Quantum transport simulations using Kwant

Anton Akhmerov

ICTP Summer School “Advances in Condensed Matter Physics”,
9 May 2019

I borrowed from a tutorial by J.B. Weston (CC-BY-SA), link at the end

Plan

1. How to solve a scattering problem?
2. How to define and solve it with Kwant?

Kwant

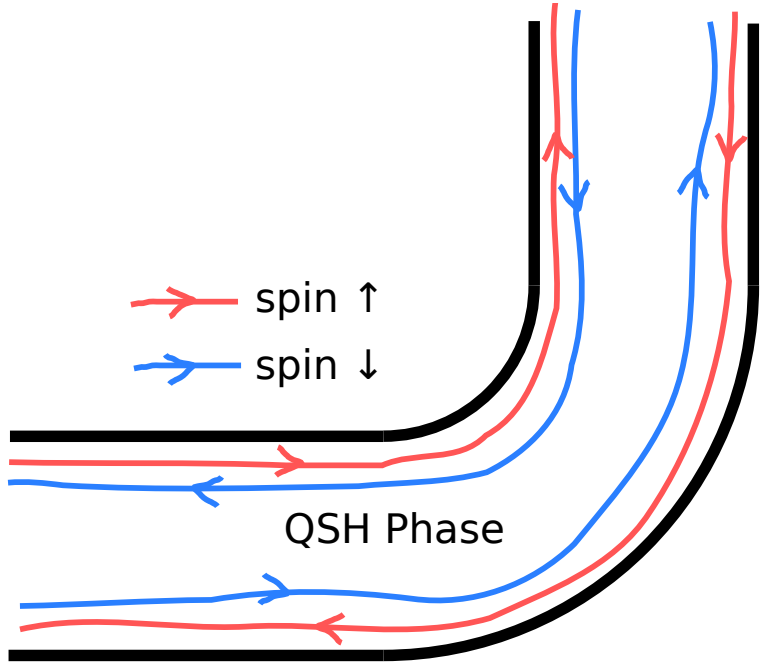
- An open source package for quantum transport
- `https://kwant-project.org`
- New version (1.4) released a couple of months ago
- Works with *tight binding* models
- Main focus on the scattering matrix formalism of quantum transport

Maintainers: Christoph Groth, Michael Wimmer, Anton Akhmerov, Xavier Waintal, Joseph Weston
Contributors: Jörg Behrmann, Paul Clisson, Mathieu Istas, Daniel Jaschke, Bas Nijholt, Michał Nowak, Viacheslav Ostroukh, Pablo Pérez Piskunow, Tómas Örn Rosdahl, Sebastian Rubbert, Rafał Skolasiński, Adrien Sorgniard, Dániel Varjas, Thomas Kloss, Pierre Carmier
Supported by: ERC, NWO, and ONR

Case study: a bend in a quantum spin Hall insulator

- Topologically protected transmission across a bend
- $k \cdot p$ Hamiltonian

$$
\begin{aligned}
H(k_x, k_y) = & \\
& + \mu - D(k_x^2 + k_y^2) \\
& + [M - B(k_x^2 + k_y^2)]\tau_z \\
& + A[k_x \sigma_z \tau_x + k_y \tau_y]
\end{aligned}
$$



Discretize

- Kwant works with *tight-binding models* and a finite number of degrees of freedom
- $\Rightarrow$ discretize space onto a square lattice with lattice spacing a

$$k_x^2 \to \frac{\partial^2}{\partial x^2} \to \frac{1}{4a^2} \begin{pmatrix} \ddots & \ddots & & & \\ \ddots & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & \ddots & \ddots \\ & & & \ddots & \ddots \end{pmatrix}$$

In Kwant

Discretize

```
import kwant

bhz_continuum = '''
    + mu * kron(sigma_0, sigma_0)
    + M * kron(sigma_0, sigma_z)
    - B * (k_x**2 + k_y**2) * kron(sigma_0, sigma_z) - D * (k_x**2 + k_y**2) * kron(sigma_0, sigma_0)
    + A * (k_x * kron(sigma_z, sigma_x) + k_y * kron(sigma_0, sigma_y))
'''
qshe_model = kwant.continuum.discretize(bhz_continuum)
```

Check the matrix elements

```
tb_matrix_elements, coords = kwant.continuum.discretize_symbolic(bhz_continuum)
tb_matrix_elements[0, 0]  # onsite
```

