10401
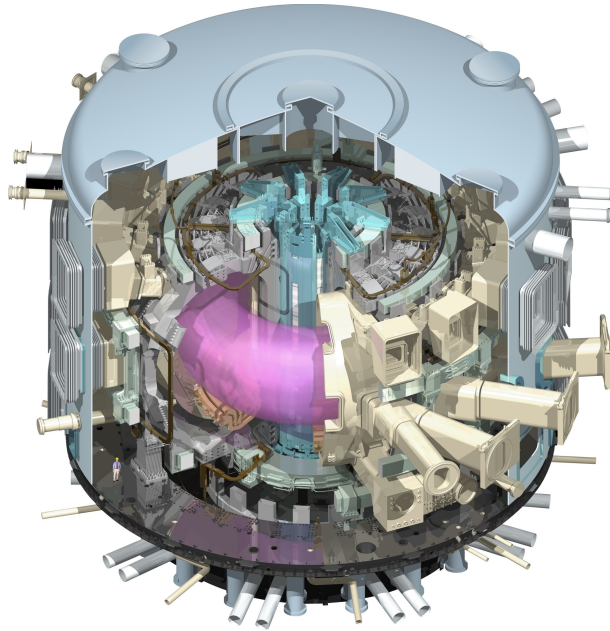
Fusion Energy and Plasma Physics

Rasmus Kronborg Finnemann Wiuff (s163977)[*] and Nicklas Kihm (s143286)[†]

Technical University of Denmark[‡]

(Dated: January 24th 2019)

Abstract: Plasma fusion for energy production is growing scientific field with interest all over the globe as clean and sustainable energy as well as growing energy demands need to be adressed. In the course 10401: Fusion energy and plasma physics we deal with some of the fundamental engineering aspects of a commercial fusion energy power plant as well as the research towards this goal. In this paper we first adress a simple model for such plant and how the model can be improved. Secondly we investigate how to diagnose plasma density with simple interferometry in the DTU reactor. Lastly we made measurements on a fusor and investigate some energy production aspects with such a device.

CONTENTS

A.   Intro

In recent years a large and quickly growing collaboration between plasma physcisists and engineers have materialised in one of the most ambitious energy producing projects ever seen. The proof-of-concept plasma fusion tokamak reactor ITER[1] in Cadarache is currently taking shape in order to adress the issue of growing energy demands and climate changes.

The mission is simple: Prove that plasma fusion is a viable source of electricity.

Whilst not being the most surmountable task, many researchers and institutions have gathered from across the world, including the Department of Physics at DTU.

In this paper, three assignments are solved as part of the course "10401 Fusion Energy and Fusion Plasma Physics". Some key aspects of fusion plasma fueled reactors are adressed and discussed in the assignments.

I.   A SIMPLE REACTOR MODEL

A.   Freidberg's simple reactor model

In the 5th chapter of the textbook by Freidberg[2], he makes a simple model for designing a fusion reactor power plant. The model uses simple geometric and electromagnetic assumptions with little involvement of plasma physics. The variables put into the model are shown in Table I. Table II shows the output quantities from the model. This model has been implemented in a Matlab script provided for the course. The script is shown included in Appendix A. As an example, the model is run with the following parameters:

$$n_{\text{flux fraction}} = 0.01 \qquad P_E = 1000 \text{ MW}$$
$$P_W = 4 \text{ MW·m}^{-2} \quad B_{\max} = 13 \text{ T} \qquad (1)$$
$$\sigma_{\max} = 3000 \text{ atm} \qquad \eta_t = 0.4$$

Note that C_F and C_I has been omitted as these serve no purpose for this assignment. It is not of interest how expensive the plant will be. Rather the geometries and physical quantities are of interest. The results from the model is given in Table II.

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24$^{\text{th}}$
2019

Symbol	Quantity
$n_{\text{flux fraction}}$	n flux in breeder end/n flux in breeder start $[]$
C_F	Fixed cost proportionality constant $[\$]$
C_I	Nuclear island cost proportionality constant $\left[\$ \cdot \text{W·m}^{-3}\right]$
P_E	Desired output power $[\text{MW}]$
P_W	Maximum wall load $\left[\text{MW·m}^{-2}\right]$
B_{max}	Magnetic field at the edge of the coil $[\text{T}]$
σ_{max}	Tensile strength of the magnetic field coils $[\text{atm}]$
η_t	Energy conversion efficiency $[]$

Table I: *Variables in the Freidberg's model*

B. Model sensitivity

At this point Freidberg has provided a model that produce some reasonable results for a power plant. It could be interesting to see how this model behaves when some vital parameters are changed. In the last section the model used the parameters shown in Eq. (1). In Eq. (2) are listed some parameter ranges.

$$P_E : 500 - 1000 \text{ MW} \quad P_W : 0 - 10 \text{ MW·m}^{-2}$$
$$B_{\text{max}} : 0 - 20 \text{ T} \quad \sigma_{\text{max}} : 2000 - 5000 \text{ atm} \tag{2}$$

Appendix B includes Matlab code that does the following:

1. Take a variable range from Eq. (2).

2. Set all other input parameters as in Eq. (1).

3. Plot the model outputs as functions of the variable range.

4. Repeat with another range from Eq. (2).

The iteration showed some interesting breakdowns in the model. When varying the desired power output there was a breakdown in magnetic field strength at the magnetic axis as well as in the normalised plasma pressure. This is shown in Fig. 1. In this case limits for what is physically possible broke and in these cases the model is unusable. The full gallery of model output is found in Appendix C.

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

Symbol	Quantity	Obtained values
b	Blanket-and-shield thickness	1.20 m
c	Magnet coil thickness	0.799 m
a	Minor radius	2.01 m
R_0	Major radius	4.96 m
A	Aspect ratio	2.4670
A_p	Plasma surface	393 m^2
V_p	Plasma volume	395 m^3
P_{dens}	Power density	4.97×10^6 W·m^{-1}
p	Plasma pressure	7.37×10^5 Pa
n	Particle density	1.53×10^{20} m^{-3}
B_0	Magnetic field at magnetic axis	4.57 T
β	Normalised plasma pressure	8.85%
$\tau_{E_{\text{min}}}$	Min confinement time for $(p \times \tau_E)_{\text{min}}$	1.14 s

Table II: *Output quantities in the model in Freidberg's[2] along with the obtained values when inserting the parameters from Eq. (1)*

C. Elliptic Cross section

Freidberg's model assumes a circular cross section of the plasma. In reality this is not the case, and as of such we will now make a more realistic, yet still approximate reactor for an elliptic plasma cross section. In describing the geometry one refers to the elongation ratio:

$$\kappa = \frac{a_{\text{max}}}{a_{\text{min}}} \tag{3}$$

With a_{max} the major radius and a_{min} the minor radius of the ellipse. This parameter ensures a true elliptic cross section as defined by the equation,

$$\frac{x^2}{a_{\text{max}}} + \frac{y^2}{a_{\text{min}}} = 1 \tag{4}$$

which can be parameterised as

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a_{\text{min}} \cos \phi \\ \kappa a_{\text{min}} \sin \phi \end{bmatrix} \tag{5}$$

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019
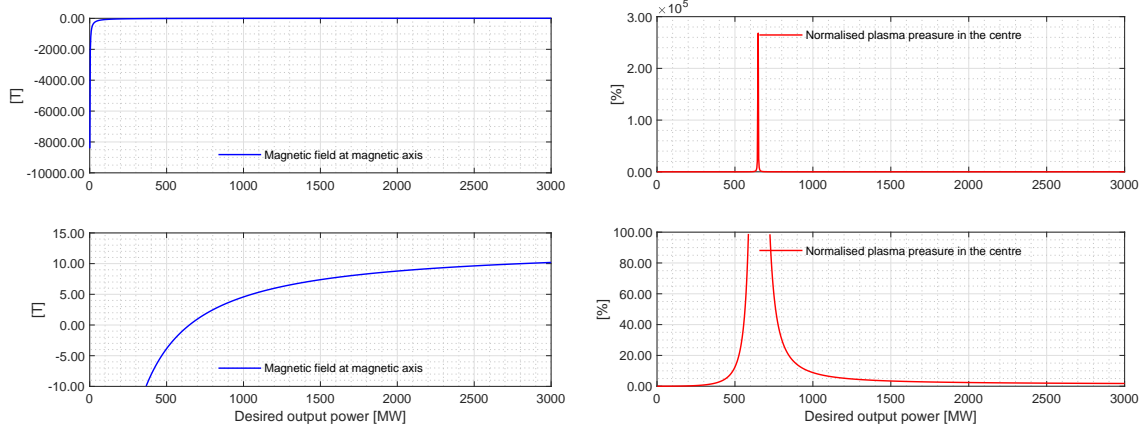
(a) The magnetic field at magnetic axis

(b) The normalised plasma pressure in the centre

Figure 1: Model breakdown when iterating over desired power output.

Meanwhile the blanket must be implemented as an ellipse or swelled ellipse. The true ellipse results in a difference of thickness in the blanket while the second results in a blanket of equal thickness throughout the structure. To this, the choice of implementing the blanket as a true ellipse has been made, since it simplifies derivations a bit. Note however, that keeping a constant thickness is the preferable option as it will reduce the engineering volume and hence the cost of the machine.

The outer layer is parameterised

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} (b + a_{min}) \cos \phi \\ \kappa(b + a_{min}) \sin \phi \end{bmatrix} \tag{6}$$

with b the blanket thickness at the minor ellipse axis. Choosing for now, $a_{min} = 2$, $\kappa = 2$ and $b = 1.2$, 5 and 6 are plotted on Figure 2 along with the variation in thickness of the blanket.

Given 5 and 6, the engineering volume can easily be derived if $c \cos(\phi)$ and $\kappa c \sin(\phi)$ is added to the x and y-direction in 6 respectively, where c is the minimum thickness of the magnetic coils that provide the torroidal field.

The cross sectional area of an ellipse is $A_e = \pi a_{min} a_{max}$ so the engineered volume becomes

$$V_I = 2 \pi R_0 \left(A_{e, outer} - A_{e, inner} \right)$$
$$= 2 \pi^2 R_0 \left((a_{min} + b + c)^2 - a_{min}^2 \right) \kappa \tag{7}$$
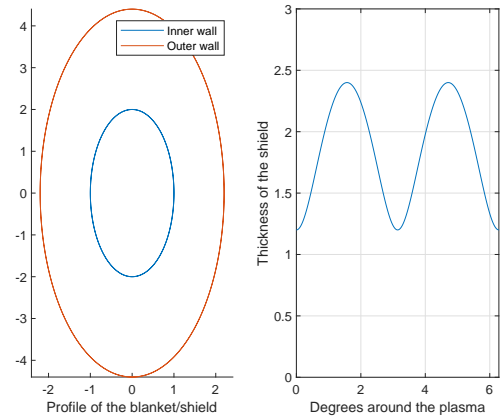
Figure 2: The profile of the blanket-and-shield and the thickness as a function of the angle around origo, ϕ.

and the plasma volume is similarly calculated as

$V_P = 2\pi^2 R_0 \kappa a_{min}^2$. The plasma surface area is a bit

more tricky, but it can be approximated as

$$S_p = 2\pi R_0 \pi \left(3\left(a_{min} + \kappa a_{min}\right) - \sqrt{\left(3 a_{min} + \kappa a_{min}\right)\left(a_{min} + 3\kappa a_{min}\right)}\right) \tag{8}$$

Using the same arguments as in the book, the B-field in the centre is surprisingly unchanged when going to the elliptical model. Now c is also approximated, or rather overestimated using a slight change to eq. (5.24) in the textbook. Since the force grows with a_{min} inserting κa_{min} instead yields an overestimation on the vertical force on the magnet. The tensile forces are the same, so the force balance leads to

$$c = \frac{2\xi}{1-\xi}(\kappa a + b) \tag{9}$$

with $\xi = B_{sic}^2 / 4\mu_0 \sigma_{max}$. These new equations are inserted in the code. The results are displayed in III. The plasma volume and surface area is of course increased as the plasma was made higher. This of course also results in a decreased power density. Overall, parameters such as B_0, β and $\tau_{E_{min}}$ while changing a bit, they were not changed significantly.

D.   Main parameters for DEMO

Setting $P_E = 2000$ in the elliptical model yields the output parameters seen in table IV. Since R_0 is directly proportional to the electric power this of course increases linearly. The other geometric output parameters regarding areas and volumes also increase as a result. β has decreased a lot, so the plasma is not confined effectively in DEMO.

E.   Designs for DEMO

With P_E, $\kappa = 2$ and $A = R_0/a_{min} = 3 \Leftrightarrow R_0 = 3 a_{min}$. This is implemented in the code and the results are displayed in V. β has increased a bit but only to 4.55%. R_0 has been forced down, so the plasma volume etc. has decreased as well. Overall it seems like a smaller R_0 while keeping a_{min} fixed is an improvement. Or in other words, $A = 3$ is more desirable than $A \approx 5$.

