{aligned} b_2 = b_1^*,\ b_4 = b_3^*,\ b_6 = b_5^*,\ b_8 = b_7^*, \\ c_2 = -c_1^*,\ c_4 = -c_3^*,\ c_6 = -c_5^*,\ c_8 = -c_7^*. \end{aligned} \tag{0.36}$$

In the upper half of the s plane the accuracy of this approximant should be sufficient for all purposes. However, for $\mathrm{Im}\, s < 0$ the errors increase as s approaches the poles c_l, and when s is large the omitted exponential term $-i2\pi^{1/2}\exp(-s^2)$ in the asymptotic series for $\mathrm{Im}\, s < 0$ may become important.

MATLAB code:

```
% ZetaPade.m, Pade approximation for plasma dispersion function Z(s)
function out=ZetaPade(z,method)
% [1] Ronnmark1982, WHAMP report & xsi.f
% [2] Martin et al 1979, 1980
% method=1; % method for expand to Im(z)<=0
b=[ -1.734012457471826E-2-4.630639291680322E-2i;
    -1.734012457471826E-2+4.630639291680322E-2i;
    -7.399169923225014E-1+8.395179978099844E-1i;
    -7.399169923225014E-1-8.395179978099844E-1i;
    5.840628642184073+9.536009057643667E-1i;
    5.840628642184073-9.536009057643667E-1i;
    -5.583371525286853-1.120854319126599E1i;
    -5.583371525286853+1.120854319126599E1i];
```

```
c=[ 2.237687789201900-1.625940856173727i;
    -2.237687789201900-1.625940856173727i;
    1.465234126106004-1.789620129162444i;
    -1.465234126106004-1.789620129162444i;
    .8392539817232638-1.891995045765206i;
    -.8392539817232638-1.891995045765206i;
    .2739362226285564-1.941786875844713i;
    -.2739362226285564-1.941786875844713i];
sqrtpi=1.772453850905516;
zc=conj(z);x=real(z);y=imag(z);
switch method
    case 1 % with correction for Im(z)<0 and Abs(Im(z))<1.0/Abs(Re(z))
        if(y>1.0/abs(x))
            sigma=0;
        elseif(abs(y)<1.0/abs(x))
            sigma=1;
        else
            sigma=2;
        end
        out=sum(b./(z-c))+sigma*1i*sqrtpi*exp(-z*z);
    case 2 % using Z(x-iy)=[Z(x+iy)]©~+2 i sqrt(¦Ð) exp(-(x-iy)^2), y>0
        if(y<0) % with correction for Im(z)<0
%           out=conj(sum(b./(zc-c)))+2.0i*sqrtpi*exp(-z*z); % seems
better?
            out=sum(b./(z-c))+2.0i*sqrtpi*exp(-z*z);
        else
            out=sum(b./(z-c));
        end
    otherwise % directly, without correction for Im(z)<0
        out=sum(b./(z-c));
end
```

Call `ZetaPade.m`:

```
format long;
z=9.8+10.0i;
ZetaPade(z,0) % without correction for Im(z)<0
ZetaPade(z,1) % with correction for Im(z)<0
ZetaPade(z,2) % with correction for Im(z)<0 and |y|<1.0/|x|
```

Results

```
ans =
 -0.049856227230207 + 0.051133797504614i
ans =
 -0.049856227230207 + 0.051133797504614i
ans =
 -0.049856227230207 + 0.051133797504614i
```

The value of [Fried1961] is Z(9.8+10.0i)=-0.0498562+0.0511338i, exactly the same at the given accuracy.

```
format long;
z=9.8-10.0i;
ZetaPade(z,0) % without correction for Im(z)<0
ZetaPade(z,1) % with correction for Im(z)<0
ZetaPade(z,2) % with correction for Im(z)<0 and |y|<1.0/|x|
```

Results

```
ans =
 -0.049856225480352 - 0.051133795382389i
ans =
 -1.747614631080070e+002 +6.363268853831691e+001i
ans =
 -1.747614631080070e+002 +6.363268853831691e+001i
```

The value of [Fried1961] is Z(9.8-10.0i)=-0.174762E03+0.636333E02i. We can see here, a correction for Im(z)<0 is a must. The difference between Pade approximation (with correction) and [Fried1961] is about 10^{-5}[9].

We should also point here: to calculate the exponential term is very slow compared with calculate the rational function, then the scheme (**WHAMP**[10]) in [Ronnmark1982] ignores the exponential term (the analytic continuation term). So, the code WHAMP is not valid for finding roots of heavy damping[11]. However, WHAMP is really a nice code which calculates general wave dispersion relation in homogeneous anisotropic multicomponent magnetized plasma. I am glad to recommend it here.

6 Faddeeva Function

Considering the relation (0.4), we can using Faddeeva function or the complex error function to calculate the plasma dispersion function.

Here is a MATLAB version of Faddeeva function[12], faddeeva.m, which is based on [Weideman1994] by FFT

```
function w = faddeeva(z,N)
% FADDEEVA  Faddeeva function
%   W = FADDEEVA(Z) is the Faddeeva function, aka the plasma dispersion
%   function, for each element of Z. The Faddeeva function is defined
as:
%     w(z) = exp(-z^2) * erfc(-j*z)
%
%   where erfc(x) is the complex complementary error function.
%
```

[9] Which is more accurate to the real value?

[10] https://launchpad.net/whamp

[11] But, for |Im(s)|<0.5|Re(s)|, the relative error is less than 2%. See [Ronnmark1982] for details.

