Simulating Planar Ion Guides (pigsim) Helper Tool Overview

Directory Overview

Parameter list (.lua)

Geometry (.gem)

Particle Definitions (.fly2)

Created by python helper tool

Don’t edit the files in “Required / ”!

Initialization Script (.lua)
(Extrude_SLIM_setup.lua)

Prewritten

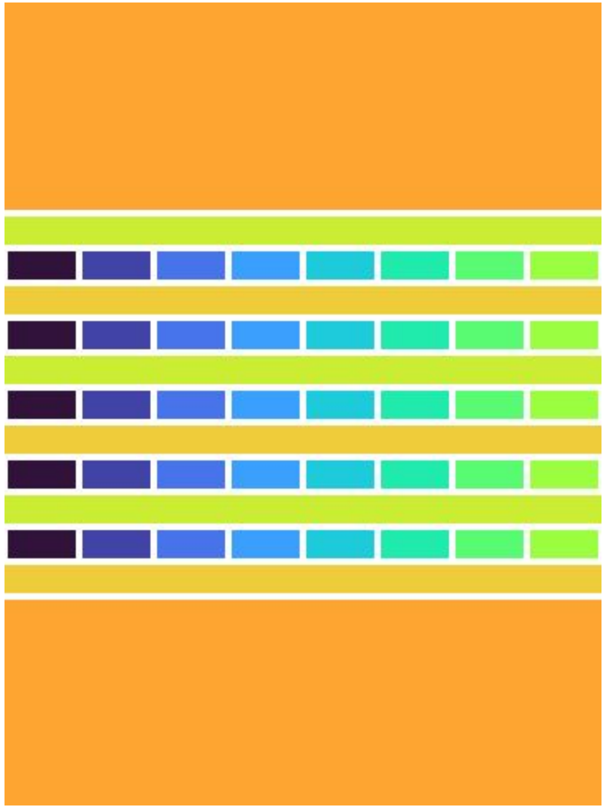
Custom simulation workspace
(Extrude_SLIM.iob +
Extrude_SLIM.lua)

Generated by setup lua

Simulation Generation Strategy

- Simple drawing functions enable electrode design and assignment
  - Currently only box electrodes, currently progressing towards more complicated shapes
  - Creation of electrode dictionary containing simulation potential assignments and colors for rendering monomer diagrams
- Electrode monomers saved as Python lists which can be easily addended to each other
  - Create polymeric pathlengths with simple rotation, translation, and addition of monomer lists
  - Easily extends to programmatic simulation creation, (*e.g.,* with a SLIM Pickins string)
- Simulation parameters determined from user inputs
  - Masses and charges of ions
  - Where along path to terminate ion flight

Monomer Creation



Major benefit of the **planar ion guide** format is modularity- define a monomer once and simply copy/paste it in a direction

SLIM is a known motif, making a few customization decisions (number of RF / TW rows, size of TW, RF, guard electrode sizes) and generating a SLIM layout is trivial

Moving on to asymmetric board pairs, alternate electrode shapes, and additional PCB features more easily achieved

Next: *Funtionzalizing to polymeric simulations*

Extrude a Path, Making Turns

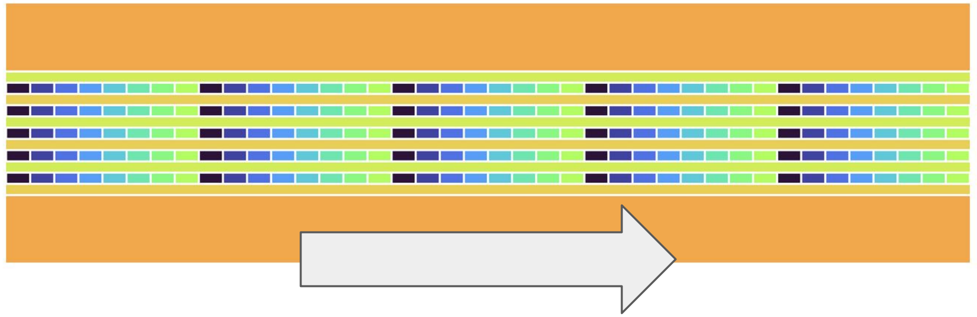
Codifying monomers makes functionalization simple:

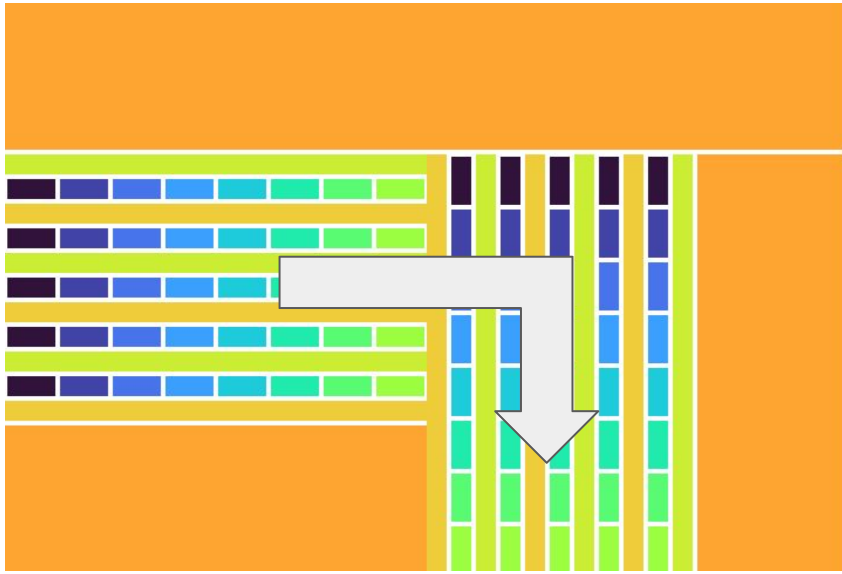
Function for extruding a straight path:

Function for simple 90º turns:

```
polymer_length = 5
print('separation length = ', polymer_length*monomerLen*pixelScale, 'mm')

fivemer_maxX, fivemer_maxY, fivemer_polys = extrudePolymer(monomer_poly_list, length = polymer_length, retXY=True)
plotPolyList(fivemer_polys, xPixels = fivemer_maxX, yPixels = fivemer_maxY)
```

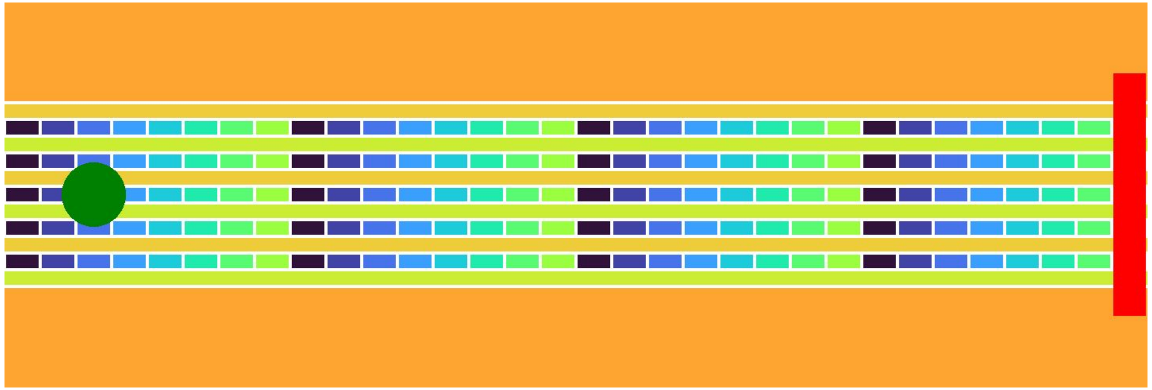
```
separation length =  45.72 mm
```

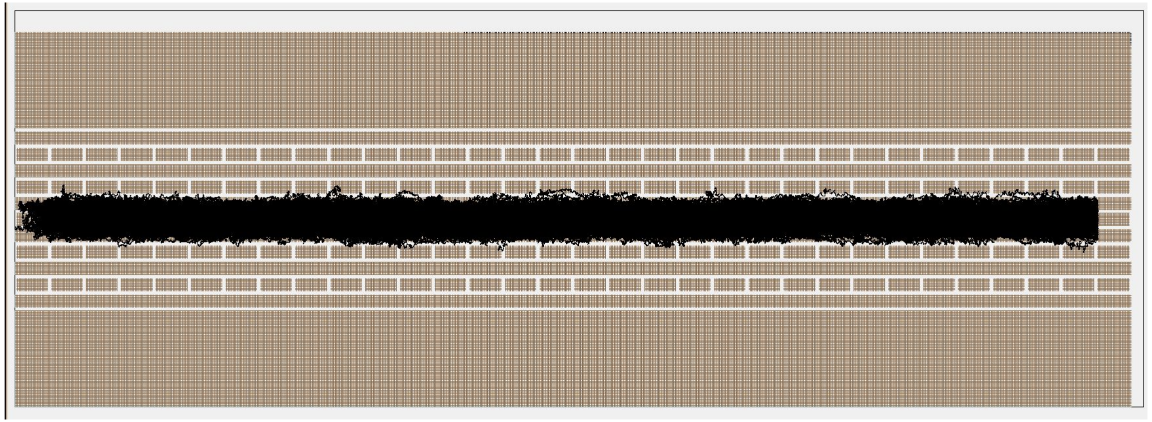




Tetramer ("FFFF")