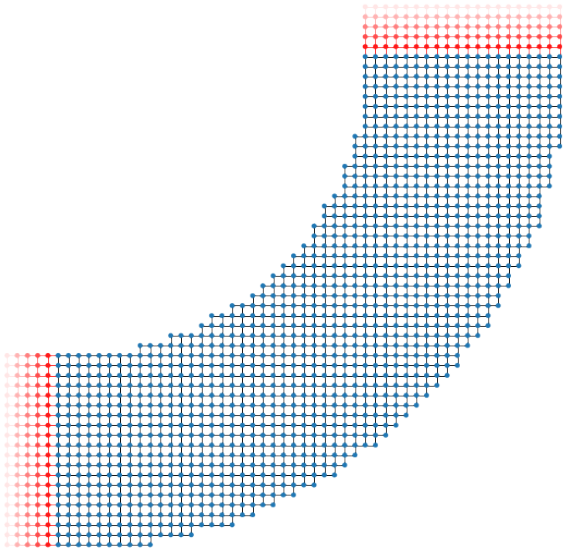
$$\begin{bmatrix} -\frac{2B}{a_y^2} - \frac{2B}{a_x^2} - \frac{2D}{a_y^2} - \frac{2D}{a_x^2} + M + \mu & 0 & 0 & 0 \\ 0 & \frac{2B}{a_y^2} + \frac{2B}{a_x^2} - \frac{2D}{a_y^2} - \frac{2D}{a_x^2} - M + \mu & 0 & 0 \\ 0 & 0 & -\frac{2B}{a_y^2} - \frac{2B}{a_x^2} - \frac{2D}{a_y^2} - \frac{2D}{a_x^2} + M + \mu & 0 \\ 0 & 0 & 0 & \frac{2B}{a_y^2} + \frac{2B}{a_x^2} - \frac{2D}{a_y^2} - \frac{2D}{a_x^2} - M + \mu \end{bmatrix}$$

Check the symmetries

```
symmetries = kwant.qsymm.find_builder_symmetries(qshe_model)
print(f"{len(symmetries)} symmetries found.")
```

```
17 symmetries found.
```

Infinite system



has the Hamiltonian:

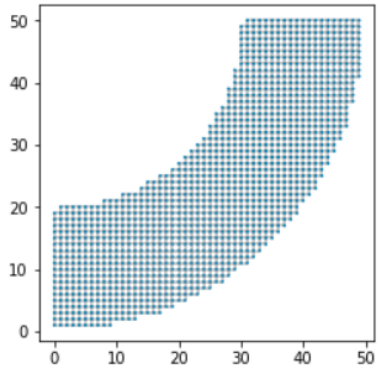
$$H = \begin{pmatrix} \ddots & V_L & & \\ V_L^\dagger & H_L & V_L & \\ & V_L^\dagger & H_L & V_L \\ & & V_L^\dagger & H_S \end{pmatrix}$$

(H_L and V_L are block-diagonal if there are many leads)

In Kwant

Scattering region

```
syst = kwant.Builder() # We'll fill this with the TB model of our scattering region

def scattering_region(site):
    x, y = site.tag  # position in integer lattice coordinates
    y = y - 50
    return (30**2 < (x**2 + y**2) < 50**2  # inside a ring
            and (x >= 0 and y <= 0))  # lower-right quadrant ofring

# Fill our scattering region with the BHZ model in a certain region of space
syst.fill(template=qshe_model, shape=scattering_region, start=(0, 10))

kwant.plot(syst);
```



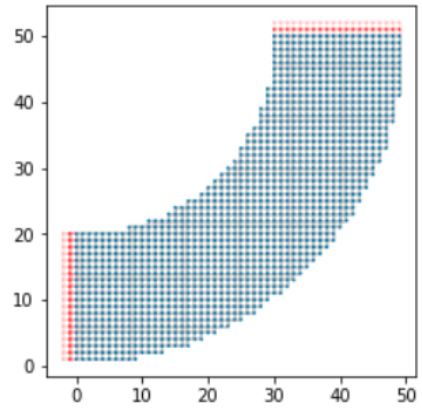
In Kwant

Make a lead

```
lead_x = kwant.Builder(
    symmetry=kwant.TranslationalSymmetry((-1, 0)),
    time_reversal=np.kron(1j * sigma_y, sigma_0),
    conservation_law=np.kron(-sigma_z, sigma_0),  # spin conservation
)

lead_x.fill(qshe_model, lambda site: 1 <= site.tag[1] <= 20, (0, 1))
```

Combine everything

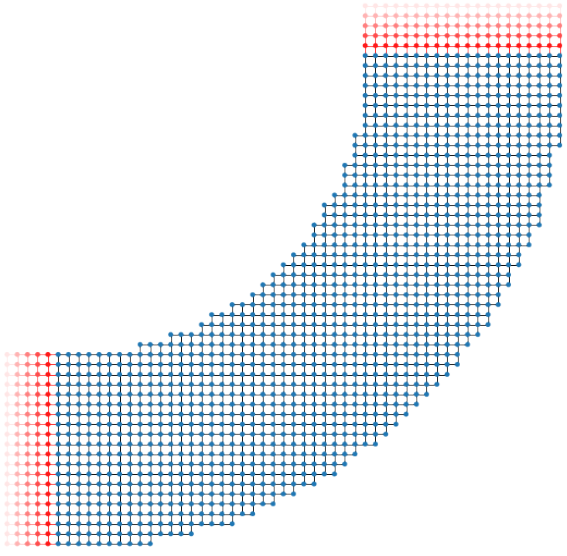
```
syst.attach_lead(lead_x)
syst.attach_lead(lead_y)

kwant.plot(syst);
```




```
fsyst = syst.finalized()  # Transform the system into an efficient form for numerics
print(fsyst)
```

```
<FiniteSystem with 1272 sites, 4888 hoppings, and parameters: ('M', 'B', 'D', 'A', 'mu')>
```