[12] I get it from http://www.mathworks.com/matlabcentral/fileexchange/22207-faddeeva-function-fft-based

```
%   W = FADDEEVA(Z,N) can be used to explicitly specify the number of
terms
%   to truncate the expansion (see (13) in [1]). N = 16 is used as default.
%
%   Example:
%       x = linspace(-10,10,1001); [X,Y] = meshgrid(x,x);
%       W = faddeeva(complex(X,Y));
%       figure;
%       subplot(121); imagesc(x,x,real(W)); axis xy square; caxis([-1
1]);
%       title('re(faddeeva(z))'); xlabel('re(z)'); ylabel('im(z)');
%       subplot(122); imagesc(x,x,imag(W)); axis xy square; caxis([-1
1]);
%       title('im(faddeeva(z))'); xlabel('re(z)'); ylabel('im(z)');
%
%   Reference:
%   [1] J.A.C. Weideman, "Computation of the Complex Error Function,"
SIAM
%       J. Numerical Analysis, pp. 1497-1518, No. 5, Vol. 31, Oct., 1994
%       Available Online: http://www.jstor.org/stable/2158232

if nargin<2, N = []; end
if isempty(N), N = 16; end

w = zeros(size(z)); % initialize output

%%%%%
% for purely imaginary-valued inputs, use erf as is if z is real
idx = real(z)==0; %
w(idx) = exp(-z(idx).^2).*erfc(imag(z(idx)));

if all(idx), return; end
idx = ~idx;

%%%%%
% for complex-valued inputs

% make sure all points are in the upper half-plane (positive imag. values)
idx1 = idx & imag(z)<0;
z(idx1) = conj(z(idx1));

M = 2*N;
M2 = 2*M;
k = (-M+1:1:M-1)'; % M2 = no. of sampling points.
```

```
L = sqrt(N/sqrt(2)); % Optimal choice of L.

theta = k*pi/M;
t = L*tan(theta/2); % Variables theta and t.
f = exp(-t.^2).*(L^2+t.^2);
f = [0; f]; % Function to be transformed.
a = real(fft(fftshift(f)))/M2; % Coefficients of transform.
a = flipud(a(2:N+1)); % Reorder coefficients.

Z = (L+1i*z(idx))./(L-1i*z(idx));
p = polyval(a,Z); % Polynomial evaluation.
w(idx) = 2*p./(L-1i*z(idx)).^2 + (1/sqrt(pi))./(L-1i*z(idx)); %
Evaluate w(z).

% convert the upper half-plane results to the lower half-plane if
necesary
w(idx1) = conj(2*exp(-z(idx1).^2) - w(idx1));
```

And the original version in [Weideman1994] is `cef.m`

```
function w = cef(z,N)
%  Computes the function w(z) = exp(-z^2) erfc(-iz) using a rational
%  series with N terms.  It is assumed that Im(z) > 0 or Im(z) = 0.
%
%                          Andre Weideman, 1995
M = 2*N;  M2 = 2*M;  k = [-M+1:1:M-1]';    % M2 = no. of sampling points.
L = sqrt(N/sqrt(2));                      % Optimal choice of L.
theta = k*pi/M; t = L*tan(theta/2);        % Define variables theta and
t.
f = exp(-t.^2).*(L^2+t.^2); f = [0; f];    % Function to be transformed.
a = real(fft(fftshift(f)))/M2;             % Coefficients of transform.
a = flipud(a(2:N+1));                     % Reorder coefficients.
Z = (L+i*z)./(L-i*z); p = polyval(a,Z);    % Polynomial evaluation.
w = 2*p./(L-i*z).^2+(1/sqrt(pi))./(L-i*z); % Evaluate w(z).
```

Note the comments in the code: Computes the function w(z) = exp(-z^2) erfc(-iz). It is assumed that Im(z) > 0 or Im(z) = 0.

To calculate Z

```
format long;
u=9.8-10.0i;N=50;
Z=faddeeva(u,N)*1i*sqrt(pi) )
Z=cef(u,N)*1i*sqrt(pi)
```

Output

```
Z =
 -1.747614631096728e+002 +6.363268853627531e+001i
Z =
 -0.049853305191875 - 0.051134588824163i
```

The result by `faddeeva.m` is very close to eight-pole Pade approximation result with correction (-1.747614631080070e+002 +6.363268853831691e+001i) of section 5. And the result by `cef.m` is close to the eight-pole Pade approximation result without correction (-0.049856225480352 - 0.051133795382389i). This means this `cef.m` is not suit for calculate the PDF at lower half-plane[13].

If we calculate Z(`9.8+10.0i`), output

```
Z =
 -0.049856227146091 + 0.051133797423976i
Z =
 -0.049856227146091 + 0.051133797423976i
```

Both of them are close to the eight-pole Pade approximation result (-0.049856227230207 + 0.051133797504614i).

7 FORTRAN Codes and with Applications

7.1 Greg Hammett's webpage

Greg Hammett lists some FORTRAN codes for calculating PDF in his webpage[14]. I just copy what he says here. You can download those codes via the links.

> Of the 3 routines here that can be used to evaluate the "Z function", the Plasma Dispersion Function Z(zeta), I'm guessing that the best one is wofz.f (or Zfun.f90, which uses subroutine wofz). At least it is based on the most recently published algorithm. I downloaded wofz.f from http://www.netlib.org/toms/680. It is based on the paper [Poppe1990]:
>
> ```
> The plasma dispersion function Z(z) is related to the w(z) function
> calculated by wofz.f by the simple relationship:
>    Z(z) = i*sqrt(Pi) * w(z)
> w(z) is sometimes called the Voight function or Faddeeva's function,
> and is related to the complementary error function by:
>    w(z) = exp(-z**2)erfc(-iz)
>    w(z) = exp(-z**2)(1 - erf(-iz))
> The plasma dispersion function  Pdf(z) = Z(z) is usually defined as
>    Z(z) = 1/SQRT(Pi) * Integral(-Inf,+Inf) exp(-t**2)/(t-z) dt ,
> (this form is valid only for z in the upper half complex plane...)
> ```
>
> [Poppe1990] discusses various ways in which this algorithm has been made significantly faster than previous algorithms. These algorithms tend to be based on various asymptotic series or continued fraction approximations (chosen to be optimal in various regions of parameter space). **I wonder if it might be possible to develop an even faster algorithm based on rational function approximations that are fit to the exact result. It would require refitting if a different level of accuracy is desired, but might give some speed advantages[15].**

[13] It seems also analytical continuous? But, at least, the analytical continuation should be different from faddeeva function.

[14] http://www.pppl.gov/~hammett/comp/src/

[15] I also wonder this.

