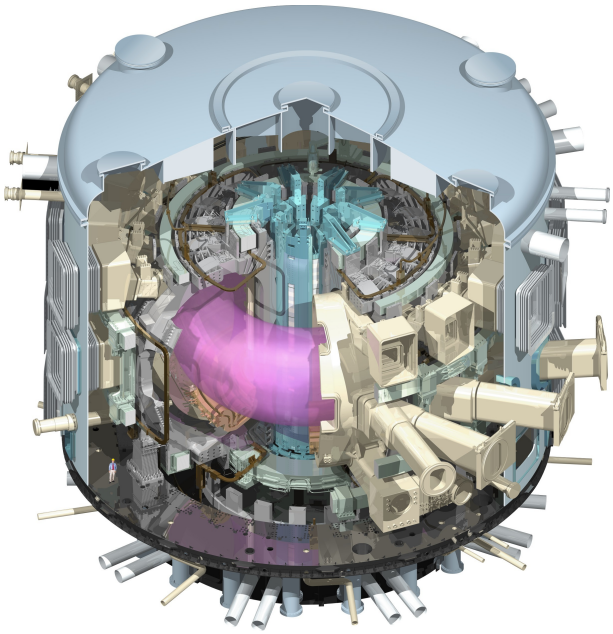
10401

Fusion Energy and Plasma Physics

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

Technical University of Denmark‡

(Dated: January 24th 2019)



Abstract: Write abstract

DTU

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 5$^{\text{th}}$ 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:

$$\begin{aligned} n_{\text{flux fraction}} &= 0.01 & P_E &= 1000 \text{ MW} \\ P_W &= 4 \text{ MW}\cdot\text{m}^{-2} & B_{\max} &= 13 \text{ T} \\ \sigma_{\max} &= 3000 \text{ atm} & \eta_t &= 0.4 \end{aligned} \tag{1}$$

Note that C_F and C_I has been ommited 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.

Symbol	Quantity
$n_{\text{flux fraction}}$	n flux in breeder end/n flux in breeder start []
C_F	Fixed cost propotionality constant [$]
C_I	Nuclear island cost propotionality constant $[\$ \cdot \text{W}{\cdot}\text{m}^{-3}]$
P_E	Desired output power [MW]
P_W	Maximum wall load $[\text{MW}{\cdot}\text{m}^{-2}]$
$B_{\max}$	Magnetic field at the edge of the coil [T]
$\sigma_{\max}$	Tensile strenght of the magnetic field coils [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 powerplant. 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.

$$\begin{aligned} P_E &: 500 - 1000 \text{ MW} & P_W &: 0 - 10 \text{ MW}{\cdot}\text{m}^{-2} \\ B_{\max} &: 0 - 20 \text{ T} & \sigma_{\max} &: 2000 - 5000 \text{ atm} \end{aligned} \tag{2}$$

Appendix B includes Matlab code that does the folowing:

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 strenght 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.

Add to this section

The full gallery of model output is found in Appendix C

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 $\mathrm{W{\cdot}m^{-1}}$
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%
$\tau_{E_{\mathrm{min}}}$	Min confinement time for $(p \times \tau_E)_{\mathrm{min}}$	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:

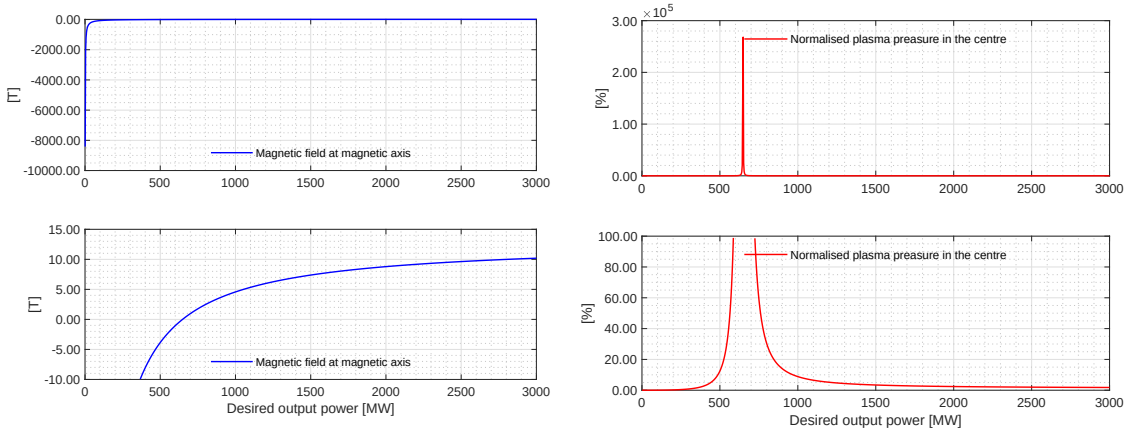
$$\kappa = \frac{a_{\mathrm{max}}}{a_{\mathrm{min}}} \tag{3}$$

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,

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

which can be parameterised as

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



(a) The magnetic field at magnetic axis

(b) The normalised plasma preasure 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 outher 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

$$\begin{aligned} V_I &= 2\,\pi\, R_0\, (A_{e,\,\text{outer}} - A_{e,\,\text{inner}}) \\ &= 2\,\pi^2\, R_0\, \left((a_{\min} + b + c)^2 - a_{\min}^2\right)\, \kappa \end{aligned} \tag{7}$$

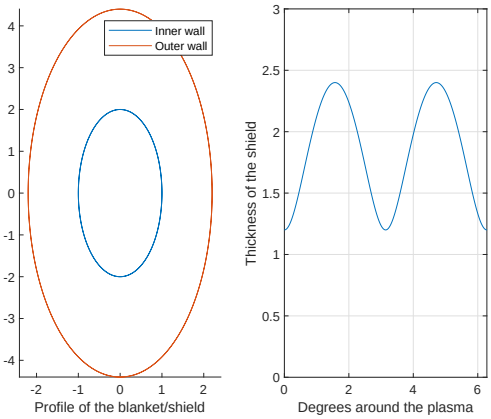


Figure 2: The profile of the blanket-and-shield and the thickness as a function of the angle around origo, ϕ.

and the plasma volume is similarly calculated as $V_\mathrm{P} = 2\,\pi^2\,R_0\,\kappa\,a_\mathrm{min}^2$. The plasma surface area is a bit more tricky, but it can be approximated as

$$S_\mathrm{p} = 2\,\pi\,R_0\,\pi\,(3\,(a_\mathrm{min} + \kappa\,a_\mathrm{min}) - \sqrt{(3\,a_\mathrm{min} + \kappa\,a_\mathrm{min})\,(a_\mathrm{min} + 3\,\kappa\,a_\mathrm{min})}) \tag{8}$$

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 $\kappa\,a_\mathrm{min}$ 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 $\xi = B_{sic}^2/4\,\mu_0\,\sigma_\mathrm{max}$. 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 $\tau_{\mathrm{E}_\mathrm{min}}$ while changing a bit, they were not changed significantly.

D. Main parameters for DEMO

Setting $P_\mathrm{E} = 2000$ 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 $A = R_0/a_\mathrm{min} = 3 \Leftrightarrow R_0 = 3\,a_\mathrm{min}$. 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 m^2
V_p	Plasma volume	793 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	4.60 T
β	Normalised plasma pressure	6.17%
$\tau_{E_{\min}}$	Min confinement time for $(p \times \tau_E)_{\min}$	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

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

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

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 $P_{\text{E}} = 2\text{GW}$

minimum R_0 is calculated. It is simply $R_0 = a_{\text{min}} + b + c$ where a_{min} is to be determined, $b = 1.2\text{m}$ and $c = \frac{2\xi}{1-\xi}(\kappa\, a_{\text{min}} + b)$. Meanwhile A is chosen to optimize 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 ($\beta = 10\%$ $a = 1.938\text{m}$) is achieved. Therefore $R_0 = 1.9378\text{m} + 1.2\text{m} + 2 \cdot 0.11/(1 - 0.11) \cdot 2.5 \cdot 1.938\text{m} + 1.2\text{m} = 4.632\text{m}$ which means $A = 4.632\text{m}/1.938\text{m} = 2.390$

Add captions

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%
$\tau_{E_{\min}}$	Min confinement time for $(p \times \tau_E)_{\min}$	1.26 s

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

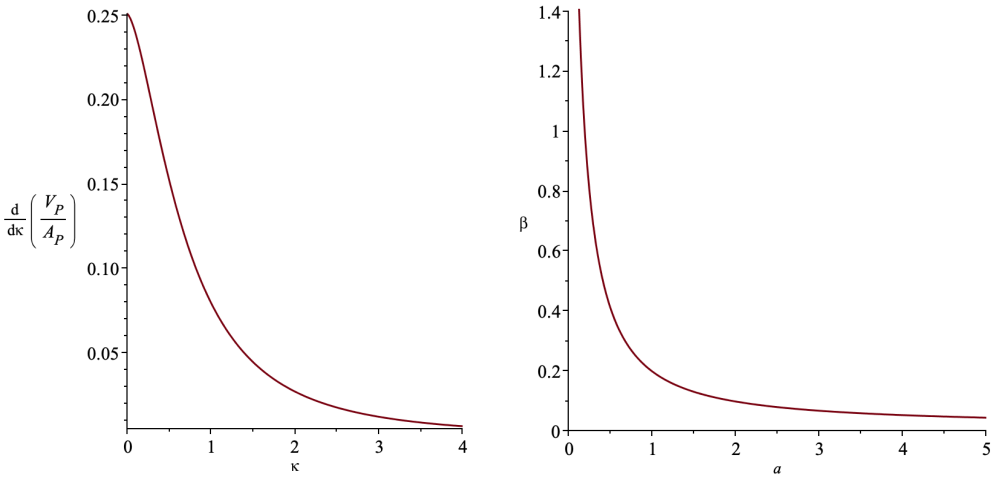
II. DIAGNOSTICS VIA INTERFEROMETRY

When operating a fusion reactor a continous 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:

