Exercise in FYS-3002 — Solutions

Eirik Rolland Enger

March 3, 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. In this context the susceptibility is a scalar and its value 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.

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 $\langle |n_\mathrm{e}(\boldsymbol{k},\omega)|^2\rangle$, 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}	[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 $h \approx 200\,\mathrm{km}$. 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.

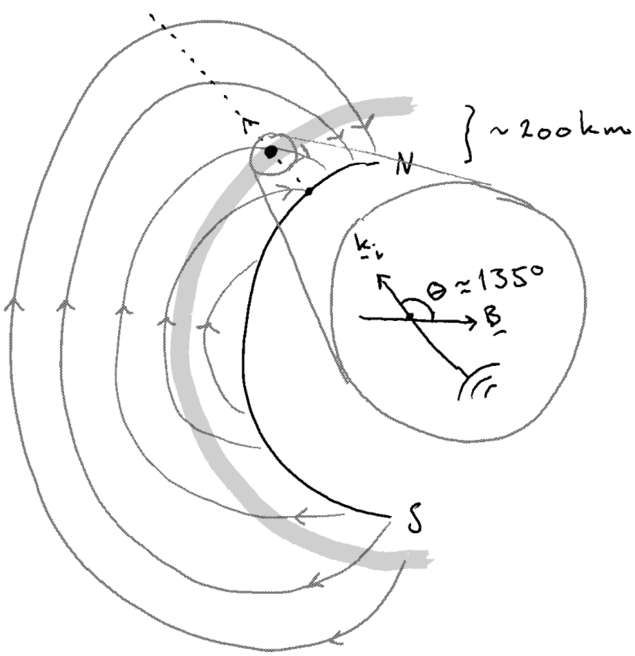

Figure 1: Sketch of a possible geometry based on the parameters given.

Task 2

Thespectrum 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.

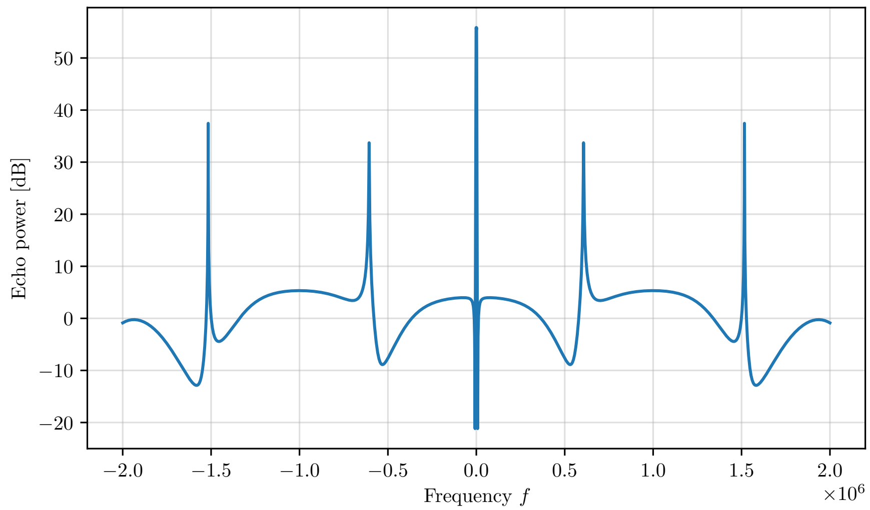

Figure 2: Power spectral density using the parameters above.

The peaks at largest frequency represent the plasma line which describe the electron population, the lines in the middle at frequency $f \approx 0.5\,\mathrm{Hz}$ 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 $T_\mathrm{e} = 2000\,\mathrm{K}$, $4000\,\mathrm{K}$, $6000\,\mathrm{K}$, and $8000\,\mathrm{K}$ 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 temperture changes.