--Greg Hammett, Oct. 27, 2006

7.2 Horne's version

This version is from Clare E. J. Watt's code v1d1code.f90[16], which provides a solver for electrostatic dispersion relation. I list the related part here just for easy reading.

```
!*****************************************************************************
      SUBROUTINE fried(zeta,z,zp)
!*****************************************************************************
!
! This subroutine calculates the plasma dispersion function and the
! derivative of the plasma dispersion function wrt zeta. The routine is
! borrowed from Richard Horne's hotray code and modified only slightly to fit
! in with this piece of code.
!
      use nrtype; use parameters
      IMPLICIT NONE
      INTEGER(I4B),PARAMETER::kc = 10
      COMPLEX(DPC),INTENT(IN)::zeta
      COMPLEX(DPC),INTENT(OUT)::z,zp
!
      REAL(DP)::x,y,x1,p_r,p_i,xyz,a,y1,t,ar,ai,ppr,ppi
      REAL(DP),PARAMETER::rpi=1.77245385090552_dp
      REAL(DP),PARAMETER::valmax=80.0_dp
!
      x=dreal(zeta)
      y=dimag(zeta)
      x1=abs(x)
      if(y) 11,10,10
   10 call wz1(x1,y,p_r,p_i)
      if(x) 12,13,13
   11 xyz=y*y-x*x
      if(xyz.ge.zero)then
      a=two*exp(xyz)
      else
      a=zero
      if(abs(xyz).lt.valmax)then
      a=two*exp(xyz)
      end if
      end if
      y1=-y
      t=x1*y1+x1*y1
      ar=a*cos(t)
      ai=a*sin(t)
```

[16] http://www.ualberta.ca/~watt/v1d1code.f90

```
      call wz1(x1,y1,p_r,p_i)
      p_r=-p_r+ar
      p_i= p_i+ai
      if(x) 12,13,13
   12 p_r=p_r
      p_i=-p_i
   13 continue
      a=p_i
      p_i=rpi*p_r
      p_r=-rpi*a
      ppr=-two*(one+x*p_r-y*p_i)
      ppi=-two*(y*p_r+x*p_i)
      z=dcmplx(p_r,p_i)
      zp=dcmplx(ppr,ppi)
      return
      END SUBROUTINE fried
!
!
!*****************************************************************************
      SUBROUTINE wz1(x,y,preel,pimag)
!*****************************************************************************
!
! This subroutine is called in the above fried routine. It is also
! borrowed from Richard Horne's hotray code.
!
      use nrtype; use parameters
      IMPLICIT NONE
      REAL(DP),INTENT(IN)::x,y
      REAL(DP),INTENT(OUT)::preel,pimag
      INTEGER(I4B),PARAMETER::kc = 10
      INTEGER(I4B)::ierr,icapn,nu,ib,nup1,n,np1
      REAL(DP)::s,h,h2,alamda,r1,r2,s1,s2,t1,t2
      REAL(DP)::c,rich1,rich2,x2
      REAL(DP),PARAMETER::coef=0.112837916709551e01_dp
      REAL(DP),PARAMETER::valmax=80.0_dp
!
      ierr=0
      h2=zero
      alamda=zero
!
! if((y.ge.0.429d01).or.(x.ge.0.533d01))go to 1
      if((y.lt.0.429e01_dp).and.(x.lt.0.533e01_dp))then
      s=(0.1e01_dp-y/0.429e01_dp)*sqrt(0.1e01_dp-x*x/0.2841e02_dp)
      h=0.16e01_dp*s
```

```
      h2=h+h
      icapn=int(0.65e01_dp+0.23e0_dp*s)
      alamda=h2**icapn
      nu=int(0.95e01_dp+0.21e02_dp*s)
! go to 2
      else
! continue
!
! Note that h2 and alamda are not defined here so set a flag
! so that if they are used due to rounding errors below a
! message is printed.
! Richard horne 16 Oct 91
!
      ierr=1
      h=zero
      icapn=0
      nu=8
      end if
! continue
      ib=0
      if((h.lt.0.1e-11_dp).or.(alamda.lt.0.1e-11_dp))ib=1
      r1=zero
      r2=zero
      s1=zero
      s2=zero
      nup1=nu+1
      do 3 n=1,nup1
      np1=nup1-n+1
      t1=y+h+dble(np1)*r1
      t2=x-dble(np1)*r2
      c=half/(t1*t1+t2*t2)
      r1=c*t1
      r2=c*t2
      ! if((h.le.zero).or.((np1-1).gt.icapn))go to 3
      if((h.gt.zero).and.((np1-1).le.icapn))then
      t1=alamda+s1
      s1=r1*t1-r2*s2
      s2=r2*t1+r1*s2
      alamda=alamda/h2
      if(ierr.eq.1)then
      PRINT*,' wz1:0: rounding error stopping'
      PRINT*,' wz1:0: rounding error stopping'
      call zexit
      end if
```

```
      end if
    3 continue
      rich1=dble(ib)
      rich2=dble(1-ib)
      pimag=coef*(rich1*r2+rich2*s2)
      if(y.eq.zero) go to 5
      preel=coef*(dble(ib)*r1+dble(1-ib)*s1)
      go to 999
    5 continue
      preel=zero
      x2=x*x
      if(x2.lt.valmax)then
      preel=exp(-x2)
      end if
  999 continue
      return
      END SUBROUTINE wz1
```

7.3 Strangeway's version

I get this version from Zhi WANG's code disfm.f[17], which is a code to solve different dispersion relations of hot plasma. This one should be the same algorithm as Horne's, but should be more clearly.

```
      function plasmaz(z)
c
c Routine to evaluate the plasma Z function.
c
c Written by R.J. Strangeway 11th Jan 1984.
c
c Original algorithm obtained from Phil Pritchett.
c
c Extension to full complex plane included.
c
      complex*16 plasmaz,z
      real*8 x,y,re,im,lambda,h,h2,s,eps,tr,ti,c,rr,ri,sr,si,cc
      integer capn
      logical b
      eps=1.d-12
      x=dabs(dble(z))
      y=dabs(dimag(z))
c
c Modified by RJS 21st Sept 1985 to allow
c transition over inaccurate points were x and y are close
c to the test values below.
```