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



(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 $\mathrm{W{\cdot}m^{-1}}$
p	Plasma pressure	1.02×10^5 Pa
n	Particle density	2.12×10^{20} $\mathrm{m^{-3}}$
B_0	Magnetic field at magnetic axis	−3.09 T
β	Normalised plasma pressure	26.9%
$\tau_{E_{\mathrm{min}}}$	Min confinement time for $(p \times \tau_E)_{\mathrm{min}}$	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:

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

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

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

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:

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

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

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

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:

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

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:

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

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

$$\frac{\Phi}{2\pi} \approx 4.48 \times 10^{-16} \left(\frac{\lambda}{\mathrm{m}}\right) n_e 0.250 \text{ m}\sqrt{2\pi} = 1.12 \times 10^{-16} \; n_e \left(\frac{\sqrt{2\pi}c}{\omega \mathrm{m}}\right)$$
$$= 8.416 \times 10^{-8} \left(\frac{n_e}{\omega \mathrm{s}}\right) \tag{24}$$

Remembering Eq. (17)

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

$$\Downarrow$$

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

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 $f = 60$GHz.

B. Beam Phaseshifts

The goal of the interferometer is to find the phaseshift between the microwave beam in vacuum and in plasma. Our suggestion is a setup involving a interferometer emitting a beam through the center 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

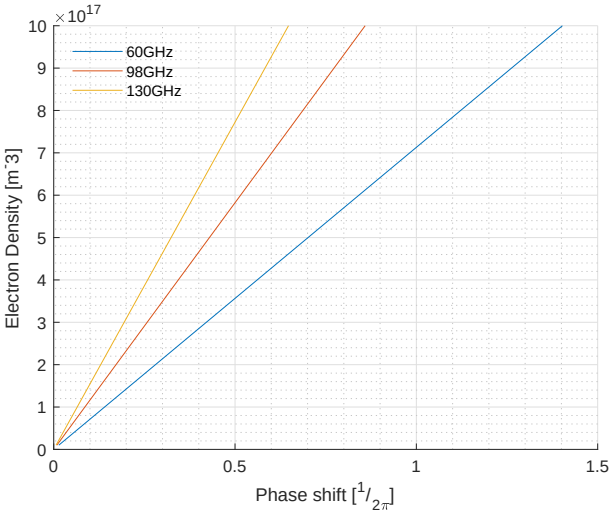


Figure 4: Electron densities for different wavelengths and phaseshifts

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 phaseshifts. 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 functino describing the waist. This equation is:

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

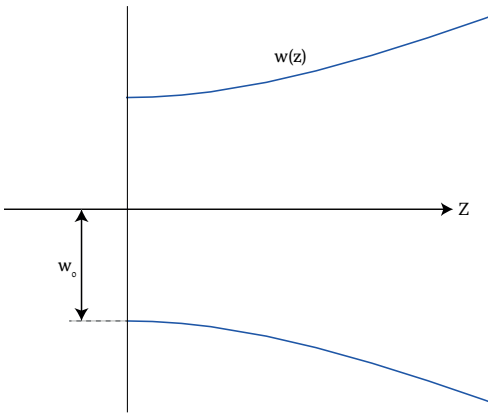


Figure 5: Sketch of Gaussian beam propagation with indication of w_0 and $w(z)$.

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

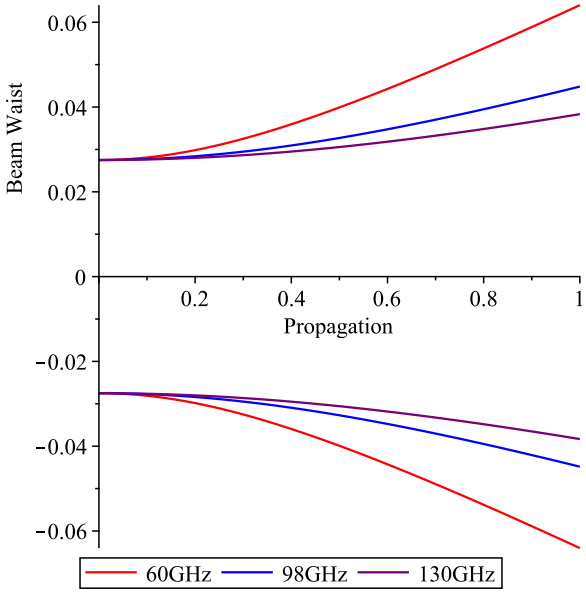


Figure 6: Beam Waist for the three microwave sources. The functions are Eqs. (32), (33) and (34).

It is undesireable 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 nesessary 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 reciever. 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:

$$\begin{aligned} r_p = 0.0550 \text{ m} \quad r = 0.125 \text{ m} \quad w_0 = 0.0275 \text{ m} \quad freq = 60 \text{ GHz} \\ d_0 = 0.200 \text{ m} \quad d_r = 0.100 \text{ m} \quad f_0 = 0.250 \text{ m} \qquad 0.200 \text{ m} \end{aligned} \tag{41}$$

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:

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

Furtermore 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 center and out, the plasma itself act as a variable lens with variable refractive indices. If the beam is emitted straight through the center, the refractive indices will only shorten and lengthen the wavelength back to normal, however if the beam is emmited off center, it will be refracted away from the center. 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.

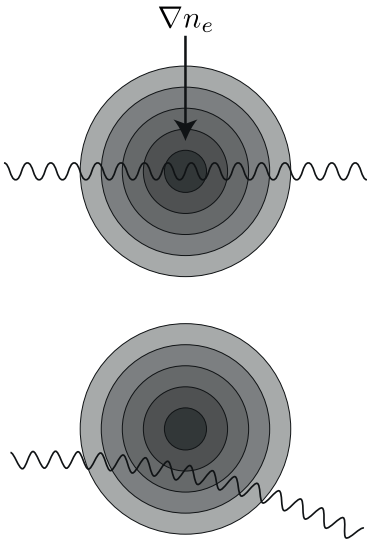


Figure 8: Refraction of the probing beam at center (top) and off center (bot-

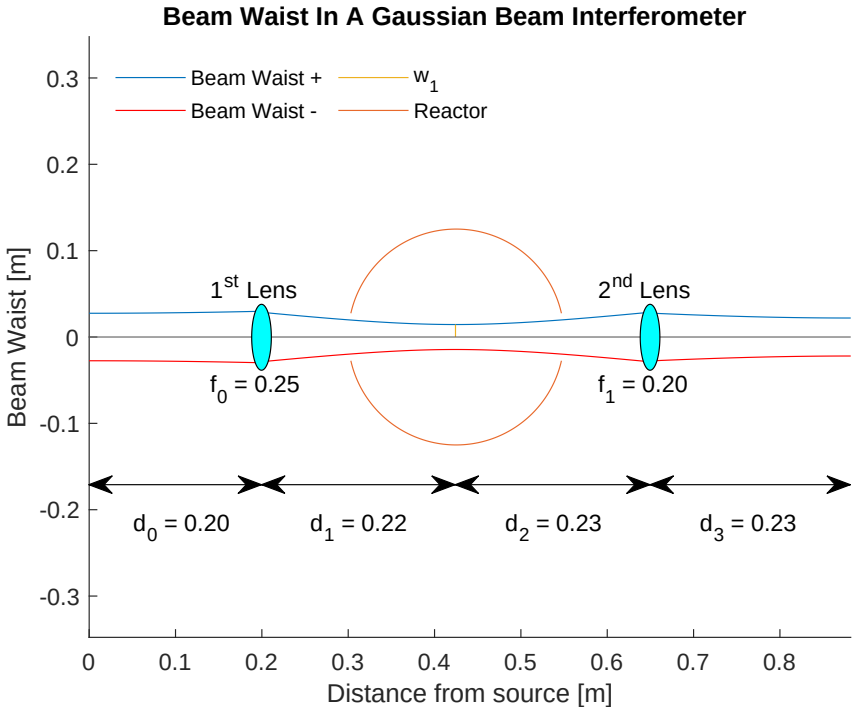


Figure 7: Gaussian beam telecsope interferometer setup with input parameters shown in Eq. (41) and output parameters shown in Eq. (42)

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.

