Tkwant: a software package for time-dependent quantum transport

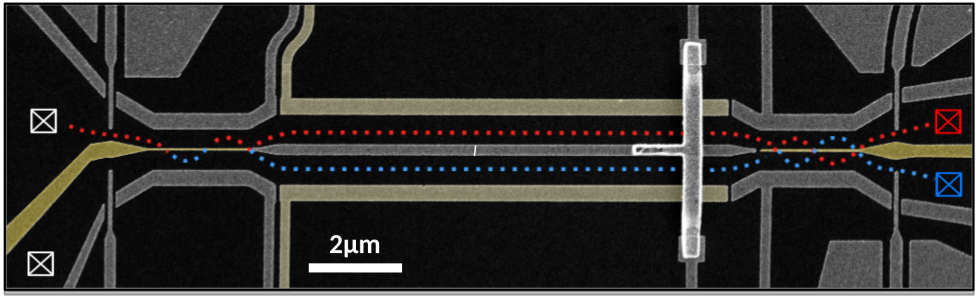
Thomas Kloss

CNRS Néel, Grenoble / France

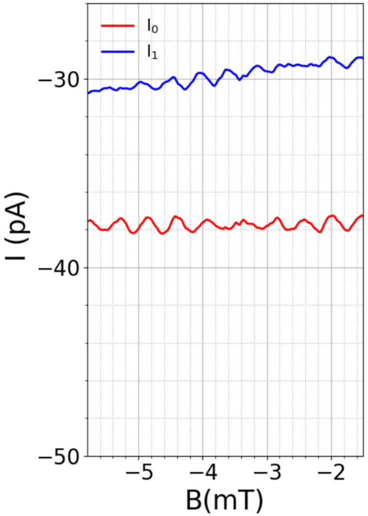
TK, Weston, Gaury, Rossignol, Groth and Waintal, *New J. Phys.* **23** 023025 (2021)

Motivation

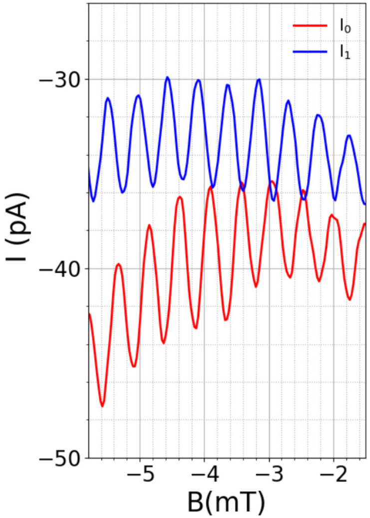
Aharonov-Bohm oscillations of „flying Qubits“ in an electronic Mach-Zehnder interferometer



static (DC)