Symbol	Quantity	Obtained values
b	Blanket-and-shield thickness	1.20 m
c	Magnet coil thickness	1.30 m
a	Minor radius	2.01 m
R_0	Major radius	4.96 m
A	Aspect ratio	2.4670
A_p	Plasma surface	609 m²
V_p	Plasma volume	793 m³
P_{dens}	Power density	2.48×10^6 W·m⁻¹
p	Plasma pressure	5.20×10^5 Pa
n	Particle density	1.08×10^{20} m⁻³
B_0	Magnetic field at magnetic axis	4.60 T
β	Normalised plasma pressure	6.17%
$\tau_{E_{\mathrm{min}}}$	Min confinement time for $(p \times \tau_E)_{\mathrm{min}}$	1.62 s

Table III:

Output quantities from the elliptical model

F.   *A* and κ as free parameters

Designing the tokamak with κ and A as free parameters has led us to try and maximise profitability from V_P/A_P and β. Profitability in this context means that increasing the size of the tokamak leads to an increase in these parameters. However, there is a point when the change vs increase in size becomes constant. This means that we do not profit from increasing the size any longer.

V_P and A_P were calculated earlier so

$$\begin{aligned}
\frac{V_\mathrm{P}}{A_\mathrm{P}} &= \frac{2\,\pi\,R_0\,\kappa\,a_{\mathrm{min}}^2}{2\,\pi\,R_0\,\pi\left(3\left(a + \kappa\,a_{\mathrm{min}}\right) - \sqrt{\left(3\,a_{\mathrm{min}} + \kappa\,a_{\mathrm{min}}\right)\left(a + 3\,\kappa\,a_{\mathrm{min}}\right)}\right)} \\
&= \frac{\kappa\,a_{\mathrm{min}}}{\pi\left(3 + 3\,\kappa - \sqrt{3 + \kappa}\,\sqrt{1 + 3\,\kappa}\right)}
\end{aligned} \tag{10}$$

The expression scales linearly with a_{min} so we set it equal to 1m since it has no effect on the choice of κ. Differentiating with respect to κ and plotting is done, with the results shown in Figure 3a. Considering the figure, $\kappa = 2.5$ is chosen as the change is close enough to zero given this value.

Now work will be done towards choosing a value for A. First for geometric reasons a

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

Symbol	Quantity	Obtained values
b	Blanket-and-shield thickness	1.20 m
c	Magnet coil thickness	1.30 m
a	Minor radius	2.01 m
R_0	Major radius	9.95 m
A	Aspect ratio	4.9512
A_p	Plasma surface	1.21×10^3 m^2
V_p	Plasma volume	1.59×10^3 m^3
P_{dens}	Power density	2.48×10^6 W·m^{-1}
p	Plasma pressure	5.20×10^5 Pa
n	Particle density	1.08×10^{20} m^{-3}
B_0	Magnetic field at magnetic axis	8.80 T
β	Normalised plasma pressure	1.69%
$\tau_{E_{\text{min}}}$	Min confinement time for $(p \times \tau_E)_{\text{min}}$	1.62 s

Table IV: *Output quantities for DEMO using the elliptical model with $P_{\text{E}} = 2$GW*

minimum R_0 is calculated. It is simply $R_0 = a_{\text{min}} + b + c$ where a_{min} is to be determined, $b = 1.2$m and $c = \frac{2\xi}{1-\xi}(\kappa \, a_{\text{min}} + b)$. Meanwhile A is chosen to optimise β. Inserting R_0 into B_0 in the textbook yields

$$B_0 = \frac{2\,\xi\,(\kappa\,a_{\text{min}} + b)\,B_{max}}{((2\,\kappa - 1)\,a_{\text{min}} + a + b)} = \frac{178.75\,a_{\text{min}} + 85.8}{55.5 + 36\,a_{\text{min}}} \tag{11}$$

Where units has been disregarded and $\xi = 0.11$, $b = 1.2$, $\kappa = 2.5$ and $B_{\text{max}} = 13$ has been inserted. Meanwhile, inserting the textbook's expression for P_{dens} into the expression for p and inserting the input parameters yields

$$p = 1.042\text{E}6\,\sqrt{\frac{1}{\kappa\,a_{\text{min}}}} \tag{12}$$

Thus β becomes

$$\beta = \frac{2148\,(a_{\text{min}} + 1.542)^2}{\sqrt{a_{\text{min}}}\,(178.8\,a_{\text{min}} + 85.8)^2} \tag{13}$$

This function is plotted in Fig. 3b and for $(\beta = 10\%\ a = 1.938$m$)$ is achieved. Therefore $R_0 = 1.9378$m $+ 1.2$m $+ 2 \cdot 0.11/(1 - 0.11) \cdot 2.5 \cdot 1.938$m $+ 1.2$m $= 4.632$m which means $A = 4.632$m$/1.938$m $= 2.390$

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

Symbol	Quantity	Obtained values
b	Blanket-and-shield thickness	1.20 m
c	Magnet coil thickness	1.30 m
a	Minor radius	2.01 m
R_0	Major radius	6.03 m
A	Aspect ratio	3
A_p	Plasma surface	738 m^2
V_p	Plasma volume	962 m^3
P_{dens}	Power density	4.09×10^6 W·m^{-1}
p	Plasma pressure	6.68×10^5 Pa
n	Particle density	1.39×10^{20} m^{-3}
B_0	Magnetic field at magnetic axis	6.07 T
β	Normalised plasma pressure	4.55%
$\tau_{E_{\text{min}}}$	Min confinement time for $(p \times \tau_E)_{\text{min}}$	1.26 s

Table V: *Output quantities for DEMO using the elliptical model with $P_{siE} = 2$, $\kappa = 2$ and setting $A = 3$*
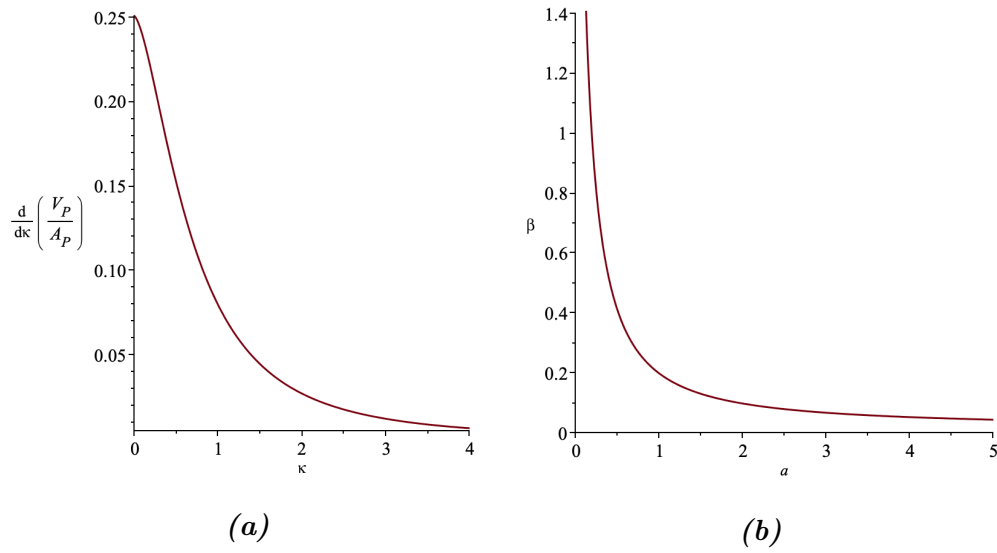
II.   DIAGNOSTICS VIA INTERFEROMETRY

When operating a fusion reactor a continuous process of diagnostics is necessary in order to optimise the plasma for the fusion process. One of the active diagnostic methods are interferometry. The goal here for this part of the paper is to measure the plasma electron density in the Danish Tokamak Undertaking reactor.

A.   Cutt-off and Source frequency

Using a interferometer one can measure the electron density n_e of the plasma. The refractive index of electromagnetic waves depend on the electron density and plasma frequency ω_p proportionally:

$$\omega_p^2 \propto n_e \tag{14}$$

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24$^{\text{th}}$
2019

(a)

(b)

Figure 3

Symbol	Quantity	Obtained values
b	Blanket-and-shield thickness	1.20 m
c	Magnet coil thickness	1.30 m
a	Minor radius	2.01 m
R_0	Major radius	2.60 m
A	Aspect ratio	1.29
A_p	Plasma surface	630 m^2
V_p	Plasma volume	414 m^3
P_{dens}	Power density	9.49×10^6 W·m^{-1}
p	Plasma pressure	1.02×10^5 Pa
n	Particle density	2.12×10^{20} m^{-3}
B_0	Magnetic field at magnetic axis	-3.09 T
β	Normalised plasma pressure	26.9%
$\tau_{E_{\text{min}}}$	Min confinement time for $(p \times \tau_E)_{\text{min}}$	1.26 s

Table VI: *Output quantities for the elliptical model after β and A has been optimised*

For O-mode plasma waves the refractive index is:

$$N_{\text{O}} = \sqrt{1 - \frac{\omega_p^2}{\omega^2}} \tag{15}$$

with ω being the probing wave frequency. The plasma will reflect the beam if the plasma frequency is larger than the beam frequency so

$$\omega > \omega_p = \sqrt{n_e \frac{e^2}{\epsilon_0 m_{e0}}} \tag{16}$$

So for a given frequency, the plasma must not exceed a critical cut off electron density:

$$n_e < n_c = \omega^2 \frac{\epsilon_0 \cdot m_{e0}}{e^2} \tag{17}$$

which gives

$$N_{\rm O} = \sqrt{1 - \frac{n_e}{n_c}} \tag{18}$$

With a probing frequency much higher than the plasma frequency and the critical density much higher than the electron density Eq. (18) can be approximated by:

$$N_{\rm O} = \sqrt{1 - \frac{n_e}{n_c}} \approx 1 - \frac{1}{2} \frac{n_e}{n_c} 01 - \frac{\omega_p^2}{2\omega^2} \tag{19}$$

With sufficient accuracy, the linear dependence of the O-mode refractive index on the electron density is obtained if the normalised quantities obey:

$$\frac{n_e}{n_c} \le 0.4 \quad \frac{\omega_p}{\omega} \le 0.6 \tag{20}$$

We must calculate the phase shift as one beam travels in vacuum by the length L_V and one wave travels in the plasma by the length L_P. The phase shift in terms of 2π is equal to the optical difference divided by the wavelength. With the refractive index in vacuum, $N_V = 1$, this yields:

$$\frac{\Phi}{2\pi} = \frac{\Delta L_{opt}}{\lambda} = \frac{\int_{x_1}^{x_2} (N_V - N_{\rm O}(x')) \, \mathrm{d}x'}{\lambda} \approx \frac{1}{2\lambda n_c} \int_0^x n_e(x') \, \mathrm{d}x'$$
$$= 4.48 \times 10^{-16} \left(\frac{\lambda}{\mathrm{m}}\right) \int_0^x \left(\frac{n_e(x')}{\mathrm{m}^{-3}}\right) \left(\frac{\mathrm{d}x'}{\mathrm{m}}\right) \tag{21}$$

Assuming a Gaussian distribution, the electron density at $\pm\infty$ is approximately equal to the densities just inside the reactor walls. Therefore

$$\int_{-\infty}^{\infty} n_e \exp\left(-\frac{(y-b)^2}{2c^2}\right) \mathrm{d}y \approx n_e \, c \, \sqrt{2\pi} \tag{22}$$

With the density at the centre given as:

$$10^{16} \mathrm{\ m}^{-3} \le n_e \le 10^{18} \mathrm{\ m}^{-3}, \tag{23}$$

N. Kihm
R. K. F. Wiuff
Course 10401
Fusion Energy and Plasma Physics
January 24$^{\text{th}}$
2019

The c in Eq. (22) is the width of the Gaussian distribution and must fit inside the reactor. The DTU tokamak has a minor diameter of 0.250 m. Thus

$$\frac{\Phi}{2\pi} \approx 4.48 \times 10^{-16} \left(\frac{\lambda}{\text{m}}\right) n_e 0.250 \text{ m}\sqrt{2\pi} = 1.12 \times 10^{-16} \ n_e \left(\frac{\sqrt{2\pi}c}{\omega \text{m}}\right)$$

$$= 8.416 \times 10^{-8} \left(\frac{n_e}{\omega \text{s}}\right) \tag{24}$$

Remembering Eq. (17)

$$\omega^2 \frac{\epsilon_0 \ m_{e0}}{e^2} = \omega^2 \frac{8.85 \times 10^{-12} \text{ F·m}^{-1} \ 9.11 \times 10^{-31} \text{ kg}}{1.60 \times 10^{-19} \text{ C}} \tag{25}$$

$$\Downarrow$$

$$n_e < 0.000314\omega^2 \tag{26}$$

Where any units has been disregarded. We want the largest possible phase shift which means that the lower the frequency the better. However cutoff must first be taken into account. Since the cutoff is given by Eq. (26) and since we want to measure densities up to 10^{18} m^{-3} the minimum frequency of the wave is

$$\frac{\omega}{2\pi} > \frac{\sqrt{\frac{10^{18} \text{ m}^{-3}}{0.000314}}}{2\pi} \tag{27}$$

$$\Downarrow$$

$$f \approx 9 \text{ GHz} \tag{28}$$

Given the available emitters, the best emitter is therefore the one with $f = 60\text{GHz}$.