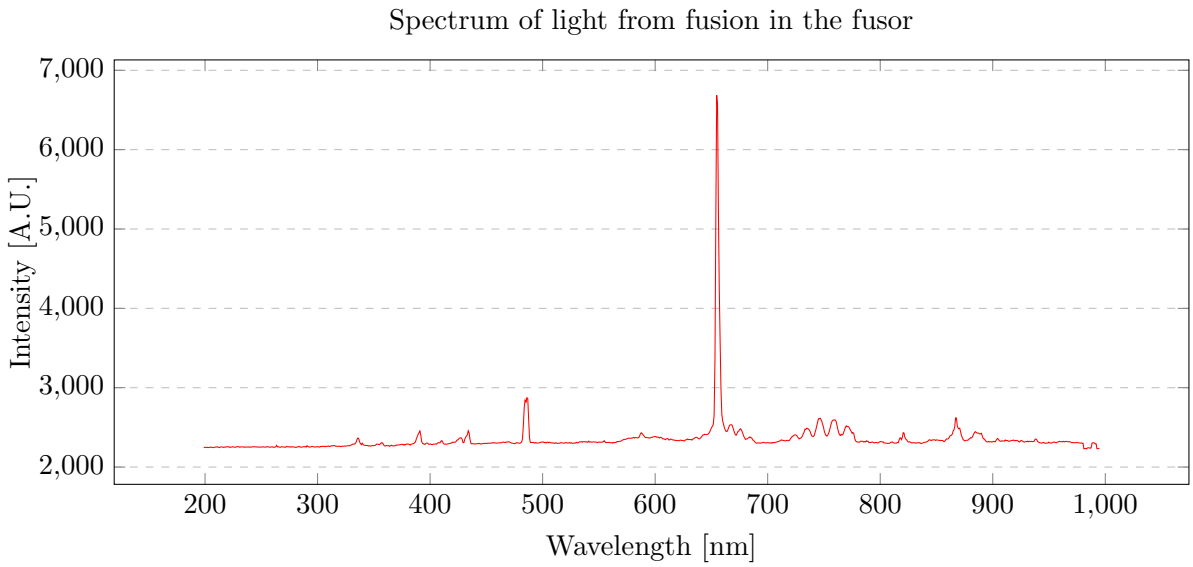


Figure 9: Spectrum of light from fusion in the fusor

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 Fig. 10, a spectral line is shown with a central wavelength of around 656 nm which correspond to an accepted value of hydrogen light emission[4]. Meanwhile the spectral lines of Deuterium only differs by a factor of 1.000272[5], which will not be detected using our methods. Two much smaller peaks at 486 nm and 433 nm where also found. These are also in agreement where the 433 nm is of by 1 nm. This is due to uncertainties, as each wavelength is found using a Gaussian fit.

Meanwhile the spectral line width is expected to increase with electric potential. This is because that increasing the potential between the electrodes means increasing the electric field which in turn increases the acceleration on the ions due to the Lorentz force. This means that more particles will be accelerated to higher speeds which increases the Doppler broadening around the normal wavelength.

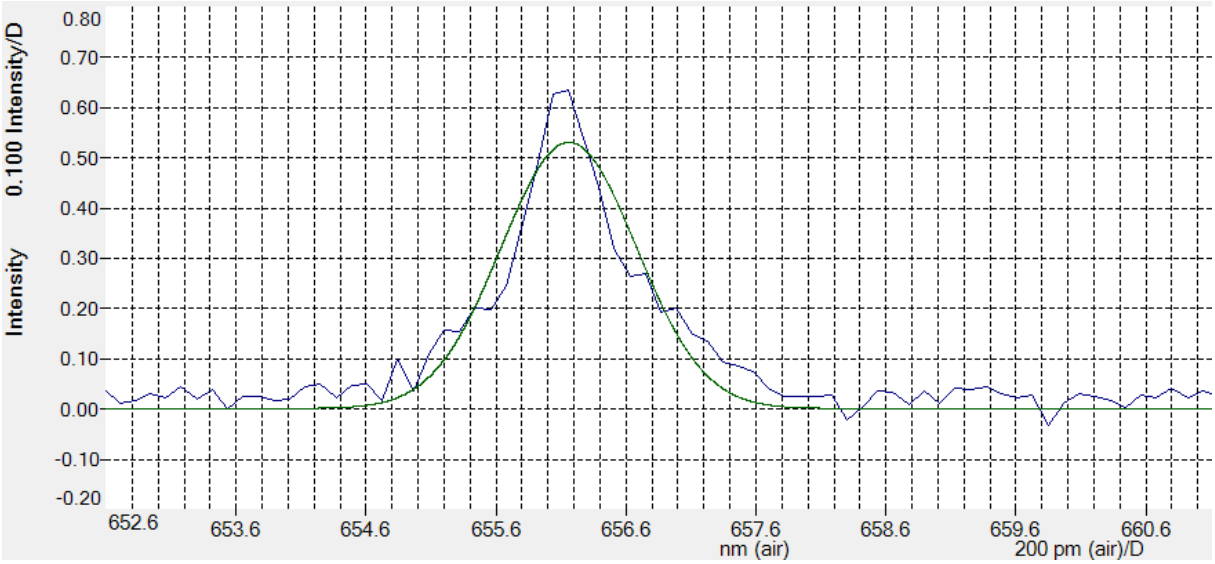


Figure 10: Spectral line for deuterium measured in the fusor.

B. Neutron production rate

The fusion reaction rate should increase with voltage and current. Assuming that the reaction rate increases linearly with current, the largest effect is seen from the voltage increase.

TODO LIST

* E-mail at s163977@student.dtu.dk

† E-mail at spacrone@live.dk

‡ Homepage of the Technical University of Denmark http://www.dtu.dk/english/;

Project Repository: https://github.com/rwiuff/10401

[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).

LIST OF FIGURES

LIST OF TABLES

LISTINGS

Appendices

Appendix A: tokamakDTU_asign_1

```
%*************************************************************************
% 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)
```

```
%TOKAMAK_DTU Function which returns the parameters of a power plant
%
% Output parameters
%------------------
% b           - Blanket/shield thickness [m]
% c           - Magnet coil thickness [m]
% a           - Minor radius [m]
% R_0         - Major radius [m]
% A           - Aspect ratio []
% A_p         - Plasma surface [m^2]
% V_p         - Plasma volume [m^3]
% P_dens      - Power density [W/m]
% p           - Plasma pressure [Pa]
% n           - Particle density [m^-3]
% B_0         - Magnetic field at magnetic axis [T]
% beta        - Plasma beta in the centre []
% tau_E_min   - Min confinement time for satisfaction of (p*tau_E)_min [s]
% C_per_watt - The cost of the powerplant [$]
%
% Input parameters
%-------------------
% n_flux_fraction - n flux in breeder end/n flux in breeder start []
% C_F             - Fixed cost propotionality constant [$]
% C_I             - Nuclear island cost propotionality constant [$W/m^3]
% P_E             - Desired output power [MW]
% P_W             - Maximum wall load [MW/m^2]
% B_max           - Magnetic field at the edge of the coil [T]
% sigma_max       - Tensile strenght of the magnetic field coils [atm]
% eta_t           - Energy conversion efficiency []



% The function starts by defining fixed constants
```

```
% Note that this is inefficient if we are looping over the function, but it
% makes the code easier to use, as these are not needed as input parameters

% Fixed constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nuclear
%--------
% Energies
E_t         = 2.5e-8; % [MeV] Energy of slow (thermal) neutron (eq 5.6)
E_n         = 14.1;   % [MeV] Neutron energy after fusion (eq 2.17)
E_a         = 3.5;    % [MeV] alpha energy after fusion (eq 2.17)
E_Li        = 4.8;    % [MeV] Heat produced by breeding Li (under eq 4.31)
% Cross section and main free paths
sigma_v_avg = 3.0e-22;% [m^3/s] DT fusion cross section @ 15keV (table 5.2)
lambda_br   = 0.0031; % [m] Breeding mean free path (under eq 5.7)
lambda_sd   = 0.055;  % [m] Mean free path from sigma_sd (eq 5.3)

% Plamsa physics
%---------------
% Parameters for infinity gain at the minimum of p tau_E (eq 4.20)
T            = 15.0;  % [keV] Temparature for obtaining min tripple product
tripple_min  = 8.3;   % [atm s] Min tripple prod to obtain Q=inf @ T=15 keV

% Natural constants
%------------------
mu_0 = 4.0*pi*1e-7;         % Vacuum permeability [T*m/A]
e    = 1.602176565e-19;   % Elementary charge [C]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Secondly we convert everything to SI units, so that the variables are
% easier to handle
```

```
% Again, this is computationally inefficient, but it suffices for our use
% Conversion to SI-units
%##########################################################################
% Conversion factors
W_per_MW     = 1.0e6;
Pa_per_atm   = 1.01325e5;
eV_per_keV   = 1.0e3;
eV_per_MeV   = 1.0e6;
J_per_eV     = e;
J_per_keV    = J_per_eV * eV_per_keV;
J_per_MeV    = J_per_eV * eV_per_MeV;
% Conversions
P_E          = P_E * W_per_MW;          % Desired output power
P_W          = P_W * W_per_MW;          % Wall Loading limit on first wall
E_t          = E_t * J_per_MeV;         % Energy of slow (thermal) neutron
E_n          = E_n * J_per_MeV;         % Neutron energy after fusion
E_a          = E_a * J_per_MeV;         % alpha energy after fusion
E_Li         = E_Li * J_per_MeV;        % Heat produced by breeding Li
sigma_max    = sigma_max * Pa_per_atm;  % Max allowable structural stress
T            = T * J_per_keV;           % Temparature for minimum p*tau_E
tripple_min  = tripple_min * Pa_per_atm;  % The Lawson parameter (p*tau_E)
%##########################################################################



% Calculate the geometrical factors
%---------------------------------
% Find the breeder thickness
b = get_b(lambda_sd, E_n, E_t, lambda_br, n_flux_fraction);
% Find the minor plasma radius and the coil thickness
[a, c] = get_a_and_c(B_max, mu_0, sigma_max, b);
% Find the major radius
R_0 = get_R_0(a, eta_t, E_n, E_a, E_Li, P_E, P_W);
```

```
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);
```

```
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
```

```
% C_p_watt = C_F + C_I*(V_I/P_E);
% Since the cost per watt contains two constants, we can minimize the
% V_I/P_E in order to optimize the cost
% Given by equation 5.20 inserted in 5.17
% Equation 5.21
% V_I_per_P_E = V_I/P_E;  % Reactor volume per power out
% a is found by setting the derivative of V_I_per_P_E = 0
% Equation 5.29
a = ((1.0 + xi)/(2.0*xi (1.0/2.0))) * b;

% Knowing xi, a, and, b, we can calculate c
% c found by comparing tensile force and magnetic force working on the coil
% Equation 5.27
c = 2*xi/(1-xi)*(a+b);
end



function       = get_R_0(                                  )
%GET_R_0 Calculate the major radius

% Divide eq 5.18 (electric power out) by
% eq 5.19 (wall loading * area = total neutron production) and solve for R0
% Equation 5.20
R_0 = (1.0/(4.0*pi (2.0)*eta_t))*(E_n/(E_n + E_a + E_Li))*(P_E/(a*P_W));
end



function          = get_P_dens(                                  )
%GET_P_DENS Calculate the power density

