Royal Military Academy
Research Group Plasma Physics

Internal report: ICRH automatic matching system on TOMAS

Arthur Adriaens

Contents

1

Overview

1.1 Introduction

This work aims to be a succesful implementation of an automatic matching algorithm for the Ion Cyclotron Resonance Heating (ICRH) antennna, a circuit drawing of this matching system is shown in figure 1.1. The antennae are facing a plasma with ever changing conditions: the density, temperature and even species (He/H/Ar) may vary; As such the capacitance of the various capacitors have to be varied every time the conditions change as to maximize power transfer.

At the time of writing this is carried out manually by first lowering the current going to the antenna by adjusting C_a, then varying capacitances of C_p and C_s, doing a frequency sweep and looking where the power transferred is maximal (i.e the measured reflected power is minimal), if the frequency at which this happens coincides with the required frequency then we stop adjusting C_s and C_p and heighten the current flowing to the antenna using C_a.

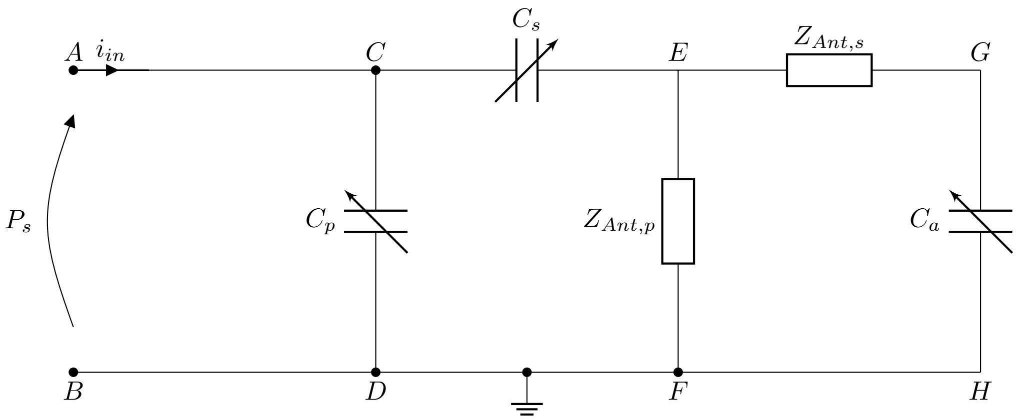


Figure 1.1: The Power input is marked P_s, the antenna can be described as having a parallel impedence $Z_{Ant,p}$ and a series impedence $Z_{Ant,s}$. The power input is assumed to have an entry impedence of $Z_0 = 50\Omega$ and to match the circuit we can play around with the variable parallel matching C_p, the series matching C_s and the pre-matching C_a capacitors

1.2 Previous work

Work has been done on this previously by F. Fasseur [1], which lays out a good exposition on the system, how changing the various capacitors affects the system and which capacitance values of C_a result in a matchable system.

K. Elinck continued on this work by implementing a neural network and training it on simulated data [2], we question wether the simulated training corresponds well with the experiment and if it's even needed to make the algorithm so complex (as e.g TEXTOR's circuit worked fine [3] which was just a simple linear algorithm) and as such we won't go into detail on his work.

1.3 system

To understand how to, in practice, implement an algorithm, the interconnectedness of the ICRH system needs to be discussed: A computer which we'll henceforth call pc1 is connected to an Arduino which can modify the capacitance of the capacitors with steppermotors. pc1 is also connected to the ICRH amplifier making it possible to select the power.

