GUISDAP Documentation

M. S. Lehtinen* and A. Huuskonen**

* Sodankylä Geophysical Observatory, Sodankylä, Finland
** Finnish Meteorological Institute, Helsinki, Finland

The calculation of the IS spectrum

The spectrum calculation is based on the general spectral formula for the non-magnetic case with any number of ions is used. All populations are assumed to be Maxwellian, with speeds and temperatures independent of each other.

$$\frac{S(\theta_0)\,\omega_0}{N_e\sigma_0}\,d\theta_0 = \frac{|y_e|^2\sum_i \eta_i \mathrm{Re}y_i + |\sum_i \mu_i y_i + ik^2D^2|^2 \mathrm{Re}y_e}{|y_e + \sum_i \mu_i y_i + ik^2D^2|^2}\frac{d\theta_0}{\pi\theta_0}$$

where $\theta_0 = \omega/\omega_0$. In the following, we need also $\psi_0 = \nu/\omega_0$, where $\nu = \nu_{in}$ is the ion-neutral collision frequency. The definition of the collision frequencies follows Schlegel (1979).

The admittances are defined by

$$y_s(\theta_s - i\psi_s) = i + \theta_s\frac{J(\theta_s)}{1-\psi_s J(\theta_s)} \equiv i + \theta_s u_s, \quad \text{with} \quad J = -iG \quad \text{and where}$$

$$\theta_s = \omega/\omega_s, \quad \omega_s = k\sqrt{\frac{2\kappa T_s}{m_s}}, \quad \psi_s = \frac{\nu_s}{\omega_s}, \quad \psi_i = \psi_0, \quad \psi_e = 0.35714\,\psi_0$$

$$G(z) = i\sqrt{\pi}e^{-z^2} + 2e^{-z^2}\int_0^z e^{t^2}dt = 2e^{-z^2}\int_{-i\infty}^z e^{t^2}dt$$

$$\eta_i = N_i/N_e, \quad \mu_i = \eta_i T_e/T_i = \frac{N_i}{N_e}\frac{T_e}{T_i}, \quad \gamma_i = \omega_0/\omega_i = \sqrt{\frac{T_0}{T_i}\frac{m_i}{m_0}}$$

$$D = \lambda_D = v_t/\omega_p = \sqrt{\frac{\kappa T_e}{m_e}}\sqrt{\frac{\epsilon_0 m_e}{N_e e^2}} = \sqrt{\frac{T_e}{T_0}}\sqrt{\frac{N_0}{N_e}}D_0, \quad D_0 = \sqrt{\frac{\epsilon_0\kappa T_0}{N_0 e^2}}, \quad \kappa = k_B.$$

The spectrum is developed to a form symmetric with respect to electron/ion populations. All parameters are relative to the reference parameters m_0, T_0 and N_0.

$$\begin{aligned}\frac{S(\theta_0)\,\omega_0}{N_0\sigma_0}\,d\theta_0 &= \frac{N_e}{N_0}\frac{\left|y(\frac{\theta_e}{\theta_0}\theta_0)\right|^2\sum_i \frac{N_i}{N_e}\gamma_i \mathrm{Re}u(\frac{\theta_i}{\theta_0}\theta_0) + \left|\sum_i \frac{N_i}{N_e}\frac{T_e}{T_i}y(\frac{\theta_i}{\theta_0}\theta_0) + i\frac{T_e}{T_0}\frac{N_0}{N_e}(kD_0)^2\right|^2\gamma_e\mathrm{Re}u(\frac{\theta_e}{\theta_0}\theta_0)}{\left|y(\frac{\theta_e}{\theta_0}\theta_0) + \sum_i \frac{N_i}{N_e}\frac{T_e}{T_i}y_i + i(kD_0)^2\frac{T_e}{T_0}\frac{N_0}{N_e}\right|^2\pi}d\theta_0 \\ &= \frac{N_e}{N_0}\frac{\left|\frac{T_0}{T_e}\frac{N_e}{N_0}y(\gamma_e\theta_0)\right|^2\frac{N_0}{N_e}\sum_i \frac{N_i}{N_0}\gamma_i\mathrm{Re}u(\gamma_i\theta_0) + \left|\sum_i \frac{T_0}{T_i}\frac{N_i}{N_0}y(\gamma_i\theta_0) + i(kD_0)^2\right|^2\gamma_e\mathrm{Re}u(\gamma_e\theta_0)}{\left|\frac{T_0}{T_e}\frac{N_e}{N_0}y(\gamma_e\theta_0) + \sum_i \frac{T_0}{T_i}\frac{N_i}{N_0}y(\gamma_i\theta_0) + i(kD_0)^2\right|^2\pi}d\theta_0\end{aligned}$$