B. Beam Phase shifts

The goal of the interferometer is to find the phase shift between the microwave beam in vacuum and in plasma. Our suggestion is a setup involving a interferometer emitting a beam through the centre of the reactor. The first measurement would be through vacuum to find the source phase. Afterwards phase measurements can be conducted with an active plasma in the reactor. So by first measuring the phase of the

Figure 4: Electron densities for different wavelengths and phaseshifts
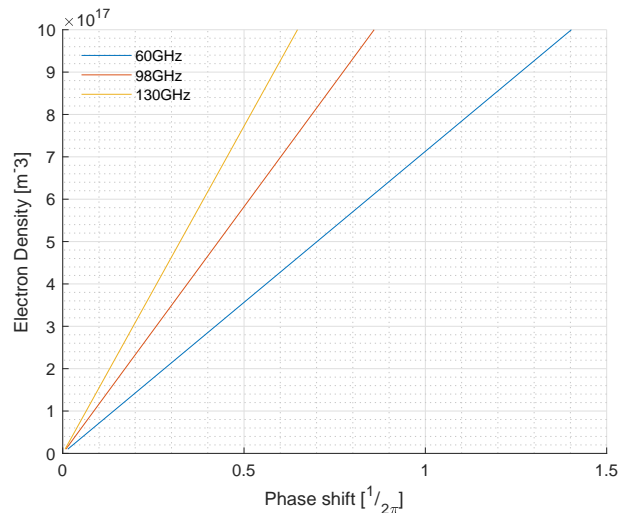
probing beam in vacuum, one can simply

measure the phase shift.

We know from Eqs. (21), (22) and (24) that:

$$\frac{\Phi}{2\pi} = 8.416 \times 10^{-8} \left(\frac{n_e}{\omega s} \right) \tag{29}$$

So for different average electron densities through the plasma, one can plot the resulting

phase shifts. Using the code in Appendix E the plot in Fig. 4 is generated. The measured

phase shift is then correlated to an electron density using Fig. 4.

C.   Evolving beam width

In our case we want to use a Gaussian microwave. Such beam propagates spatially as shown in Fig. 5. w_0 is the beam initial beam waist determined by the source and $w(z)$ is the function describing the waist. This equation is:

$$w(z) = w_0(z) \sqrt{1 + \left(\frac{\lambda z}{\pi w_0^2} \right)^2} \tag{30}$$

Figure 5: *Sketch of Gaussian beam propagation with indication of w_0 and $w(z)$.*

From this equation one can track the spatial propagation of the wave. In this project three sources are given. The initial beam waist is set to $w_0 = 0{\cdot}0275$m corresponding with half a reactor port. With

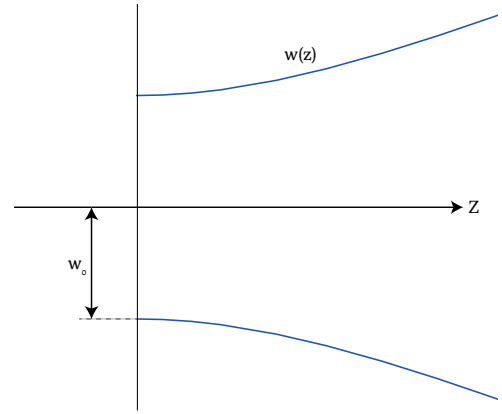$$\frac{\lambda z}{\pi w_0^2} = \frac{9.542690316 \times 10^7 z}{w_0^2 f} \tag{31}$$

, and f being the frequency, the sources' beam waists are given as such:

$$60 \text{ GHz}: \quad w(z) = 0.0275 \sqrt{\left(1 + \frac{9.542690316 \times 10^7 z}{0.0275^2 \, 60 \times 10^9}\right)^2} \tag{32}$$
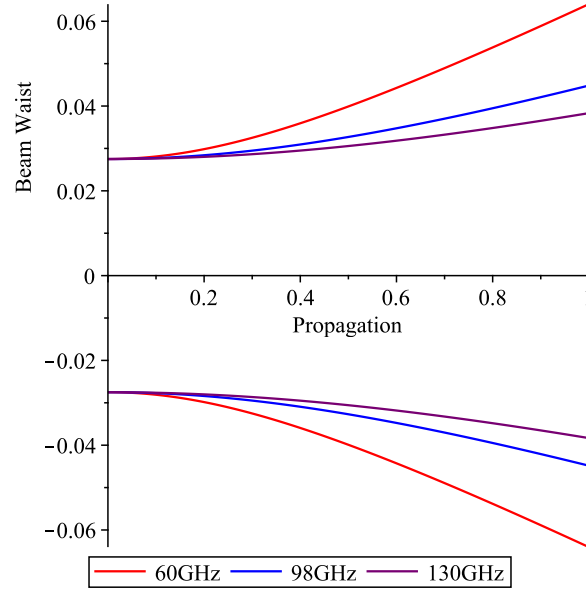
$$98 \text{ GHz}: \quad w(z) = 0.0275 \sqrt{\left(1 + \frac{9.542690316 \times 10^7 z}{0.0275^2 \, 98 \times 10^9}\right)^2} \tag{33}$$

$$130 \text{ GHz}: \quad w(z) = 0.0275 \sqrt{\left(1 + \frac{9.542690316 \times 10^7 z}{0.0275^2 \, 130 \times 10^9}\right)^2} \tag{34}$$

For the three given sources in this project, the beam waist has been plotted on Fig. 6

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24$^{\text{th}}$
2019

Figure 6: *Beam Waist for the three microwave sources. The functions are Eqs.* (32), (33) *and* (34).

It is undesirable for some of the beam to not reach the output port and instead propagate into the reactor wall, as this will result in a lesser signal strength. Therefore it can be necessary to deploy a Gaussian telescope.

D. Gaussian beam telescope interferometer

In order to control the beam waist, one can use a Gaussian beam telescope arrangement around the reactor. A lens is placed between the source and the reactor input port and again between the output port and the receiver. From chapter 5 in "Fusion Plasma Diagnostics with mm-Waves"[3] the authors explains how such an arrangement can be obtained. Starting with the focal lengths of the lenses,

$$d = f_0 + f_1 \tag{35}$$

, where f_0 and f_1 are focal lengths, and d are the distance between the lenses. The resulting narrowest beam waist after the second lens are given as:

$$w_2 = \frac{f_1}{f_0} w_0 \tag{36}$$

This making the transformation wavelength independent[3] (Eq 5.118). The distance to this waist is given as:

$$d_3 = \frac{f_1}{f_0} \left(f_0 + f_1 - \frac{f_1}{f_0} d_0 \right) \tag{37}$$

The waist in between the lenses is given as:

$$w_1 = \frac{\lambda f_0}{\pi w_0} \tag{38}$$

And the distance from the first lens to this waist is:

$$d_1 = \left(\frac{\frac{d_0}{f_0} - 1}{\frac{w_0^2 \pi}{f_0 \lambda} + \left(\frac{d_0}{f_0} - 1 \right)^2} + 1 \right) f_0 \tag{39}$$

Thus the distance from w_1 to the second lens is:

$$d_2 = d - d_1 \tag{40}$$

Knowing these variables and utilising Eq. (30) we can model a complete setup. Using the Matlab code in Appendix D, the following parameters were input:
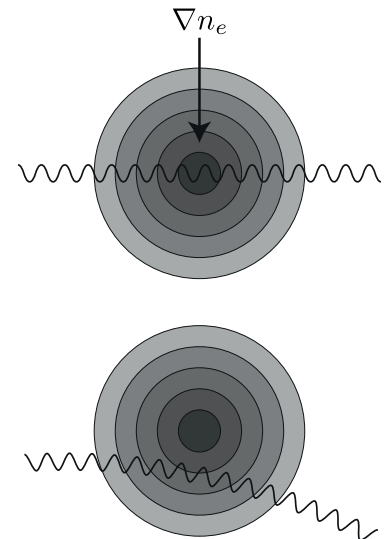
$$r_p = 0.0550 \text{ m} \quad r = 0.125 \text{ m} \quad w_0 = 0.0275 \text{ m} \quad freq = 60 \text{ GHz}$$
$$d_0 = 0.200 \text{ m} \quad d_r = 0.100 \text{ m} \quad f_0 = 0.250 \text{ m} \qquad 0.200 \text{ m} \tag{41}$$

With r_p being the tokamak port radius, r the tokamak minor radius and d_r the distance between the first lens and the reactor wall. In the case of the Danish Tokamak Undertaking, two ports opposing each other is ideal for use with interferometry. In this example, the port radius, r_p, and the minor radius r are based on the ST-25 in the basement of building 309 at DTU. The script gave the results:

$$w_1 = 0.0145 \text{ m} \quad w_2 = 0.0220 \text{ m}$$
$$d_1 = 0.224 \text{ m} \quad d_2 = 0.226 \text{ m} \quad d_3 = 0.232 \text{ m} \tag{42}$$

Furthermore the code plots a sketch of the desired interferometer setup. The sketch can be seen in Fig. 7.

Lastly it should be mentioned that the beam waist confinement does not matter if the wave is refracted into the side of the reactor. As the plasma density varies from the centre and out, the plasma itself act as a variable lens with variable refractive indices. If the beam is emitted straight through the centre, the refractive indices will only shorten and lengthen the wavelength back to normal, however if the beam is emitted off centre, it will be refracted away from the centre. In such a setup, further calculations need to be conducted to determine variation in signal strength, phase and wavelength. The two described setups are depicted on Fig. 8.

Figure 8: *Refraction of the probing beam at centre (top) and off centre (bot-*

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

Beam Waist In A Gaussian Beam Interferometer
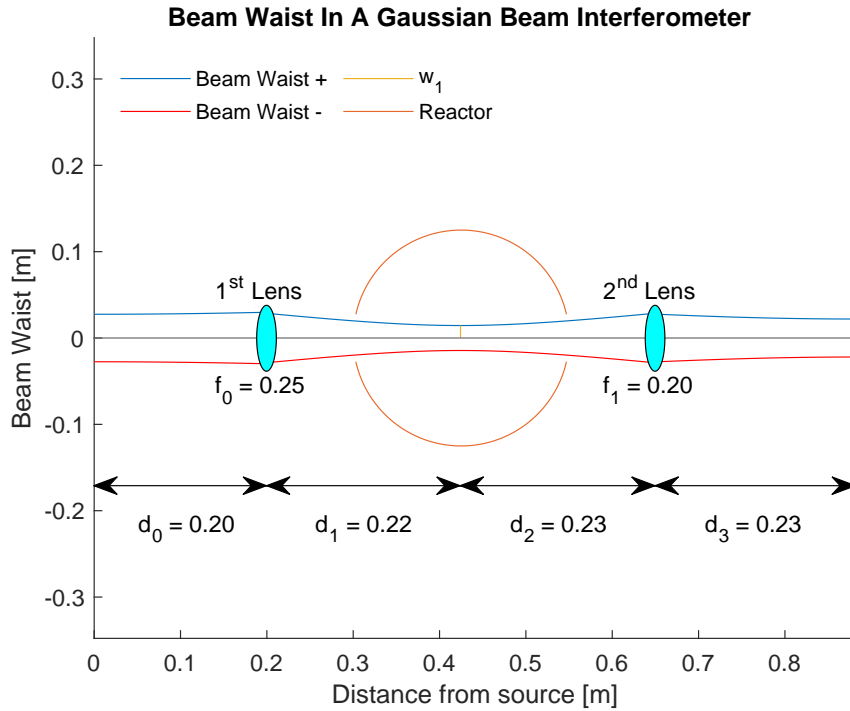
Figure 7: *Gaussian beam telescope interferometer setup with input parameters shown in Eq. (41) and output parameters shown in Eq. (42)*

III.   FUSOR EXERCISE

Studying the Inertial electrostatic confinement fusor at DTU. Measurements of neutron counts and the spectral line width are made as a function of the voltage and current. To this, the emission spectrum of the gas are also measured.

A.   Plasma light emission and spectral line

When doing spectroscopy on a plasma of an unknown gas. Optical dispersion splits up the spectrum of light into lines which are then recorded using a detector. This detector measures the frequency/wavelength of the light and the intensity of the light. Thus a plot with the wavelength along the x-axis vs the intensity along the y-axis are produced. For our recordings a typical spectrum looked like that of Fig. 9.