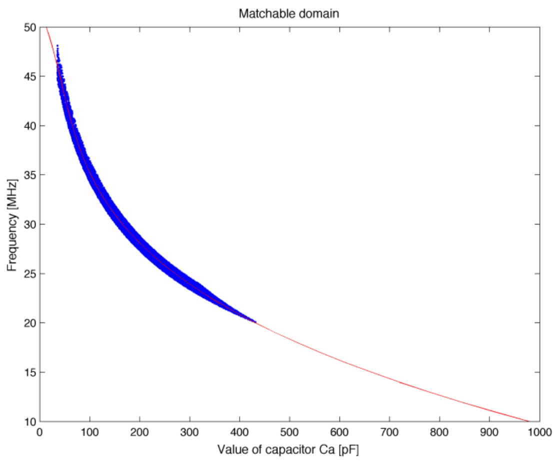
The DAQ, however, is a second computer: pc2. We can thus set the various parameters on pc1 but only see what they imply on pc2, making it necessary to transfer data between the two, which makes the automatic matching a little more complex.

2

TOMAS matching algorithm

2.1 F. Fasseur's work

F. Fasseur took the idea of the matching algorithm developed for TEXTOR [4] and applied it on TOMAS, here I'll quickly discuss the main pointers: Wirst he did an analysis to the matchable domain from which he found a curve telling us, given a certain frequency f in the domain [20MHz:48MHz] which C_a value should make the system matchable:



Of which I analysed the inverse to get the relation

$$C_a = -3.3094 \times 10^{-6} f^5 + 5.17212 \times 10^{-4} f^4 - 5.10998 \times 10^{-2} f^3 + 3.5211 f^2 - 132.074 f + 2000.87 \tag{2.1}$$

In the region 20MHz to 45MHz[1]. Note that the capacitance seems to be related to the amount of steps as (from the capacitor spec page):

$$Steps = 0.04781 C_a - 3.68067 \tag{2.2}$$

[1] it's worth mentioning that lower frequencies should also be matchable if the power is capped at 3000W but we won't concern ourselves with the added complexity

He also designed the following error variables:

$$\epsilon_g = sin(2\beta l_2)(|V_1| - |V_+|) - sin(2\beta l_1)(|V_2| - |V_+|) \tag{2.3}$$

$$\epsilon_b = cos(2\beta l_2)(|V_1| - |V_+|) - cos(2\beta l_1)(|V_2| - |V_+|) \tag{2.4}$$

Whom are similar to the ones used on TEXTOR. Here V_1 and V_2 are the voltages at distances l_1 and l_2 in the power line and V_+ the measured forward voltage (using a differential coupler). Here β is the so called "phase constant", as we're dealing with a coax cable on TOMAS[2], our cutoff frequency can be said to be 0 (as the wavelength will be order 30m whilst the perimeter is some 30cm) so we simply have

$$\boxed{\beta = \frac{\omega}{c} = \frac{2\pi f}{c}} \tag{2.5}$$

These error values are each proportional to a different capacitor:

$$\epsilon_g \quad \propto C_s \tag{2.6}$$
$$\epsilon_b \quad \propto C_p \tag{2.7}$$

The algorithm was simulated for ± 3 turns from the matching point.

2.2 Improvement

Fasseur's work assumes three voltmeters: V_1, V_2 and V_+. These only measure amplitude in his setup. in his work he linearizes the equations, this was needed in times of TEXTOR (when computations needed to be done using OpAmps) but a quick calculation shows that this isn't needed anymore when dealing with the TOMAS setup which has 4 voltmeters. The voltage along the line has the form (from transmission line theory):

$$V_i = V_0^+(e^{i\beta l_i} + \Gamma_L e^{i\beta l_i}) \tag{2.8}$$

with $\Gamma_L = u + iv$ the reflection coefficient of the load (the antenna). from this we can derive that

$$\frac{|V_i|^2}{|V^+|^2} = 1 + u^2 + v^2 + 2u\cos(2\beta l_i) + 2v\sin(2\beta l_i) \tag{2.9}$$

Now, using the absolute voltage measurements we can construct a system of linear equations:

$$\begin{aligned}
u^2 + v^2 + 2u\cos(2\beta l_1) + 2v\sin(2\beta l_1) &= \tfrac{|V_1|^2}{|V^+|^2} - 1 \\
u^2 + v^2 + 2u\cos(2\beta l_2) + 2v\sin(2\beta l_2) &= \tfrac{|V_2|^2}{|V^+|^2} - 1 \\
u^2 + v^2 + 2u\cos(2\beta l_3) + 2v\sin(2\beta l_3) &= \tfrac{|V_3|^2}{|V^+|^2} - 1
\end{aligned}$$