For collisional and cases and with Doppler velocities taken into account the arguments in the y and u functions should be replaced by

$$\gamma_i\theta_0 \rightarrow \gamma_i(\theta_0 - v_i) - i\psi_i \quad \text{and} \quad \gamma_e\theta_0 \rightarrow \gamma_e(\theta_0 - v_e) - i\psi_e$$

The calculation of the spectrum for arbitrary drifts, temperatures or an arbitrary number of ions is listed below.

GUPHD:guisdap:m150:spec.m

```
1     % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2     %
3     % Incoherent scatter spectrum calculation
4     %
5     % function s=spec(nin0,tit0,mim0,psi,vi,kd2,om,pldfvv);
6     function s=spec(nin0,tit0,mim0,psi,vi,kd2,om,pldfvv);
7
8
9       j=sqrt(-1);
10      npar=length(mim0); nion=npar-1;
11      nom=length(om);
12      oma=zeros(length(om),npar);
13      gam=sqrt(mim0./tit0);
14      for i=1:nion+1,
15        oma(:,i)=(om-vi(i))*gam(i);
16        ua(:,i)=oma(:,i)-j*psi(i)/sqrt(tit0(i));
17      end;
18      ua=pldfi(ua);
19      for i=1:nion+1,
20        ua(:,i)=ua(:,i)./(j-(psi(i)/sqrt(tit0(i))*ua(:,i)));
21        ya(:,i)=j+oma(:,i).*ua(:,i);
22      end;
23      s=(abs(ya(:,npar)*(nin0(npar)/tit0(npar))).^2).*...
24                      (real(ua(:,1:nion))*...
25                      (gam(1:nion).*nin0(1:nion))' );
26      s=s+(abs(j*kd2+ya(:,1:nion)*(nin0(1:nion)./tit0(1:nion))').^2).*...
27                       real(ua(:,npar)*...
28                       (gam(npar)*nin0(npar)) );
29      s=(s./(abs(j*kd2+(ya*(nin0./tit0)')).^2))/pi;
```

The input parameters to the spectral calculation (and the corresponding mex-version) are defined by

$$\begin{aligned}
\texttt{nin0(i)} &= N_i/N_0 \\
\texttt{tit0(i)} &= T_i/T_0 \\
\texttt{mim0(i)} &= m_i/m_0 \\
\texttt{psi(i)} &= \psi_i \\
\texttt{kd2(i)} &= (k_{scatt}D_0)^2 \\
\texttt{vi(i)} &= v_i/v_0
\end{aligned}$$

The argument `om` is the scaled variable $\theta_0 = \omega/\omega_0$. The last argument `pldfvv` is the plasma dispersion function coefficient table used by the mex-version of spec. In the m-file version it is not actually necessary at all, becaus the m-files use the global variable `pldfv` instead, but it is included here to make the call of the m-file `spec.m` identical to the corresponding call to the mex-file `spec.mex`.

The value $\theta = \pm 1$ with one ion and no drifts or collisions is exactly the ion line position.

The calculations in `spec.m` perform the calculations in the general formulas shown. There is some gimmickry involved to vectorize the operation, including just a single call to the `pldf` routine.

In analysis spectrum calculations go through a transformation function `transf(p)`, whose input is a vector of the parameters actually fitted and whose output should be a series of MATLAB vectors specifying the different ion species suitable for input to the `spec` function. The ions are indexed starting from 1 and the last element in the vectors is for the electrons.

The following form is for two ion species with equal temperatures and drifts; namely the molecular ion with mass 30.5 u, the O^+ with mass 16 u; also, for example, H^+ with mass 1 u and metallic ions, like Fe^+ with mass 56 may be added. It is also straightforward to change the transformation function to use different masses or different ways of representing the parameters, but so far this is up to the user and GUISDAP 1.50 does not automatically offer this possibility.

