

Exercise in FYS-3002 — Solutions

Eirik Rolland Enger

March 12, 2021

1 Problems

Problem 1

Task 1

What is the physical interpretation of χ in eq. (3)?

Equation (3) is what is given in FH (4.35). The parameter χ describe the susceptibility of the medium, that is, the electron population and the ion population in the plasma. When we have no magnetic field the susceptibility is a scalar while as a magnetic field is introduced the susceptibility is a tensor. However, only motion parallel to the applied force represents work done on the system, so if the force is in the z direction we need only one component of the tensor. The value of χ describe how prone the medium is of being magnetized by an external magnetic field, as such, we get information about the long term behaviour of the plasma through the susceptibility function. The χ functions incorporate the effects of the Coulomb interactions.

Task 2

Explain what the term $f_{\alpha,1}$ in eq. (2) describe.

This equation describe the perturbation of the phase space function and is the same as FH (4.26) only written out in full and using the Fourier transformed spatial variable and Laplace transformed temporal variable.

Problem 2

Using any one of the expressions for $\langle |n_e(\mathbf{k}, \omega)|^2 \rangle$, write a program that calculates the power spectral density. The program should accept a number of input parameters:

- f_r Radar frequency
- n_e Electron number density
- B Magnetic field strength
- m_i Ion mass
- T_e Electron temperature
- T_i Ion temperature
- θ Aspect angle (the angle between the radar pointing direction and the magnetic field.)

The code should be well commented and included as an appendix.

Explain the code. Some things to consider:

- Where in the code were the different equations solved?
- How was the numerical calculation implemented?

The code itself should be well commented and included as an appendix.

Hint: Before you integrate all the way to infinity: where along the axis of integration does most of the information lie?

Problem 3

We will now look at some specific parameters using our program. Run your program with the parameters given as:

Parameter	Unit	Value
f_r	[Hz]	430×10^6
n_e	$[\text{m}^{-3}]$	2×10^{10}
B	[T]	3.5×10^{-5}
m_i	[amu]	16
T_e	[K]	200
T_i	[K]	200
θ	[°]	135

for frequencies $f \in [-2 \times 10^6, 2 \times 10^6]$.

Task 1

Where could an experiment with these parameters be done? Make a sketch that includes the radar beam and the magnetic field line. Assume that the radar points directly upwards, i.e., towards zenith.

The radar beam will be scattered at some height, say $h \approx 200$ km. The aspect angle is the angle between the incident wave vector and the magnetic field line at the altitude where the radar beam is scattered.

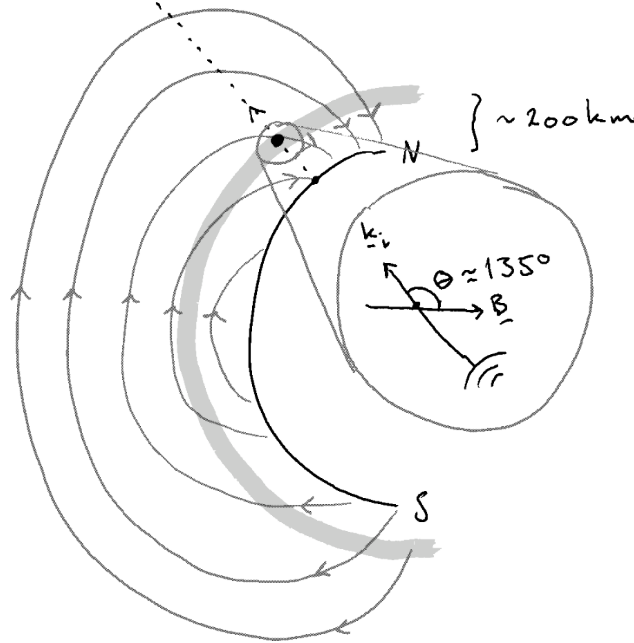


Figure 1: Sketch of a possible geometry based on the parameters given.

Task 2

The spectrum is plotted for frequencies $f \in [-2 \times 10^6, 2 \times 10^6]$; relative to an observer at the radar location, which way does the features found at positive frequencies is the spectrum move?