Polygon map for tetramer polymer, ion definitions – Generated by helper tool

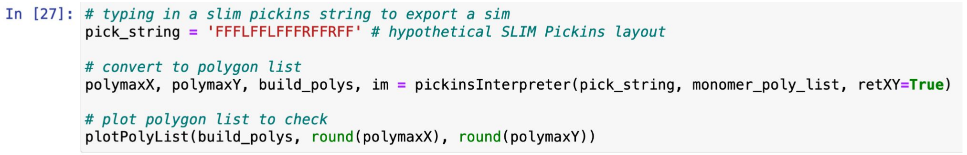




Programmatic SIMION generation

Pickins Style Layout / Predetermined Fly Files

Use polymer functions for quick translations of Slim Pickins designs:

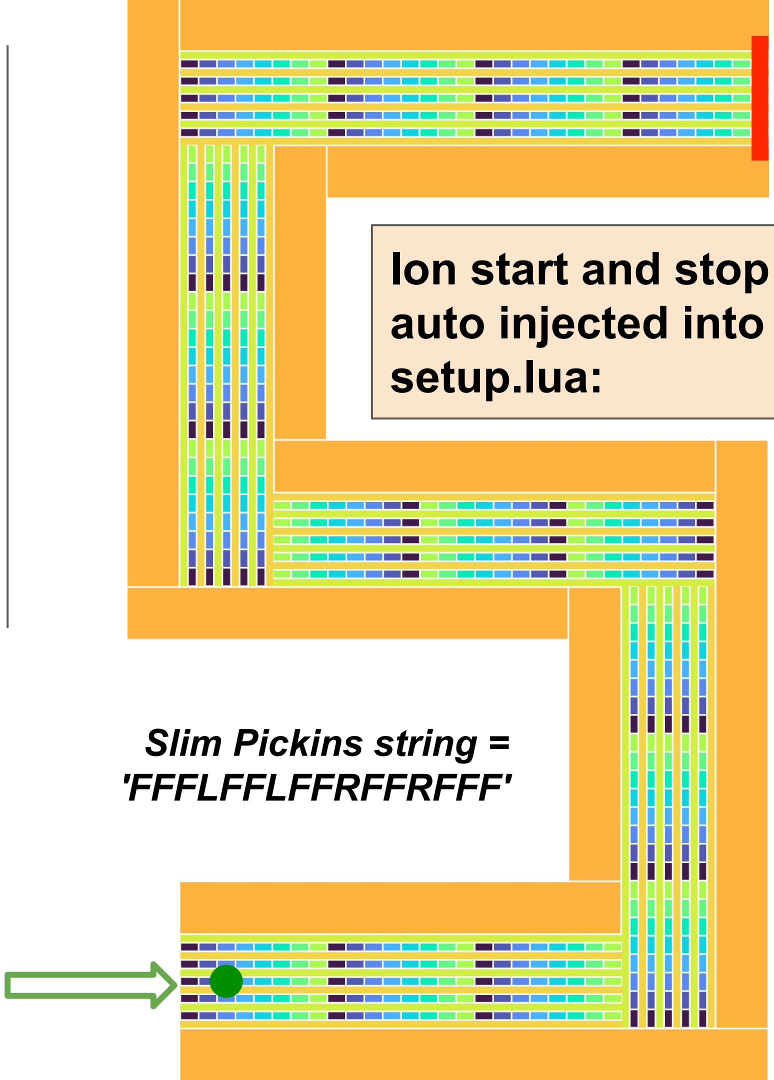


```
In [27]: # typing in a slim pickins string to export a sim
         pick_string = 'FFFLFFLFFFRFFFRFF' # hypothetical SLIM Pickins layout

         # convert to polygon list
         polymaxX, polymaxY, build_polys, im = pickinsInterpreter(pick_string, monomer_poly_list, retXY=True)

         # plot polygon list to check
         plotPolyList(build_polys, round(polymaxX), round(polymaxY))
```

Out[27]:

Enter masses, charges, number of ions per, and a function can automatically determine where ions should **start**

Another function lets the user select a polymer index along the SLIM path to **terminate** the ions at (defaults to the last monomer in the sequence, as shown here)