pulses



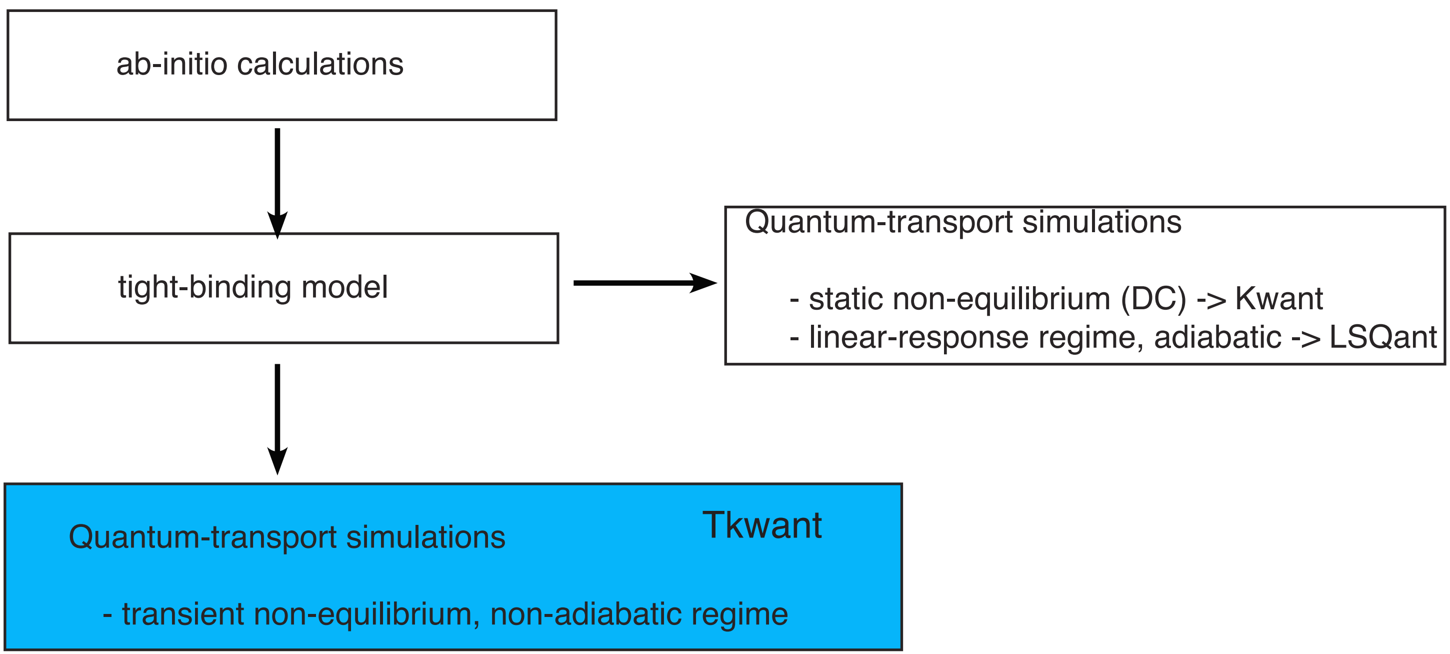
New physical effects in the transient regime. We aim for a simulation tool to describe them.

C. Bauerle group at Neel/Grenoble, measurements by S. Ouacel.

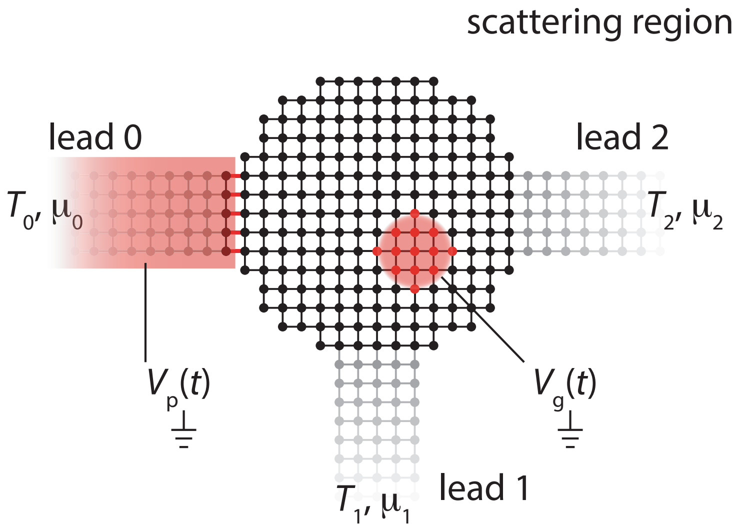
Motivation

New erea of experiments

- Time-resolved measurements
- GHz / THz regime
- Coherent single-electron sources
- Levitons, flying Qubits