Structures that move towards the Earth will result in structures in the power spectrum at positive frequencies.

Task 3

Plot the resulting power spectra calculated by the program and explain what the different peaks represent.

The plot from problem 3 should result in something similar to fig. 2.

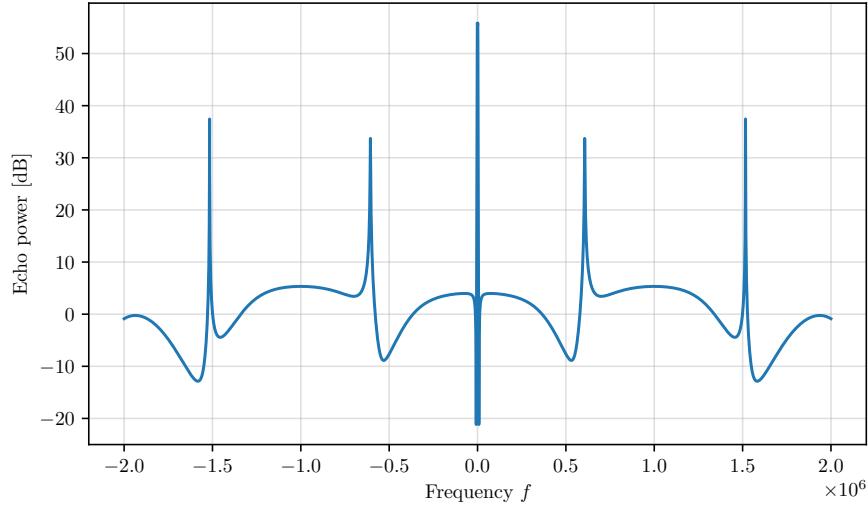


Figure 2: Power spectral density using the parameters above.

The peaks at largest frequency represent the plasma line which describe the electron population, the lines in the middle at frequency $f \approx 0.5$ Hz are gyro lines that appear due to the aspect angle we use (not parallel to the magnetic field line) and the lines at zero frequency are the ion lines.

Problem 4

In this exercise we will use the parameters:

Parameter	Unit	Value
f_r	[Hz]	933×10^6
n_e	$[m^{-3}]$	2×10^{11}
B	[T]	5×10^{-5}
m_i	[amu]	16
T_i	[K]	2000
θ	[°]	180

Task 1

Calculate the power spectral density on $f \in [3.5 \times 10^6, 7 \times 10^6]$ for $T_e = 2000$ K, 4000 K, 6000 K, and 8000 K and plot the power spectra.

The plots from the above should look similar to fig. 3.

Task 2

Explain the changes that can be seen as the electron temperature changes.

