Background

Code to calculate the sheath admittance

ytot(w,wci,bx,xi,j)

where normalized (dimensionless) variables are used throughout

w = rf frequency
wci = ion cyclotron frequency
bx = normal component of b, (bx = 1 for perp. sheath)
xi = e Vrf / Te where Vrf = 0 to peak amplitude (not peak-to-peak)
j = Jdc/(ni0 e cs) = normalized DC current density flowing through the sheath

Individual contributions are

ye(bx,xi) = electron admittance
yd(w,wci,bx,xi) = displacement admittance
yi(w,wci,bx,xi) = ion admittance

The sheath impedance is z = 1/ytot. To convert dimensionless impedance z to CGS or SI units see

J. R. Myra and D. A. D'Ippolito, Physics of Plasmas 22, 062507 (2015), Appendix A
http://dx.doi.org/10.1063/1.4922848

The code is based upon work described in

Physics-based parametrization of the surface impedance for radio frequency sheaths
J. R. Myra
Physics of Plasmas 24, 072507 (2017)
http://dx.doi.org/10.1063/1.4990373

This version (v3.2) implements an upgrate to include the JDC as described in

Effect of net direct current on the properties of radio frequency sheaths: simulation and cross-code comparison
J. R. Myra, M.T. Elias, D. Curreli, and T. G. Jenkins
submitted to Nucl. Fusion
http://www.lodestar.com/LRCreports/Myra_DC_current_carrying_RF_sheaths_LRC-20-186.pdf

version history

v3.2

replaces 1-j with 1-j/upar0 where j is the DC current. This produces better quality fits than v3.1. For ye, change the leading coefficient from 1.161585 to 1.05704

v3.1

replaces 1-j with upar0-j where j is the DC current

v3

v3 adds the optional independent variable jdc describing the net dc current through the sheath, using the model discussed in “RF sheaths that carry net DC current” This model is justified for the low frequency limit, however here it is applied in general. Tests will reveal how well it does and what modifications are needed.

v3 also generalizes to arbitrary μ, where μ is a global parameter. The only changes occur in ff[ ] and niw[ ]. Although ff[ ] should depend on μ only through the additive term Log[μ], unfortunately, the fits were done for μ = 24.17 with the Log[μ] in the numerator of a more complication fraction. Nevertheless, the way the generalization is done here should be correct.

v2

v2 protects constant parameters inside Module[ ] functions to prevent trouble when the same names are used by calling codes.

v1

functions extracted from

database analysis.nb

important note:

The order of the arguments in these functions is different that in database analysis.nb. It is consistent with the python version of these functions in sheath_param.py

ξ is the 0 - peak rf voltage throughout

---

fits

all fits done for Deuterium with μ = 24.17 but here generalized analytically to arbitrary μ

In[2]:=
```
μ = 24.17 (* default value, can override in main code *)
```
Out[2]=
```
24.17
```

In[3]:=
```
upar0 = 1.1;
```

phi0avg

In[4]:=
```
ff[ξ_, j_] := Module[{a1, a2, a3, b1, b2, phipars},
  phipars = {a1 → 3.7028540109659485`, a2 → 3.8199078552022536`,
    b1 → 1.133522930502336`, b2 → 1.2417092549137196`, a3 → 2. b2 / π};
  (Log[μ] + ξ a1 + ξ^2 a2 + ξ^3 a3) / (1 + ξ b1 + ξ^2 b2) - Log[1 - j / upar0] + Log[μ / 24.17] //. phipars
 ]
```

In[5]:=
```
ff[ξ_] := ff[ξ, 0]
```

In[6]:=
```
gg[ω_] := Module[{c0, c1, ggpars},
  ggpars = {c0 → 0.966463, c1 → 0.141639};
  c0 + c1 * Tanh[ω] /. ggpars
 ]
```

In[7]:=
```
phi0avg[ω_, ξ_, j_] := ff[gg[ω] ξ, j]
```

In[8]:=
```
phi0avg[ω_, ξ_] := phi0avg[ω, ξ, 0]
```

ni1avg

niw = ion density at the wall for a static sheath
vv is the total dc potential drop

In[9]:=
```
niw[Ω_, bx_, vv_] := Module[{arg, ggg, Φp, ΩΦ, d0, d1, d2, d3, d4, v1, fff, nipars},
  nipars = {d0 → 0.7944430930529499`, d1 → 0.803531266389172`,
    d2 → 0.18237897510951012`, d3 → 0.9957212047604492`, v1 → 1.4555923231100891`};
  arg = Sqrt[(μ^2 bx^2 + 1) / (μ^2 + 1)];
  fff = (-Log[arg]) / (1 + d3 Ω^2);
  Φp = vv - fff;
  If[Φp < 0, Φp = 0; Print["Φp<0"]];
  ΩΦ = Ω Φp^(1/4);
  d4 = d2^2 / (μ^2 d0^2 - d2^2);
  d0 / (d2 + Φp^(1/2)) ((bx^2 + d4 + d1^2 ΩΦ^(2 v1)) / (1 + d4 + d1^2 ΩΦ^(2 v1)))^(1/2) /. nipars
 ]
```

niwω is the ion density at the wall for an rf sheath
ξ is the 0-peak rf voltage

In[10]:=
```
niwω[ω_, Ω_, bx_, ξ_, j_] := Module[{phiavg, philowω, phimod, k0, k1, niωpars},
  niωpars = {k0 → 3.7616962640756197`, k1 → 0.2220204461728174`};
  phiavg = phi0avg[ω, ξ, j];
  philowω = k0 + k1 (ξ - k0) - Log[1 - j / upar0] /. niωpars;
  phimod = philowω + (phiavg - philowω) Tanh[ω];
  Re[niw[Ω, bx, phimod]]
 ]
```