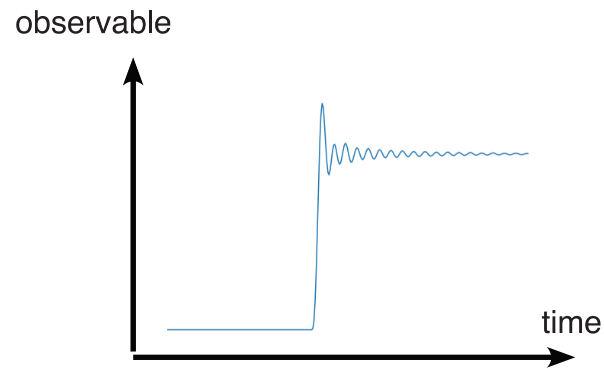


Simulating quantum nanolectronics



tight-binding approximation

$$\hat{\mathbf{H}}(t) = \sum_{i,j} \mathbf{H}_{ij}(t)\hat{c}_i^\dagger \hat{c}_j$$



input

chemical potentials μ
temperatures T
Hamiltonian $H_{ij}(t)$

→

output

time-dependent manybody observables as e.g.

densities $n_i(t) = \langle c_i^\dagger c_i \rangle(t)$

currents $I_{ij}(t) = \Im(H_{ij}\langle c_i^\dagger c_j \rangle(t))$

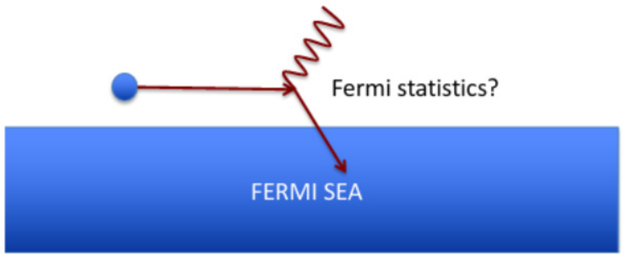
Green functions $G_{ij}^<(t,t') = i\langle c_j^\dagger(t') c_i(t)\rangle$

Infinite system and Pauli principle

infinite matrix

$$i\partial_t \psi_{\alpha E}(t,i) = \sum_j \mathbf{H}_{ij}(t)\psi_{\alpha E}(t,j),$$

$$\hat{\mathbf{H}}(t) = \sum_{i,j} \mathbf{H}_{ij}(t)\hat{c}_i^\dagger \hat{c}_j$$



The problem is manybody even without interactions!

Wave function formalism (equivalent to Keldysh)

thermal equilibrium

1. calculate **stationary** scattering states

$$\mathbf{H}(t=0)\psi_{\alpha E} = E\psi_{\alpha E}.$$

perturbation $t = 0$

out-of-equilibrium

2. evolve **time-dependent** scattering states

$$i\partial_t \psi_{\alpha E}(t,i) = \sum_j \mathbf{H}_{ij}(t)\psi_{\alpha E}(t,j),$$

$$\psi_{\alpha E}(t<t_0,i) = \psi_{\alpha E}(i)e^{-iEt}$$

3. calculation of obervables

$$n_i(t) \equiv \langle \hat{c}_i^\dagger \hat{c}_i \rangle(t) = \sum_\alpha \int \frac{dE}{2\pi} f_\alpha(E)|\psi_{\alpha E}(t,i)|^2$$

$$f_\alpha(E) = \frac{1}{e^{(E-\mu_\alpha)/k_B T_\alpha} + 1}$$

time

Gaury, Weston, Santin, Houzet, Groth and Waintal, Phys. Rep. 534, 1 (2014).

Nonequilibrium Green function approach (NEGF)

The wave function approach is mathematically equivalent to the Keldysh Green function approach

$$i\partial_t G^R(t,t') = \mathbf{H}_{\bar{0}\bar{0}}(t)G^R(t,t') + \int du \ \Sigma^R(t,u)G^R(u,t')$$

$$G^<(t,t') = \int du \ dv \ G^R(t,u)\Sigma^<(u,v)[G^R(t',v)]^\dagger$$

Both formulations related via

$$G^<_{ij}(t,t') = \sum_\alpha \int \frac{dE}{2\pi} i f_\alpha(E)\psi_{\alpha E}(t,i)\psi^*_{\alpha E}(t',j),$$
$$G^R_{ij}(t,t') = -i\theta(t-t')\sum_\alpha \int \frac{dE}{2\pi}\psi_{\alpha E}(t,i)\psi^*_{\alpha E}(t',j)$$