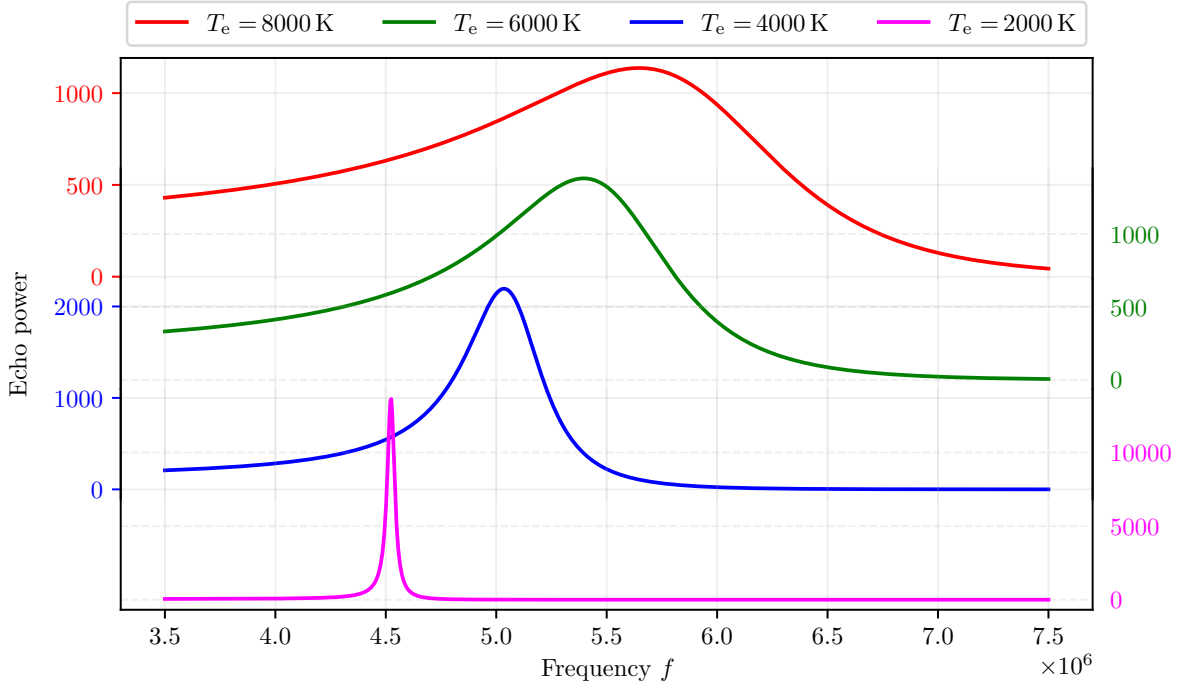


Figure 3: Power spectral density plot obtained with the parameters for problem 4.

This is showing the upshifted plasma line. The equation describing the real part of the plasma line wave resonance frequency is

$$\omega_{\Re,e} = [\omega_{pe}^2(1 + 3k^2\lambda_D^2) + \Omega_e^2 \sin^2 \theta]^{1/2} = [\omega_{pe}^2 + 3k^2v_{th,e}^2 + \Omega_e^2 \sin^2 \theta]^{1/2}$$

and we see that with increasing temperature the thermal velocity will increase, thus increasing the resonance frequency as seen in the fig. 3.

Task 3

Explain what the assumption $k^2\lambda_D^2 \ll 1$ is referring to. Is this assumption valid for all temperatures? Why/why not?

We also see from the figure that the width of the plasma line gets wider as the temperature is increased. The assumption $k^2\lambda_D^2 \ll 1$ is usually applied when solving the IS spectrum, implying weak Landau damping. It describe the kind of scattering we are interested in, which is scattering from the larger structures with length scale equal to the Debye length. But with increasing temperature this assumption is no longer valid (the Debye length is proportional to the square root of the temperature) and the Landau damping gets stronger. More power is distributed to the shoulders and therefore the peak power is also decreased to maintain the same power of the plasma line for all temperatures.

Problem 5

You should now be able to experiment a bit for yourself.

Task 1

Explain which parameter(s) that needs to be changed to obtain a similar plot as is shown in fig. 4.

We recognize this as the same as the plot in the compendium by Bjørnå (last page, top panel). If we change the ion temperature and keep the ratio between ion and electron temperature the same we get the desired result. Specifically, we can use $T_i = 100, 200, 300, 400$ K and a ratio to electron temperature given as $T_e/T_i = 1.5$.

Task 2

Reproduce the plot in fig. 4 using your own program. Include values on the axis and labels to all spectra.