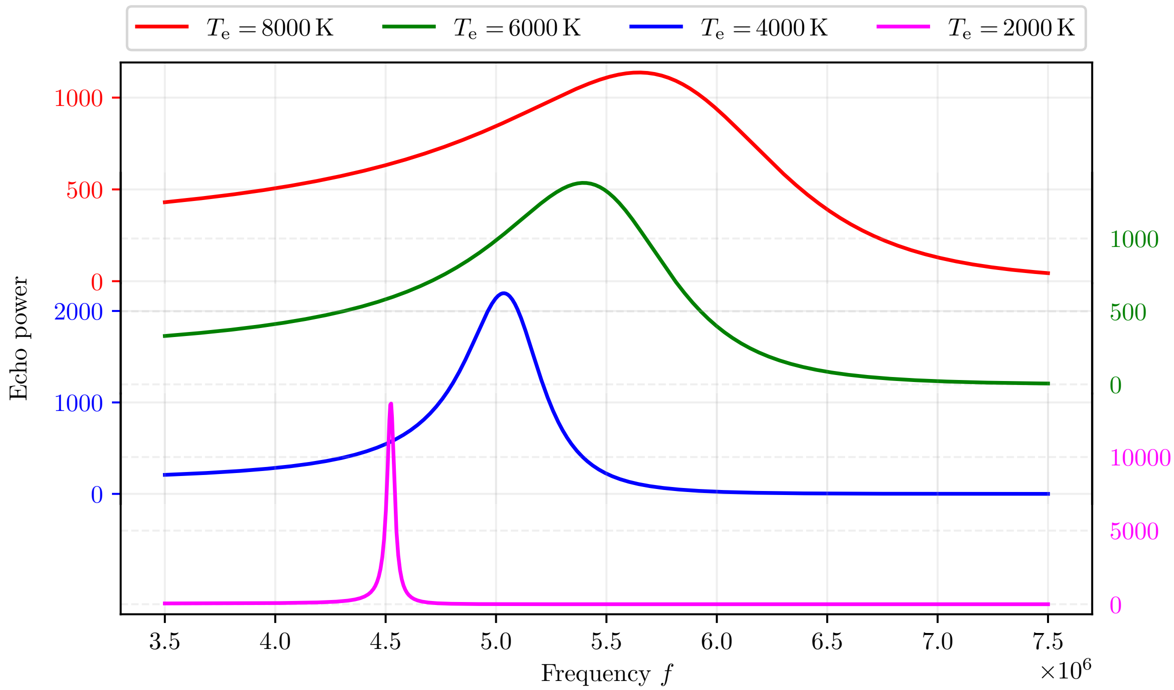

Figure 3: Power spectral density plot obtained with the parameters for problem 4.

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

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

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 $k^2\lambda_\mathrm{D}^2 \ll 1$ 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 $k^2\lambda_\mathrm{D}^2 \ll 1$ 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 $T_\mathrm{i} = 100, 200, 300, 400\,\mathrm{K}$ and a ratio to electron temperature given as $T_\mathrm{e}/T_\mathrm{i} = 1.5$.