```

```
% The power density is found by the sum of the power from the alphas plus
% the power from the neutrons, divided by the plasma volume
% Equation 5.35
P_dens = (E_a + E_n)/(E_a + E_n + E_Li)*P_E/(eta_t*V_p);
end



function       = get_B_0(               )
%GET_B_0 Calculte the magnetic field strength on the magnetic axis

% B_max is found in the edge of the magnet (at R = R_0-a-b)
% B0 is the magnetic field at R0
% As B propto 1/R. we have that B_0/B_max = (R_0-a-b)/R_0, which leads to
% Equation 5.42
B_0 = ((R_0-a-b)/R_0)*B_max;
end



function        = get_beta(            )
%GET_beta Calculte the magnetic field strength on the magnetic axis

% Plasma beta in the center (kinetical pressure over magnetical pressure):
% Equation 5.43
beta = p / (B_0 2/(2.0*mu_0));
end



function     = get_p(                                )
%GET_P Calculate the plasma pressure
```

```
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
```

Appendix B: IterateTokamakDTU

```
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] = ...
```

```
                tokamakDTU_asign_1(0.01, 1, 2, 1000, x(i), 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 == 3
        x = linspace(10, 20, 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, 1000, 4, x(i), 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 == 4
        x = linspace(2000, 5000, 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, 1000, 4, 13, x(i), 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
    end
    printout = (sprintf('Iterated tokamakDTU_asign_1.m %d of 4', d));
    disp(printout)
    b = p0(:, 1);
    c = p0(:, 2);
    a = p0(:, 3);
    R_0 = p0(:, 4);
```

```
A = p0(:, 5);
A_p = p0(:, 6);
V_p = p0(:, 7);
P_dens = p0(:, 8);
p = p0(:, 9);
n = p0(:, 10);
B_0 = p0(:, 11);
beta = p0(:, 12);
beta = beta * 100;
tau_E_min = p0(:, 13);
C_per_watt = p0(:, 14);

f1 = figure;
ptau = plot(x, tau_E_min, 'color', 'r', 'LineWidth', 1);
xlabel(titl(1));
ylabel('[s]');
ytickformat('%.2f');
grid on
grid minor
lgd = legend([ptau], ...
    { Min confinement time for satisfaction of (p tau_E)_{min} });% Figure legend
legend('boxoff')
legend('Location', 'north')
figcount = 1;
printout = (sprintf('Created figure %d of 8', figcount));
disp(printout)

f2 = figure;
pb = plot(x, b, 'color', 'r', 'LineWidth', 1);
hold on
pc = plot(x, c, '--', 'color', 'b', 'LineWidth', 1);
pa = plot(x, a, '-.', 'color', 'm', 'LineWidth', 1);
```

```
95      pcpw = plot(x, C_per_watt, 'color', 'g', 'LineWidth', 1);
96      hold off
97      xlabel(titl(1));
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(1));
125     ytickformat('%.2f');
126     lgd = legend([pr_0, pA], ...
```

```
        { Major, radius ,  Aspect, ratio, [] });% Figure legend
    legend('boxoff')
    legend('Location', 'east')
    grid on
    grid minor
    figcount = 3;
    printout = (sprintf('Created figure %d of 8', figcount));
    disp(printout)

    f4 = figure;
    pA_p = plot(x, A_p, 'LineWidth', 1);
    hold on
    pV_p = plot(x, V_p, '--', 'LineWidth', 2);
    hold off
    xlabel(titl(1));
    ytickformat('%.2f');
    lgd = legend([pA_p, pV_p], ...
        { Plasma, surface, [m 2] ,  Plasma, volume, [m 3] });% Figure legend
    legend('boxoff')
    legend('Location', 'east')
    grid on
    grid minor
    figcount = 4;
    printout = (sprintf('Created figure %d of 8', figcount));
    disp(printout)

    f5 = figure;
    pP_dens = plot(x, P_dens, 'color', 'r', 'LineWidth', 1);
    xlabel(titl(1));
    ylabel('[W m^-1]');
    ytickformat('%.2f');
    grid on
```

```
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(1));
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);
```

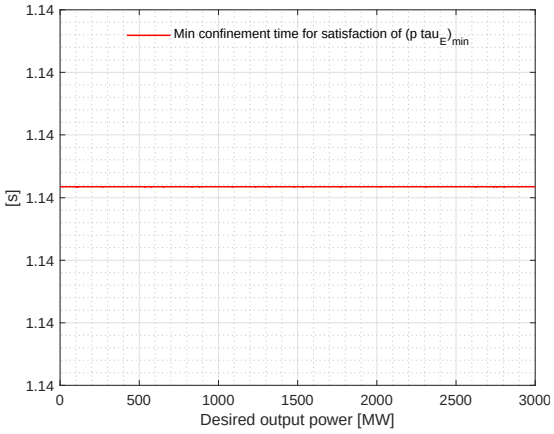
```
        pB_0 = plot(x, B_0, 'color', 'b', 'LineWidth', 1);
        ylabel('[T]');
        ytickformat('%.2f');
        grid on
        grid minor
        lgd = legend([pB_0], ...
        { Magnetic field at magnetic axis });% Figure legend
        legend('boxoff')
        legend('Location', 'south')
        subplot(2, 1, 2);
        B_0(B_0 <= -10) = NaN;
        pB_0 = plot(x, B_0, 'color', 'b', 'LineWidth', 1);
        xlabel(titl(1));
        ylabel('[T]');
        ytickformat('%.2f');
        grid on
        grid minor
        lgd = legend([pB_0], ...
        { Magnetic field at magnetic axis });% Figure legend
        legend('boxoff')
        legend('Location', 'south')
    else
        f7 = figure;
        pB_0 = plot(x, B_0, 'color', 'b', 'LineWidth', 1);
        xlabel(titl(1));
        ylabel('[T]');
        ytickformat('%.2f');
        grid on
        grid minor
        lgd = legend([pB_0], ...
        { Magnetic field at magnetic axis });% Figure legend
        legend('boxoff')
```

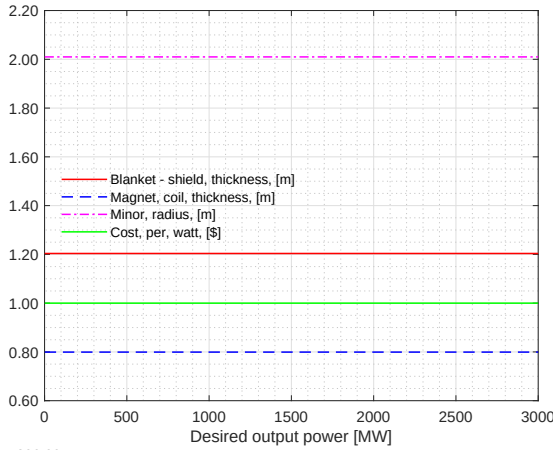
```
        legend('Location', 'south')
    end
    figcount = 7;
    printout = (sprintf('Created figure %d of 8', figcount));
    disp(printout)
    if l == 1 || l == 2
        f8 = figure;
        subplot(2, 1, 1);
        pbeta = plot(x, beta, 'color', 'r', 'LineWidth', 1);
        ylabel('[%]');
        ytickformat('%.2f');
        grid on
        grid minor
        lgd = legend([pbeta], ...
        { Normalised plasma preasure in the centre });% Figure legend
        legend('boxoff')
        legend('Location', 'north')
        subplot(2, 1, 2);
        beta(beta >= 100) = NaN;
        pbeta = plot(x, beta, 'color', 'r', 'LineWidth', 1);
        xlabel(titl(l));
        ylabel('[%]');
        ytickformat('%.2f');
        grid on
        grid minor
        lgd = legend([pbeta], ...
        { Normalised plasma preasure in the centre });% Figure legend
        legend('boxoff')
        legend('Location', 'north')
    else
        f8 = figure;
        pbeta = plot(x, beta, 'color', 'r', 'LineWidth', 1);
```