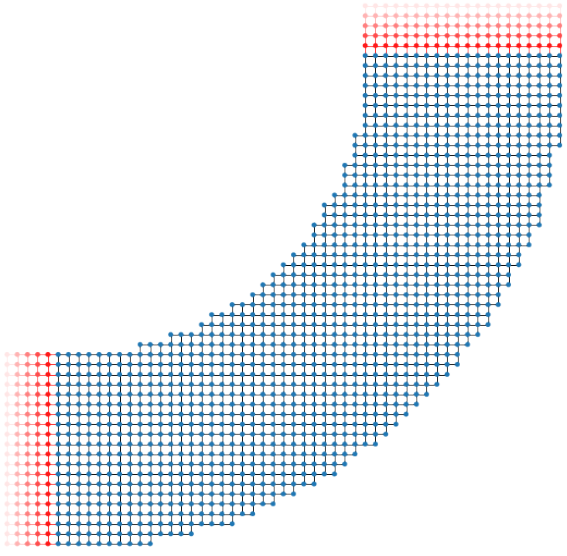
Scattering problem



Now to get conductance and other observables, we only need to calculate Σ_{lead}, G^R, $G^<$, and we're done.

(that is what one mostly hears)

Scattering problem

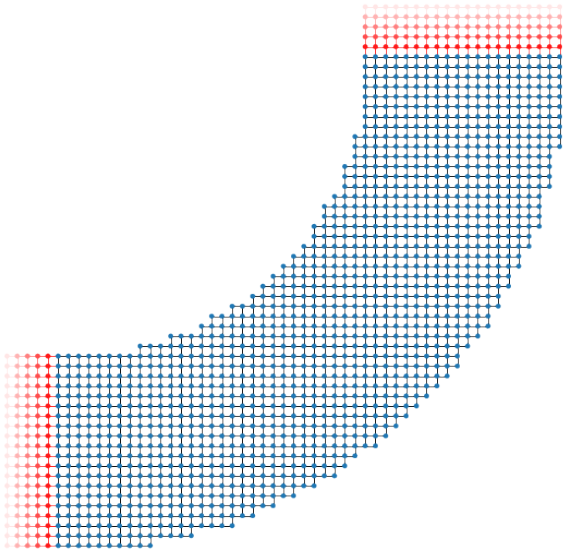


Now to get conductance and other observables, we only need to calculate Σ_{lead}, G^R, $G^<$, and we're done.

(that is what one mostly hears)

. . . But there is a more intuitive definition.

Scattering problem



Problem statement:

$$H = \begin{pmatrix} \ddots & V_L & & \\ V_L^\dagger & H_L & V_L & \\ & V_L^\dagger & H_L & V_L \\ & & V_L^\dagger & H_S \end{pmatrix}, \quad \psi = \begin{pmatrix} \vdots \\ \psi_2 \\ \psi_1 \\ \psi_S \end{pmatrix}$$

We are solving $(H - E)\psi = 0$ with a *fixed* E.

Modes in the lead

Since the leads are translationally invariant, we decompose the lead wave function into eigenvectors of translation (lead modes)

$$\psi_j = \lambda^j \psi_0,$$

Modes in the lead

Since the leads are translationally invariant, we decompose the lead wave function into eigenvectors of translation (lead modes)

$$\psi_j = \lambda^j \psi_0,$$

Now substitute into H:

$$V_L \psi_0 + H\lambda\psi_0 + V_L^\dagger \lambda^2 \psi_0 = 0,$$

Modes in the lead

Since the leads are translationally invariant, we decompose the lead wave function into eigenvectors of translation (lead modes)

$$\psi_j = \lambda^j \psi_0,$$

Now substitute into H:

$$V_L \psi_0 + H\lambda\psi_0 + V_L^\dagger \lambda^2 \psi_0 = 0,$$

or in a matrix form:

$$\begin{pmatrix} -V_L^{-1} H_L & -V_L^{-1} \\ V_L^\dagger & 0 \end{pmatrix} \begin{pmatrix} \psi_0 \\ V_L^\dagger \psi_1 \end{pmatrix} = \lambda^{-1} \begin{pmatrix} \psi_0 \\ V_L^\dagger \psi_1 \end{pmatrix}$$

Modes in the lead

Since the leads are translationally invariant, we decompose the lead wave function into eigenvectors of translation (lead modes)

$$\psi_j = \lambda^j \psi_0,$$

Now substitute into H:

$$V_L \psi_0 + H\lambda\psi_0 + V_L^\dagger \lambda^2 \psi_0 = 0,$$

or in a matrix form:

$$\begin{pmatrix} -V_L^{-1}H_L & -V_L^{-1} \\ V_L^\dagger & 0 \end{pmatrix} \begin{pmatrix} \psi_0 \\ V_L^\dagger \psi_1 \end{pmatrix} = \lambda^{-1} \begin{pmatrix} \psi_0 \\ V_L^\dagger \psi_1 \end{pmatrix}$$