Task 2

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

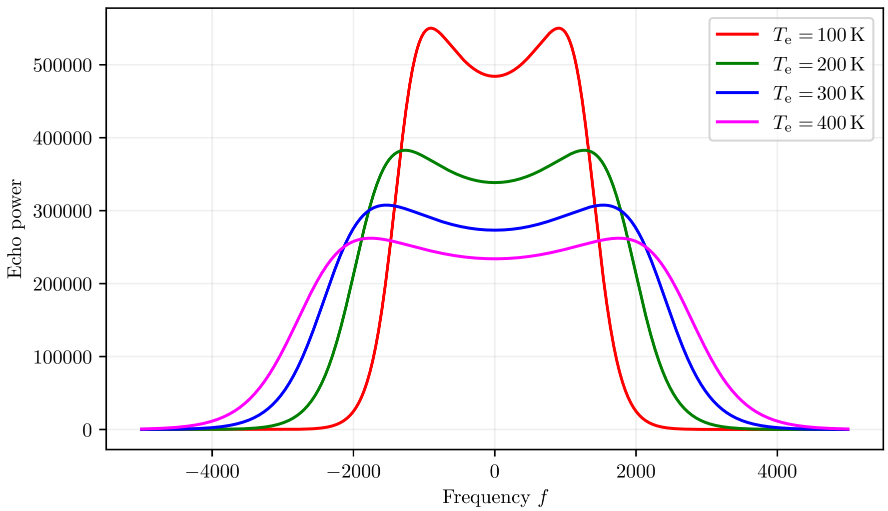

Figure 4: Power spectral density plot

A Appendix

A.1 Main program

isr.py solves the incoherent scatter spectrum.

```
import sys
import numpy as np
import scipy.constants as const
import scipy.integrate as si
import matplotlib.pyplot as plt

import config as cf
plt.rcParams.update({
    'text.usetex': True,
    'font.family': 'DejaVu Sans',
    'axes.unicode_minus': False,
})

def isr(params=None) -> np.ndarray:
    """Calculate an incoherent scatter spectrum.

    Parameters
    ----------
    params : dict
        Give the physical parameters of the IS experiment.
        'N_f' -- number of data points along the frequency axis
        'f_min' -- lower bound of frequency axis
        'f_max' -- upper bound of frequency axis
        'f' -- radar frequency
        'T_e' -- electron temperature
        'T_i' -- ion temperature
        'n_e' -- electron number density
        'B' -- magnetic field strength
        'aspect' -- aspect angle
        'M' -- ion mass in amu

    Returns
    -------
    np.ndarray, np.ndarray
        The first array if the frequency axis while
        the second is the echo power at each frequency
    """
    # Set physical parameters
    if params is None:
        N_f = cf.N_F
        f_min = -2e6
        f_max = 2e6
        f = cf.f  # 1/s - radar frequency
        T_e = cf.T_e  # K - electron temperature
        T_i = cf.T_i  # K - ion temperature
        n_e = cf.n_e  # 1/m^3 - electron number density
        B = cf.B  # T - magnetic field strength (towards Earth)
        aspect = cf.aspect  # degree - radar pointing direction to magnetic field line
        aspect = np.pi / 180 * aspect
```

```
        M_amu = cf.M  # amu - ion mass
        M = M_amu * (const.m_p + const.m_n) / 2  # Convert to kg
    else:
        N_f = params['N_f']
        f_min = params['f_min']
        f_max = params['f_max']
        f = params['f']
        T_e = params['T_e']
        T_i = params['T_i']
        n_e = params['n_e']
        B = params['B']
        aspect = params['aspect']
        aspect = np.pi / 180 * aspect
        M_amu = params['M']
        M = M_amu * (const.m_p + const.m_n) / 2  # Convert to kg
    nu = 0  # 1/s - collision frequency

    # Calculate constants
    k = - 4 * np.pi * f / const.c
    l_D = debye(T_e, n_e)
    w_c = gyro('e', B)
    W_c = gyro('i', B, M_amu)

    # Susceptibility
    f_ax = np.linspace(f_min, f_max, N_f)  # Frequency axis
    # Integration variable of Gordeyev
    y_e = np.linspace(0, 1.5e-4**(1 / cf.ORDER), cf.N_Y)**cf.ORDER
    y_i = np.linspace(0, 1.5e-2**(1 / cf.ORDER), cf.N_Y)**cf.ORDER
    G_e = maxwellian_integrand(y_e, nu, k, aspect, T_e, w_c, const.m_e)
    G_i = maxwellian_integrand(y_i, nu, k, aspect, T_i, W_c, M)
    Fe = F(f_ax, y_e, nu, G_e)
    Fi = F(f_ax, y_i, nu, G_i)

    Xp = (1 / (2 * l_D**2 * k**2))**(1 / 2)
    chi_e = 2 * Xp**2 * Fe
    chi_i = 2 * Xp**2 * Fi

    # Calculate the IS spectrum
    with np.errstate(divide='ignore', invalid='ignore'):
        IS = n_e / (np.pi * 2 * np.pi * f_ax) * \
                (np.imag(Fe) * np.abs(1 + chi_i)**2 + np.imag(Fi) * np.abs(chi_e)**2) / \
                (np.abs(1 + chi_e + chi_i)**2)

    return f_ax, IS

def F(f_ax:np.ndarray, y:np.ndarray, nu:float, G:np.ndarray) -> np.ndarray:
    """Calculate the helper function 'F'.

    Parameters
    ----------
    f_ax : np.ndarray
        The frequency axis
    y : np.ndarray
        Axis of integration in the Gordeyev integral
```

```python
    nu : float or int
        The collision frequency
    G : np.ndarray
        The integrand in the Gordeyev integral

    Returns
    -------
    np.ndarray
        The 'F' function
    """
    # Calculate the F functions that include susceptibility
    a = np.array([])
    for f in f_ax:
        w = 2 * np.pi * f
        sint = my_integration_method(w, y, G)
        a = np.r_[a, sint]

    func = 1 + (1j * 2 * np.pi * f_ax + nu) * a
    return func

def maxwellian_integrand(
        y:np.ndarray,
        nu:float,
        k:float,
        aspect:float,
        T:float,
        w_c:float,
        m:float
    ) -> np.ndarray:
    """Calculate a Maxwellian integrand for a Gordeyev integral.

    Parameters
    ----------
    y : np.ndarray
        Axis of integration in the Gordeyev integral
    nu : float or int
        The collision frequency
    k : float
        The radar wave number
    aspect : float
        The aspect anngle in radians
    T : float
        Temperature
    w_c : float
        Gyro frequency
    m : float
        Mass in kg

    Returns
    -------
    np.ndarray
        The Maxwellian integrand to be used in the Gordeyev integral
    """
    G = np.exp(- y * nu -
```

```
            k**2 * np.sin(aspect)**2 * T * const.k /
            (m * w_c**2) * (1 - np.cos(w_c * y)) -
            .5 * (k * np.cos(aspect) * y)**2 * T * const.k / m)

    return G

def my_integration_method(w:float, y:np.ndarray, G:np.ndarray) -> np.ndarray:
    """A simple wrapper for integrating of the Gordeyev integral.

    Parameters
    ----------
    w : float
        The angular/signed frequency to be evaluated
    y : np.ndarray
        Axis of integration in the Gordeyev integral
    G : np.ndarray
        The integrand in the Gordeyev integral

    Returns
    -------
    np.ndarray
        The value of the Gordeyev integral at each frequency data points
    """
    val = np.exp(1j * w * y) * G
    sint = si.simps(val, y)
    return sint

def debye(T:float, n:float) -> float:
    """Calculate the Debye length for elecrons.

    Parameters
    ----------
    T : float
        Temperature
    n : float
        Electron number density

    Returns
    -------
    float
        The Debye length
    """
    ep0 = 1e-9 / 36 / np.pi
    l_D = (ep0 * const.k * T / (n * const.e**2))**(1 / 2)
    return l_D

def gyro(p:str, B:float, m=16) -> float:
    """Calculate the gyro frequency of a particle species.

    Parameters
    ----------
    p : str
        A string specifying the particle species
        ({'e', 'i'} are recognized as electrons and ions)
```

```
    B : float
        The magnetic field strength
    m : float (default: 16)
        Particle mass in amu

    Returns
    -------
    float
        The gyro frequency
    """
    if p == 'e':
        w = const.e * B / const.m_e
    elif p == 'i':
        w = const.e * B / (m * (const.m_p + const.m_n) / 2)
    else:
        sys.exit(f'I do not know what kind of particle {p} is.')
    return w

def plot(dB=True):
    x, y = isr()
    y = 10 * np.log10(y) if dB else y

    plt.figure(figsize=(7,4))
    plt.plot(x, y)
    plt.xlabel('Frequency $f$')
    plt.ylabel('Echo power [dB]')
    plt.grid(alpha=0.4)
    # plt.savefig('gyrolines.pdf')
    plt.show()

if __name__ == '__main__':
    print('Calculating isr spectrum...')
    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.