Writing this into a matrix:

$$\begin{bmatrix}
1 & 1 & 2\cos(2\beta l_1) & 2\sin(2\beta l_1) & \frac{|V_1|^2}{|V^+|^2} - 1 \\
1 & 1 & 2\cos(2\beta l_2) & 2\sin(2\beta l_2) & \frac{|V_2|^2}{|V^+|^2} - 1 \\
1 & 1 & 2\cos(2\beta l_3) & 2\sin(2\beta l_3) & \frac{|V_3|^2}{|V^+|^2} - 1
\end{bmatrix}$$

[2] arguably not a very bendy one but not a waveguide

Defining

$$\boxed{C_i := \cos(2\beta l_i)} \quad \boxed{S_i := \sin(2\beta l_i)} \tag{2.10}$$

And

$$\boxed{\frac{|V_i|^2}{|V^+|^2} - 1 =: \mathcal{V}_i} \tag{2.11}$$

to help write everything clearer we now have

$$\begin{bmatrix} 1 & 1 & 2C_1 & 2S_1 & \mathcal{V}_1 \\ 1 & 1 & 2C_2 & 2S_2 & \mathcal{V}_2 \\ 1 & 1 & 2C_3 & 2S_3 & \mathcal{V}_3 \end{bmatrix}$$

Which gives (substracting the $u^2 + v^2$ terms)

$$\begin{bmatrix} 2(C_1 - C_2) & 2(S_1 - S_2) & (\mathcal{V}_1 - \mathcal{V}_2) \\ 2(C_3 - C_2) & 2(S_3 - S_2) & (\mathcal{V}_3 - \mathcal{V}_2) \end{bmatrix}$$

solving for u gives:

$$\boxed{u = \frac{1}{2} \frac{(\mathcal{V}_1 - \mathcal{V}_2) - (\mathcal{V}_3 - \mathcal{V}_2)\left(\frac{S_1 - S_2}{S_3 - S_2}\right)}{(C_1 - C_2) - (C_3 - C_2)\left(\frac{S_1 - S_2}{S_3 - S_2}\right)}} \tag{2.12}$$

and for v:

$$\boxed{v = \frac{1}{2} \frac{(\mathcal{V}_1 - \mathcal{V}_2) - (\mathcal{V}_3 - \mathcal{V}_2)\left(\frac{C_1 - C_2}{C_3 - C_2}\right)}{(S_1 - S_2) - (S_3 - S_2)\left(\frac{C_1 - C_2}{C_3 - C_2}\right)}} \tag{2.13}$$

All values shown here are measurable, making it possible to find both u and v, so from now on we'll use the variables u and v as if we have measured them.

Matching occurs when $\Gamma_L = 0$ and, as the admittance is given by

$$y = \frac{1 - \Gamma}{1 + \Gamma} = \frac{1 - u^2 - v^2}{(1 + u)^2 + v^2} - i\frac{2v}{(1 + u)^2 + v^2} := g + ib \tag{2.14}$$

Matching occurs when $\Re\{y\} := g = 1$ and $\Im\{y\} := b = 0$. So, to probe how close we are to matching, as a first error value we can take b:

$$\boxed{\epsilon_b = -\frac{2v}{(1 + u)^2 + v^2}} \tag{2.15}$$

and as a second error we'll take $1 - g$:

$$\boxed{\epsilon_g = 1 - \frac{1 - u^2 - v^2}{(1 + u)^2 + v^2}} \tag{2.16}$$

So finally, as we know both u and v, to match the system the change in capacitance will have to vary as:

$$\Delta C_s \propto \epsilon_g \tag{2.17}$$

$$\Delta C_p \propto \epsilon_b \tag{2.18}$$

As both variable capacitors are linear, we can say that the amount of steps required can be gives by

$$\Delta C_s := S\epsilon_g \tag{2.19}$$

$$\Delta C_p := P\epsilon_b \tag{2.20}$$

With S and P constants we may determine from running a simulation or by doing experiments.