Disadvantage of Green function approach

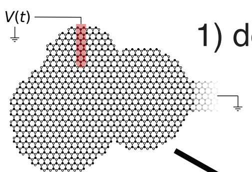
- numerically much more expensive: scaling *sites*^{3} x *time*^{2}

- numerical stability

Gaury, Weston, Santin, Houzet, Groth and Waintal, Phys. Rep. 534, 1 (2014).

Tkwant

Tkwant: A Python package for time-dependent quantum transport
nonstationary version of the *Kwant* code (Groth, Wimmer, Akhmerov, and Waintal, New J. Phys. 16, 063065 (2014))

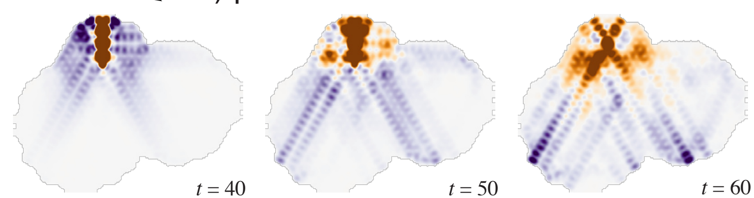


1) define the system

2) write a simple script

```
import tkwant
import kwant

syst = make_system()

current_operator = kwant.operator.Current(syst)

state = tkwant.manybody.State(syst, tmax=1000)

for time in range(1000):
    state.evolve(time)
    current = state.evaluate(current_operator)
```

3) perform the actual simulation



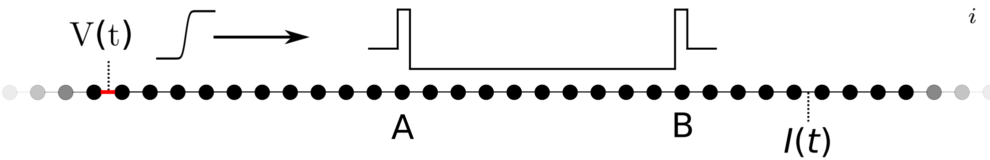
example: graphene flake

- Python package (opensource, BSD license)
- simple and flexible
- adaptive numerical routines
- MPI parallelized
- performant, linear scaling: *size* x *time*