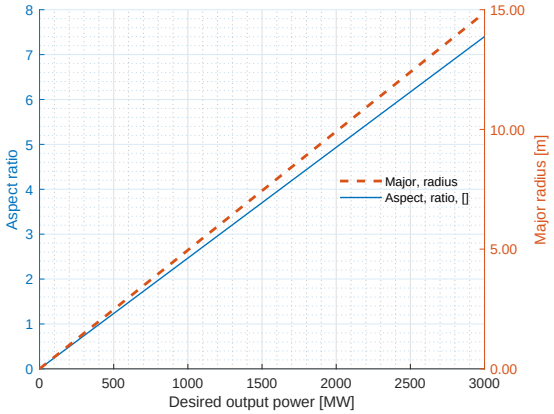
```
        xlabel(titl(l));
        ylabel('[%]');
        ytickformat('%.2f');
        grid on
        grid minor
        lgd = legend([pbeta], ...
        { Normalised plasma preasure in the centre });% Figure legend
        legend('boxoff')
        legend('Location', 'north')
    end
    figcount = 8;
    printout = (sprintf('Created figure %d of 8', figcount));
    disp(printout)
    % lgd = legend([pb, pc, pa, pr_0, pA, pA_p, pV_p,...
    %     pP_dens, p, pn, pB_0, pbeta, ptau, pcpw], ...
    %     {"Blanket-shield thickness [m]", "Magnet coil thickness [m]"...
    %     "Minor radius [m]", "Major radius [m]", "Aspect ratio []"...
    %     "Plasma surface [m^2]", "Plasma volume [m^3]", "Power density [W m^-1]"...
    %     "Plasma pressure [Pa]", "Particle density [m^-3]",...
    %     "Magnetic field at magnetic axis [T]", "Plasma beta in the centre []"...
    %     "Min confinement time for satisfaction of (p tau_E)_min [s]",...
    %     "The cost of the powerplant [$]"});% Figure legend


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


    for i = 1:8
        foldername = sprintf('../MatlabFigures/%s', foldertitl(l));
        epsfilename = sprintf('%s.eps', fignames(i));
        fullfilename = fullfile(foldername, epsfilename);
```

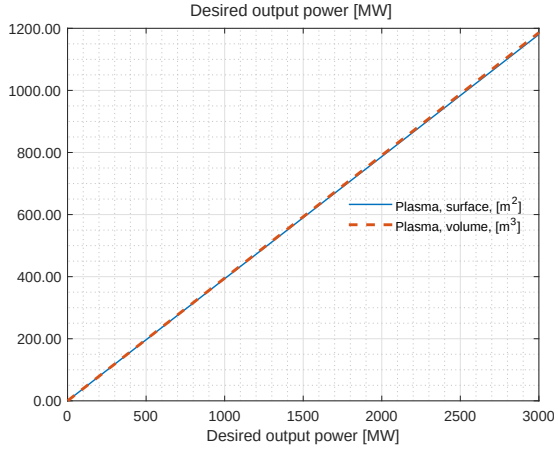
```
287         saveas(figfigs(i), fullfilename, 'epsc')
288         printout = (sprintf('Saved figure %d of 8', i));
289         disp(printout)
290     end
291
292     close all
293     printout = (sprintf('Completed iteration %d of 4', d));
294     disp(printout)
295 end
```

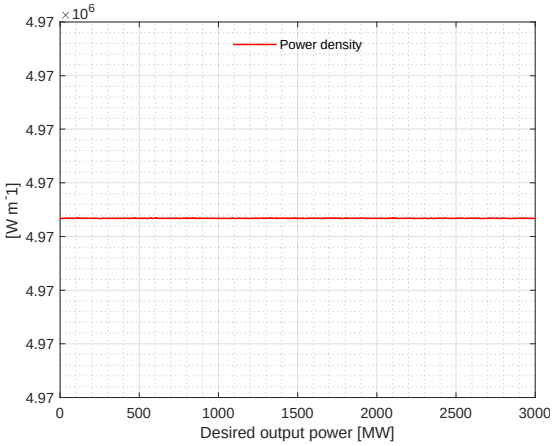
Appendix C: Iterations over Freidberg's model