[17] ftp://igpp.ucla.edu/pub/space_physics/simulation_codes/disp.tar.gz

```
c
c The transition from larger expansion to smaller expansion
c is given by CDABS(Z)=7.D0
c I have verified that the two different expansions give the
c same answer to 1.d-13 at CDABS(Z)=7.d0
c
!! if (y.lt.4.29d0.and.x.lt.5.33d0) then
!! s=(1.d0-y/4.29d0)*dsqrt(1.d0-x*x/28.41d0)
      if (y.lt.8.58d0.and.x.lt.10.66d0.and.cdabs(z).lt.7.d0) then
      s=(1.d0-y/8.58d0)*dsqrt(1.d0-x*x/113.64d0)
      h=1.6d0*s
      h2=2.d0*h
      capn=6.d0+23.d0*s+.5d0
      nu=9.d0+21.d0*s+.5d0
      lambda=h2**capn
      else
      h=0.d0
      capn=0
      nu=8
      end if
      b=h.eq.0.d0.or.lambda.lt.eps
      rr=0.d0
      ri=0.d0
      sr=0.d0
      si=0.d0
      nup=nu+1
      do 100 i=1,nup
      n=nup-i
      np1=n+1
      tr=y+h+np1*rr
      ti=x-np1*ri
      c=.5d0/(tr*tr+ti*ti)
      rr=c*tr
      ri=c*ti
      if (h.gt.0.and.n.le.capn) then
      tr=lambda+sr
      sr=rr*tr-ri*si
      si=ri*tr+rr*si
      lambda=lambda/h2
      end if
  100 continue
      cc=1.12837916709551d0
      if (y.lt.eps) then
c
```

```
c Pure real argument, include pole.
c
      re=dexp(-x*x)
      else
      if (b) then
      re=rr*cc
      else
      re=sr*cc
      end if
      end if
      if (b) then
      im=ri*cc
      else
      im=si*cc
      end if
c
c extend to y < 0.
c
      if (dimag(z).lt.0.d0) then
      c=2.d0*dexp(y*y-x*x)
      re=c*dcos(2.d0*x*y)-re
      im=c*dsin(2.d0*x*y)+im
      end if
c
c extend to x < 0.
c
      if (dble(z).lt.0.d0) then
      im=-im
      end if
      re=2.d0*re/cc
      im=2.d0*im/cc
      plasmaz=dcmplx(-im,re)
      return
      end
c
      function plasmazlg(z)
c
c Routine to evaluate the plasma Z function for large argument.
c If the argument is large, then routine uses asymptotic expansion,
c otherwise routine calls PLASMAZ to evaluate the Z function.
c
c Written by R.J. Strangeway 25th Oct 1985.
c
c Extension to full complex plane included.
```

```
c
      complex*16 plasmaz,z,z2,plasmazlg,zp,zpp,cc
      real*8 xx,x,y,dfact,eps
      eps=1.d-12
      x=cdabs(z)
      xx=dreal(z)
      y=dimag(z)
      if (x.lt.10.d0) then
      plasmazlg=plasmaz(z)
      return
      end if
      ipw=dint(dlog10(x))+1
      irz=18/ipw
      z2=z*z
      zpp=1.d0/z
      zp=-zpp
      do 100 i=0,irz
      dfact=dfloat(i)+.5d0
      zpp=zpp*dfact/z2
      zp=zp-zpp
  100 continue
      if (dabs(y).lt.eps) then
      cc=dcmplx(0.d0,1.772453850905516027 3d0)*dexp(-xx*xx)
      else if (y.lt.0.d0) then
      cc=2.d0*1.772453850905516027 3d0*dexp(y*y-xx*xx)*
     + dcmplx(dsin(2.d0*xx*y),dcos(2.d0*xx*y))
      else
      cc=dcmplx(0.d0,0.d0)
      end if
      zp=zp+cc
      plasmazlg=zp
      return
      end
c
      function plasmaplg(z)
c
c Routine to evaluate the first derivative of the plasma Z function
c for large argument. Note that the routine actually returns -Z'/2.
c i.e. Z' = -2.*PLASMAPLG
c If the argument is large, then routine uses asymptotic expansion,
c otherwise routine calls PLASMAZ to evaluate the Z function.
c
c Written by R.J. Strangeway 25th Oct 1985.
c
```

```
c Extension to full complex plane included.
c
      complex*16 plasmaz,z,z2,plasmaplg,zp,zpp,cc
      real*8 xx,x,y,dfact,eps
      eps=1.d-12
      x=cdabs(z)
      xx=dreal(z)
      y=dimag(z)
      if (x.lt.10.d0) then
      plasmaplg=1.d0+z*plasmaz(z)
      return
      end if
      ipw=dint(dlog10(x))+1
      irz=18/ipw
      z2=z*z
      zpp=1.d0
      zp=0.d0
      do 100 i=0,irz
      dfact=dfloat(i)+.5d0
      zpp=zpp*dfact/z2
      zp=zp-zpp
  100 continue
      if (dabs(y).lt.eps) then
      cc=dcmplx(0.d0,1.7724538509055160273d0)*dexp(-xx*xx)
      else if (y.lt.0.d0) then
      cc=2.d0*1.7724538509055160273d0*dexp(y*y-xx*xx)*
     + dcmplx(dsin(2.d0*xx*y),dcos(2.d0*xx*y))
      else
      cc=dcmplx(0.d0,0.d0)
      end if
      cc=cc*z
      zp=zp+cc
      plasmaplg=zp
      return
      end
```

7.4 HSL Mathematical Software Library

This version is provided in HSL Library[18], which computes the real and imaginary parts of the PDF. If y≥2.75 or if y≥2 and x≥4 an asymptotic continued fraction due to Fried and Conte is used, otherwise if x≥6.25 a rational approximation from Abramowitz and Stegun is used, otherwise a Taylor series is used.

The single precision version

```
CALL FC12A(X,Y,ZR,ZI,ZPR,ZPI)
```

[18] http://www.hsl.rl.ac.uk/archive/

The double precision version

```
CALL FC12AD(X,Y,ZR,ZI,ZPR,ZPI)
```

Accuracies: approx. 10^{-6} absolute.

See FC12 document[19] for details. The code is very short and available at no cost.