2.3 Practical implementation

Applying the algorithm would thus look as follows:

1. use equations 2.1 and 2.2 to move the prematching capacitor according to the selected frequency.
2. lower the power to 1000W as to not damage any components whilst tuning and start the ICRH antenna
3. Record the various needed measurements over some sufficiently big time Δt (to reduce noise) and send them over from matlab to the python gui
4. Compute the various errors, stop the algorithm if either
   - $\epsilon_b < \Delta_b$ and $\epsilon_g < \Delta_g$
   - $\Gamma_L <$ some percentage
5. As the response of both the parallel and series capacitor are linear, compute the steps using Steps $C_s \propto \epsilon_g$ and Steps $C_p \propto \epsilon_b$
6. adjust the capacitors
7. back to 3.

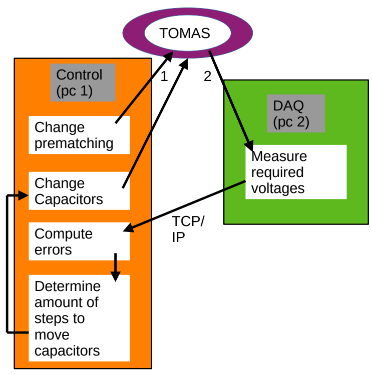


Figure 2.1: Overview of the system, pc2 will communicate it's data with pc1 through TCP/IP whilst pc1 is triggered using a biasing voltage from pc2

After sufficient loops, the errors will fluctuate less than the predefined values Δ_g and Δ_b or the reflection coeffeciënt is sufficiently small and the matching algorithm may stop.