Spectrum of light from fusion in the fusor
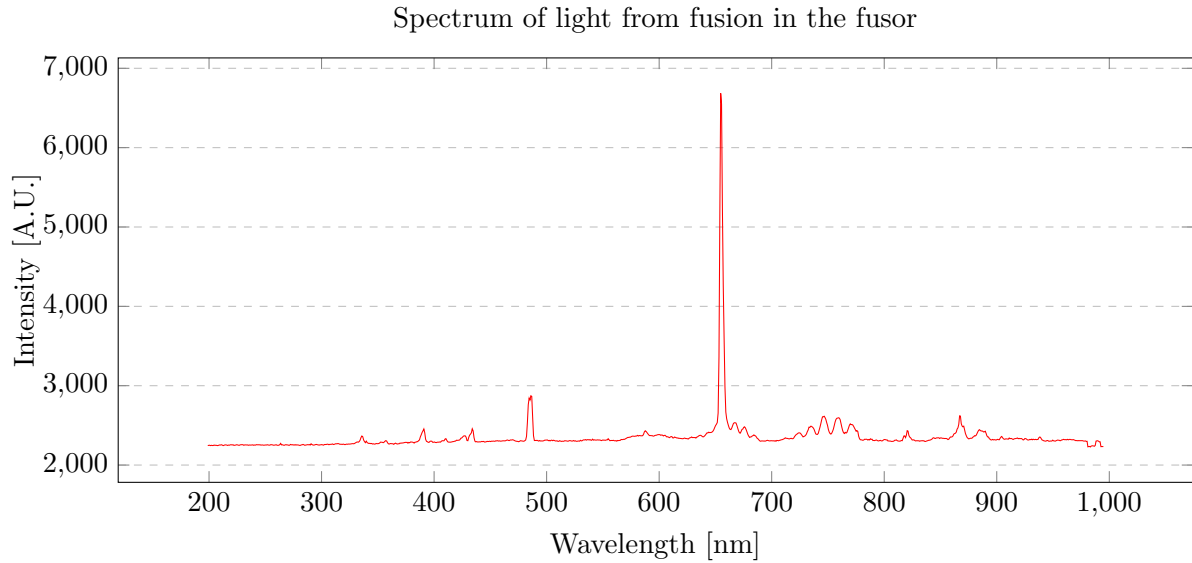
***Figure 9:*** *Spectrum of light from fusion in the fusor*

Each peak correspond with a constituent gas in the fusor. The peaks present on the spectrum are subject to broadening effects. Considering the Doppler effect, each photon's frequency will be shifted since the particles (which are the emitters) are moving at fast speeds (the Doppler shift increases with speed). Therefore if each particle where moving in the same direction the entire intensity peak would be shifted. But since the particles are moving more or less isotropic around the centre of the device, the peaks are not shifted but rather broadened around the normal emission wavelength. To this it is also noted that the Doppler broadening increases if particles are moving faster.

Considering **??**, a spectrum is shown with central wavelengths around 656 nm, 486 nm and 433 nm which correspond to accepted values of hydrogen light emission wavelengths[4]. Meanwhile the spectral lines of Deuterium only differs by a factor of 1.000272[5], which will not be detected using our methods. There are other peaks but those peaks are due to noise from impurities etc.

Since the speed of the particles increases with the voltage we expect the line broadening to do the same since $\sigma = \lambda_0/c\, v_{1D}$. While there seems to be enormous uncertainties on the hydrogen measurements, Fig. 10 shows a tendency to increase for both hydrogen and deuterium. Disregarding the outliers the hydrogen also seems to have a wider spectral line than deuterium which makes sense due to the mass difference between the species. Meanwhile the uncertainties seen might be due to differences in pressure which leads to differences in kinetic energy lost per time. Other uncertainties are most likely also present. For the width of a spectral line, it is known that:
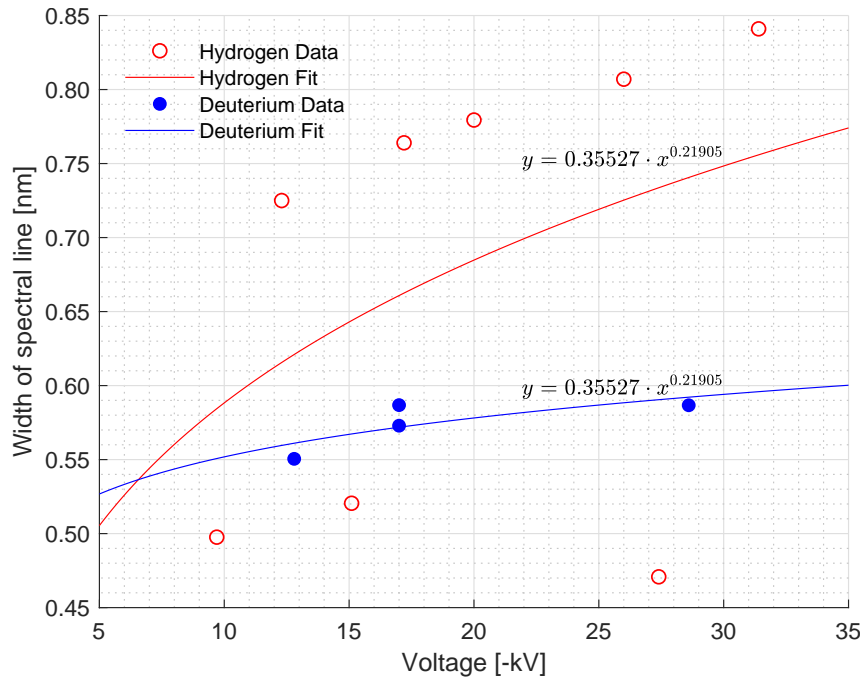
Figure 10: *The spectral line width as a function of*

$$\sigma = \Delta\lambda = \frac{\lambda_0}{c}v_{1D} \tag{43}$$

So for the obtained widths the velocities can be estimated via Eq. (43). Once these points are obtained a curve is fitted to it. The fit is assummed to be linear as velocity is proportional to voltage applied. The kinetic energies are also estimated via:

$$KE = \frac{1}{2}mv^2 \tag{44}$$

These points are also estimated and plotted along with a fit. The fit is assumed to be some constant times the square root of the voltage as as isolated from Eq. (44). This is done for both hydrogen and deuterium. The plot can be seen in Fig. 11. The four fits were computed to be:

$$\text{Velocity for hydrogen:} \quad v = 3.12 \times 10^4 \sqrt{x} + 1.72 \times 10^5 \tag{45}$$

$$\text{Velocity for deuterium:} \quad v = 0.7666 \times 10^4 \sqrt{x} + 2.295 \times 10^5 \tag{46}$$

$$\text{Kinetic energy for hydrogen:} \quad KE = 11.7x + 287.7 \tag{47}$$

$$\text{Kinetic energy for deuterium:} \quad KE = 4.387x + 637.3 \tag{48}$$

B.   Neutron production rate

The fusion reaction rate should increase with voltage and current where approximately every second reaction produce a neutron. Measurements of the neutron production rate

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019
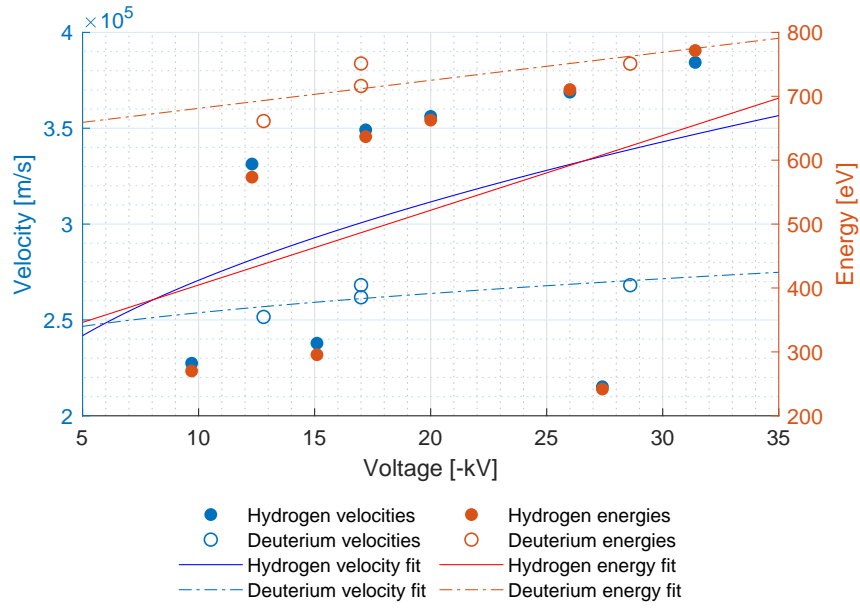
Figure 11: *The estimated velocities and kinetic energies for hydrogen and deuterium as functions of applied voltage*

are made and seen in Section III B. It is important to note that only 5 measurements are made, so the sample size will be very small. There are 3 parameters that varies throughout the measurements which makes it hard to say anything about the data. There is an unknown dependency on the pressure since an increasing pressure will decelerate the particles overall, but it will also increase the amount of fuel available. The pressure varies from 1.40 μbar to 2 μbar. This should have an effect on the neutron rate, but let us disregard that fact for now. The data suggests that a current of 6 mA is needed to get decent reaction rates when the voltage is in the ranges measured. On top of that, the data also suggests that the voltage needed is between 24 and 28. Note that the last suggestion is made at an approximately constant pressure.

Calculating the total neutron flux at the detector and the total neutron production rate is trivial given Cts./s, $\eta_{\text{eff.}} = 15$, $A_{\text{det}} = 6 \times 10^{-5}$ m^2 and $R_{\text{det}} = 0.500$ m.

$$\Gamma_{\text{n}} = \frac{\text{Cts.}}{\text{s}} \frac{1}{\eta_{\text{det}}} \frac{1}{A_{\text{det}}} \tag{49}$$

And n/s $= 4 \pi R_{\text{det}} \Gamma_{\text{n}}$.

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

p(μBar)	I(mA)	V(-kV)	P(W)	Cts./s	$\Gamma_n(s^{-1}{\cdot}m^{-2})$	n/s	Q(10^{-6})
2	12	17	204	0.689	0.766×10^5	2.41×10^5	1.38
1.7	9	24	216	0.891	0.990×10^5	3.11×10^5	5.36
1.6	7	28	196	2.106	2.34×10^5	7.35×10^5	13.96
1.4	3	32	96	0.295	0.328×10^5	1.03×10^5	3.99
1.4n	6	33	198	1.725	1.92×10^5	6.02×10^5	1.13

Table VII: *Table containing the measured parameters for deuterium.*

1 https://www.iter.org/.

2 Jeffrey P. Freidberg, *Plasma physics and fusion energy* (Cambridge University Press, 2007).

3 Hans-Jürgen Hartfuß and Thomas Geist, *Fusion Plasma Diagnostics with mm-Waves* (Wiley-VCH, 2013).

4 "Hydrogen spectral series" - Wikipedia (Visited January 22nd).

5 "Deuterium#Spectroscopy" - Wikipedia (Visited January 22nd).