. . . or in a more stable form and reduced basis:

$$\begin{pmatrix} iA^\dagger \tilde{H}^{-1} B & -A^\dagger \tilde{H}^{-1} B \\ -1 + iB^\dagger \tilde{H}^{-1} B & -B^\dagger \tilde{H}^{-1} B \end{pmatrix} \begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \end{pmatrix} = \lambda^{-1} \begin{pmatrix} A^\dagger \tilde{H}^{-1} A & -1 - iA^\dagger \tilde{H}^{-1} A \\ B^\dagger \tilde{H}^{-1} A & -iB^\dagger \tilde{H}^{-1} A \end{pmatrix} \begin{pmatrix} \tilde{\psi}_0 \\ \tilde{\psi}_1 \end{pmatrix},$$

$$\tilde{H} = H_L + iAA^\dagger + iBB^\dagger, \quad V_L = AB^\dagger, \quad \tilde{\psi}_0 = B^+ \psi_0, \quad \tilde{\psi}_1 = A^+ \psi_1$$

Separating the modes

Split all the eigenvectors into incoming, outgoing and evanecsent, such that:

$$\langle \psi_{\text{in}} | \hat{I} | \psi_{\text{in}} \rangle = 1$$

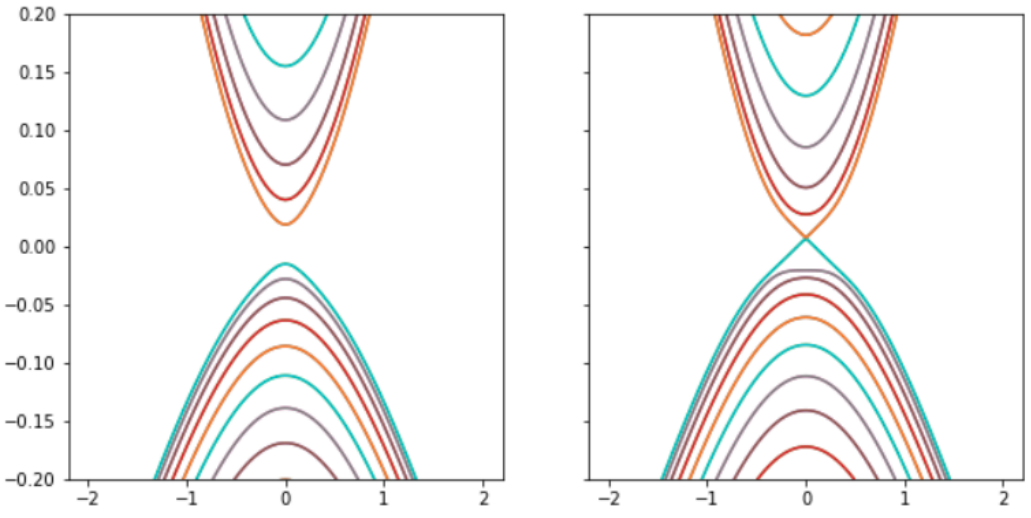
$$\langle \psi_{\text{out}} | \hat{I} | \psi_{\text{out}} \rangle = -1$$

$$\langle \psi_{\text{evan}} | \hat{I} | \psi_{\text{out}} \rangle = 0, \quad |\lambda_{\text{evan}}| < 1$$

In Kwant

Dispersion

```
topo_params = dict(A=0.09, B=-0.18, D=-0.065, M=-0.02, mu=0)  # Parameters for topological phase
trivial_params = dict(A=0.09, B=-0.18, D=-0.065, M=0.01, mu=0)  # Parameters for trivial phase

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5), sharey=True)
kwant.plotter.bands(flead_x, momenta=np.linspace(-2, 2, 101), params=trivial_params, show=False, ax=ax1)
kwant.plotter.bands(flead_x, momenta=np.linspace(-2, 2, 101), params=topo_params, show=False, ax=ax2)
ax1.set_ylim(-.2, .2);
```