2.4 Python code

```
def MatchICRH(self):
  ####################################
  #            constants             #
  ####################################

  #length from amplifier to voltmeters 1-3
  l1 = ?
  l2 = ?
  l3 = ?
  #speed of light in m/s
  c = 299792458
  # not matched yet
  matched = False

  if self.matchICRH_entr == "":
      print("frequency must be specified")
      return

  ####################################
  # move C_a to the correct position #
  ####################################
  f = self.matchICRH_entr.get()
  C_a = -3.3094*e-6*f**5 5.17212e-4*f**4 - 5.10998e-2*f**3 + 3.5211*f**2 - 132.074*f + 2000.87 #calculate required capacitance
  Stepfactor = 10 #amount of steps in a full revolution
  StepToMoveTo = round(Stepfactor*(0.04781*C_a - 3.68067)) -  self.minPos[0]#The 'A' limits are from minPos to maxPos whilst the original c_a to steps assumes starting at 0
  cmd = ""
  cmd += "A " + moveAto + " "
  if cmd == "":
      print("Some error occured whilst trying to move the capacitor")
      return
  # Communicate the desired position to Arduino
  print(cmd)
  self.arduino.write(cmd.encode())
  time.sleep(2) #buffer,
  # Retrieve communication from Arduino
  if "Error" in newPos.decode():
      print(newPos.decode())
      # After the error, Arduino will communicate the new positions
  print("Theposition of C_a is:")
  print(newPos.decode())
  self.f.write(newPos.decode().strip()+"\n")
  # Update the information on the GUI
  posStrs = newPos.decode().split(" ")
  for i in range(0, len(posStrs)):
      if posStrs[i] == "A": #check to make sure, but probably not needed
    self.posA_lbl.config(text="A: " + posStrs[i + 1].strip())

  #######################################
  #           Connect to DAQ            #
  #######################################
  # note that we are the server and labview is the client
  server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #start basic server
  DAQip = '192.168.?.?' #adress of DAQ
  server.bind((DAQip,8089)) #bind DAQ adress to port 8089
  server.listen(1) #listen for 1 connection (only the DAQ)

  #######################################
```

```
#           Start ICRH on 10dBm         #
#######################################
self.dev.IC_enable()
self.dev.setICFixedFrequency(f*10**6) #in Hz
Pin = 5 #should be low eneough
print('setting IC power to ' + str(Pin))
# Set the IC voltage amplitude to Pin (in dBm) and the voltage ofsett to 0 V, limited at 10 dBm.
self.dev.setICPower(float(Pin))  # dBm
# Update information on GUI
self.ICpower_lbl.config(text="The IC power is " + str(self.dev.GetICPower()))

#######################################
#            Match the system           #
#######################################
message = ""
while str(message) == "":
    conn, addr = server.accept() #accept connections
    doDAQ() #trigger DAQ
    message = conn.recv() #receive the averaged voltages as one csv line: V1,V2,V3,V+,V-
message_listform = message.split(",")
V = [float(i) for i in message_listform]
print("initial measured voltages:")
print(V)
ReflCoeff = V[4]/V[3]
#Stop matching if reflection is < 10%
if  ReflCoeff < 0.1:
    matched = True

while matched == False:
    #compute the \mathcal{V}_i's
    V1 = (V[0]**2)/(V[-1]**2) - 1
    V2 = (V[1]**2)/(V[-1]**2) - 1
    V3 = (V[2]**2)/(V[-1]**2) - 1
    #compute beta
    beta = 2*np.pi*f/c
    #compute the cosines and sines
    C_1 = np.cos(2*beta*l1)
    S_1 = np.sin(2*beta*l1)
    C_2 = np.cos(2*beta*l2)
    S_2 = np.sin(2*beta*l2)
    C_3 = np.cos(2*beta*l3)
    S_3 = np.sin(2*beta*l3)
    #even though we'll scale this later, we'll keep the original definition of u and v
    u = (1/2)*(((V1 - V2) - (V3 - V2)*(S_1 - S_2)/(S_3 - S_2))/((C1 - C2) - (C3 - C2)*(S_1 - S_2)/(S_3 - S_2)))
    v = (1/2)*(((V1 - V2) - (V3 - V2)*(C_1 - C_2)/(C_3 - C_2))/((S1 - S2) - (S3 - S2)*(C_1 - C_2)/(C_3 - C_2)))
    error_b = -2*v/((1+u)**2 + v**2)
    error_g = 1-(1-u**2-v**2)/((1+u)**2 + v**2)
    #we might scale the following two variables whilst testing:
    S = 1
    P = 1
    SeriesSteps = S*error_g
    ParallelSteps = S*error_g
    movePto = myPos[1] + ParallelSteps
    moveSto = myPos[2] + SeriesSteps
    cmd = ""
    cmd += "P " + movePto + " "
    cmd += "S " + moveSto
```

```
      print("Instructing Arduino:")
      print(cmd)
      self.arduino.write(cmd.encode())
      time.sleep(2)
      message = ""
      while str(message) == "":
    conn, addr = server.accept() #accept connections
    doDAQ() #trigger DAQ
    message = conn.recv() #receive the averaged voltages as V1,V2,V3, V+ and V-
      message_listform = message.split(",")
      V = [float(i) for i in message_listform]
      print("measured voltages:")
      print(V)
      ReflCoeff = V[4]/V[3]
      #Stop matching if reflection is < 10%
      if  ReflCoeff < 0.1:
    matched = True

  print("MATCHED! The Reflection co fficient is now {}".format(ReflCoeff))
  server.close()
```

Bibliography

[1] F. Fasseur, “Study of an rf-system for plasma production on tomas.”

[2] K. Elinck, “Implementation of an icrf matching algorithm on tomas,” 2019.

[3] F. Durodié and M. Vervier, “First results of the automatic matching device for textor’s icrh system,” in *Fusion Technology 1992*, C. FERRO, M. GASPAROTTO, and H. KNOEPFEL, Eds. Oxford: North-Holland, 1993, pp. 477–480. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444899958500886

[4] ——, “Design of an automatic matching device for textor’s icrh system,” *Elsevier*, pp. 1186–1190, 1991.