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24$^{\text{th}}$
2019

LIST OF FIGURES

LIST OF TABLES

LISTINGS

Appendices

Appendix A: tokamakDTU_asign_1

```matlab
%*********************************************************************
% Name:          tokamakDTU
%
% Version:       1.0
%
% Purpose:       Contains the function 'tokamakDTU' which gives parameters
%                for a tokamak fusion power plant as output based on a
%                simplified model. The equations used are derived in
%                chapter 5 in Friedberg, Plasma physics and Fusion
%                Energy, 2007 (all references are referring thereto).
%
% To do (NOT for 10401 - Fusion Energi students):
%                1. Rewrite the code to a class (this is not done on purpose so
%                   that the code is more readable for students not familiar
%                   with classes). Can from that merge the files which takes
%                   R_0/a and/or the ellipticity as an input into one file.
%
% Changelog:
%                1. December 2014:
%                   Written by Michael Løiten based on a similar code written
%                   as a bachelor project by Elias Pagh Sentius
%                   mailto: mmag@fysik.dtu.dk
%*********************************************************************

function [b, c, a, R_0, A, A_p, V_p, P_dens, p, n, B_0, beta, tau_E_min,...
          C_per_watt] =...
    tokamakDTU_asign_1(...
        n_flux_fraction, C_F, C_I, P_E, P_W, B_max, sigma_max, eta_t)
```

```
29   %TOKAMAK_DTU Function which returns the parameters of a power plant

30   %

31   % Output parameters

32   %------------------

33   % b          - Blanket/shield thickness [m]

34   % c          - Magnet coil thickness [m]

35   % a          - Minor radius [m]

36   % R_0        - Major radius [m]

37   % A          - Aspect ratio []

38   % A_p        - Plasma surface [m^2]

39   % V_p        - Plasma volume [m^3]

40   % P_dens     - Power density [W/m]

41   % p          - Plasma pressure [Pa]

42   % n          - Particle density [m^-3]

43   % B_0        - Magnetic field at magnetic axis [T]

44   % beta       - Plasma beta in the centre []

45   % tau_E_min  - Min confinement time for satisfaction of (p*tau_E)_min [s]

46   % C_per_watt - The cost of the powerplant [$]

47   %

48   % Input parameters

49   %------------------

50   % n_flux_fraction - n flux in breeder end/n flux in breeder start []

51   % C_F             - Fixed cost propotionality constant [$]

52   % C_I             - Nuclear island cost propotionality constant [$W/m^3]

53   % P_E             - Desired output power [MW]

54   % P_W             - Maximum wall load [MW/m^2]

55   % B_max           - Magnetic field at the edge of the coil [T]

56   % sigma_max       - Tensile strenght of the magnetic field coils [atm]

57   % eta_t           - Energy conversion efficiency []

58

59

60   % The function starts by defining fixed constants
```

```matlab
61  % Note that this is inefficient if we are looping over the function, but it
62  % makes the code easier to use, as these are not needed as input parameters
63
64  % Fixed constants
65  %#########################################################################
66  % Nuclear
67  %--------
68  % Energies
69  E_t         = 2.5e-8; % [MeV] Energy of slow (thermal) neutron (eq 5.6)
70  E_n         = 14.1;   % [MeV] Neutron energy after fusion (eq 2.17)
71  E_a         = 3.5;    % [MeV] alpha energy after fusion (eq 2.17)
72  E_Li        = 4.8;    % [MeV] Heat produced by breeding Li (under eq 4.31)
73  % Cross section and main free paths
74  sigma_v_avg = 3.0e-22;% [m^3/s] DT fusion cross section @ 15keV (table 5.2)
75  lambda_br   = 0.0031; % [m] Breeding mean free path (under eq 5.7)
76  lambda_sd   = 0.055;  % [m] Mean free path from sigma_sd (eq 5.3)
77
78  % Plamsa physics
79  %---------------
80  % Parameters for infinity gain at the minimum of p tau_E (eq 4.20)
81  T           = 15.0;  % [keV] Temparature for obtaining min tripple product
82  tripple_min = 8.3;   % [atm s] Min tripple prod to obtain Q=inf @ T=15 keV
83
84  % Natural constants
85  %------------------
86  mu_0 = 4.0*pi*1e-7;      % Vacuum permeability [T*m/A]
87  e    = 1.602176565e-19;  % Elementary charge [C]
88  %#########################################################################
89
90
91  % Secondly we convert everything to SI units, so that the variables are
92  % easier to handle
```

```matlab
93   % Again, this is computationally inefficient, but it suffices for our use

94   % Conversion to SI-units

95   %#########################################################################

96   % Conversion factors

97   W_per_MW     = 1.0e6;

98   Pa_per_atm   = 1.01325e5;

99   eV_per_keV   = 1.0e3;

100  eV_per_MeV   = 1.0e6;

101  J_per_eV     = e;

102  J_per_keV    = J_per_eV * eV_per_keV;

103  J_per_MeV    = J_per_eV * eV_per_MeV;

104  % Conversions

105  P_E          = P_E * W_per_MW;          % Desired output power

106  P_W          = P_W * W_per_MW;          % Wall Loading limit on first wall

107  E_t          = E_t * J_per_MeV;         % Energy of slow (thermal) neutron

108  E_n          = E_n * J_per_MeV;         % Neutron energy after fusion

109  E_a          = E_a * J_per_MeV;         % alpha energy after fusion

110  E_Li         = E_Li * J_per_MeV;        % Heat produced by breeding Li

111  sigma_max    = sigma_max * Pa_per_atm;  % Max allowable structural stress

112  T            = T * J_per_keV;           % Temparature for minimum p*tau_E

113  tripple_min  = tripple_min * Pa_per_atm;  % The Lawson parameter (p*tau_E)

114  %#########################################################################

115

116

117  % Calculate the geometrical factors

118  %---------------------------------

119  % Find the breeder thickness

120  b = get_b(lambda_sd, E_n, E_t, lambda_br, n_flux_fraction);

121  % Find the minor plasma radius and the coil thickness

122  [a, c] = get_a_and_c(B_max, mu_0, sigma_max, b);

123  % Find the major radius

124  R_0 = get_R_0(a, eta_t, E_n, E_a, E_Li, P_E, P_W);
```

```matlab
125   % Find the resulting geometrical factors
126   A   = R_0/a;                      % Aspect Ratio
127   A_p = (2.0*pi*a)*(2.0*pi*R_0);    % Plasma surface area
128   V_p = (pi*a (2.0))*(2.0*pi*R_0);  % Plasma volume
129
130
131   % Calculate the plasma physics parameters
132   %---------------------------------------
133   % Find the power density in the plasma
134   P_dens = get_P_dens(E_a, E_n, E_Li, P_E, eta_t, V_p);
135   % Find the plasma pressure
136   p = get_p(E_a, E_n, P_dens, T, sigma_v_avg);
137   % Calculate the density from the definition of p under eq 5.36
138   n = p/(2.0*T);
139   % Find the magnetic field strength on the magnetic axis
140   B_0 = get_B_0(R_0,a,b,B_max);
141   % Find the plasma beta on the magnetic axis
142   beta = get_beta(p, B_0, mu_0);
143   % Find the minimum reqiured confinement time from the definition of the
144   % minimum tripple product.
145   % NOTE: A higher confinement time is advantegous, and could in principle
146   % yield a smaller (and cheaper) reactor. However, the effect is not
147   % included in this model
148   tau_E_min = tripple_min/p;
149
150
151   % Calculate the cost
152   % (details about the cost can be found in the function get_a_and_c)
153   %-------------------
154   % Find the volume of the nuclear island
155   % (the material surrounding the plasma)
156   V_I = get_V_I(R_0,a,b,c);
```

```matlab
157    % Find the reactor volume per power out
158    % In the current model, this is the only non-constant in the expression for
159    % cost per watt
160    V_I_per_P_E = V_I/P_E;
161    C_per_watt = get_C_per_watt(C_F, C_I, V_I_per_P_E);
162    end
163
164
165
166    function      = get_b(                                              )
167    %GET_B Calculates b from the need of slowing down and breeding neutrons
168
169    % Thickness of the moderator-breeding region so that 1 - n_flux_fraction
170    % have slowed down and undergone a breeding reaction
171    % [m]
172    % Equation 5.10
173    delta_x = 2.0*lambda_sd*...
174            log( 1.0-(1.0/2.0)*(E_n/E_t) (1.0/2.0)*...
175                (lambda_br/lambda_sd)*log( n_flux_fraction )...
176              );
177
178    % Set b from delta_x
179    % Friedberg argues above equation 5.11 that b should be between 1 and 1.5 m
180    % Therefore a self chose constant is set to 0.38
181    self_chosen_constant = 0.32;
182    b = delta_x + self_chosen_constant;
183    end
184
185
186
187
188    function        = get_a_and_c(                              )
```

```
189   %GET_A_AND_C Calculates a and c
190
191   % c is obtained from requiring that the magnets are so thin that they are
192   % on the limit of the tensile strenght
193   % a is obtained from minimizing the costs
194
195   % xi defined when making the magnetic coil c as thin as possible
196   % Under equation 5.27
197   xi = B_max (2.0) / (4.0*mu_0*sigma_max);
198
199   % a is found from optimization of the cost, where
200   % total cost = fixed cost + nuclear island cost
201   %...........
202   % Fixed cost
203   %...........
204   % K_F = Fixed cost for building, turbines, generators etc (also applies to
205   % fusion, fission, fossil)
206   % Assumption: The fixed cost is proportional to power output:
207   % Equation 5.13
208   % K_F = C_F*P_E;
209   %...........
210   % Nuclear island cost (mainly cost of magnets, blanket and shield)
211   %............
212   % Assumption: The proportional to reactor volume:
213   % Equation 5.14
214   % K_I = C_I*V_I;
215   % Equation 5.15
216   % V_I = 2.0*pi^(2.0) * R_0 * ( (a+b+c)^(2.0) - a^(2.0) ); % Reactor volume
217   %..............
218   % Cost per watt:
219   %..............
220   % Defined as C_p_watt = (K_F + K_I)/P_E, rewritten to
```

```matlab
221   % C_p_watt = C_F + C_I*(V_I/P_E);

222   % Since the cost per watt contains two constants, we can minimize the

223   % V_I/P_E in order to optimize the cost

224   % Given by equation 5.20 inserted in 5.17

225   % Equation 5.21

226   % V_I_per_P_E = V_I/P_E;  % Reactor volume per power out

227   % a is found by setting the derivative of V_I_per_P_E = 0

228   % Equation 5.29

229   a = ((1.0 + xi)/(2.0*xi (1.0/2.0)))) * b;

230

231   % Knowing xi, a, and, b, we can calculate c

232   % c found by comparing tensile force and magnetic force working on the coil

233   % Equation 5.27

234   c = 2*xi/(1-xi)*(a+b);

235   end

236

237

238

239   function       = get_R_0(                              )

240   %GET_R_0 Calculate the major radius

241

242   % Divide eq 5.18 (electric power out) by

243   % eq 5.19 (wall loading * area = total neutron production) and solve for R0

244   % Equation 5.20

245   R_0 = (1.0/(4.0*pi (2.0)*eta_t))*(E_n/(E_n + E_a + E_Li))*(P_E/(a*P_W));

246   end

247

248

249

250   function           = get_P_dens(                          )

251   %GET_P_DENS Calculate the power density

252
```

```matlab
253    % The power density is found by the sum of the power from the alphas plus
254    % the power from the neutrons, divided by the plasma volume
255    % Equation 5.35
256    P_dens = (E_a + E_n)/(E_a + E_n + E_Li)*P_E/(eta_t*V_p);
257    end
258
259
260
261    function        = get_B_0(                )
262    %GET_B_0 Calculte the magnetic field strength on the magnetic axis
263
264    % B_max is found in the edge of the magnet (at R = R_0-a-b)
265    % B0 is the magnetic field at R0
266    % As B propto 1/R. we have that B_0/B_max = (R_0-a-b)/R_0, which leads to
267    % Equation 5.42
268    B_0 = ((R_0-a-b)/R_0)*B_max;
269    end
270
271
272
273    function         = get_beta(          )
274    %GET_beta Calculte the magnetic field strength on the magnetic axis
275
276    % Plasma beta in the center (kinetical pressure over magnetical pressure):
277    % Equation 5.43
278    beta = p / (B_0 2/(2.0*mu_0));
279    end
280
281
282
283    function      = get_p(                    )
284    %GET_P Calculate the plasma pressure
```

```matlab
285
286    % Found from solving the sum of neutron and alpha power for n, and multiply
287    % the result with T
288    % Equation 5.37
289    p = ( (16.0/(E_a + E_n)) * P_dens ) (1.0/2.0)*...
290        (T (2.0)/sigma_v_avg) (1.0/2.0);
291    end
292
293
294
295    function        = get_V_I(          )
296    %GET_V_I Calculate the volume of the material surrounding the plasma
297
298    % Equation 5.15
299    V_I = 2.0*pi (2.0) * R_0 * ( (a+b+c) (2.0) - a (2.0) );
300    end
301
302
303
304    function             = get_C_per_watt(                )
305    %GET_C_PER_WATT Calculates the cost for one watt out from the power plant
306
307    % For details in how the cost is derived, see comments in the function
308    % get_a_and_c
309
310    C_per_watt = C_F + C_I*(V_I_per_P_E);
311    end
```

```matlab
close all

clear all


titl = [ Desired output power [MW] ,  Maximum wall load [MW, m -2] , ...
        Magnetic field at the edge of the coil [T] , ...
        Tensile strenght of the magnetic field coils [atm] ];
foldertitl = [ PE ,  PW ,  Bmax ,  sigmamax ];


for d = 1:4
    l = d; % Choose iteration: 1 = P_E, 2 = P_W, 3 = B_max, 4 = sigma_max
    p0 = [];
    x = [];
    mkdir('../MatlabFigures', foldertitl(l))
    printout = (sprintf('Created folder %s', foldertitl(l)));
    disp(printout)
    if l == 1
        x = linspace(1, 3000, 1000);
        for i = 1:length(x)
            [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
                n, B_0, beta, tau_E_min, C_per_watt] = ...
                tokamakDTU_asign_1(0.01, 1, 2, x(i), 4, 13, 3000, 0.4);
            q = [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
                n, B_0, beta, tau_E_min, C_per_watt];
            p0 = cat(1, p0, q);
        end
    elseif l == 2
        x = linspace(1, 10, 1000);
        for i = 1:length(x)
            [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
                n, B_0, beta, tau_E_min, C_per_watt] = ...
```

```matlab
31                tokamakDTU_asign_1(0.01, 1, 2, 1000, x(i), 13, 3000, 0.4);
32            q = [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
33                n, B_0, beta, tau_E_min, C_per_watt];
34            p0 = cat(1, p0, q);
35        end
36    elseif l == 3
37        x = linspace(10, 20, 1000);
38        for i = 1:length(x)
39            [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
40                n, B_0, beta, tau_E_min, C_per_watt] = ...
41                tokamakDTU_asign_1(0.01, 1, 2, 1000, 4, x(i), 3000, 0.4);
42            q = [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
43                n, B_0, beta, tau_E_min, C_per_watt];
44            p0 = cat(1, p0, q);
45        end
46    elseif l == 4
47        x = linspace(2000, 5000, 1000);
48        for i = 1:length(x)
49            [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
50                n, B_0, beta, tau_E_min, C_per_watt] = ...
51                tokamakDTU_asign_1(0.01, 1, 2, 1000, 4, 13, x(i), 0.4);
52            q = [b, c, a, R_0, A, A_p, V_p, P_dens, p, ...
53                n, B_0, beta, tau_E_min, C_per_watt];
54            p0 = cat(1, p0, q);
55        end
56    end
57    printout = (sprintf('Iterated tokamakDTU_asign_1.m %d of 4', d));
58    disp(printout)
59    b = p0(:, 1);
60    c = p0(:, 2);
61    a = p0(:, 3);
62    R_0 = p0(:, 4);
```