Modes at $E = 0$

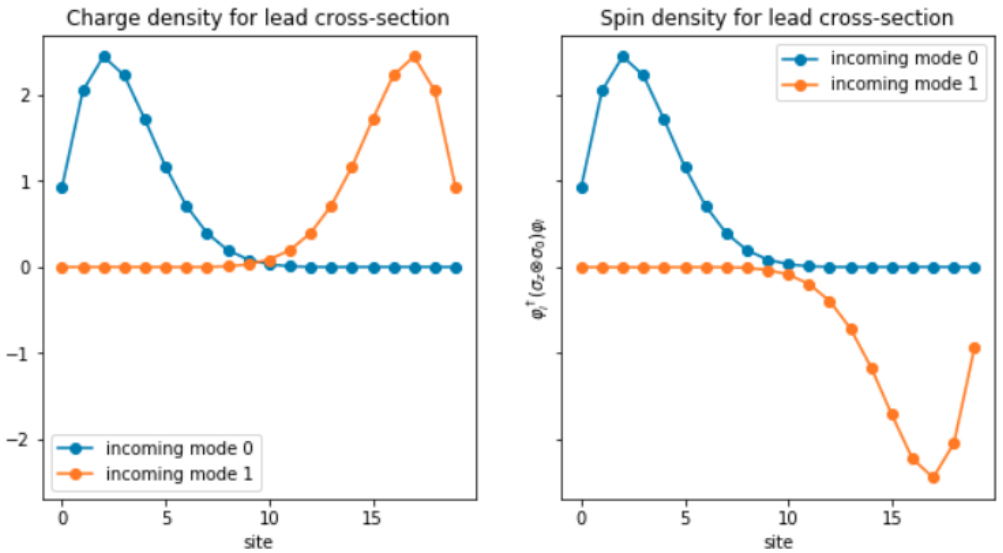
```
propagating_modes, _ = flead_x.modes(energy=0, params=topo_params)

print(propagating_modes.wave_functions.shape)
print(propagating_modes.velocities)
print(propagating_modes.momenta)
```

```
(80, 4)
[-0.08354984 -0.08354984  0.08354984  0.08354984]
[ 0.08609145  0.08609145 -0.08609145 -0.08609145]
```

In Kwant

Mode wave functions

```
phi0, phi1 = propagating_modes.wave_functions[:, :2].transpose()

density = kwant.operator.Density(fsyst.leads[0]) # calculate |φ_i|**2 for each site (summing degrees of freedom)
spin_density = kwant.operator.Density(fsyst.leads[0], np.kron(sigma_z, sigma_0))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5), sharey=True)
plot_densities(density(phi0), density(phi1), title='Charge', ax=ax1)
plot_densities(spin_density(phi0), spin_density(phi1), title='Spin', ax=ax2)
```



Equations to solve

Substitute the lead modes into the Hamiltonian and get a linear system:

$$\begin{pmatrix} -U_{\text{out}} & 1 \\ V_L^\dagger U_{\text{out}} \Lambda_{\text{out}} & H_{\text{S}} \end{pmatrix} \begin{pmatrix} \mathbf{S} \\ \psi_S \end{pmatrix} = \begin{pmatrix} U_{\text{in}} \\ -V_{\text{L}}^\dagger U_{\text{in}} \Lambda_{\text{in}}, \end{pmatrix} \tag{1}$$

with U_{in} and U_{out} wave functions of incoming and outgoing modes, and $\Lambda \equiv \text{diag}(\lambda_i)$.

Equations to solve

Substitute the lead modes into the Hamiltonian and get a linear system:

$$\begin{pmatrix} -U_{\rm out} & 1 \\ V_L^\dagger U_{\rm out}\Lambda_{\rm out} & H_{\rm S} \end{pmatrix} \begin{pmatrix} \mathbf{S} \\ \psi_S \end{pmatrix} = \begin{pmatrix} U_{\rm in} \\ -V_{\rm L}^\dagger U_{\rm in}\Lambda_{\rm in}, \end{pmatrix} \tag{1}$$

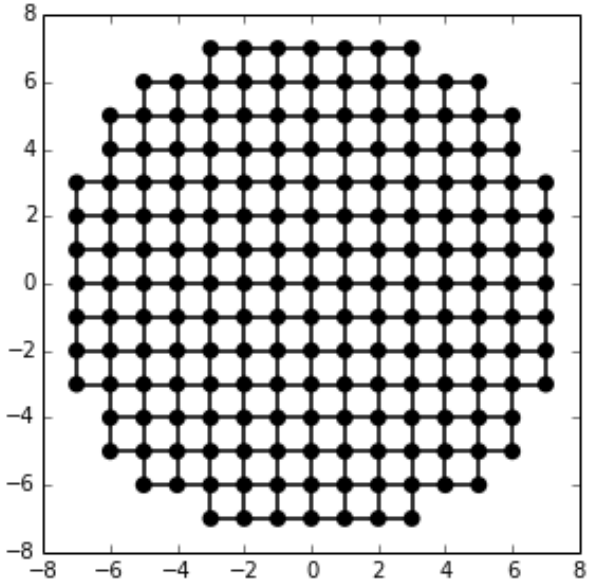
with $U_{\rm in}$ and $U_{\rm out}$ wave functions of incoming and outgoing modes, and $\Lambda \equiv \mathrm{diag}(\lambda_i)$.