GUPHD:guisdap:m150:transf.m

```
1     % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
```

```
2       %
3       % transf.m
4       %
5       % Function to transform user parameters to spectrum parameters
6       %
7       % [nin0,tit0,mim0,psi,vi]=transf(p)
8       %
9         function [nin0,tit0,mim0,psi,vi]=transf(p)
10
11      m1 = 30.5*1.6605402e-27;
12      m2 = 16*1.6605402e-27;
13      me = 9.1093897e-31;
14      m0 = m1;
15      %
16          psi0=p(4);                          % psi0
17      %
18          nin0(1)=p(1)*(1-p(6));              % N1/Ne
19          nin0(2)=p(1)*p(6);                  % N2/Ne
20          nin0(3)=p(1);                       % Ne/N0;
21          tit0(1)=p(2);                       % T1/T0
22          tit0(2)=p(2);                       % T2=T1
23          tit0(3)=p(2)*p(3);                  % Te/T0
24          mim0(1)=m1/m0;                      % m1/m0
25          mim0(2)=m2/m0;                      % m2/m0
26          mim0(3)=me/m0;                      % me/m0 from amu
27          vi(1)=p(5);                         % v1/v0; v0=phase velocity om0/kscatt
28          vi(2)=p(5);                         % v2/v0
29          vi(3)=p(5);                         % ve/v0
30          psi(1)=psi0;                        % psi1
31          psi(2)=psi0;                        % psi2=psi1 for all temperatures
32          psi(3)=psi0*0.35714;                % psie independently of Te/Ti
33        end
```

The following function is used to interpolate the plasma dispersion function values. Lagrangian interpolation is used. If the function values y_i are given at argument values x_i, the Lagrange interpolation polynomial is given by

$$P(x) = A(x) \sum_{j=1}^{N} \frac{y_j}{(x - x_j) \prod_{i \neq j} (x_i - x_j)} + R_N \quad \text{with} \quad A(x) = \prod_{i=1}^{N} (x - x_i).$$

The error term is given in a number of cases by

$$R_2 \approx 0.125\ \Delta^2 \approx 8.15\, 10^{-4}$$
$$R_4 \approx 0.024\ \Delta^4 \approx 1.41\, 10^{-5}$$
$$R_6 \approx 0.0042 \Delta^6 \approx 7.00\, 10^{-8}$$
$$R_8 \approx 0.0011 \Delta^8 \approx 2.96\, 10^{-9},$$

where the finite differences are approximately given by $\Delta^n \approx f^{(n)} (\Delta x)^n$. The numerical values calculated correspond to the maximum value for the real part of the plasma dispersion function with $\Delta x = 0.075$. Four-point interpolation is used. If the user wants to used six-point interpolation instead, the parameter `accur` must be set to 6 below.

GUPHD:guisdap:m150:pldfi.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % pldfi.m
4       %
5       % Plasma dispersion function interpolation
6       %
7       % res=pldfi(z)
8       %
9         function res=pldfi(z)
10      %
11      global pldfv path_GUP
12
13      if max(size(pldfv))==0,
14        eval(canon(['load ' path_GUP 'matfiles:pldfv.mat']));
15      end
16
17       % t_start(5)
18        accur=4;
19        res=z;
20        z=z(:);
21        f1=find( abs(real(z)) >=3.9 | imag(z) <= -3.5 );
```