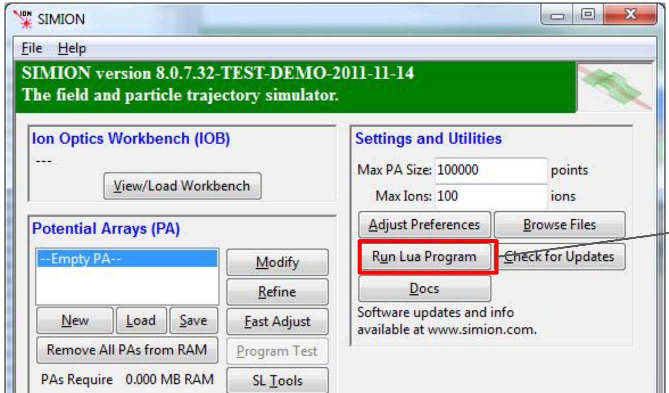


Ion start and stop auto injected into setup.lua:

Slim Pickins string = 'FFFLFFLFFRFFFRFFF'

SIMION Initialization

1. Running the setup lua from SIMION initializes the ion bench (.iob)
   a. This script takes longer for bigger geometries
2. After it completes, load your custom fly file (one is generated in the associated ipynb), then save the workspace as “project_name.iob”
3. In many versions of SIMION, adjustable parameters will not appear until ions have been flown at least once, so click “Fly’m”



Run “Extrude_SLIM_setup.lua”

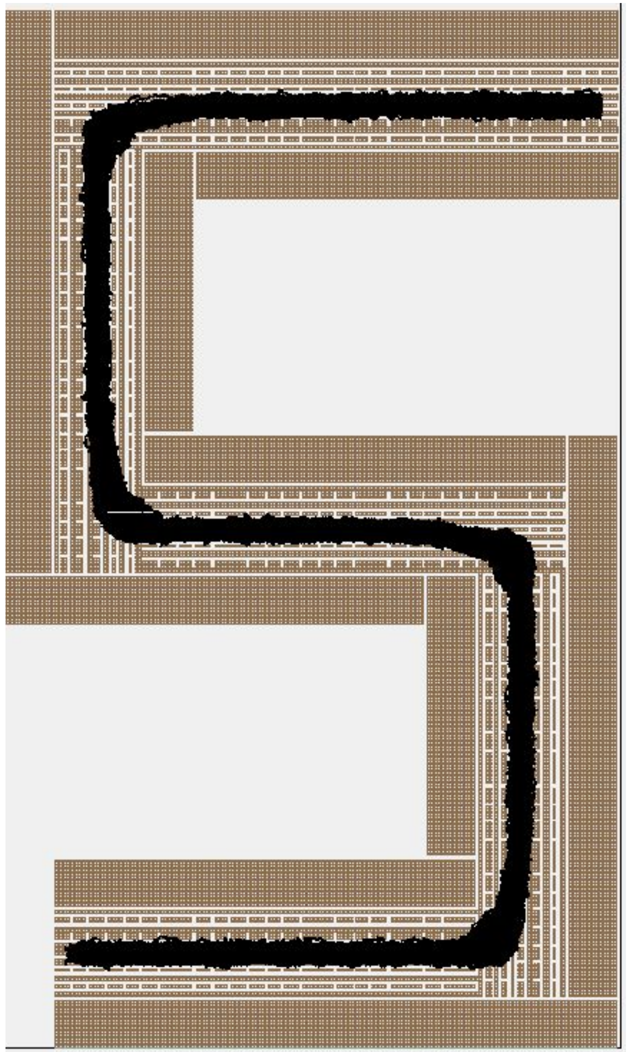
(this runtime increases for larger builds, the larger example takes ~25 minutes to build on my machine)

Flying Ions

Default workflow has the ions start and stop at the inlet and end of the SLIM polymer

Can alter the fly file from the ipynb or SIMION menus without touching the other files or needing to re-init, easy to change masses, n ions

Unfortunately changing the stop location currently requires a re-init because it is plugged into the setup.lua. Future functions will directly alter the project.lua



Future Directions

1. Greatly simplify, reduce dependencies on setup lua script
    a. Python functions that directly edit the project.lua eliminate restarts
2. Improve drawing functions + GUI access
    a. Additional shapes, rounded electrodes
3. Board-combining script to automatically couple multiple SIMION arrays
    a. Generalize pickins functions to alternative monomers
    b. Add KiCad tool for laying out saved polymers (*e.g.,* multipass junction or splitter)
4. Improve visualization of SIMION data
    a. Filtered feedback of ion current over polygon models
    b. Simulated visualization of confining fields