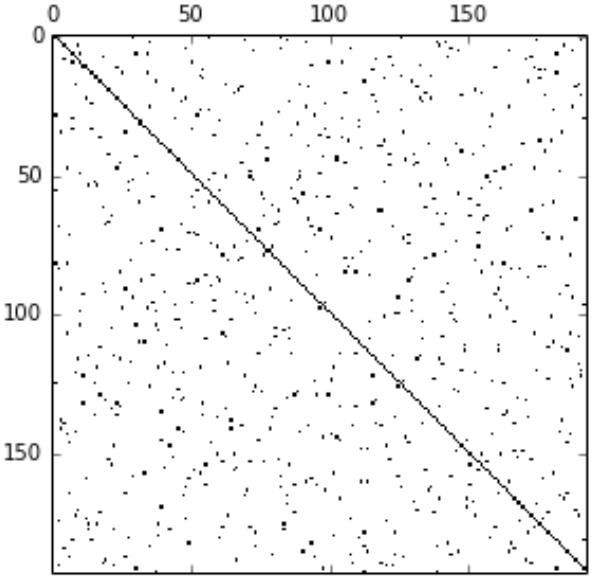
Next: write down these linear equations and solve them.

(NB: if we start by eliminating **S**, the rhs becomes $H_S + \Sigma$)

Solving a scattering problem

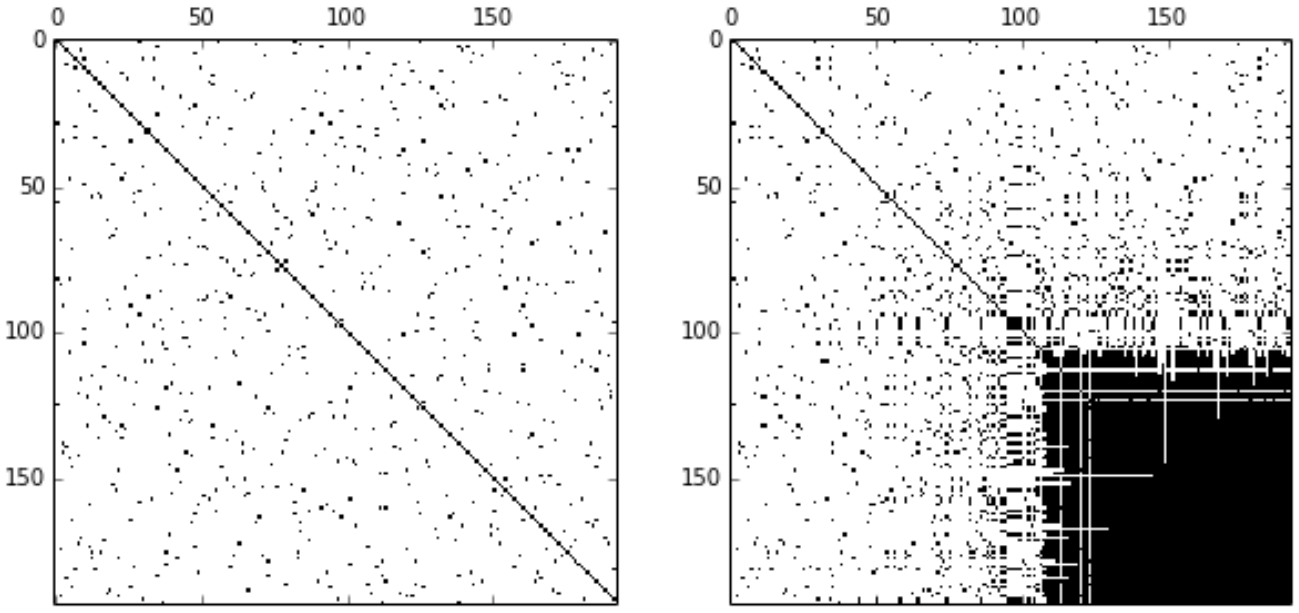
The scattering region is large and its Hamiltonian is sparse.