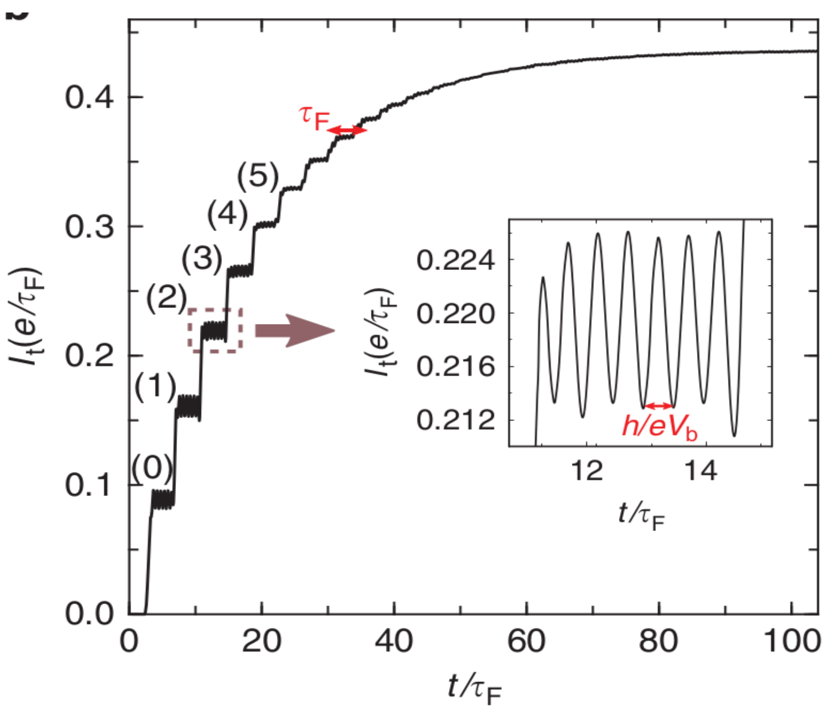
https://tkwant.kwant-project.org/

Example: Fabry-Perot interferometer

$$\hat{\mathbf{H}}(t) = \sum_{i}^{N_s+1} \epsilon_i \hat{c}_i^\dagger \hat{c}_i - \sum_{-\infty}^{\infty} \hat{c}_{i+1}^\dagger \hat{c}_i - [e^{i\phi(t)} - 1]\hat{c}_1^\dagger \hat{c}_0 + \text{h.c.}$$



$$V(t) = \begin{cases} 0, & \text{for } t < 0 \\ \frac{V_b}{2}\left(1 - \cos\left(\frac{\pi t}{\tau}\right)\right), & \text{for } 0 \le t \le \tau \\ V_b, & \text{for } t > \tau \end{cases}$$



Gaury, Weston, and Waintal, Nat. Commun. 6, 6524 (2015).

Tkwant simulation script for the Fabry-Perot interferometer

TK, Weston, Gaury, Rossignol, Groth and Waintal, *New J. Phys.* **23** 023025 (2021).

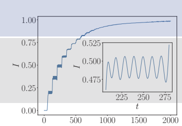
```
import tkwant
import kwant
from math import sin, pi
import matplotlib.pyplot as plt


def make_fabry_perot_system():
    # Define an empty tight-binding system on a square lattice.
    lat = kwant.lattice.square(norbs=1)
    syst = kwant.Builder()

    # Central scattering region.
    syst[(lat(x, 0) for x in range(80))] = 0
    syst[lat.neighbors()] = -1
    # Backgate potential.
    syst[(lat(x, 0) for x in range(5, 75))] = -0.0956
    # Barrier potential.
    syst[[lat(4, 0), lat(75, 0)]] = 5.19615

    # Attach lead on the left- and on the right-hand side.
    sym = kwant.TranslationalSymmetry((-1, 0))
    lead = kwant.Builder(sym)
    lead[(lat(0, 0))] = 0
    lead[lat.neighbors()] = -1
    syst.attach_lead(lead)
    syst.attach_lead(lead.reversed())

    return syst, lat


# Phase from the time integrated voltage V(t).
def phi(time):
    vb, tau = 0.6, 30.
    if time > tau:
        return vb * (time - tau / 2.)
    return vb / 2. * (time - tau / pi * sin(pi * time / tau))


times = range(2000)

# Make the system and add voltage V(t) to the left lead (index 0).
syst, lat = make_fabry_perot_system()
tkwant.leads.add_voltage(syst, 0, phi)
syst = syst.finalized()

# Define an operator to measure the current after the barrier.
hoppings = [(lat(78, 0), lat(77, 0))]
current_operator = kwant.operator.Current(syst, where=hoppings)

# Set occupation T = 0 and mu = -1 for both leads.
occup = tkwant.manybody.lead_occupation(chemical_potential=-1)

# Initialize the time-dependent manybody state.
state = tkwant.manybody.State(syst, tmax=max(times),
                              occupations=occup)

# Loop over timesteps and evaluate the current.
currents = []
for time in times:
    state.evolve(time)
    current = state.evaluate(current_operator)
    currents.append(current)

# Plot the normalized current vs. time.
plt.plot(times, currents / currents[-1])
plt.show()
```

$$\hat{\mathbf{H}}(t) = \sum_{i}^{N_s+1} \epsilon_i \hat{c}_i^\dagger \hat{c}_i - \sum_{-\infty}^{\infty} \hat{c}_{i+1}^\dagger \hat{c}_i - [e^{i\phi(t)} - 1]\hat{c}_1^\dagger \hat{c}_0 + \text{h.c.}$$

$$V(t) = \begin{cases} 0, & \text{for } t < 0 \\ \frac{V_\text{b}}{2}\left(1 - \cos\left(\frac{\pi t}{\tau}\right)\right), & \text{for } 0 \le t \le \tau \\ V_\text{b}, & \text{for } t > \tau \end{cases}$$