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24ᵗʰ
2019

```matlab
63      A = p0(:, 5);

64      A_p = p0(:, 6);

65      V_p = p0(:, 7);

66      P_dens = p0(:, 8);

67      p = p0(:, 9);

68      n = p0(:, 10);

69      B_0 = p0(:, 11);

70      beta = p0(:, 12);

71      beta = beta * 100;

72      tau_E_min = p0(:, 13);

73      C_per_watt = p0(:, 14);

74

75      f1 = figure;

76      ptau = plot(x, tau_E_min, 'color', 'r', 'LineWidth', 1);

77      xlabel(titl(l));

78      ylabel('[s]');

79      ytickformat('%.2f');

80      grid on

81      grid minor

82      lgd = legend([ptau], ...

83          { Min confinement time for satisfaction of (p tau_E)_{min} });% Figure legend

84      legend('boxoff')

85      legend('Location', 'north')

86      figcount = 1;

87      printout = (sprintf('Created figure %d of 8', figcount));

88      disp(printout)

89

90      f2 = figure;

91      pb = plot(x, b, 'color', 'r', 'LineWidth', 1);

92      hold on

93      pc = plot(x, c, '--', 'color', 'b', 'LineWidth', 1);

94      pa = plot(x, a, '-.', 'color', 'm', 'LineWidth', 1);
```

```matlab
95      pcpw = plot(x, C_per_watt, 'color', 'g', 'LineWidth', 1);
96      hold off
97      xlabel(titl(l));
98      ytickformat('%.2f');
99      lgd = legend([pb, pc, pa, pcpw], ...
100         { Blanket - shield, thickness, [m] ,  Magnet, coil, thickness, [m] , ...
101          Minor, radius, [m] , ...
102          Cost, per, watt, [ ] });% Figure legend
103     legend('boxoff')
104     if l == 3
105         legend('Location', 'north')
106     else
107         legend('Location', 'west')
108     end
109     grid on
110     grid minor
111     figcount = 2;
112     printout = (sprintf('Created figure %d of 8', figcount));
113     disp(printout)
114
115     f3 = figure;
116     hold on
117     yyaxis left
118     pA = plot(x, A, 'LineWidth', 1);
119     ylabel('Aspect ratio')
120     yyaxis right
121     pr_0 = plot(x, R_0, '--', 'LineWidth', 2);
122     ylabel('Major radius [m]')
123     hold off
124     xlabel(titl(l));
125     ytickformat('%.2f');
126     lgd = legend([pr_0, pA], ...
```

```matlab
127         { Major, radius ,  Aspect, ratio, [] });% Figure legend
128     legend('boxoff')
129     legend('Location', 'east')
130     grid on
131     grid minor
132     figcount = 3;
133     printout = (sprintf('Created figure %d of 8', figcount));
134     disp(printout)
135
136     f4 = figure;
137     pA_p = plot(x, A_p, 'LineWidth', 1);
138     hold on
139     pV_p = plot(x, V_p, '--', 'LineWidth', 2);
140     hold off
141     xlabel(titl(l));
142     ytickformat('%.2f');
143     lgd = legend([pA_p, pV_p], ...
144         { Plasma, surface, [m 2] ,  Plasma, volume, [m 3] });% Figure legend
145     legend('boxoff')
146     legend('Location', 'east')
147     grid on
148     grid minor
149     figcount = 4;
150     printout = (sprintf('Created figure %d of 8', figcount));
151     disp(printout)
152
153     f5 = figure;
154     pP_dens = plot(x, P_dens, 'color', 'r', 'LineWidth', 1);
155     xlabel(titl(l));
156     ylabel('[W m^-1]');
157     ytickformat('%.2f');
158     grid on
```

```matlab
159     grid minor
160      lgd = legend([pP_dens], ...
161         { Power density });% Figure legend
162     legend('boxoff')
163     legend('Location', 'north')
164     figcount = 5;
165     printout = (sprintf('Created figure %d of 8', figcount));
166     disp(printout)
167
168     f6 = figure;
169     yyaxis left
170     pn = plot(x, n, 'LineWidth', 1);
171     ylabel('Particle density [m^-3]')
172     ytickformat('%.2f');
173     yyaxis right
174     p = plot(x, p, '--', 'LineWidth', 2);
175     ylabel('Plasma pressure [Pa]')
176     ytickformat('%.2f');
177     xlabel(titl(l));
178     lgd = legend([pn, p], ...
179         { Particle, density ,  Plasma, pressure, [Pa] });% Figure legend
180     legend('boxoff')
181     legend('Location', 'north')
182     grid on
183     grid minor
184     figcount = 6;
185     printout = (sprintf('Created figure %d of 8', figcount));
186     disp(printout)
187
188     if l == 1
189         f7 = figure;
190         subplot(2, 1, 1);
```

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

```matlab
191         pB_0 = plot(x, B_0, 'color', 'b', 'LineWidth', 1);
192         ylabel('[T]');
193         ytickformat('%.2f');
194         grid on
195         grid minor
196         lgd = legend([pB_0], ...
197         { Magnetic field at magnetic axis });% Figure legend
198         legend('boxoff')
199         legend('Location', 'south')
200         subplot(2, 1, 2);
201         B_0(B_0 <= -10) = NaN;
202         pB_0 = plot(x, B_0, 'color', 'b', 'LineWidth', 1);
203         xlabel(titl(l));
204         ylabel('[T]');
205         ytickformat('%.2f');
206         grid on
207         grid minor
208         lgd = legend([pB_0], ...
209         { Magnetic field at magnetic axis });% Figure legend
210         legend('boxoff')
211         legend('Location', 'south')
212     else
213         f7 = figure;
214         pB_0 = plot(x, B_0, 'color', 'b', 'LineWidth', 1);
215         xlabel(titl(l));
216         ylabel('[T]');
217         ytickformat('%.2f');
218         grid on
219         grid minor
220         lgd = legend([pB_0], ...
221         { Magnetic field at magnetic axis });% Figure legend
222         legend('boxoff')
```
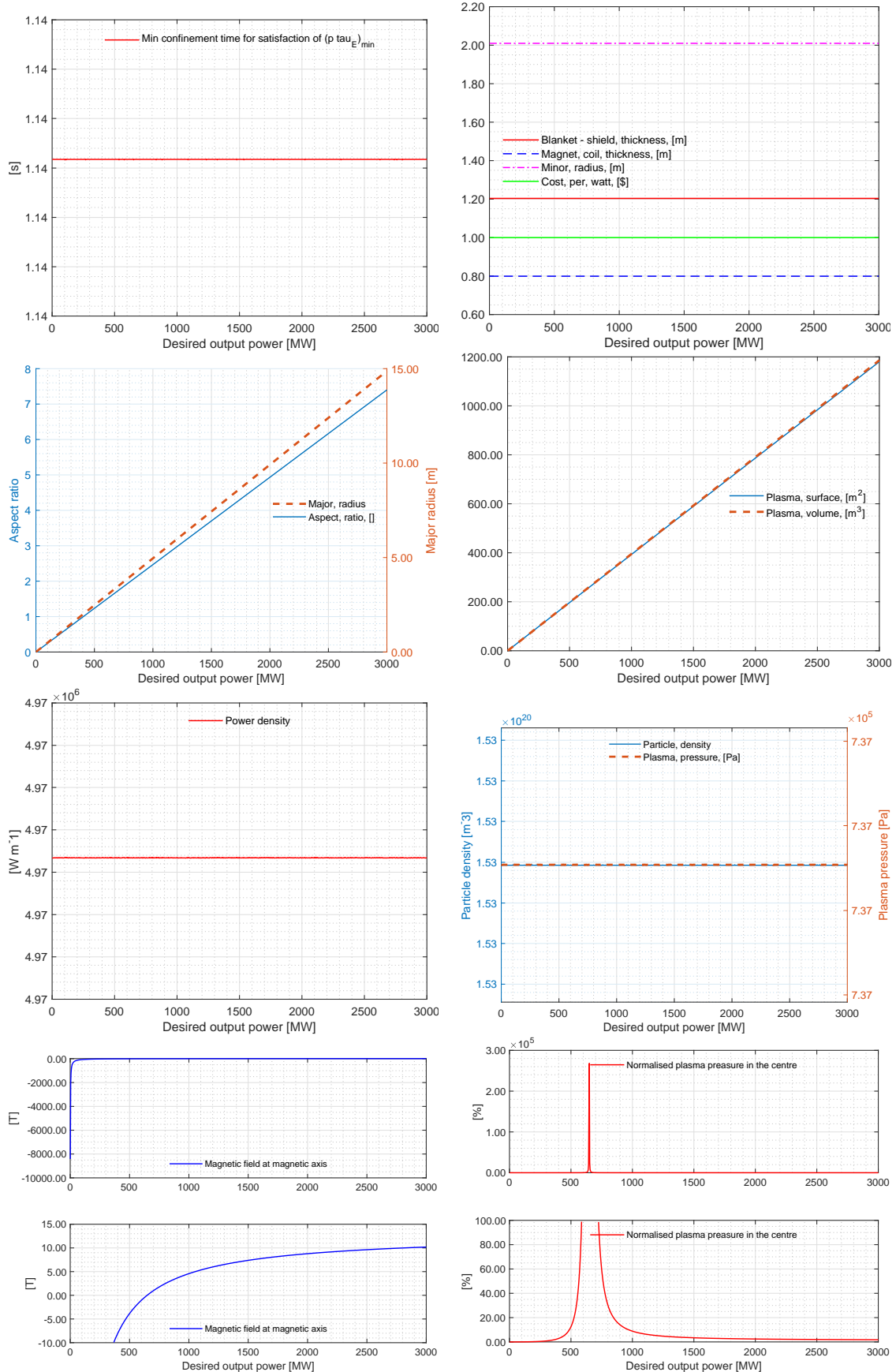
```matlab
223        legend('Location', 'south')
224    end
225    figcount = 7;
226    printout = (sprintf('Created figure %d of 8', figcount));
227    disp(printout)
228    if l == 1 || l == 2
229        f8 = figure;
230        subplot(2, 1, 1);
231        pbeta = plot(x, beta, 'color', 'r', 'LineWidth', 1);
232        ylabel('[%]');
233        ytickformat('%.2f');
234        grid on
235        grid minor
236        lgd = legend([pbeta], ...
237        { Normalised plasma preasure in the centre });% Figure legend
238        legend('boxoff')
239        legend('Location', 'north')
240        subplot(2, 1, 2);
241        beta(beta >= 100) = NaN;
242        pbeta = plot(x, beta, 'color', 'r', 'LineWidth', 1);
243        xlabel(titl(l));
244        ylabel('[%]');
245        ytickformat('%.2f');
246        grid on
247        grid minor
248        lgd = legend([pbeta], ...
249        { Normalised plasma preasure in the centre });% Figure legend
250        legend('boxoff')
251        legend('Location', 'north')
252    else
253        f8 = figure;
254        pbeta = plot(x, beta, 'color', 'r', 'LineWidth', 1);
```