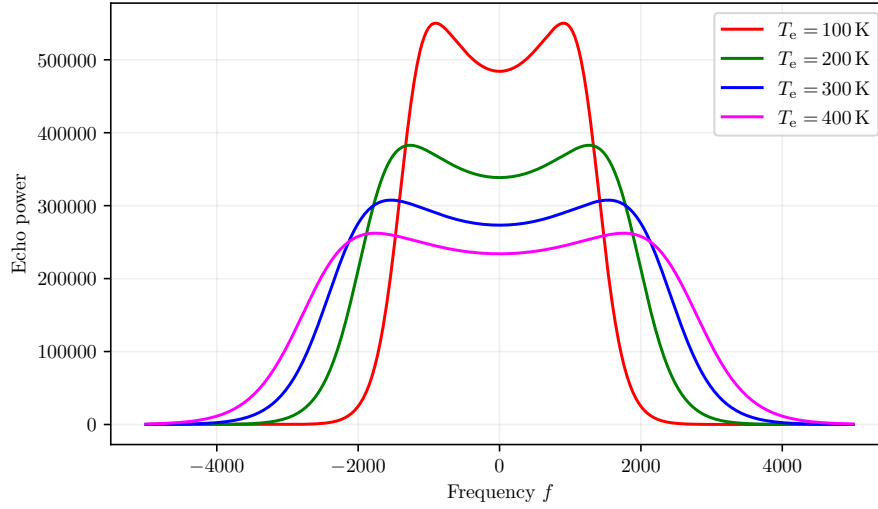


Figure 4: Power spectral density plot

A Appendix

A.1 Main program

isr.py solves the incoherent scatter spectrum.

```
1 import sys
2 import numpy as np
3 import scipy.constants as const
4 import scipy.integrate as si
5 import matplotlib.pyplot as plt
6
7 import config as cf
8 plt.rcParams.update({
9     'text.usetex': True,
10    'font.family': 'DejaVu Sans',
11    'axes.unicode_minus': False,
12 })
13
14 def isr(params=None) -> np.ndarray:
15     """Calculate an incoherent scatter spectrum.
16
17     Parameters
18     -----
19     params : dict
20         Give the physical parameters of the IS experiment.
21         'N_f' -- number of data points along the frequency axis
22         'f_min' -- lower bound of frequency axis
23         'f_max' -- upper bound of frequency axis
24         'f' -- radar frequency
25         'T_e' -- electron temperature
26         'T_i' -- ion temperature
27         'n_e' -- electron number density
28         'B' -- magnetic field strength
29         'aspect' -- aspect angle
30         'M' -- ion mass in amu
31
32     Returns
33     -----
34     np.ndarray, np.ndarray
35         The first array is the frequency axis while
36         the second is the echo power at each frequency
37     """
38     # Set physical parameters
39     if params is None:
40         N_f = cf.N_F
41         f_min = -2e6
42         f_max = 2e6
43         f = cf.f # 1/s - radar frequency
44         T_e = cf.T_e # K - electron temperature
45         T_i = cf.T_i # K - ion temperature
46         n_e = cf.n_e # 1/m^3 - electron number density
47         B = cf.B # T - magnetic field strength (towards Earth)
48         aspect = cf.aspect # degree - radar pointing direction to magnetopause
49         aspect = np.pi / 180 * aspect
```

```

50     M_amu    = cf.M                                # amu          - ion mass
51     M        = M_amu * (const.m_p + const.m_n) / 2 # Convert to kg
52 else:
53     N_f      = params['N_f']
54     f_min    = params['f_min']
55     f_max    = params['f_max']
56     f        = params['f']
57     T_e      = params['T_e']
58     T_i      = params['T_i']
59     n_e      = params['n_e']
60     B        = params['B']
61     aspect   = params['aspect']
62     aspect   = np.pi / 180 * aspect
63     M_amu    = params['M']
64     M        = M_amu * (const.m_p + const.m_n) / 2 # Convert to kg
65 nu = 0      # 1/s - collision frequency
66
67 # Calculate constants
68 k      = - 4 * np.pi * f / const.c
69 l_D    = debye(T_e, n_e)
70 w_c    = gyro('e', B)
71 W_c    = gyro('i', B, M_amu)
72
73 # Susceptibility
74 f_ax   = np.linspace(f_min, f_max, N_f) # Frequency axis
75 # Integration variable of Gordeyev
76 y_e    = np.linspace(0, 1.5e-4*(1 / cf.ORDER), cf.N_Y)**cf.ORDER
77 y_i    = np.linspace(0, 1.5e-2*(1 / cf.ORDER), cf.N_Y)**cf.ORDER
78 G_e    = maxwellian_integrand(y_e, nu, k, aspect, T_e, w_c, const.m_e)
79 G_i    = maxwellian_integrand(y_i, nu, k, aspect, T_i, W_c, M)
80 Fe     = F(f_ax, y_e, nu, G_e)
81 Fi     = F(f_ax, y_i, nu, G_i)
82
83 Xp     = (1 / (2 * l_D**2 * k**2))**(1 / 2)
84 chi_e  = 2 * Xp**2 * Fe
85 chi_i  = 2 * Xp**2 * Fi
86
87 # Calculate the IS spectrum
88 with np.errstate(divide='ignore', invalid='ignore'):
89     IS = n_e / (np.pi * 2 * np.pi * f_ax) * \
90         (np.imag(Fe) * np.abs(1 + chi_i)**2 + np.imag(Fi) * np.abs(chi_e)**2) / \
91         (np.abs(1 + chi_e + chi_i)**2)
92
93     return f_ax, IS
94
95 def F(f_ax:np.ndarray, y:np.ndarray, nu:float, G:np.ndarray) -> np.ndarray:
96     """Calculate the helper function 'F'.
97
98     Parameters
99     -----
100     f_ax : np.ndarray
101         The frequency axis
102     y : np.ndarray
103         Axis of integration in the Gordeyev integral