7.5 Another Version

I get this Fortran77 version from Prof. Jia-qi DONG[20], which is checked to be effective. But, I haven't understood what algorithm it is.

```
      complex function zp(u)
c a simple version to compute the plasma dispersion function
      complex u,z,u2,azp,azpold,usqm
      na=10
      if(cabs(u).ge.(4.)) go to 3
      usqm=-u**2
      if(real(usqm).gt.(200.)) usqm=cmplx(200.,0.)
      zp=cmplx(0.,1.)*1.77245385090551602729816748*cexp(usqm)
      u2=-2.*u**2
      azp=-2.*u
      do 2 n=1,100
      zp=zp+azp
    2 azp=azp*u2/(2.*n+1.)
      zp=zp+azp
      go to 11
    3 z=1./u
      if(aimag(u).le.(0.)) go to 10
      zp=0.
      go to 20
   10 continue
      usqm=-u**2
      if(real(usqm).gt.(200.)) usqm=cmplx(200.,0.)
      zp=cmplx(0.,1.)*1.77245385090551602729816748*cexp(usqm)
c 1 format(76h argument u of subroutine zeta has too large a negative
c 1imaginary part, u= ,1e14.7,3h + ,1e14.7,2h i)
c write(6,1) u
      if(aimag(u).lt.(0.)) zp=2.*zp
   20 azp=z
      u2=.5*z**2
      do 25 n=1,na
      zp=zp-azp
      azpold=azp
      azp=(2.*n-1.)*azp*u2
      if (cabs(azp) .ge. cabs(azpold)) go to 11
```

[19] http://www.hsl.rl.ac.uk/archive/specs/fc12.pdf
[20] http://ifts.zju.edu.cn/iftsnew/index.php?CONTENT=jiaqidong&LANG=CN&NAME=jiaqidong

```
 25 continue
    zp=zp-azp
 11 continue
    return
    end
```

7.6 More papers

For examples, [Bravo-Ortega1987], [Jimenez-Mier2001], [Newberger1986], [Mamedov2009], [Percival1998], [Robinson1988], [Sato1984] and [McCabe1984].
There should be more references.

8 Mathematica

To calculate the Z function in *Mathematica* is extremely easy. We need only two lines:

```
x=9.8+10.0*I;
Z= I*Sqrt[Pi]*Exp[-x^2] (1 + Erf[I*x]);
```

Output:

```
 -0.0498562 +0.00511338i
```

x=9.8-10.0*I

```
-174.761 + 63.6327i
```

This means *Mathematica* has considered the analytical continuation of the complex error function.

Table1. A table to compare different algorithms

x	9.8+10.0*I	9.8-10.0*I
[Freid1961]	-0.0498562+ 0.0511338i	-0.174762E03+ 0.636333E02i
8-pole (with correction)	-0.04985622723020 7 + 0.05113379750461 4i	-1.74761463108007 0e+002 + 6.36326885383169 1e+001i
`faddeeva.m`	-0.04985622714609 1 + 0.05113379742397 6i	-1.74761463109672 8e+002 + 6.36326885362753 1e+001i
Mathematica	-0.0498562 + 0.00511338i	-174.761 + 63.6327i

9 Illustrations

This part is mainly generated via PPLU[21] (Plasma Physics Learning Utility), which uses `faddeeva.m` to calculate the plasma dispersion function.

```
% Surfc plot of Re(z) and Im(z)
xmin=-2; xmax=2; dx=0.1; ymin=-2; ymax=2; dy=0.1;
[X,Y]=meshgrid(xmin:dx:xmax,ymin:dy:ymax);
Zz=faddeeva(complex(X,Y))*1i*sqrt(pi);
surfc(xmin:dx:xmax,ymin:dy:ymax,real(Zz));axis xy square; caxis([-1
```

[21] http://ifts.zju.edu.cn/forum/viewtopic.php?f=18&t=461

```
1]);
xlabel('x');ylabel('y');title('Re(z)');
figure;surfc(xmin:dx:xmax,ymin:dy:ymax,imag(Zz));axis xy square;
caxis([-1 1]);
xlabel('x');ylabel('y');title('Im(z)');
```

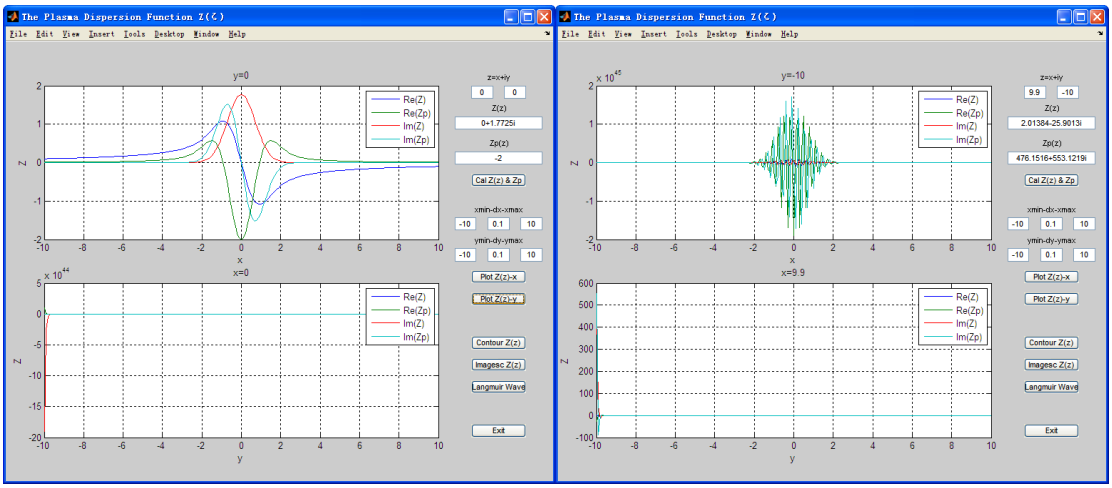


Fig1. Plot of Re(Z), Im(Z), Re(Z'), and Im(Z'), with y=0, x=0, y=-10, x=9.9.