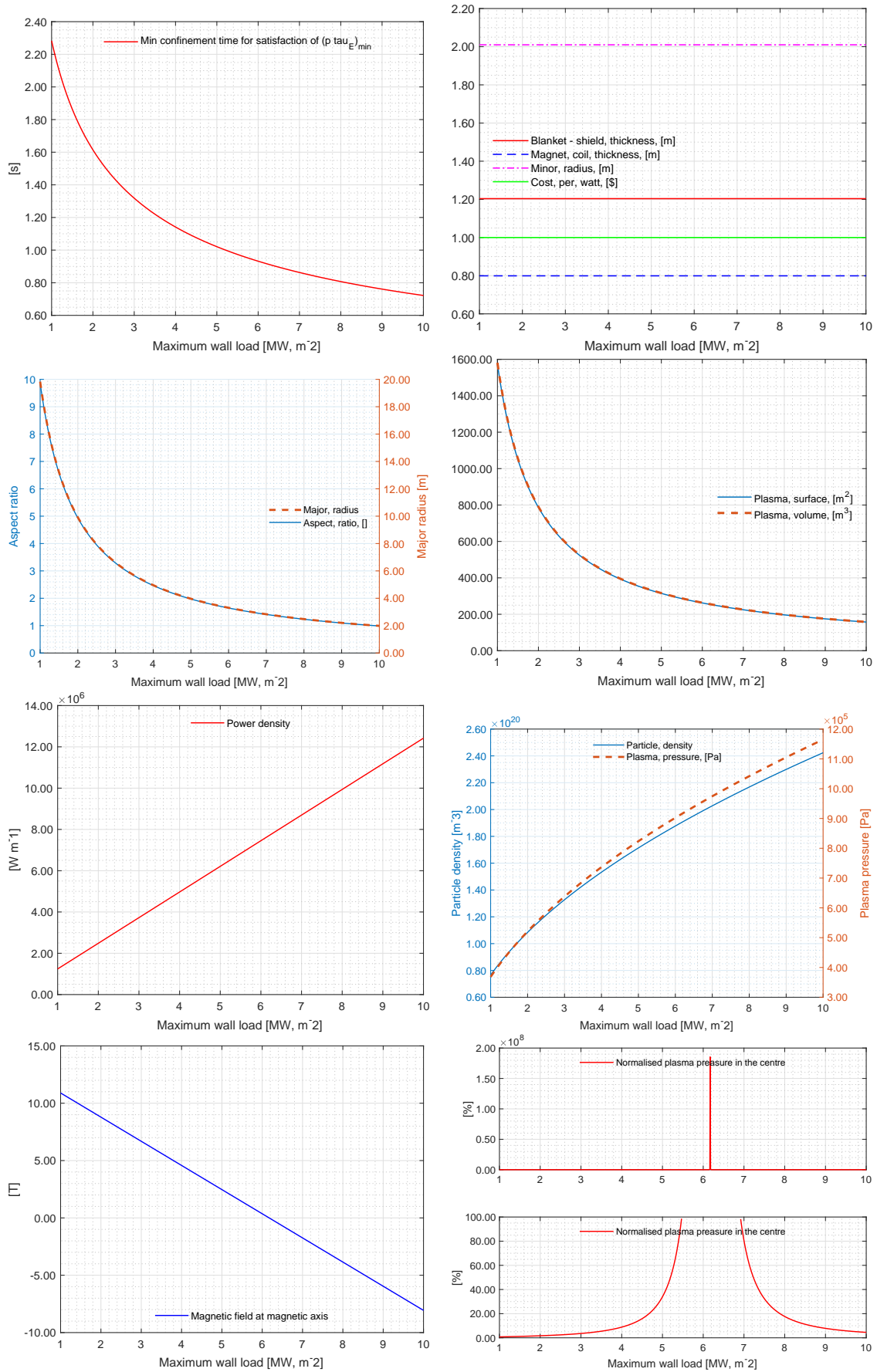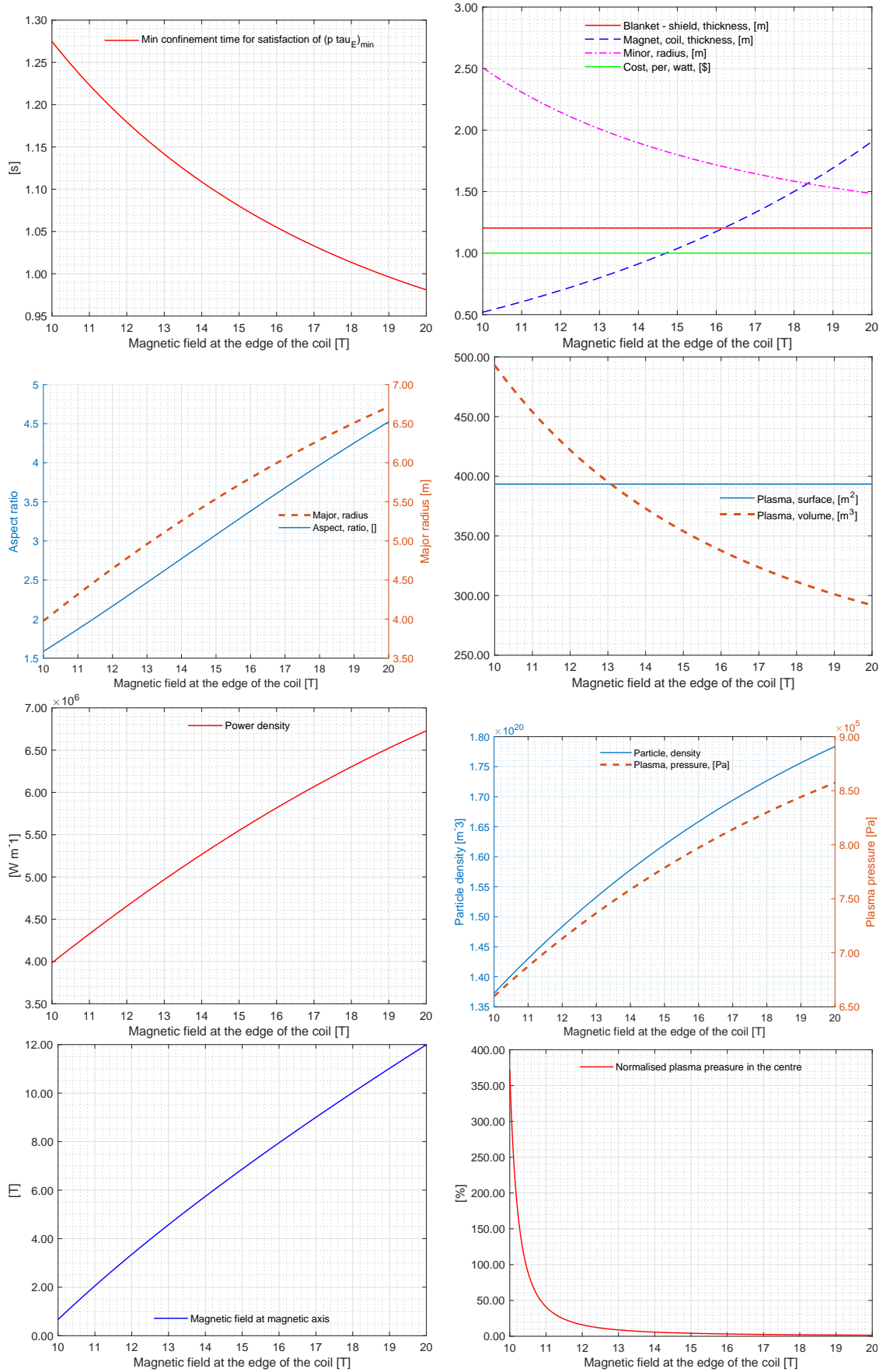```matlab
255   xlabel(titl(l));

256   ylabel('[%]');

257   ytickformat('%.2f');

258   grid on

259   grid minor

260   lgd = legend([pbeta], ...

261   { Normalised plasma preasure in the centre });% Figure legend

262   legend('boxoff')

263   legend('Location', 'north')

264   end

265   figcount = 8;

266   printout = (sprintf('Created figure %d of 8', figcount));

267   disp(printout)

268   % lgd = legend([pb, pc, pa, pr_0, pA, pA_p, pV_p,...

269   %   pP_dens, p, pn, pB_0, pbeta, ptau, pcpw], ...

270   %   {"Blanket-shield thickness [m]", "Magnet coil thickness [m]"...

271   %   "Minor radius [m]", "Major radius [m]", "Aspect ratio []"...

272   %   "Plasma surface [m^2]", "Plasma volume [m^3]", "Power density [W m^-1]"...

273   %   "Plasma pressure [Pa]", "Particle density [m^-3]",...

274   %   "Magnetic field at magnetic axis [T]", "Plasma beta in the centre []"...

275   %   "Min confinement time for satisfaction of (p tau_E)_min [s]",...

276   %   "The cost of the powerplant [$]"});% Figure legend

277

278

279   fignames = [f1 , f2 , f3 , f4 , f5 , f6 , f7 , f8];

280   figfigs = [f1, f2, f3, f4, f5, f6, f7, f8];

281

282

283   for i = 1:8

284   foldername = sprintf('../MatlabFigures/%s', foldertitl(l));

285   epsfilename = sprintf('%s.eps', fignames(i));

286   fullfilename = fullfile(foldername, epsfilename);
```

```
287         saveas(figfigs(i), fullfilename, 'epsc')

288         printout = (sprintf('Saved figure %d of 8', i));

289         disp(printout)

290     end

291

292     close all

293     printout = (sprintf('Completed iteration %d of 4', d));

294     disp(printout)

295 end
```

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

Appendix C: Iterations over Freidberg's model

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019

N. Kihm
R. K. F. Wiuff

Course 10401
Fusion Energy and Plasma Physics

January 24th
2019
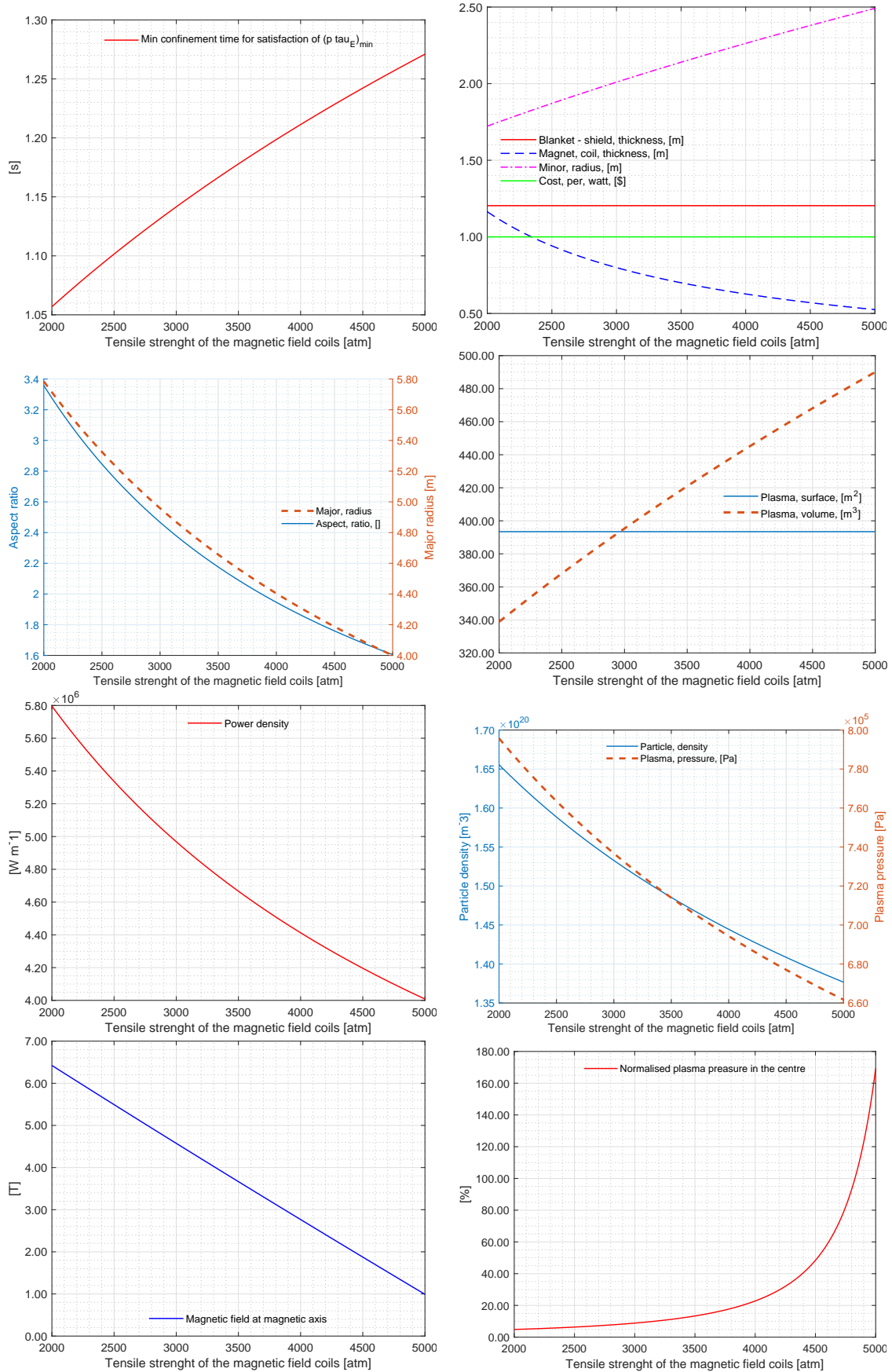
Appendix D: interferometer.m

```matlab
1   close all

2   clear all

3

4   mkdir('../MatlabFigures', 'Interferometer'); % Create save directory

5

6   %-------------------------Input Parameters----------------------------%

7   r_p = 0.055; % Reactor port opening [m]

8   w_0 = 0.0275; % Initial Beam Waist [m]

9   r = 0.125; % Minor tokamak radius [m]

10  freq = 60; % Frequency of probe beam [GHz]

11

12  d_0 = 0.20; % Distance between source and first lens

13  d_r = 0.10; % Distance between first lens and reactor wall

14  f_0 = 0.25; % Focal length of first lens

15  f_1 = 0.20; % Focal length of second lens

16  %---------------------------------------------------------------------%

17

18  %----------Distances and Lowest Beam Waists Calculations--------------%

19  c = 299792458; % The speec of light [m/s]

20  lambda = c / (freq * 10 9); % Calculates the wavelength [m]

21  d = f_0 + f_1; % Calculates the distance between lenses

22  w_1 = (lambda * f_0) / ...