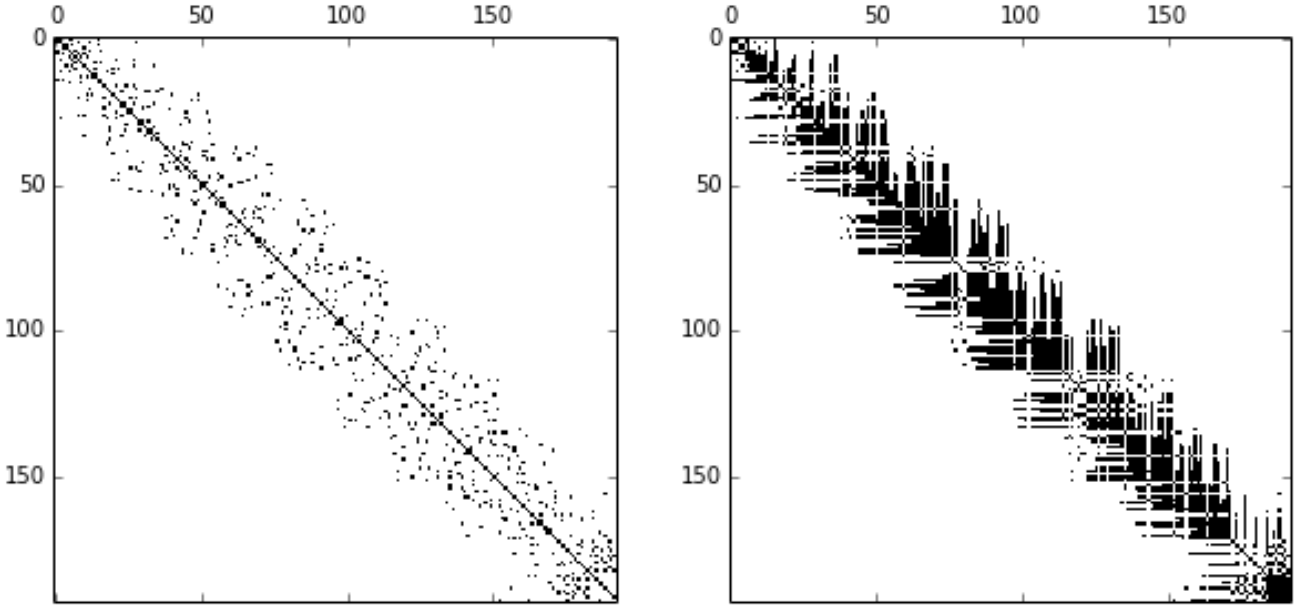
Solving a scattering problem

Naive solution: gaussian elimination
(dense LU decomposition, cost L^6)



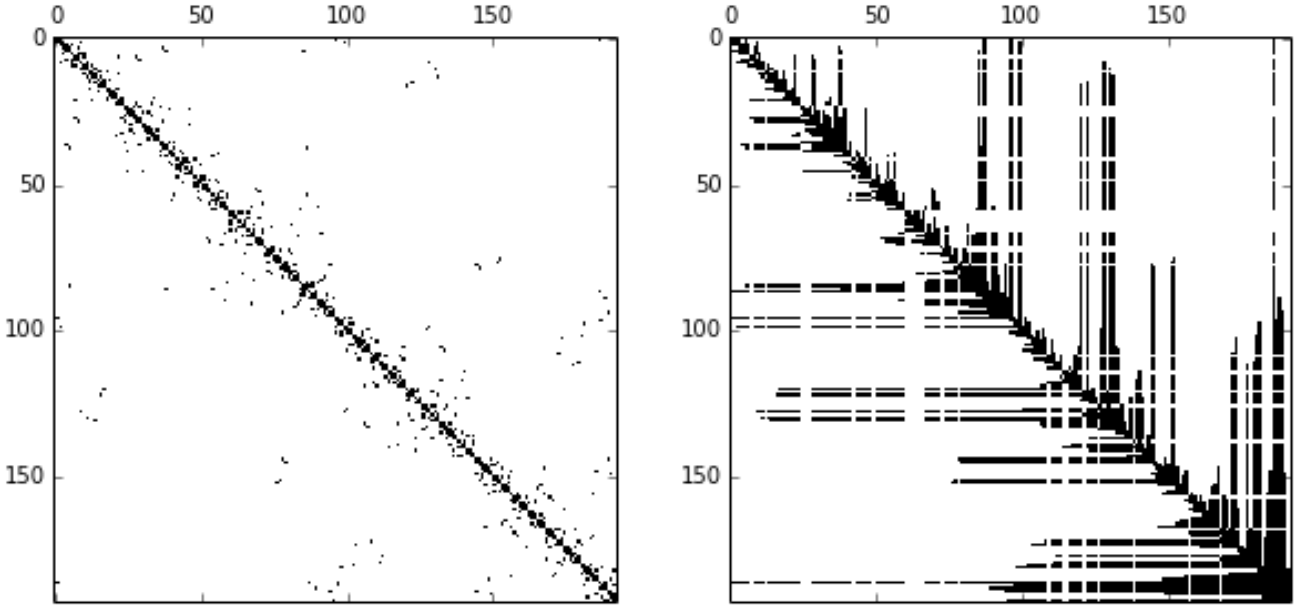
Solving a scattering problem

Better option: reshuffle to reduce bandwidth
("recursive Green's functions", cost L^4)



Solving a scattering problem

What specialized libraries do
(nested dissection, cost L^3, $\sim 10\times$ better):



In Kwant

Scattering matrix

```
S = kwant.smatrix(fsyst, energy=0, params=topo_params)

print(np.round(S.data, 2))
print(S.transmission(1, 0))  # Tr(t . t^†)
```

```
[[-0.  -0.j    0.  +0.j    0.27-0.96j  0.  +0.j  ]
 [-0.  +0.j   -0.  -0.j   -0.  +0.j    0.5 +0.86j]
 [ 0.5 +0.86j  0.  +0.j   -0.  -0.j    0.  +0.j  ]
 [-0.  +0.j    0.27-0.96j -0.  +0.j   -0.  -0.j  ]]
1.9999957932536172
```