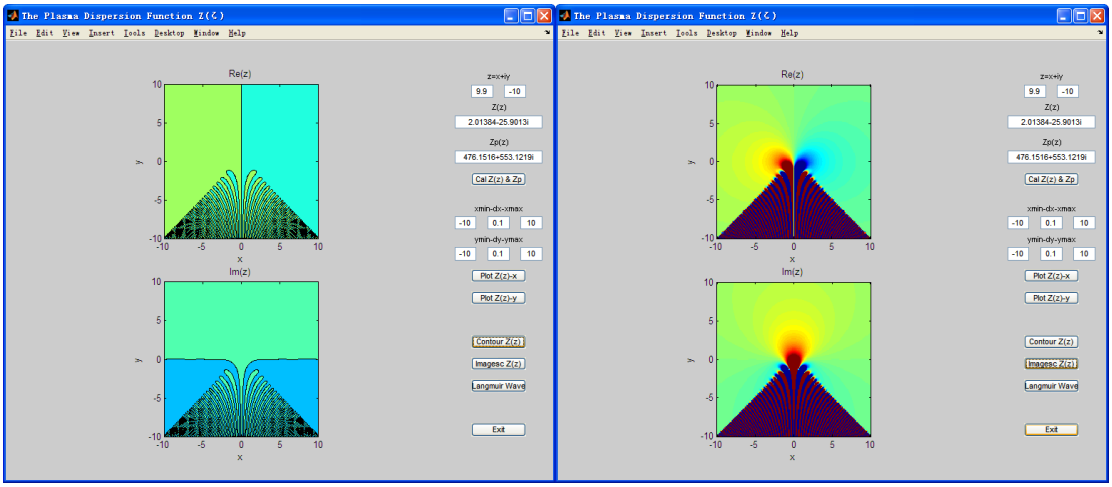


Fig2. Contour Plot of Re(z) and Im(z) in [-10, 10]×[-10, 10]

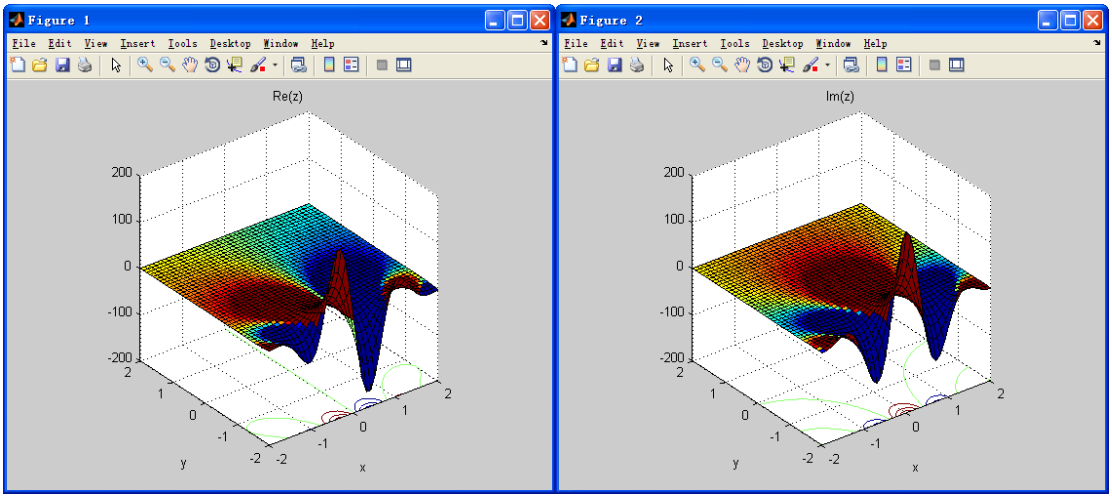


Fig3. Surfc Plot of Re(z) and Im(z) in [-2, 2]×[-2, 2] , with analytical continuation

If we do not add the correction term,

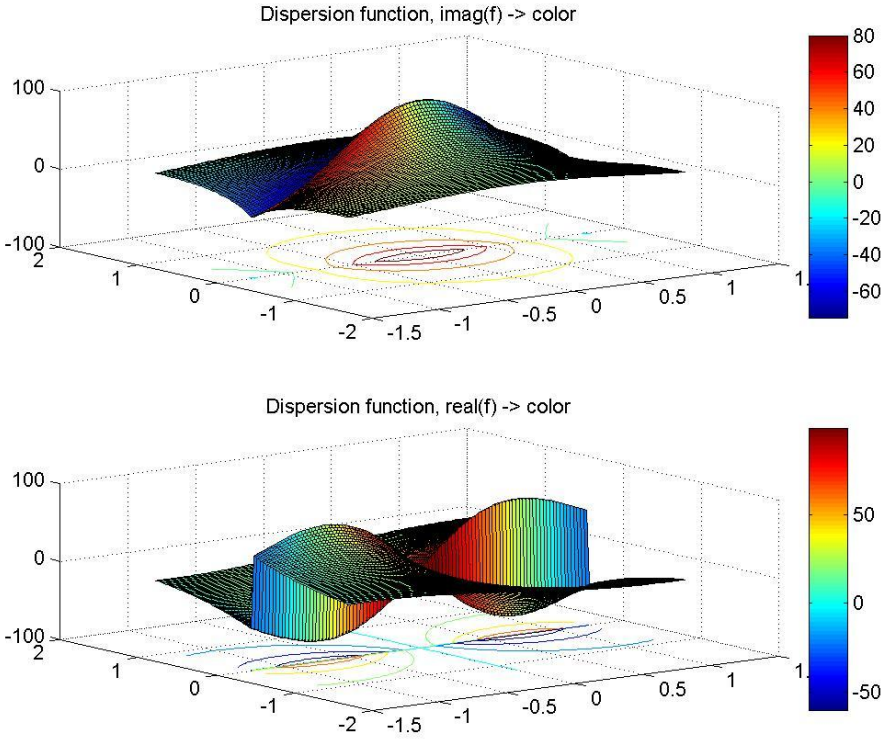


Fig4. Surfc Plot of Re(z) and Im(z) in [-2, 2]×[-2, 2], without analytical continuation

We can see the function is discontinuous at y=0. This is why we need the extra exponential term.

We can also see the uncontinuity from (0.5), $x \to x_0, y \to 0^+$,

$$Z(x_0 - i0^+) - Z^*(x_0 + i0^+) = 2i\pi^{1/2} \exp[-x_0^2] \neq 0, \tag{0.37}$$

Then, if we do not add the analytical continuation,

$$\Rightarrow \begin{cases} \mathrm{Re}(Z), & \text{continuous} \\ \mathrm{Im}(Z), & \text{discontinuous} \end{cases}. \tag{0.38}$$

This result is consists with Fig4.

