Working with impurity in UEDGE:

1. Restore a pure-D case and ensure it is converged
2. Manually turn on the Li species to allocate the right size arrays
    - bbb.isimpon = 6
    - com.nzsp[0]=3
    - com.ngsp += 1
    - bbb.allocate()
3. Initialize the Li densities to negligible values to be as close to the initial state as possible
    - bbb.nis[:,:,2:]=1e10
    - bbb.ngs[:,:,1]=1e10
4. Now, save the state to a file (savefile="Li"+".hdf5"; hdf5_save(savefile))
5. Update the input file with the Li settings, and have the input file read the newly created save, as it has the right dimensions
    - Here, make sure the correct sputtering and reaction rates are used when setting up Li (such as, isph_sput = 1 (with cion = 3))
6. Converging the hydrogenic species only, using the very low 1e10 Li densities prescribed (bbb.isnion[2:]=0, bbb.isngon[1:]=0).
7. Then, turn them on one-by-one and reconverge the solution, until we have all Li species turned on at steady state.
8. Use the following functions to help diagnose the mode of failure in uedge:

```
from uedge.rundt import UeRun

UeRun().failureanalysis("./dtrun_7_last_ii2.hdf5")
UeRun().convergenceanalysis("./dtrun_7_last_ii2.hdf5")
```

9. These will display the equation and location of the failure, and the evolution of the initial fnrm. If the same equation fails in the same location, you might consider modifying the grid in that region, or tweaking the settings of the equation.

Found a converged case with Li on 8/29/2024 with the following setting (/home/islam9/Desktop/UEDGE/NSTX/NSTX_g128638/Li_Charge_State/BC_n6e19_Pe3Pi3e6_Dn_nonuniform_INGRID_Li)

- bbb.isnion = all D and Li activated
- bbb.isupon = all activated

- bbb.isteon = 0
- bbb.istion =1
- bbb.isngon[1]=1
- bbb.isupgon[0]=1
- wall and plate temp = 300 K
- bbb.recycp[1]=recycw[1]=1e-10

Sputtering model in UEDGE:

```
if (isph_sput(igsp) .ge. 1) then  # use fits for phys sput
            do ifld = ipsputt_s, ipsputt_e
             eng_sput = ( 0.5*mi(ifld)*up(ixt1,iy,ifld)**2 +
     .                  ti(ixt,iy) + zi(ifld)*
     .                  kappar(iy,jx)*te(ixt,iy) )/ev
             if(zi(ifld)>0.) sputflxrb(iy,igsp,jx) = sputflxrb(iy,igsp,jx) +
     .                          fnix(ixt1,iy,ifld)*
     .                  fphysyrb(igsp,jx)*yld96(matp,matt,eng_sput)
            enddo
```

hflux = main ion flux

```
if (sputtrb(iy,igsp,jx) .ge. 0. .or.
     .                   abs(sputflxrb(iy,igsp,jx)).gt.0.) then
            t0 = max(cdifg(igsp)*tg(ixt1,iy,igsp), tgmin*ev)
            vxn = 0.25 * sqrt( 8*t0/(pi*mg(igsp)) )
            areapl = isoldalbarea*sx(ixt1,iy) + (1-isoldalbarea)*sxnp(ixt1,iy)
              zflux = - sputtrb(iy,igsp,jx) * hflux -
     .                  sputflxrb(iy,igsp,jx) -
```

. **recyrb(iy,igsp,jx) * zflux** +

```
elseif (sputtrb(iy,igsp,jx).ge.-9.9) then # neg. sputtrb ==> albedo
                t0 = max(cdifg(igsp)*tg(ixt1,iy,igsp), tgmin*ev)
                vxn = 0.25 * sqrt( 8*t0/(pi*mg(igsp)) )
                yldot(iv) =  nurlxg*( fngx(ixt1,iy,igsp) -
     .            (1+sputtrb(iy,igsp,jx))*ng(ixt,iy,igsp)*vxn*sx(ixt1,iy) )
     .                              / (vxn*sx(ixt1,iy)*n0g(igsp))
              else                          # sputtrb < -9.9 ==> fix dens
                yldot(iv) = -nurlxg*(ng(ixt,iy,igsp)-ngplatrb(igsp,jx))/
     .                                    n0g(igsp)
```

Let’s apply

```
 bbb.sputtlb[:,1,0]= np.array([0.00035000000611196195
0.00035000000408360937
0.00035000000426459345
0.00035000000457294244
0.00035000000507665155
0.00035000000592696454
0.00035000000074453072
0.00035000001042076895
0.00035000001822193573
0.00035000000357118248
0.00035000000714504597
0.00035000003287960487
```

0.00035001199547605587

0.0003810977309257621

0.008748845175577825

0.017785793893790282

0.0019673432857559014

0.0005502488305213001

0.00036722158099745814

0.0003511516861049338

0.0003500737664674755

0.0003500077592164666

0.00035000075332857606

0.00035000006376646763

0.00035000000491663063

0.00035000000061107184])