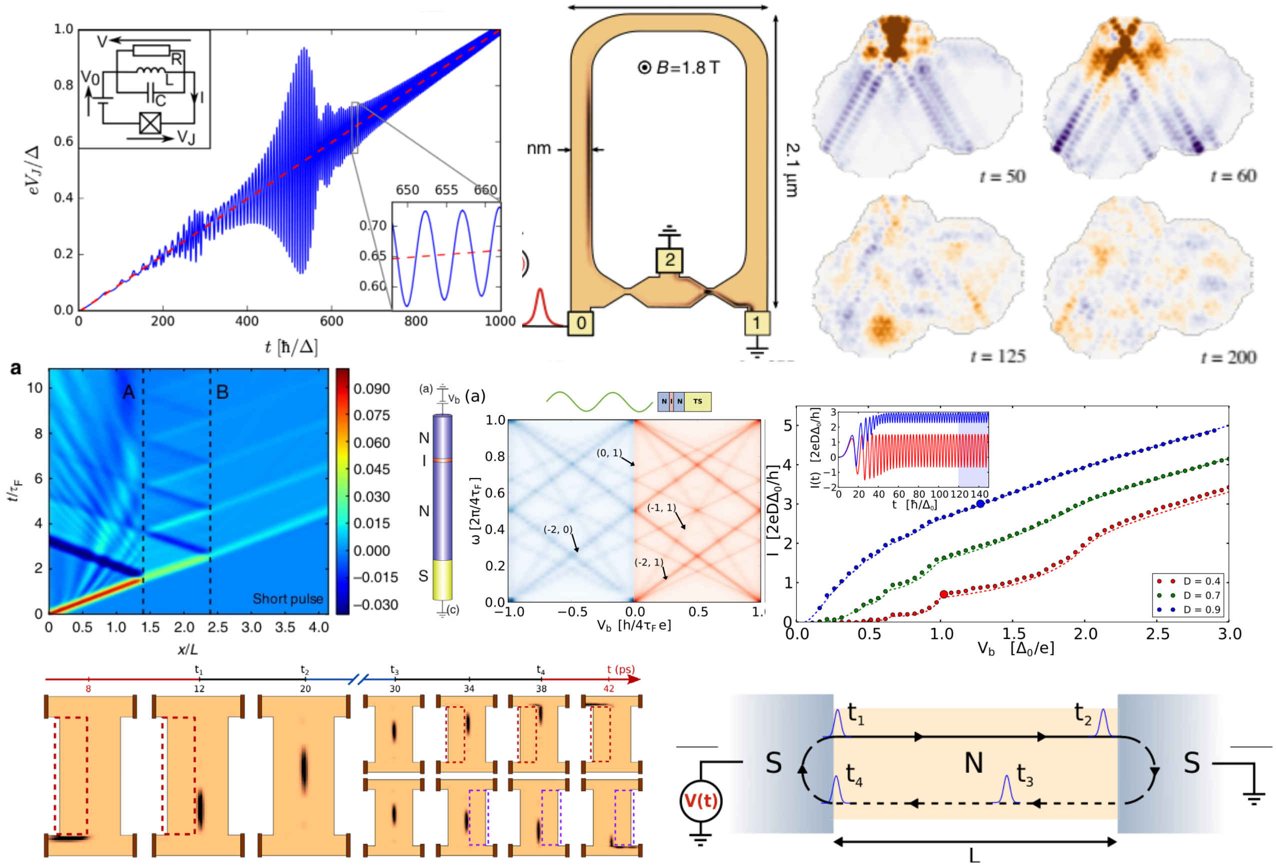
$$j_i(t) = \langle c_i^\dagger c_{i-1} \rangle (t)$$

actual time-dependent simulation



Script stays as close as possible to the analytical approach and to physical intuition.