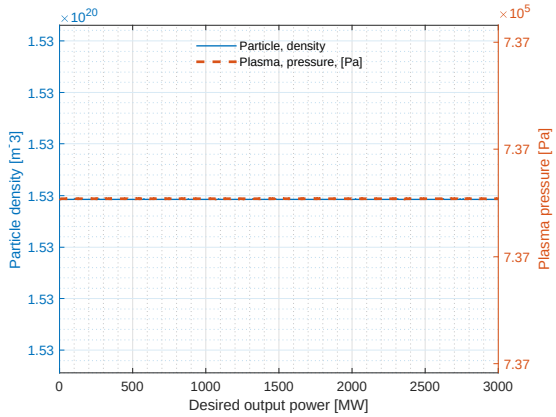


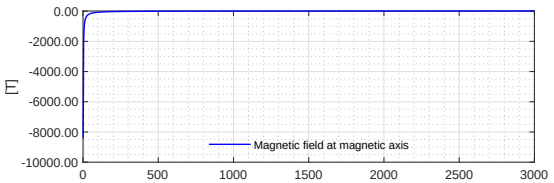


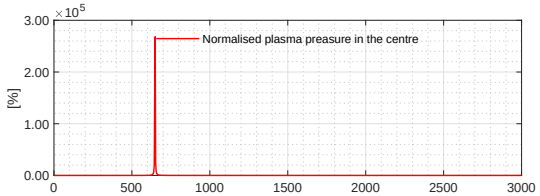


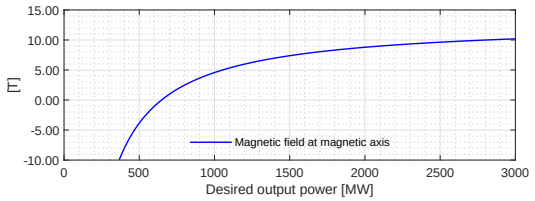


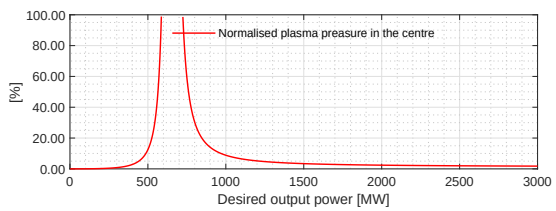


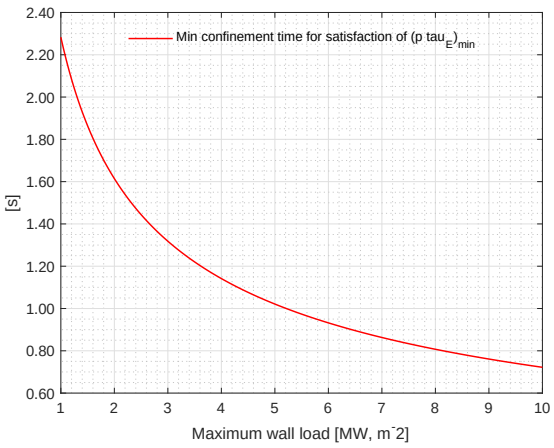


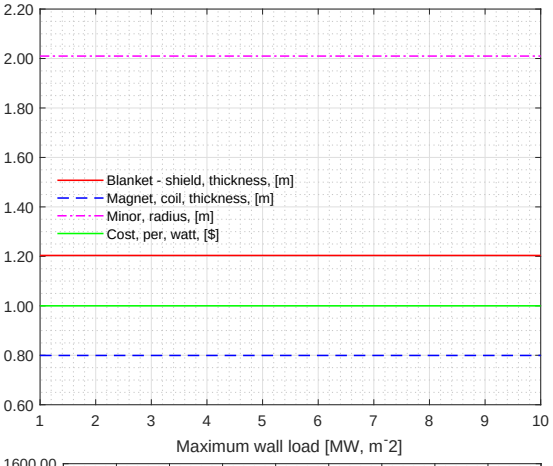


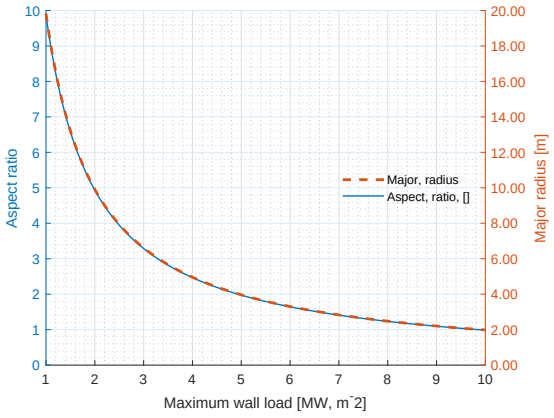


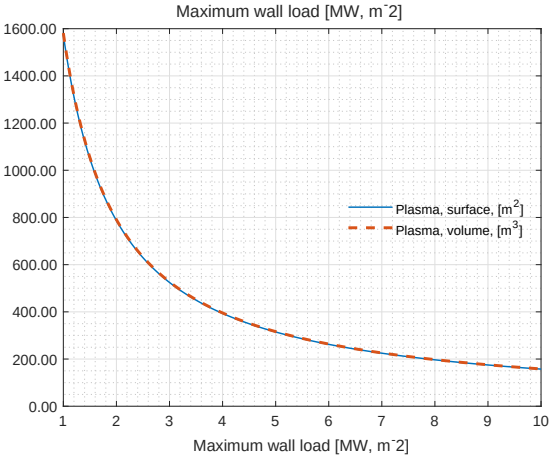


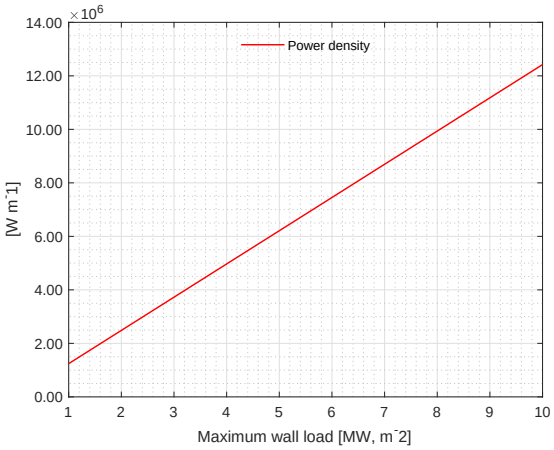


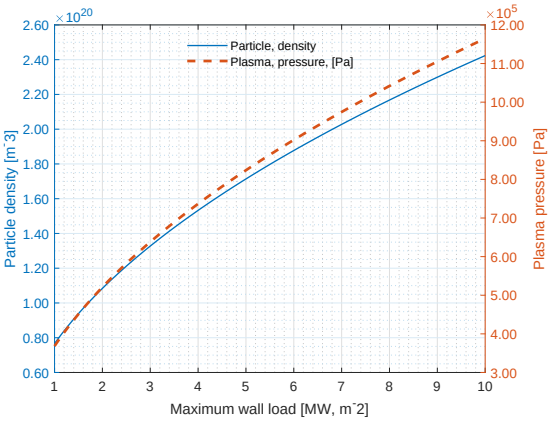

Min confinement time for satisfaction of (p tau_E)_min
[s]
Maximum wall load [MW, m^-2]

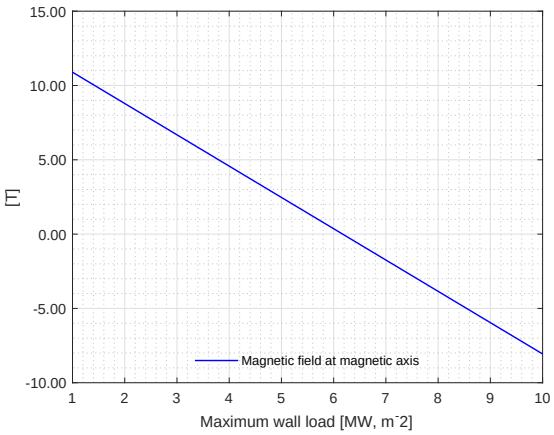

Blanket - shield, thickness, [m]
Magnet, coil, thickness, [m]
Minor, radius, [m]
Cost, per, watt, [$]
Maximum wall load [MW, m^-2]

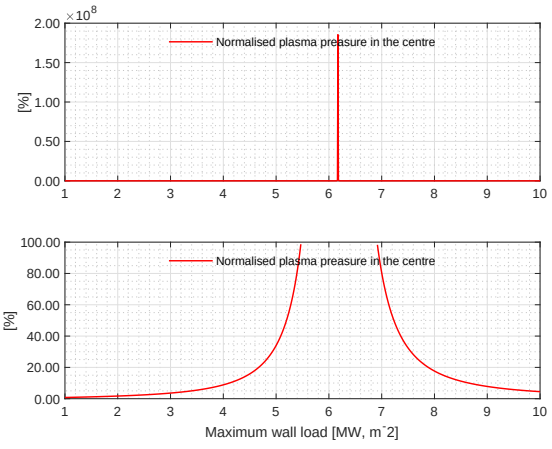

Major, radius
Aspect, ratio, []
Aspect ratio
Major radius [m]
Maximum wall load [MW, m^-2]

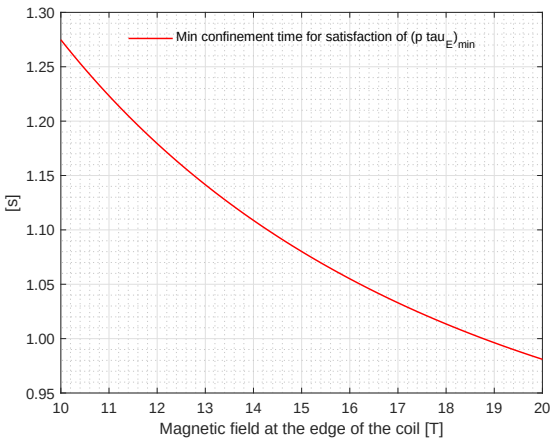

Plasma, surface, [m^2]
Plasma, volume, [m^3]
Maximum wall load [MW, m^-2]

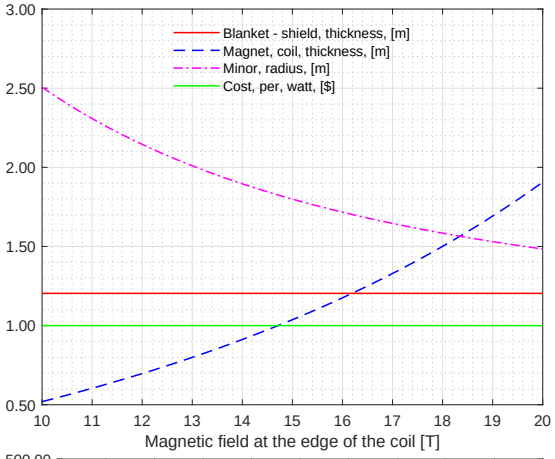

×10^6
Power density
[W m^-1]
Maximum wall load [MW, m^-2]

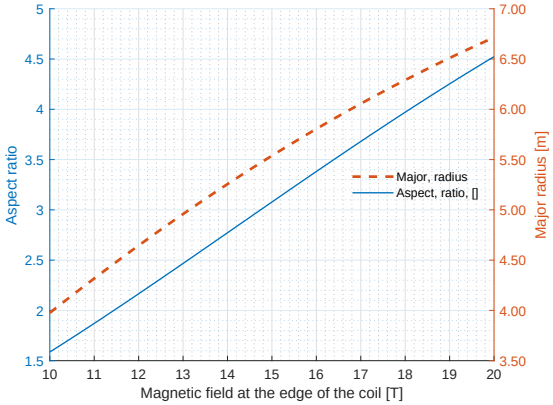

×10^20
×10^5
Particle, density
Plasma, pressure, [Pa]
Particle density [m^-3]
Plasma pressure [Pa]
Maximum wall load [MW, m^-2]

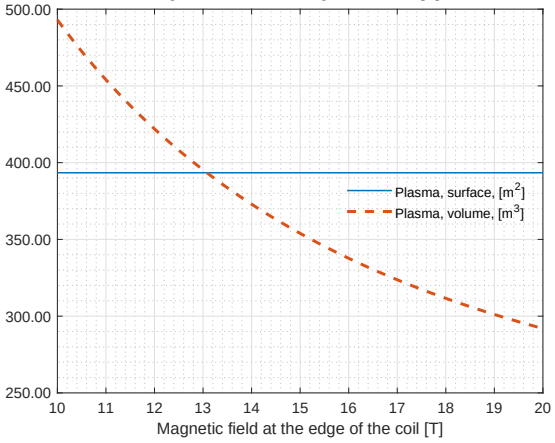

Magnetic field at magnetic axis
[T]
Maximum wall load [MW, m^-2]

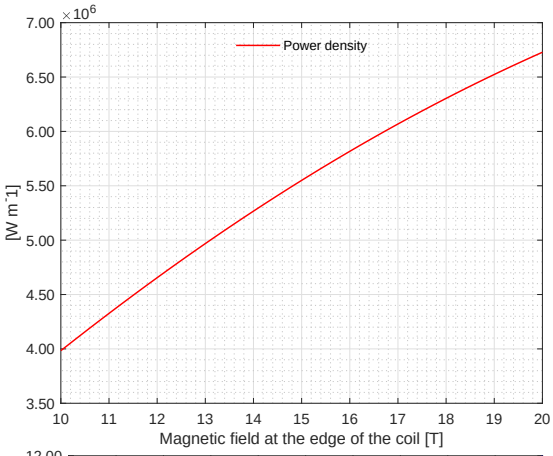

×10^8
Normalised plasma preasure in the centre
[%]
Normalised plasma preasure in the centre
[%]
Maximum wall load [MW, m^-2]

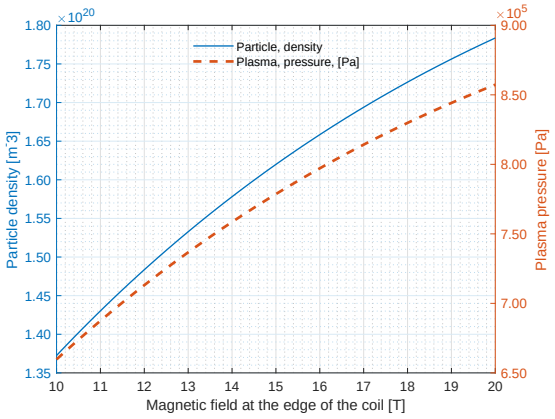

Min confinement time for satisfaction of (p tau$_E$)$_{min}$
[s]
Magnetic field at the edge of the coil [T]