In[11]:=
```
niwω[ω_, Ω_, bx_, ξ_] := niwω[ω, Ω, bx, ξ, 0]
```

yd

In[12]:=
```
yd[ω_, Ω_, bx_, ξ_, j_] := Module[{phi0a, niwωa, Δ, s0, ydpars},
  ydpars = {s0 → 1.1241547327789232`};
  phi0a = phi0avg[ω, ξ, j];
  niwωa = niwω[ω, Ω, bx, ξ, j];
  Δ = Sqrt[phi0a / niwωa];
  (-I s0 ω) / Δ /. ydpars
 ]
```

In[13]:=
```
yd[ω_, Ω_, bx_, ξ_] := yd[ω, Ω, bx, ξ, 0]
```

ye

In[14]:=
```
he[ξ_] := Module[{h1, h2, g1, g2, g3, hepars},
  hepars = {h1 → 0.607405123251634`, h2 → 0.3254965671158986`,
    g1 → 0.6243920388599393`, g2 → 0.5005946718280853`, g3 → (π / 4.) h2};
  (1 + ξ h1 + ξ^2 h2) / (1 + ξ g1 + ξ^2 g2 + ξ^3 g3) //. hepars
 ]
```

yepars = {h0 -> 1.161585};

ye[bx_, ξ_] := h0 Abs[bx] he[ξ] /. yepars

In[15]:=
```
ye[bx_, ξ_, j_] := 1.05704235 Abs[bx] he[ξ] (1 - j / upar0)
```

In[16]:=
```
ye[bx_, ξ_] := ye[bx, ξ, 0]
```

In[17]:=
```
ye[0.4, 3.6]
```

Out[17]=
```
0.144533
```

yi

yipars = {p0 -> 1.0555369617763768`, p1 -> 0.7976591020008023`, p2 -> 1.47404874815277`, p3 -> 0.8096145628336325`};

In[18]:=
```
yi[ω_, Ω_, bx_, ξ_, j_] := Module[
  {phi0a, niwωa, ωcup, γcup, ϵ, gsmall, yi0, parp0, parp1, parp2, parp3, yipars},
  yipars = {parp0 → 1.0555369617763768`, parp1 → 0.7976591020008023`,
    parp2 → 1.47404874815277`, parp3 → 0.8096145628336325`};
  phi0a = phi0avg[ω, ξ, j];
  niwωa = niwω[ω, Ω, bx, ξ, j];
  ωcup = parp3 ω / Sqrt[niwωa];
  γcup = Abs[bx] / (niwωa Sqrt[phi0a]);
  ϵ = 0.0001;
  (* gg=Which[bx==1,1,ω==Ω,N[10^99],True,(ω^2-bx^2Ω^2)/(ω^2-Ω^2)]; *)
  gsmall = (ω^2 - bx^2 Ω^2 + I ϵ) / (ω^2 - Ω^2 + I ϵ);
  yi0 = niwωa / Sqrt[phi0a];
  parp0 yi0 I ωcup / (ωcup^2 / gsmall - parp1 + I parp2 γcup ωcup) /. yipars
]
```

In[19]:=
```
yi[ω_, Ω_, bx_, ξ_] := yi[ω, Ω, bx, ξ, 0]
```

ytot and ztot

In[20]:=
```
ytot[ω_, Ω_, bx_, ξ_, j_] := yi[ω, Ω, bx, ξ, j] + ye[bx, ξ, j] + yd[ω, Ω, bx, ξ, j]
ytot[ω_, Ω_, bx_, ξ_] := ytot[ω, Ω, bx, ξ, 0]
```

In[22]:=
```
ztot[ω_, Ω_, bx_, ξ_, j_] := 1 / ytot[ω, Ω, bx, ξ, j]
ztot[ω_, Ω_, bx_, ξ_] := ztot[ω, Ω, bx, ξ, 0]
```

---

expected test values

for j = 0 and μ = 24.17

In[24]:=
```
ff[11.5]
```

Out[24]= 9.83552

9.83553

In[25]:=
```
phi0avg[0.4, 6.]
```

Out[25]= 6.43176

6.43176

In[26]:=
```
niw[.2, .3, 13]
```

Out[26]= 0.0764645

0.0764645

In[27]:= **niwω[0.2, 0.3, 0.4, 13]**

Out[27]= 0.140776

0.140776

In[28]:= **yd[0.2, 0.3, 0.4, 13]**

Out[28]= 0. − 0.0257383 i

0. - 0.0257383 I

In[29]:= **he[10.5]**

Out[29]= 0.120617

0.120617

In[30]:= **ye[0.4, 3.6]**

Out[30]= 0.144533

0.144533

In[31]:= **yi[0.2, 0.3, 0.4, 13]**

Out[31]= 0.00654387 − 0.0137273 i

0.00654387 - 0.0137273 I

In[32]:= **ytot[0.2, 0.3, 0.4, 13]**

Out[32]= 0.0477729 − 0.0394656 i

0.0477729-0.0394656 *i*