```
import numpy as np
import matplotlib.pyplot as plt
import isr
import py_plot as pylt

def prob3():
    params = {
            'N_f': int(1e3), 'f_min': -2e6, 'f_max': 2e6, 'f': 430e6, 'n_e': 2e10,
            'B': 3.5e-5, 'M': 16, 'T_e': 200, 'T_i': 200, 'aspect': 135
            }
    x, y = isr.isr(params)
    y = 10 * np.log10(y)

    plt.figure(figsize=(7,4))
    plt.plot(x, y)
    plt.xlabel('Frequency $f$')
```

```python
    plt.ylabel('Echo power [dB]')
    plt.grid(alpha=0.4)
    # plt.savefig('gyrolines.pdf')
    plt.show()

def prob4():
    params = {
            'N_f': int(1e3), 'f_min': 3.5e6, 'f_max': 7.5e6, 'f': 933e6, 'n_e': 2e11,
            'B': 5e-5, 'M': 16, 'T_e': 2000, 'T_i': 2000, 'aspect': 180
            }

    T_E = [8000, 6000, 4000, 2000]
    data = []
    for T_e in T_E:
        params['T_e'] = T_e
        x, y = isr.isr(params)
        data.append((x, y))

    lab = [r'$T_\mathrm{e}=\,$' + f'{t_e}' + r'$\,\mathrm{K}$' for t_e in T_E]
    plt.figure('temps', figsize=(7,4))
    pylt.ridge_plot(data, 'squeeze', 'grid', xlabel='Frequency $f$', \
            ylabel='Echo power', labels=lab, figname='temps')
    # for d, l in zip(data, lab):
    #     plt.plot(d[0], d[1], label=l)
    # plt.legend()
    plt.savefig('temps.pdf')
    plt.show()

def prob5():
    params = {
            'N_f': int(1e3), 'f_min': - 5e3, 'f_max': 5e3, 'f': 430e6, 'n_e': 2e10,
            'B': 5e-5, 'M': 16, 'T_e': 300, 'T_i': 200, 'aspect': 180
            }

    T_I = [400, 300, 200, 100]
    data = []
    for T_i in T_I:
        params['T_i'] = T_i
        params['T_e'] = T_i * 1.5
        x, y = isr.isr(params)
        data.append((x, y))

    lab = [r'$T_\mathrm{e}=\,$' + f'{t_e}' + r'$\,\mathrm{K}$' for t_e in T_I]
    plt.figure('temps', figsize=(7,4))
    c = ['r', 'g', 'b', 'magenta']
    plt.grid(True, which="major", ls="-", alpha=0.2)
    for (x, y), col, l in zip(data[::-1], c, lab[::-1]):
        plt.plot(x, y, f'{col}', label=l)
    plt.legend()
    # plt.tick_params(axis='both', which='both', labelbottom=False, labelleft=False)
    plt.xlabel(r'Frequency $f$')
    plt.ylabel('Echo power')
    # pylt.ridge_plot(data, 'squeeze', 'grid', xlabel='Frequency $f$', \
    #         ylabel='Echo power [dB]', labels=lab, figname='temps', ylim=(- 7e4, 6e5))
```

```python
    # plt.savefig('ionline_soln.pdf')
    plt.show()

if __name__ == '__main__':
    prob4()
```