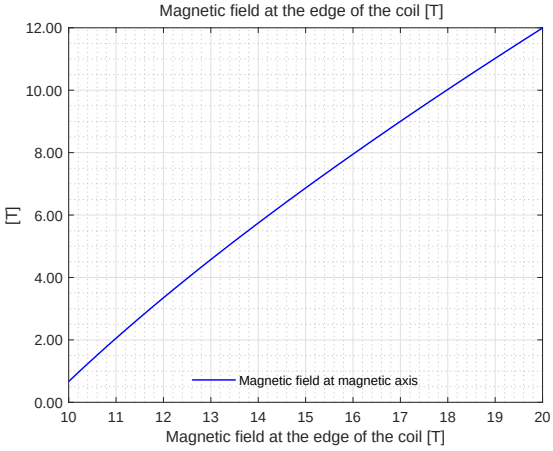

Blanket - shield, thickness, [m]
Magnet, coil, thickness, [m]
Minor, radius, [m]
Cost, per, watt, [$]
Magnetic field at the edge of the coil [T]

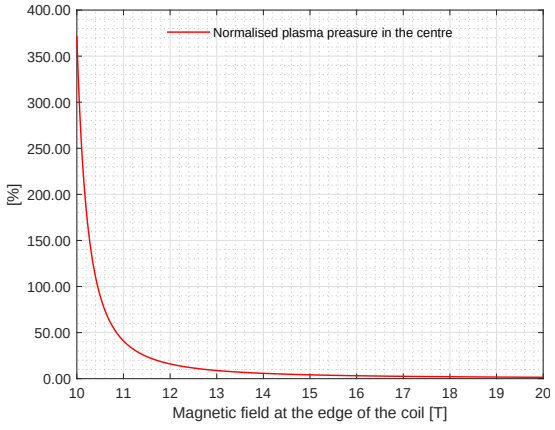

Major, radius
Aspect, ratio, []
Aspect ratio
Major radius [m]
Magnetic field at the edge of the coil [T]

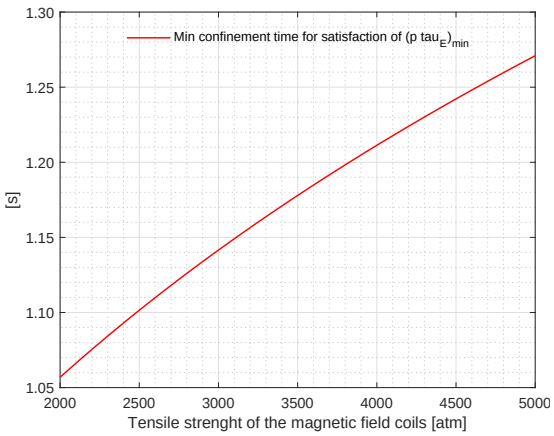

Plasma, surface, [m^2]
Plasma, volume, [m^3]
Magnetic field at the edge of the coil [T]

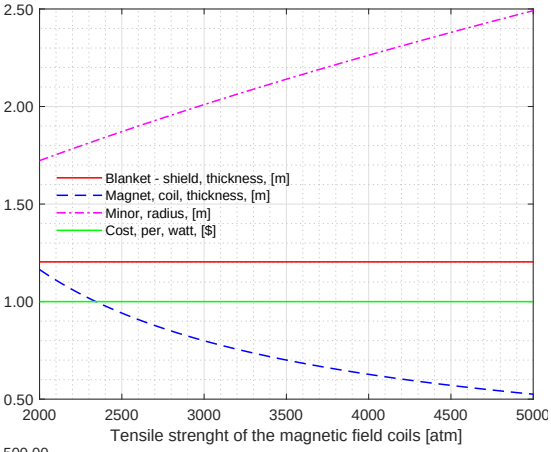

×10^6
Power density
[W m^{-1}]
Magnetic field at the edge of the coil [T]

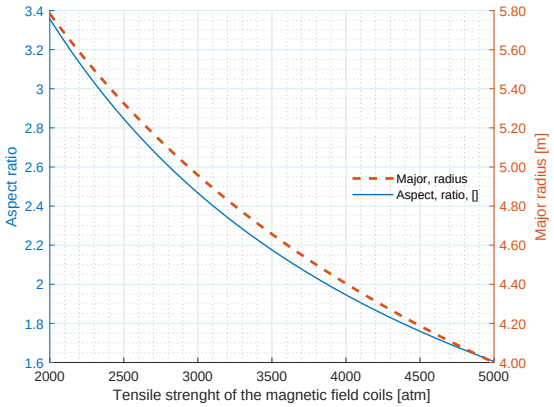

×10^{20}
×10^5
Particle, density
Plasma, pressure, [Pa]
Particle density [m^{-3}]
Plasma pressure [Pa]
Magnetic field at the edge of the coil [T]

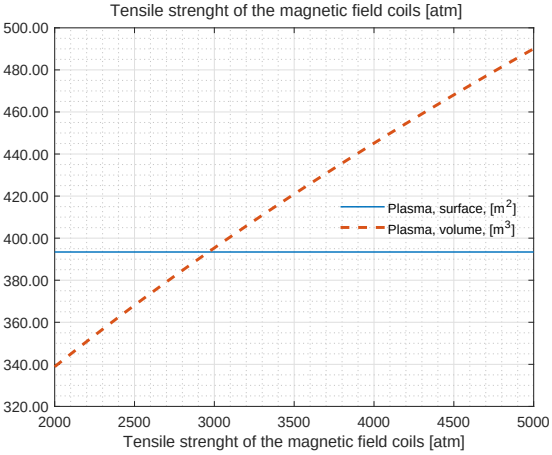

Magnetic field at magnetic axis
[T]
Magnetic field at the edge of the coil [T]

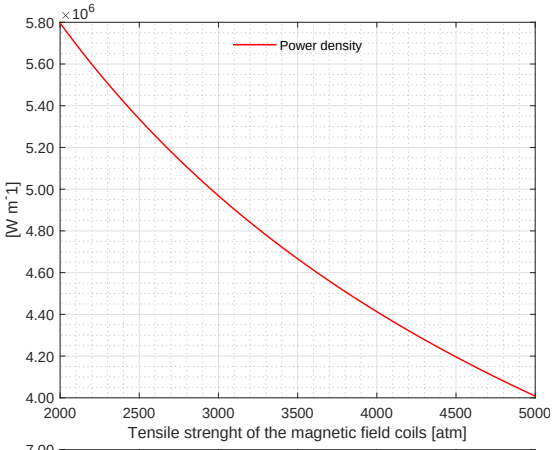

Normalised plasma preasure in the centre
[%]
Magnetic field at the edge of the coil [T]

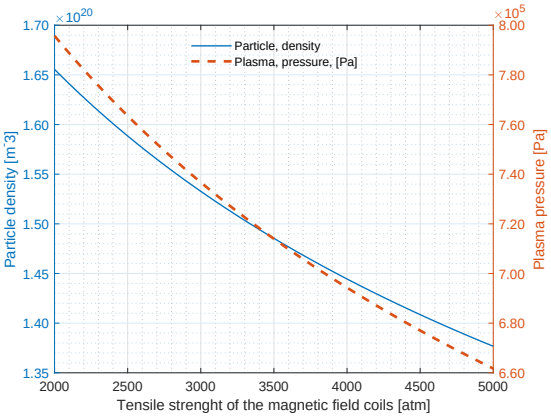

Min confinement time for satisfaction of (p tau_E)_min
[s]
Tensile strenght of the magnetic field coils [atm]

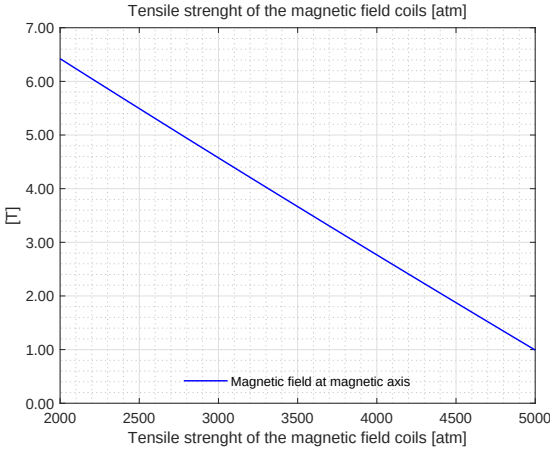

Blanket - shield, thickness, [m]
Magnet, coil, thickness, [m]
Minor, radius, [m]
Cost, per, watt, [$]
Tensile strenght of the magnetic field coils [atm]