23      (pi * w_0);% Calculates the beam waist between lenses

24  d_1 = (((d_0 / f_0) - 1) / ((w_0 (2) * pi) / ...

25      (f_0 * lambda) + ((d_0 / f_0) - 1) (2)) + 1) * ...

26      f_0;% Distance after 1st lens to lowest beam waist

27  w_2 = (f_1 / f_0) * w_0; % Calculates w_2 after 2nd second lens

28  d_2 = d - d_1; % Distance from w_2 to second lens

29  d_3 = f_1 / f_0 *...

30      (f_0 + f_1 -...
```

```matlab
31        (f_1 / f_0) * d_0);% Distance to lowest beam waist after 2nd lens

32   %------------------------------Page 47------------------------------------%

33

34   %----------------------Beam Waist Datapoints--------------------------%

35   x_0 = linspace(0, d_0, 1000);% Plotpoints till first lens

36   y_0 = [];

37   for i = 1:1000

38       w = w_0 * ...

39           sqrt(1+((lambda * x_0(i)) / ...

40           (pi * w_0 2)) 2);% Calculates the beam waist from source

41       y_0 = horzcat(y_0, w);

42   end

43

44   x_1 = linspace(0, d_1, 1000); % Plotpoints from first lens to w_1

45   y_1 = [];

46   for i = 1:1000

47       w = w_1 * ...

48           sqrt(1+((lambda * x_1(i)) / ...

49           (pi * w_1 2)) 2);% Beam waist between w_1 and the 1st lens

50       y_1 = horzcat(y_1, w);

51   end

52   y_1 = fliplr(y_1); % Flips the y-plot points as the beam waist is declining

53   y = horzcat(y_0, y_1);

54

55   x_2 = linspace(0, d_2, 1000); % Plotpoints from w_1 to second lens

56   y_2 = [];

57   for i = 1:1000

58       w = w_1 * ...

59           sqrt(1+((lambda * x_2(i)) / ...

60           (pi * w_1 2)) 2);% Beam waist between w_1 and the 2nd lens

61       y_2 = horzcat(y_2, w);

62   end
```

```
63
64    y = horzcat(y, y_2);

65

66    x_3 = linspace(0, d_3, 1000); % Plotpoints from 2nd lens to w_2
67    y_3 = [];
68    for i = 1:1000
69        w = w_2 * ...
70            sqrt(1+((lambda * x_3(i)) / ...
71            (pi * w_2 2)) 2);% Evolving beam waist after the 2nd lens to w_2
72        y_3 = horzcat(y_3, w);
73    end
74    y_3 = fliplr(y_3); % Flips the y-plot points as the beam waist is declining
75    y = horzcat(y, y_3);

76

77    for i = 1:1000
78        x_1(i) = x_1(i) + d_0; % Creates x-axis data points
79    end
80    for i = 1:1000
81        x_2(i) = x_2(i) + d_0 + d_1; % Creates x-axis data points
82    end
83    for i = 1:1000
84        x_3(i) = x_3(i) + d_0 + d_1 + d_2; % Creates x-axis data points
85    end
86    x = horzcat(x_0, x_1, x_2, x_3);

87

88    Opy = -y; % Flip beam waist to plot 'Beam Waist -'
89    %-----------------------------------------------------------------------------%

90

91    %--------------------Beam Waist Propagation Plot------------------------%
92    l = figure; % Creates figure
93    hold on
94    axis equal
```

```matlab
95   p1 = plot(x, y); % Plots Beam Waist+

96   p2 = plot(x, Opy, 'Color', 'red'); % Plots Beam Waist-

97   %p3 = xline(d_0, '--');

98   %xline(d_0+d_1+d_2, '--');

99   axPos = get(gca, 'Position'); % Get normalised axis coordinates

100  xMinMax = xlim; % Get normalised axis coordinates

101  yMinMax = ylim; % Get normalised axis coordinates

102  zAnn = axPos(1) + ((0 - xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...

103      * axPos(3);% Defines points relative to axis in normalised coordinates

104  d0Ann = axPos(1) + ((d_0 - xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...

105      * axPos(3);% Defines points relative to axis in normalised coordinates

106  d1Ann = axPos(1) + ((d_0 + d_1 - xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...

107      * axPos(3);% Defines points relative to axis in normalised coordinates

108  d2Ann = axPos(1) + ((d_0 + d_1 + d_2 - ...

109      xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...

110      * axPos(3);% Defines points relative to axis in normalised coordinates

111  d3Ann = axPos(1) + ((d_0 + d_1 + d_2 + d_3 - ...

112      xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...

113      * axPos(3);% Defines points relative to axis in normalised coordinates

114  yAnn = axPos(2) + ((0 - yMinMax(1)) / (yMinMax(2) - yMinMax(1))) ...

115      * axPos(4);% Defines points relative to axis in normalised coordinates

116  annotation('doublearrow', ...

117      [zAnn, d0Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_0

118  annotation('doublearrow', ...

119      [d0Ann, d1Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_1

120  annotation('doublearrow', ...

121      [d1Ann, d2Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_2

122  annotation('doublearrow', ...

123      [d2Ann, d3Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_3

124  annotation('ellipse', ...

125      [d0Ann - 0.01, yAnn - y_0(1000) * 1.5, 0.02, y_0(1000) * 3], ...

126      'FaceColor', 'cyan');% Draws 1st lens
```

N. Kihm
R. K. F. Wiuff
Course 10401
Fusion Energy and Plasma Physics
January 24th
2019

```matlab
127   annotation('ellipse', ...
128       [d2Ann - 0.01, yAnn - y_0(1000) * 1.5, 0.02, y_0(1000) * 3], ...
129       'FaceColor', 'cyan');% Draws 2nd lens
130   text(d_0/4, -0.22, sprintf('d_0 = %0.2f', d_0)) % Distance annotation
131   text(d_0+(d_1 / 4), -0.22, ...
132       sprintf('d_1 = %0.2f', d_1))% Distance annotation
133   text(d_0+d_1+(d_2 / 4), -0.22, ...
134       sprintf('d_2 = %0.2f', d_2))% Distance annotation
135   text(d_0+d_1+d_2+(d_3 / 4), -0.22, ...
136       sprintf('d_3 = %0.2f', d_3))% Distance annotation
137   text(d_0-0.06, 0.06, '1^{st} Lens') % Lens annotation
138   text(d_0-0.06, -0.06, sprintf('f_0 = %0.2f', f_0)) % Lens annotation
139   text(d_0+d_1+d_2-0.06, 0.06, '2^{nd} Lens') % Lens annotation
140   text(d_0+d_1+d_2-0.06, -0.06, ...
141       sprintf('f_1 = %0.2f', f_1))% Lens annotation
142   w1x = (d_0 + d_1); % w_1 coordinate
143   w1x = repelem(w1x, 100); % w_1 x-coordinate array
144   w1y = linspace(0, w_1); % w_1 y-coordinate array
145   p4 = plot(w1x, w1y); % Plots w_1
146   yline(0, '-'); % Plots optical axis
147   xlabel('Distance from source [m]'); % x-axis label
148   ylabel('Beam Waist [m]'); % y-axis label
149   title('Beam Waist In A Gaussian Beam Interferometer'); % Figure title
150
151   topstart = 2 * asin(r_p/(2 * r)) / 2; % Calculates reactor plot coordinates
152   topend = (2 * pi - 2 * topstart) / 2; % Calculates reactor plot coordinates
153   bottomstart = topend + topstart * 2; % Calculates reactor plot coordinates
154   bottomend = 2 * pi - topstart; % Calculates reactor plot coordinates
155
156   top = linspace(topstart, ...
157       topend, 100);% Creates 100 datapoints for the top half reactor cutout
158   x_t = d_0 + d_r + r + r * cos(top); % x-parameter of circle
```

```matlab
159  y_t = r * sin(top); % y-parameter of circle
160  bottom = linspace(bottomstart, ...
161      bottomend, 100);% Creates 100 datapoints for the bottom half reactor
162  x_b = d_0 + d_r + r + r * cos(bottom); % x-parameter of circle
163  y_b = r * sin(bottom); % y-parameter of circle
164  p5 = plot(x_t, y_t, 'color', ...
165      [0.911, 0.4100, 0.1700]);% Plot top half reactor
166  plot(x_b, y_b, 'color', ...
167      [0.911, 0.4100, 0.1700]);% Plot bottom half reactor
168  lgd = legend([p1, p2, p4, p5], ...
169      {'Beam Waist +', 'Beam Waist -', 'w_1', 'Reactor'});% Figure legend
170  legend('boxoff')
171  legend('Location', 'northwest')
172  lgd.NumColumns = 2;
173  hold off
174  %------------------------------------------------------------------------%
175
176  %-----------------Saving figure as Encapsulated Postscript---------------%
177  epsfilename = 'Interferometer.eps'; % Savename for the figure
178  foldername = sprintf('../MatlabFigures/Interferometer'); % Folder path
179  fullfilename = fullfile(foldername, epsfilename); % Filename path
180  saveas(l, fullfilename, 'epsc') % Save the figure as eps
181  %------------------------------------------------------------------------%
182
183  %------------------------Print Outputs-----------------------------------%
184  w_1
185  w_2
186  d_1
187  d_2
188  d_3
189  %------------------------------------------------------------------------%
```

Appendix E: PhaseShiftDensity.m

```matlab
close all;

clear all;


mkdir('../MatlabFigures', 'PhaseShift'); % Create save directory


%------------------------Input Parameters----------------------------%
f_0 = 60; % 1st beam frequency [GHz]

f_1 = 98; % 2nd beam frequency [GHz]

f_2 = 130; % 3rd beam frequency [GHz]


n_0 = 10 16; % Lower electron density

n_1 = 10 18; % Hihger electron density

%-------------------------------------------------------------------%


c = 299792458; % The speec of light [m/s]


y = linspace(n_0, n_1, 1000); % Datapoints between n_0 and n_1

Phi0 = []; % Phase shift array

Phi1 = []; % Phase shift array

Phi2 = []; % Phase shift array


for i = 1:1000 % Calculates Phase Shifts for frequency = f_0

    Phi0 = horzcat(Phi0, 8.416e-8*(y(i) / (f_0 * 10 9)));

end

for i = 1:1000 % Calculates Phase Shifts for frequency = f_1

    Phi1 = horzcat(Phi1, 8.416e-8*(y(i) / (f_1 * 10 9)));

end

for i = 1:1000 % Calculates Phase Shifts for frequency = f_2

    Phi2 = horzcat(Phi2, 8.416e-8*(y(i) / (f_2 * 10 9)));

end
```

```matlab
l = figure; % Creates figure

hold on

p0 = plot(Phi0, y); % Plot electron denisty as function of phase shift

p1 = plot(Phi1, y); % Plot electron denisty as function of phase shift

p2 = plot(Phi2, y); % Plot electron denisty as function of phase shift

xlabel('Phase shift [^{1}/_{2\pi}]'); % x-axis label

ylabel('Electron Density [m^-3]'); % y-axis label

%title('Electron Density as function of measured phase shift'); % Figure title

lgd = legend([p0, p1, p2], ...
    {'60GHz', '98GHz', '130GHz'});% Figure legend

legend('boxoff')

legend('Location', 'northwest')

grid on

grid minor

hold off


%------------------Saving figure as Encapsulated Postscript--------------%

epsfilename = 'PhaseShift.eps'; % Savename for the figure

foldername = sprintf('../MatlabFigures/PhaseShift'); % Folder path

fullfilename = fullfile(foldername, epsfilename); % Filename path

saveas(l, fullfilename, 'epsc') % Save the figure as eps

%-----------------------------------------------------------------------%
```