10 Applications in Solving Dispersion Relation

10.1 The dispersion relation of Langmuir wave

The dispersion relation

$$D(\omega, k) = 1 - \frac{\omega_p^2}{k^2} \int_{-\infty}^{\infty} \frac{\partial_v f_0}{v - \omega / k} dv = 0, \tag{0.39}$$

f_0 is the distribution function at $t = 0$. When $v \to \pm\infty$, $f_0 \to 0$, then (0.39) reduce to

$$D(\omega,k)=1-\frac{\omega_p^2}{k^2}\int_{-\infty}^{\infty}\frac{f_0}{(v-\omega/k)^2}dv=0, \tag{0.40}$$

When f_0 is Maxwellian distribution

$$f_0=(\frac{m}{2\pi kT})^{1/2}exp(-\frac{mv^2}{2kT}), \tag{0.41}$$

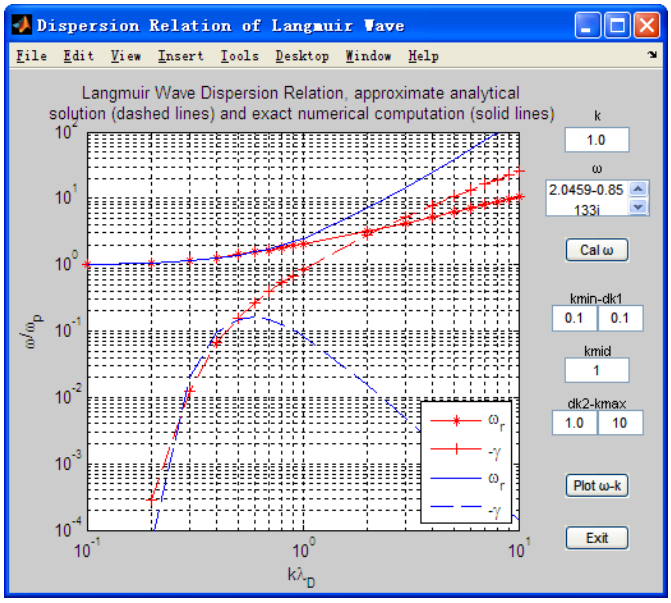
$D(\omega,k)=0$ is equivalent to $D(\zeta,k)=0$

$$D(\zeta,k)=1+\frac{1}{(k\lambda_D)^2}[1+\zeta Z(\zeta)], \tag{0.42}$$

10.2 The solution

Equation (0.42) can be solved by Newton iterative method with proper initial guess. However, we use Matlab function `fsolve` directly here

```
clear;clc;
zeta=@(x)faddeeva(x)*1i*sqrt(pi);
f=@(x,k)1+k*k+x*zeta(x);w=[];
kmin=0.1;dk1=0.1;kmid=1;dk2=1;kmax=10.0;
k=[kmin:dk1:kmid,(kmid+dk2):dk2:kmax];
for kk=k
    options=optimset('Display','off');
    x=fsolve(f,1-0.1i,options,kk)*sqrt(2)*kk;
    w=[w,x];
end
wre=real(w);wie=imag(w);
wrt=1.0+1.5.*k.*k;
wit=-sqrt(pi/8).*exp(-1.0./(2.0.*k.^2)-1.5)./(k.^3);
loglog(k,wre,'-*r',k,-wie,'+r--',k,wrt,'b-',k,-wit,'b--');
legend('\omega_r','-\gamma','\omega_r','-\gamma','Location','SouthEas
t');grid on;
title(strcat('Langmuir Wave Dispersion Relation, approximate analytical
'...
    ,10,'solution (dashed lines) and exact numerical computation (solid
lines)'));
xlabel('k\lambda_D');ylabel('\omega/\omega_p');
xlim([kmin,kmax]);
ylim([0.0001,100]);
```



Langmuir Wave Dispersion Relation[22], approximate analytical solution (dashed lines) and exact numerical computation (solid lines), via PPLU

We see here, the approximation analytical solution won't be enough when $k\lambda_D \sim 1$. This is why we need calculate PDF as exactly as possible.

10.3 Beam Plasma

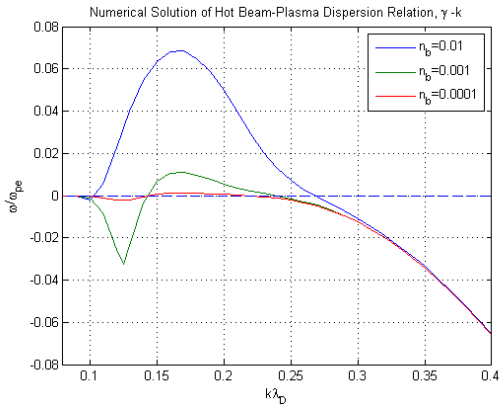
Distribution function

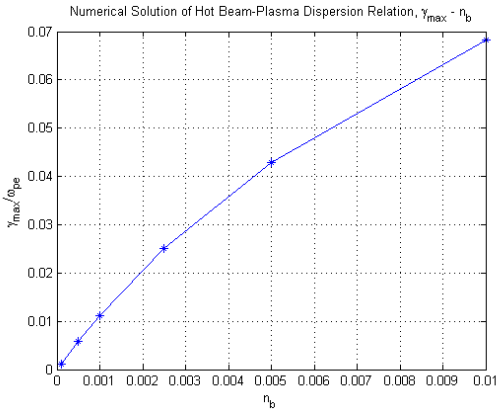
$$f_0 = (1-n_b)(\frac{m}{2\pi T_e})^{3/2}\exp(-\frac{mv^2}{2T_e}) + n_b(\frac{m}{2\pi T_b})^{3/2}\exp[-\frac{m(v-v_d)^2}{2T_b}], \qquad (0.43)$$

The dispersion relation

$$D(\omega,k) = 1 + \frac{\omega_{pe}^2}{k^2 v_{Te}^2}[1+\xi_e Z(\xi_e)] + \frac{\omega_{pb}^2}{k^2 v_{Tb}^2}[1+\xi_b Z(\xi_b)] = 0, \qquad (0.44)$$
$$\xi_\alpha = (\omega - kv_{d\alpha})/kv_{T\alpha}.$$

The solution is illustrated below.





See also [Sydora2003].

[22] Also call collisionless Landau damping.