```

```

104     nu : float or int
105         The collision frequency
106     G : np.ndarray
107         The integrand in the Gordeyev integral
108
109     Returns
110     -----
111     np.ndarray
112         The 'F' function
113     """
114     # Calculate the F functions that include susceptibility
115     a = np.array([])
116     for f in f_ax:
117         w = 2 * np.pi * f
118         sint = my_integration_method(w, y, G)
119         a = np.r_[a, sint]
120
121     func = 1 + (1j * 2 * np.pi * f_ax + nu) * a
122     return func
123
124 def maxwellian_integrand(
125     y:np.ndarray,
126     nu:float,
127     k:float,
128     aspect:float,
129     T:float,
130     w_c:float,
131     m:float
132 ) -> np.ndarray:
133     """Calculate a Maxwellian integrand for a Gordeyev integral.
134
135     Parameters
136     -----
137     y : np.ndarray
138         Axis of integration in the Gordeyev integral
139     nu : float or int
140         The collision frequency
141     k : float
142         The radar wave number
143     aspect : float
144         The aspect anngle in radians
145     T : float
146         Temperature
147     w_c : float
148         Gyro frequency
149     m : float
150         Mass in kg
151
152     Returns
153     -----
154     np.ndarray
155         The Maxwellian integrand to be used in the Gordeyev integral
156     """
157     G = np.exp(- y * nu -

```



```

158         k**2 * np.sin(aspect)**2 * T * const.k /
159         (m * w_c**2) * (1 - np.cos(w_c * y)) -
160         .5 * (k * np.cos(aspect) * y)**2 * T * const.k / m)
161
162     return G
163
164 def my_integration_method(w:float, y:np.ndarray, G:np.ndarray) -> np.ndarray:
165     """A simple wrapper for integrating of the Gordeyev integral.
166
167     Parameters
168     -----
169     w : float
170         The angular/signed frequency to be evaluated
171     y : np.ndarray
172         Axis of integration in the Gordeyev integral
173     G : np.ndarray
174         The integrand in the Gordeyev integral
175
176     Returns
177     -----
178     np.ndarray
179         The value of the Gordeyev integral at each frequency data points
180     """
181     val = np.exp(1j * w * y) * G
182     sint = si.simps(val, y)
183     return sint
184
185 def debye(T:float, n:float) -> float:
186     """Calculate the Debye length for electrons.
187
188     Parameters
189     -----
190     T : float
191         Temperature
192     n : float
193         Electron number density
194
195     Returns
196     -----
197     float
198         The Debye length
199     """
200     ep0 = 1e-9 / 36 / np.pi
201     l_D = (ep0 * const.k * T / (n * const.e**2))**(1 / 2)
202     return l_D
203
204 def gyro(p:str, B:float, m=16) -> float:
205     """Calculate the gyro frequency of a particle species.
206
207     Parameters
208     -----
209     p : str
210         A string specifying the particle species
211         ({'e', 'i'} are recognized as electrons and ions)