In Kwant

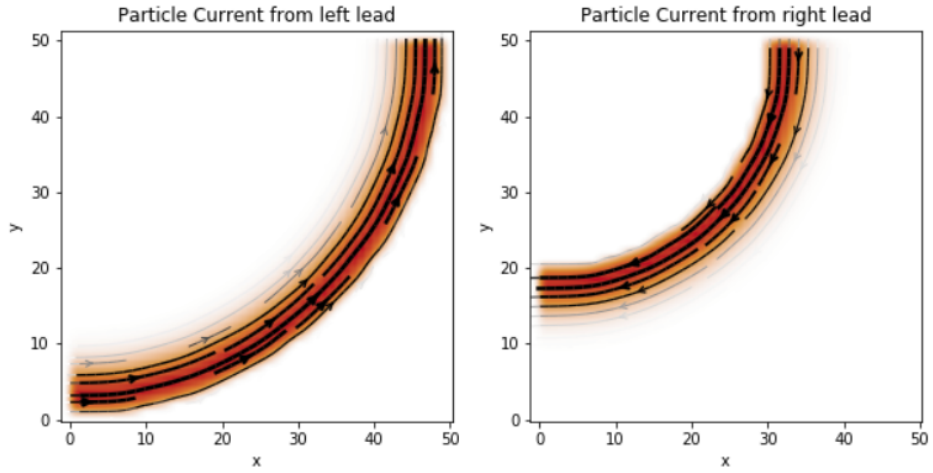
Scattering matrix

```
S = kwant.smatrix(fsyst, energy=0, params=topo_params)

print(np.round(S.data, 2))
print(S.transmission(1, 0))  # Tr(t . t^†)
```

```
[[-0.  -0.j    0.  +0.j    0.27-0.96j  0.  +0.j  ]
 [-0.  +0.j   -0.  -0.j   -0.  +0.j    0.5 +0.86j]
 [ 0.5 +0.86j  0.  +0.j   -0.  -0.j    0.  +0.j  ]
 [-0.  +0.j    0.27-0.96j -0.  +0.j   -0.  -0.j  ]]
1.9999957932536172
```

Current density

```
current = kwant.operator.Current(fsyst).bind(params=topo_params)

(ax0, ax1) = prepare_axes('Particle Current');
kwant.plotter.current(fsyst, current(sl0) , ax=ax0)
kwant.plotter.current(fsyst, current(sr0), ax=ax1)
```



In Kwant

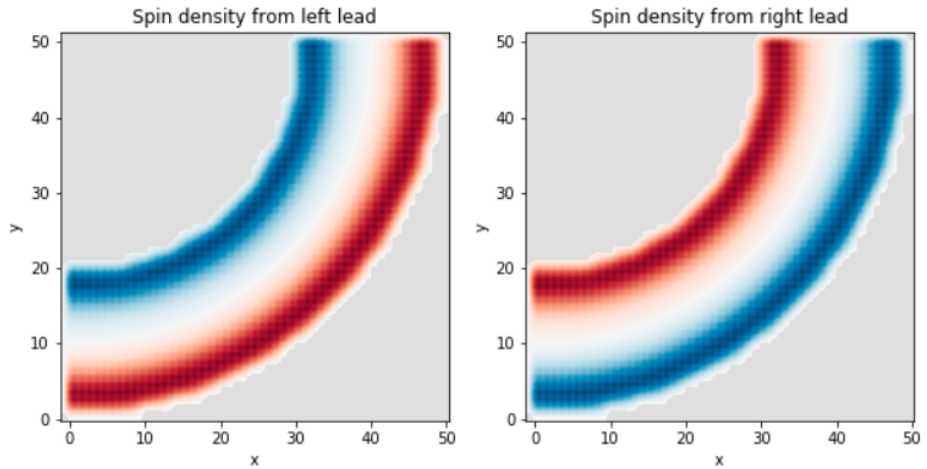
Scattering matrix

```
S = kwant.smatrix(fsyst, energy=0, params=topo_params)

print(np.round(S.data, 2))
print(S.transmission(1, 0))  # Tr(t . t^†)
```

```
[[-0.  -0.j    0.  +0.j    0.27-0.96j  0.  +0.j  ]
 [-0.  +0.j   -0.  -0.j   -0.  +0.j    0.5 +0.86j]
 [ 0.5 +0.86j  0.  +0.j   -0.  -0.j    0.  +0.j  ]
 [-0.  +0.j    0.27-0.96j -0.  +0.j   -0.  -0.j  ]]
1.9999957932536172
```

Spin density

```
spin_density = kwant.operator.Density(fsyst, spinz_operator).bind(params=topo_params)

(ax0, ax1) = prepare_axes('Spin density')
kwant.plotter.density(fsyst, spin_density(sl0) + spin_density(sl1), cmap='RdBu_r', ax=ax0)
kwant.plotter.density(fsyst, spin_density(sr0) + spin_density(sr1), cmap='RdBu_r', ax=ax1)
```



Conclusions

- Write down a problem + feed it to a good solver = problem solved
- Kwant does this with quantum transport
- Try a live version at
  `https://tiny.cc/maryland-kwant-tutorial`
  (I used code and images from it)

The end.
Questions?