10.4 More applications

See the codes mentioned in section 7, e.g., WHAMP, v1d1code.f90 and disfm.f.

11 Summary and Discussion

11.1 Summary

As a summary, to calculate PDF, we mainly have three types of algorithms:

- Continued fraction.
- Approximation by using poles (e.g., Pade approximation).
- FFT (complex error function).

And the auxiliary algorithms:

- Numerical integration of the differential equation (0.6).
- Taylor series.

But, we should attentive that these algorithms have not considered the analytical continuation for the y<0 plane, then which is only valid for upper half-plane (the growth wave). For weak damping, these algorithms can still be used, but the result may not be accurate. And, for heavy damping, these algorithms should give wrong answers. However, we can add the extra correction term $i\pi^{1/2}\sigma\exp(-\zeta^2)$ by ourselves when apply it.

Although I list several codes here, I recommend HSL's version and `faddeeva.m` for that they are short and clear and also with the correction to lower half-plane.

11.2 Discussion

I list two questions here which I have no answer yet:

- ✧ Can we get arbitrary accuracy value of $Z(\zeta)$ for arbitrary ζ up to now? It seems still a challenge.
- ✧ How to calculate the Landau contour numerically for non-Maxwellian distribution, i.e., how we do the analytical continuation? If, we integrate directly for –infinity to +infinity, the result is only valid for upper half-plane, because the definition is on the upper half-plane[23].

12 References

[Baker1975] Baker, G. A., Essentials of Pade Approximants, Academic Press Inc, 1975, 316.

[Bravo-Ortega1987] Bravo-Ortega, A.; Swanson, D. G. & Glasser, A. H., Asymptotic approximation for the dispersion relation of a hot magnetized plasma, Journal of Plasma Physics, 1987, 38, 275-286.

[Cuyt2008] Cuyt, A. A. M.; Petersen, V.; Verdonk, B.; Waadeland, H.; Jones, W. B.; Backeljauw, F. & Bonan-Hamada, C., Handbook of continued fractions for special functions, Springer, 2008, 430.

[Fadeeva1954] V. N. Fadeeva and N. M. Terent'ev, Tables of Values of the Probability Integral for

[23] If you do not understand the question here, please read the Appendix C of [Nicholson1983] carefully first.

Complex Arguments, State Publishing House for Technical Theoretical Literature, Moscow, 1954.

[Fried1961] Fried, B. D. & Conte, S. D., The Plasma Dispersion Function—THE HILBERT TRANSFORM OF THE GAUSSIAN, Academic Press, New York and London, 1961. Erratum: Math. Comp. v. 26, 1972, no. 119, p. 814. Reviews and Descriptions of Tables and Books, Math. Comp., v. 17, 1963, pp. 94-95.

[Fried1968] Fried, B. D.; Hedrick, C. L. & Mccune, J., Two-Pole Approximation for the Plasma Dispersion Function, Physics of Fluids, AIP, 1968, 11, 249-252.

[Henrici] P. Henrici , Quotient difference algorithm, Natl. Bureau Standards, U. S. Appl. Math. Ser. 49.

[Huba2009] Huba, J., NRL PLASMA FORMULARY, The Office of Naval Research, 2009.

[Jimenez-Mier2001] Jimenez-Mier, J., An approximation to the plasma dispersion function, Journal of Quantitative Spectroscopy and Radiative Transfer, 2001, 70, 273 – 284.

[Landau1946] Landau, L. D., On the vibration of the electronic plasma, Journal of Physics, 1946, 10, 25.

[Mamedov2009] Mamedov, B., Analytical Evaluation of the Plasma Dispersion Function Using Binomial Coefficients and Incomplete Gamma Functions, Contributions to Plasma Physics, WILEY-VCH Verlag, 2009, 49, 36-39.

[Martin1979] Martin, P. & Gonzalez, M. A., New two-pole approximation for the plasma dispersion function Z, Physics of Fluids, AIP, 1979, 22, 1413-1414.

[Martin1980] Martin, P.; Donoso, G. & Zamudio-Cristi, J., A modified asymptotic Pade method Application to multipole approximation for the plasma dispersion function Z, Journal of Mathematical Physics, AIP, 1980, 21, 280-285.

[McCabe1984] McCabe, J. H., Continued fraction expansions for the plasma dispersion function, Journal of Plasma Physics, 1984, 32, 479-485.

[Nemeth1981] Nemeth, G.; Ag, A. & Paris, G., Two-sided Pade approximations for the plasma dispersion function, Journal of Mathematical Physics, AIP, 1981, 22, 1192-1195.

[Newberger1986] Newberger, B. S., Efficient numerical computation of the plasma dispersion function, Computer Physics Communications, 1986, 42, 305 – 311.

[Nicholson1983] Nicholson, D. R., Introduction to Plasma Theory, Wiley, 1983.

[Percival1998] Percival, D. J. & Robinson, P. A., Generalized plasma dispersion functions, J. Math. Phys., 1998, 39, 3678.

[Poppe1990] Poppe, G. P. M. & Wijers, C. M. J., More efficient computation of the complex error function, ACM Trans. Math. Softw., ACM, 1990, 16, 38-46.

[Robinson1988] Robinson, P. A. & Newman, D. L., Approximation of the dielectric properties of Maxwellian plasmas: dispersion functions and physical constraints, Journal of Plasma Physics, 1988, 40, 553-566.

[Ronnmark1982] Kjell Ronnmark, WHAMP – Waves in Homogeneous, Anisotropic Multi component Plasmas, KGI Report NO. 179, June 1982.

[Salzer1951] H. E. Salzer, Formulas for calculating the error function of a complex variable, Math. Tables Aids Comput. 35, 67, 1951.

[Sato1984] Sato, M., Transformation approximation for the plasma dispersion function and application to electrostatic waves, Journal of Plasma Physics, 1984, 31, 325-331.

[Sydora2003] Sydora, R. D., Low Noise Electrostatic and Electromagnetic Delta-f Particle-in-Cell Simulation of Plasmas, 2003, 109.

[Weideman1994] Weideman, J. A. C., Computation of the Complex Error Function, SIAM Journal on Numerical Analysis, SIAM, 1994, 31, 1497-1518.

2011-10-09 19:35
huashengxie@gmail.com
Hua-sheng XIE(谢华生)