```

```

212     B : float
213         The magnetic field strength
214     m : float (default: 16)
215         Particle mass in amu
216
217     Returns
218     -----
219     float
220         The gyro frequency
221     """
222     if p == 'e':
223         w = const.e * B / const.m_e
224     elif p == 'i':
225         w = const.e * B / (m * (const.m_p + const.m_n) / 2)
226     else:
227         sys.exit(f'I do not know what kind of particle {p} is.')
228     return w
229
230 def plot(dB=True):
231     x, y = isr()
232     y = 10 * np.log10(y) if dB else y
233
234     plt.figure(figsize=(7,4))
235     plt.plot(x, y)
236     plt.xlabel('Frequency $f$')
237     plt.ylabel('Echo power [dB]')
238     plt.grid(alpha=0.4)
239     # plt.savefig('gyrolines.pdf')
240     plt.show()
241
242 if __name__ == '__main__':
243     print('Calculating isr spectrum...')
244     plot()

```

A.2 Solutions script

The script `soln.py` uses `isr.py` with different input parameters and solves the problems / creates the plots seen in all exercises.

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  import isr
4  import py_plot as pylt
5
6  def prob3():
7      params = {
8          'N_f': int(1e3), 'f_min': -2e6, 'f_max': 2e6, 'f': 430e6, 'n_e': 2e10,
9          'B': 3.5e-5, 'M': 16, 'T_e': 200, 'T_i': 200, 'aspect': 135
10         }
11     x, y = isr.isr(params)
12     y = 10 * np.log10(y)
13
14     plt.figure(figsize=(7,4))
15     plt.plot(x, y)
16     plt.xlabel('Frequency $f$')

```

```

17     plt.ylabel('Echo power [dB]')
18     plt.grid(alpha=0.4)
19     # plt.savefig('gyrolines.pdf')
20     plt.show()
21
22 def prob4():
23     params = {
24         'N_f': int(1e3), 'f_min': 3.5e6, 'f_max': 7.5e6, 'f': 933e6, 'n_e': 2e11,
25         'B': 5e-5, 'M': 16, 'T_e': 2000, 'T_i': 2000, 'aspect': 180
26     }
27
28     T_E = [8000, 6000, 4000, 2000]
29     data = []
30     for T_e in T_E:
31         params['T_e'] = T_e
32         x, y = isr.isr(params)
33         data.append((x, y))
34
35     lab = [r'$T_{\mathrm{e}}=\backslash,\$' + f'\{t_e\}' + r'\backslash,\mathrm{K}\$' for t_e in T_E]
36     plt.figure('temps', figsize=(7,4))
37     pyplt.ridge_plot(data, 'squeeze', 'grid', xlabel='Frequency $f$', \
38                     ylabel='Echo power', labels=lab, figname='temps')
39     # for d, l in zip(data, lab):
40     #     plt.plot(d[0], d[1], label=l)
41     # plt.legend()
42     plt.savefig('temps.pdf')
43     plt.show()
44
45 def prob5():
46     params = {
47         'N_f': int(1e3), 'f_min': - 5e3, 'f_max': 5e3, 'f': 430e6, 'n_e': 2e10,
48         'B': 5e-5, 'M': 16, 'T_e': 300, 'T_i': 200, 'aspect': 180
49     }
50
51     T_I = [400, 300, 200, 100]
52     data = []
53     for T_i in T_I:
54         params['T_i'] = T_i
55         params['T_e'] = T_i * 1.5
56         x, y = isr.isr(params)
57         data.append((x, y))
58
59     lab = [r'$T_{\mathrm{e}}=\backslash,\$' + f'\{t_e\}' + r'\backslash,\mathrm{K}\$' for t_e in T_I]
60     plt.figure('temps', figsize=(7,4))
61     c = ['r', 'g', 'b', 'magenta']
62     plt.grid(True, which="major", ls="--", alpha=0.2)
63     for (x, y), col, l in zip(data[:-1], c, lab[:-1]):
64         plt.plot(x, y, f'\{col\}', label=l)
65     plt.legend()
66     # plt.tick_params(axis='both', which='both', labelbottom=False, labelleft=False)
67     plt.xlabel(r'Frequency $f$')
68     plt.ylabel('Echo power')
69     # pyplt.ridge_plot(data, 'squeeze', 'grid', xlabel='Frequency $f$', \
70     #                 ylabel='Echo power [dB]', labels=lab, figname='temps', ylim=(- 7e4, 6e5))

```

```
71     # plt.savefig('ionline_soln.pdf')
72     plt.show()
73
74 if __name__ == '__main__':
75     prob4()
```