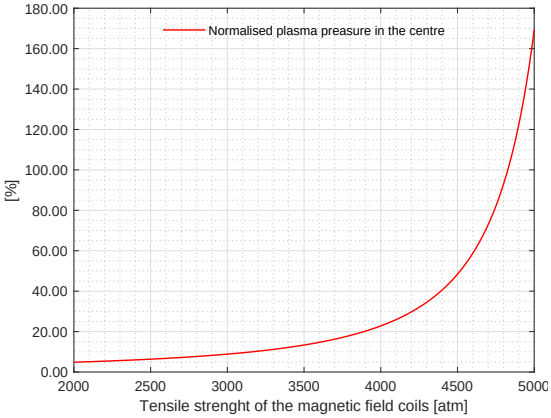

Major, radius
Aspect, ratio, []
Aspect ratio
Major radius [m]
Tensile strenght of the magnetic field coils [atm]


Plasma, surface, [m^2]
Plasma, volume, [m^3]
Tensile strenght of the magnetic field coils [atm]


Power density
[W m^-1]
Tensile strenght of the magnetic field coils [atm]


Particle, density
Plasma, pressure, [Pa]
Particle density [m^-3]
Plasma pressure [Pa]
Tensile strenght of the magnetic field coils [atm]


Magnetic field at magnetic axis
[T]
Tensile strenght of the magnetic field coils [atm]


Normalised plasma preasure in the centre
[%]
Tensile strenght of the magnetic field coils [atm]

Appendix D: interferometer.m

```
close all
clear all

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

%-------------------------Input Parameters-------------------------------%
r_p = 0.055; % Reactor port opening [m]
w_0 = 0.0275; % Initial Beam Waist [m]
r = 0.125; % Minor tokamak radius [m]
freq = 60; % Frequency of probe beam [GHz]

d_0 = 0.20; % Distance between source and first lens
d_r = 0.10; % Distance between first lens and reactor wall
f_0 = 0.25; % Focal length of first lens
f_1 = 0.20; % Focal length of second lens
%------------------------------------------------------------------------%

%----------Distances and Lowest Beam Waists Calculations-----------------%
c = 299792458; % The speec of light [m/s]
lambda = c / (freq * 10 9); % Calculates the wavelength [m]
d = f_0 + f_1; % Calculates the distance between lenses
w_1 = (lambda * f_0) / ...
    (pi * w_0);% Calculates the beam waist between lenses
d_1 = (((d_0 / f_0) - 1) / ((w_0 (2) * pi) / ...
    (f_0 * lambda) + ((d_0 / f_0) - 1) (2)) + 1) * ...
    f_0;% Distance after 1st lens to lowest beam waist
w_2 = (f_1 / f_0) * w_0; % Calculates w_2 after 2nd second lens
d_2 = d - d_1; % Distance from w_2 to second lens
d_3 = f_1 / f_0 *...
    (f_0 + f_1 -...
```

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

%------------------------Beam Waist Datapoints-----------------------------%
x_0 = linspace(0, d_0, 1000);% Plotpoints till first lens
y_0 = [];
for i = 1:1000
    w = w_0 * ...
        sqrt(1+((lambda * x_0(i)) / ...
        (pi * w_0 2)) 2);% Calculates the beam waist from source
    y_0 = horzcat(y_0, w);
end

x_1 = linspace(0, d_1, 1000); % Plotpoints from first lens to w_1
y_1 = [];
for i = 1:1000
    w = w_1 * ...
        sqrt(1+((lambda * x_1(i)) / ...
        (pi * w_1 2)) 2);% Beam waist between w_1 and the 1st lens
    y_1 = horzcat(y_1, w);
end
y_1 = fliplr(y_1); % Flips the y-plot points as the beam waist is declining
y = horzcat(y_0, y_1);

x_2 = linspace(0, d_2, 1000); % Plotpoints from w_1 to second lens
y_2 = [];
for i = 1:1000
    w = w_1 * ...
        sqrt(1+((lambda * x_2(i)) / ...
        (pi * w_1 2)) 2);% Beam waist between w_1 and the 2nd lens
    y_2 = horzcat(y_2, w);
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
```

```
p1 = plot(x, y); % Plots Beam Waist+
p2 = plot(x, Opy, 'Color', 'red'); % Plots Beam Waist-
%p3 = xline(d_0, '--');
%xline(d_0+d_1+d_2, '--');
axPos = get(gca, 'Position'); % Get normalised axis coordinates
xMinMax = xlim; % Get normalised axis coordinates
yMinMax = ylim; % Get normalised axis coordinates
zAnn = axPos(1) + ((0 - xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...
    * axPos(3);% Defines points relative to axis in normalised coordinates
d0Ann = axPos(1) + ((d_0 - xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...
    * axPos(3);% Defines points relative to axis in normalised coordinates
d1Ann = axPos(1) + ((d_0 + d_1 - xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...
    * axPos(3);% Defines points relative to axis in normalised coordinates
d2Ann = axPos(1) + ((d_0 + d_1 + d_2 - ...
    xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...
    * axPos(3);% Defines points relative to axis in normalised coordinates
d3Ann = axPos(1) + ((d_0 + d_1 + d_2 + d_3 - ...
    xMinMax(1)) / (xMinMax(2) - xMinMax(1))) ...
    * axPos(3);% Defines points relative to axis in normalised coordinates
yAnn = axPos(2) + ((0 - yMinMax(1)) / (yMinMax(2) - yMinMax(1))) ...
    * axPos(4);% Defines points relative to axis in normalised coordinates
annotation('doublearrow', ...
    [zAnn, d0Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_0
annotation('doublearrow', ...
    [d0Ann, d1Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_1
annotation('doublearrow', ...
    [d1Ann, d2Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_2
annotation('doublearrow', ...
    [d2Ann, d3Ann], [yAnn - 0.2, yAnn - 0.2]);% Annotates d_3
annotation('ellipse', ...
    [d0Ann - 0.01, yAnn - y_0(1000) * 1.5, 0.02, y_0(1000) * 3], ...
    'FaceColor', 'cyan');% Draws 1st lens
```

```
annotation('ellipse', ...
    [d2Ann - 0.01, yAnn - y_0(1000) * 1.5, 0.02, y_0(1000) * 3], ...
    'FaceColor', 'cyan');% Draws 2nd lens
text(d_0/4, -0.22, sprintf('d_0 = %0.2f', d_0)) % Distance annotation
text(d_0+(d_1 / 4), -0.22, ...
    sprintf('d_1 = %0.2f', d_1))% Distance annotation
text(d_0+d_1+(d_2 / 4), -0.22, ...
    sprintf('d_2 = %0.2f', d_2))% Distance annotation
text(d_0+d_1+d_2+(d_3 / 4), -0.22, ...
    sprintf('d_3 = %0.2f', d_3))% Distance annotation
text(d_0-0.06, 0.06, '1^{st} Lens') % Lens annotation
text(d_0-0.06, -0.06, sprintf('f_0 = %0.2f', f_0)) % Lens annotation
text(d_0+d_1+d_2-0.06, 0.06, '2^{nd} Lens') % Lens annotation
text(d_0+d_1+d_2-0.06, -0.06, ...
    sprintf('f_1 = %0.2f', f_1))% Lens annotation
w1x = (d_0 + d_1); % w_1 coordinate
w1x = repelem(w1x, 100); % w_1 x-coordinate array
w1y = linspace(0, w_1); % w_1 y-coordinate array
p4 = plot(w1x, w1y); % Plots w_1
yline(0, '-'); % Plots optical axis
xlabel('Distance from source [m]'); % x-axis label
ylabel('Beam Waist [m]'); % y-axis label
title('Beam Waist In A Gaussian Beam Interferometer'); % Figure title

topstart = 2 * asin(r_p/(2 * r)) / 2; % Calculates reactor plot coordinates
topend = (2 * pi - 2 * topstart) / 2; % Calculates reactor plot coordinates
bottomstart = topend + topstart * 2; % Calculates reactor plot coordinates
bottomend = 2 * pi - topstart; % Calculates reactor plot coordinates

top = linspace(topstart, ...
    topend, 100);% Creates 100 datapoints for the top half reactor cutout
x_t = d_0 + d_r + r + r * cos(top); % x-parameter of circle
```

```
y_t = r * sin(top); % y-parameter of circle
bottom = linspace(bottomstart, ...
    bottomend, 100);% Creates 100 datapoints for the bottom half reactor
x_b = d_0 + d_r + r + r * cos(bottom); % x-parameter of circle
y_b = r * sin(bottom); % y-parameter of circle
p5 = plot(x_t, y_t, 'color', ...
    [0.911, 0.4100, 0.1700]);% Plot top half reactor
plot(x_b, y_b, 'color', ...
    [0.911, 0.4100, 0.1700]);% Plot bottom half reactor
lgd = legend([p1, p2, p4, p5], ...
    {'Beam Waist +', 'Beam Waist -', 'w_1', 'Reactor'});% Figure legend
legend('boxoff')
legend('Location', 'northwest')
lgd.NumColumns = 2;
hold off
%--------------------------------------------------------------------------%

%------------------Saving figure as Encapsulated Postscript---------------%
epsfilename = 'Interferometer.eps'; % Savename for the figure
foldername = sprintf('../MatlabFigures/Interferometer'); % Folder path
fullfilename = fullfile(foldername, epsfilename); % Filename path
saveas(1, fullfilename, 'epsc') % Save the figure as eps
%--------------------------------------------------------------------------%

%----------------------------Print Outputs---------------------------------%
w_1
w_2
d_1
d_2
d_3
%--------------------------------------------------------------------------%
```

Appendix E: PhaseShiftDensity.m

```
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
```

```

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