```
22        f2=find( abs(real(z)) < 3.9 & imag(z) > -3.5 & imag(z) <0.0375 );
23        f3=find( imag(z) >= 0.0375);
24        if max(size(f1))>0, z(f1)=pldfas(z(f1),accur+2); end
25        if max(size(f3))>0, fprintf('calculating pldf\n'),z(f3)=pldf(z(f3)); end
26      %
27        if max(size(f2)) >0,
28          x=abs(real(z(f2)))/0.075+4;
29          f1=find(real(z(f2)) < 0);
30          y=-imag(z(f2))/0.075;
31          p=x-floor(x)-sqrt(-1)*(y-round(y));
32          x=floor(x)+61*round(y);          % index for linear pldfv table lookup
33          if accur==4
34            y=(p-1).*(p-2).*( (-p/3).*pldfv(x-1) + (p+1).*pldfv(x) );
35            y=(y+p.*(p+1).*( (2-p).*pldfv(x+1) + ((p-1)/3).*pldfv(x+2)) )*.5;
36          elseif accur==6
37            p(find(abs(p-round(p))<=1e-99))=p(find(abs(p-round(p))<=1e-99))+2e-99;
38            y=(pldfv(x+3)./(p-3)-pldfv(x-2)./(p+2))/10+pldfv(x+1)./(p-1);
39            y=y+(pldfv(x-1)./(p+1)-pldfv(x+2)./(p-2))/2-pldfv(x)./p;
40            y=y.*(p-3).*(p-2).*(p-1).*p.*(p+1).*(p+2)/12;
41          end
42          y(f1) = -conj( y(f1) );
43          z(f2)=y;
44        end                                  % if max(size(f2)) >0
45        res(:)=z;
46      %t_stop(5)
```

The definition of the plasma dispersion function G used here is

$$\mathrm{pldf}(z) = 2e^{-z^2}\int_0^z e^{t^2} + i\sqrt{\pi}e^{-z^2} = 2i\int_0^\infty e^{-t^2-2iz}dt = 2ie^{-z^2}\int_{iz}^\infty e^{-t^2}dt,$$

and for the set of values specified by the input vector z, the function returns a set of values specified by the output vector.

GUPHD:guisdap:m150:pldf.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % pldf.m
4       %
5       % complex plasma dispersion function
6       %
7       % accuracy:  8 numbers in the whole complex plane
8       %            (main algorithm accurate to 1e-12, accuracy is
9       %            limited by subroutine cerfexp and numerical accuracy)
10      %
11      % method:    approximation described by Salzer in
12      %            Math. Tables Aids Comput., Vol. 5 (1951) pp. 67-70.
13      %            The same formula appears in
14      %            Abramovitch-Stegun p.299 formula 7.1.29.
15      %            (the exponentials in the formulas have been written
16      %            in a slightly modified way to get rid of unnecessary
17      %            overflows)
18      %                                          Markku Lehtinen 10.3.1979
19      %
20      % res=pldf(z)
21      %
22        function res=pldf(z)
23      %
24      %  if max(size(z))==1
25      %    res=0;
26      %  else
27      %    res=zeros(z);
28      %  end
29        [nrow ncol]=size(z);
30        res=zeros(nrow, ncol);
31        j=sqrt(-1);
32        for irow=1:nrow
33          for icol=1:ncol
34            z1=-j*z(irow,icol);
35            x=real(z1);
36            y=imag(z1);
37            if(abs(2*x*y)<30000)
38              cs=cos(2*x*y);
39              sn=sin(2*x*y);
40            else
41              cs=1; sn=0;
42            end
43            pisqr=sqrt(pi);
44            fn=max(1,floor(abs(2*y))-11):floor(abs(2*y))+11;
45            term1=exp(-fn.*fn/4-y*y);
```

```
46          if max(abs(term1))<1e-100, term1=zeros(fn); end
47          term2=exp(-(fn/2-y).*(fn/2-y))/2;
48          term3=exp(-(fn/2+y).*(fn/2+y))/2;
49          factor=ones(fn)./(fn.*fn+4*x*x);
50          sume=sum(factor.*(term1*cs-term2-term3));
51          sumf=sum(factor.*(term1*2*x*sn+fn.*(term2-term3)));
52          if (abs(x*y)<1e-4)
53            e1=(-exp(-y*y)*x*y*y/2+2*x*sume)/pisqr;
54            f1=(exp(-y*y)*y/2+sumf)/pisqr;
55          else
56             e1=(exp(-y*y)*(cs-1)/4/x+2*x*sume)/pisqr;
57             f1=(exp(-y*y)*sn/4/x+sumf)/pisqr;
58          end
59          if abs(e1)<1e-100, e1=0; end
60          z1=2*j*(e1+j*f1)+j*pisqr*exp(-y*y)*(cs+j*sn)*cerfexp(-x);
61          res(irow,icol)=z1;
62        end
63      end
```

The following function is necessary at some argument values:

GUPHD:guisdap:m150:cerfexp.m

```
1     % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2     %
3     % cerfx.m
4     %
5     % Approximation for the function  (1-erf(x))*exp(x**2) with
6     % accuracy 1e-8 for all x.  Method: power series for
7     % erf(x) for small x, asymptotic continued fractions expansion
8     % for large x (Abramovitz-Stegun formula 7.1.14)
9     %                                              Markku Lehtinen 2.7.1979
10    %
11    % res=cerfexp(x)
12    %
13      function res=cerfexp(x)
14    %
15      x1=abs(x);
16      sqx=x*x;
17         if (x1<1.)
18            tail=0.;
19            term=sqx*sqx/10.;
20            erf=1.-sqx/3.+term;
21            for n=3:12
22               term=-term*sqx*(2*n-1.)/(2*n+1.)/n;
23               tail=tail+term;
24            end
25            erf=(erf+tail)*x*2./sqrt(pi);
26            res=(1-erf)*exp(sqx);
27          else
28            dum=0;
29            for n=(55:-1:1)
30               dum=n/2/(dum+x1);
31             end
32             res=1/(dum+x1)/sqrt(pi);
33             if (x<0), res=2*exp(sqx)-res; end
34          end
```