Gallery of Tkwant examples



- *Mach-Zehnder and Fabry-Perot electronic interferometers*
- *Floquet topological insulator*
- *Quantum Hall regime*
- *Quantum noise*
- *Heat transport and thermoelectic effect*
- *Multiple Andreev reflections*
- *Skyrmion dynamics*
- *Graphene relaxation dynamics*
- *Plasmons and Luttinger liquids*

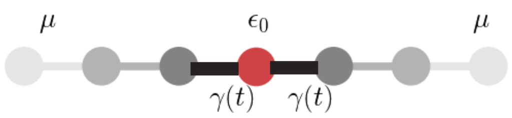
Green functions

Arbitrary Green functions can be calculated from the wavefunction

$$G^{<}_{ij}(t,t') = \sum_{\alpha} \int \frac{dE}{2\pi} i f_{\alpha}(E) \psi_{\alpha E}(t,i) \psi^{*}_{\alpha E}(t',j),$$

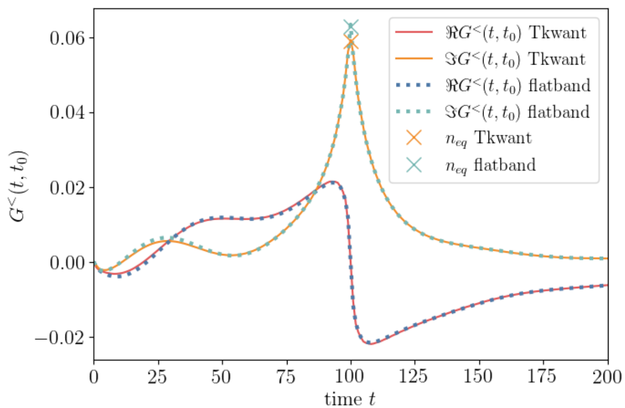
$$G^{R}_{ij}(t,t') = -i\theta(t-t') \sum_{\alpha} \int \frac{dE}{2\pi} \psi_{\alpha E}(t,i) \psi^{*}_{\alpha E}(t',j)$$

Example: Lesser Green function on an impurity after a sudden parameter change



$$H = -\sum_{i=-\infty}^{\infty} [\gamma_i(t) c^{\dagger}_{i+1} c_i + \text{h.c.}] + \epsilon_0 c^{\dagger}_0 c_0$$

$$\gamma_i(t) = 1, \gamma_0(t) = \gamma_1(t) = \gamma\theta(t)$$

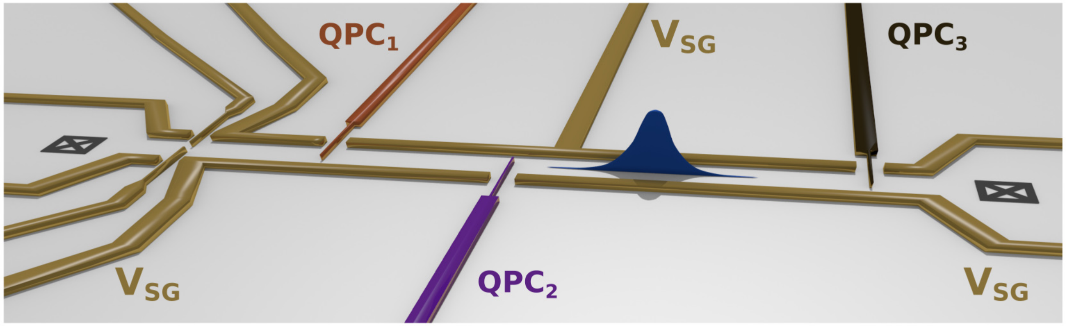


Analytical solution (flatband approximation)