The asymptotic formula for the plasma dispersion function is

$$\mathtt{pldf}(z) \approx z^{-1}(1 + 1/z^2 + 1 \cdot 3/(z^2)^2 + 1 \cdot 3 \cdot 5/(z^2)^2 + \ldots),$$

and the expansion is valid for z not close to positive y-axis. We must require that $|\arg z - \pi/2| > \pi/4$.

GUPHD:guisdap:m150:pldfas.m

```
1     % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2     %
3     % pldfas.m
4     %
5     % asymptotic formula for the plasma dispersion function
6     % n is the order of the expansion
7     %
8     % res=pldfas(z,n)
9     %
10      function res=pldfas(z,n)
11    %
12      if max(size(z))==0, res=[]; return, end
13      term=1;
14      res=1;
15      for i=1:n
16        term=term*(2*i-1)./(2*z.^2);
17        res=res+term;
18      end
```

```
19        res=res./z;
```

The following function can be used to create different tabulations of the plasma dispersion function

GUPHD:guisdap:m150:pldftab.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % pldftab.m
4       %
5       % function to create the plasma dispersion function table
6       %
7       % res=pldftab(dx,dy,nx,ny,nx1,ny1)
8       %
9         function res=pldftab(dx,dy,nx,ny,nx1,ny1)
10      %
11        res=zeros(nx*ny,1);
12        for i=1:nx
13          disp(i);
14          for j=1:ny
15            res(i+(j-1)*nx,1)=pldf( (nx1+i-1)*dx +sqrt(-1)*(ny1+j-1)*(-dy) );
16          end
17        end
```

The following `m-file` was used to create the present pldfv. The tabulation interval is 0.075 in both directions, three x-values below zero, first y-value zero, 61 x-values leading to last value being (61-4)*.075=4.275 and 48 y-values leading to last value being -47*0.075=-3.525.

pldftabb.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % script file to create and save the plasma dispersion function table
4       %
5         pldfv=pldftab(0.075,0.075,61,48,-3,0);
6         save pldfv pldfv;
```

In the mex version of `spec.m` the same spectrum is calculated, but slightly different ways of plasma dispersion function interpolation is used. Lagrangian interpolation is used in this case also, but the interpolation is not calculated directly from the grid `pldfv`. Instead, coefficients of the interpolation polynimial are precalculated and stored in a variable `pldfvv`. Moreover, what is stored is not actually polynomial coefficients but instead coefficients a_i in an expansion of the form

$$\mathrm{pldf}(z) = ((a_1 z + a_2)z + a_3)z + a_4$$

in each of the interpolation cells (with z relative to a corner point). The table `pldfvv` then becomes four times as large as the table `pldfv`, but with present memories this is not a problem and calculations become much faster.

pldfvv_make.m

```
1       % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2       %
3       % find a polynomial
4       %  p(z)=((a1*z+a2)*z+a3)*z+a4
5       % with p(-1)=f-1,   p(0)=f0,   p(1)=f1   and  p(2)=f2
6       Q=[..
7       -1 1 -1 1
8        0 0  0 1
9        1 1  1 1
10       8 4  2 1];
11      Q=inv(Q);
12
13      f=zeros(16*4+3, 16*4+1);
14      for m=-1:1:(16*4+1),
15        for n=0:1:(16*4),
16           z=(m-j*n)/16;
17           f(m+2,n+1)=pldf(z);
18        end;
19        disp(m);
20      end;
21
22      pldfvv=zeros(64*65*4,1);
23      for n=0:1:(16*4),
24        for m=0:1:(16*4-1),
25          pldfvv(64*4*n+4*m + (1:4))  = Q*f( (m+1):(m+4) , n+1 );
26        end;
27        disp(n);
28      end;
29      save pldfvv pldfvv
```