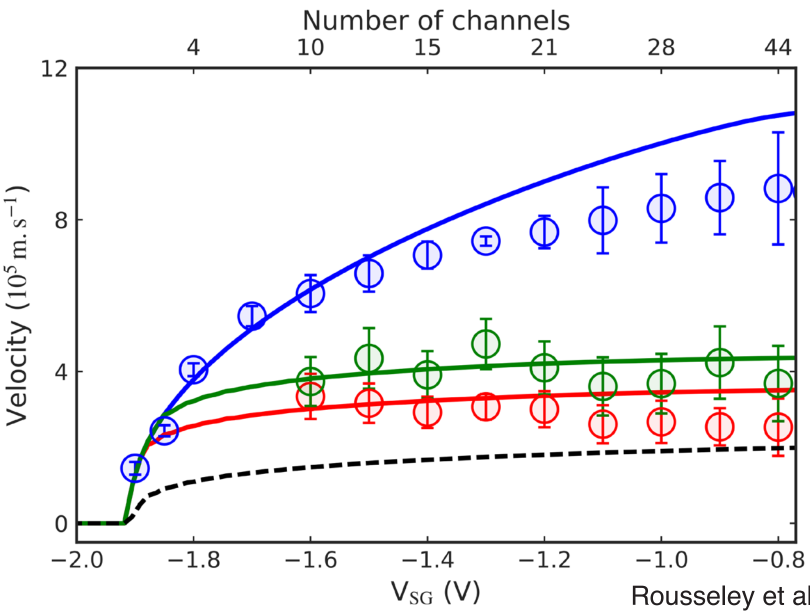
$$G^{<}_{00}(t,t') = \theta(t)\theta(t') \frac{i\Gamma}{\pi} e^{-i\epsilon_0(t-t')} \int_{-\infty}^{0} d\omega \frac{1}{(\omega-\epsilon_0)^2 + \Gamma^2} \left(e^{-i(\omega-\epsilon_0)t} - e^{-\Gamma t}\right) \left(e^{i(\omega-\epsilon_0)t'} - e^{-\Gamma t'}\right), \qquad \Gamma = 2\gamma^2$$

Example form Tkwant tutorial

Bosonic excitations in few-electron pulses



electrostatic DC simulation



Rousseley et al., Nat Comm, 9, 2811 (2018).

Interacting system behaves as Luttinger liquid.
Pulses travel faster with plasmon velocity, which is faster than the Fermi velocity.

Solving self-consistent equations

Hubbard-like Hamiltonian

$$H = \sum_{<ij>,\sigma} \gamma_{ij} c^\dagger_{i\sigma} c_{j\sigma} + U\theta(t) \sum_i c^\dagger_{i\uparrow} c_{i\uparrow} c^\dagger_{i\downarrow} c_{i\downarrow}$$

Mean-field decoupling

$$H^{\mathrm{HF}} = \sum_{<ij>,\sigma} \gamma_{ij} c^\dagger_{i\sigma} c_{j\sigma} + U \sum_{i,\sigma} \langle c^\dagger_{i\sigma} c_{i\sigma} \rangle (t) c^\dagger_{i\bar{\sigma}} c_{i\bar{\sigma}}$$

$$\langle c^\dagger_{i\sigma} c_{i\sigma} \rangle (t) = \sum_\alpha \int \frac{dE}{2\pi} f_{0,\alpha}(E) |\psi_{\sigma,\alpha,E}(i,t)|^2$$

A direct numerical implementation of above equations would be extremely inefficient!

Reason: very different timescales

$$\langle c^\dagger_i c_i \rangle (t) \sim d\tau \qquad \psi_\alpha(i,t) \sim dt$$

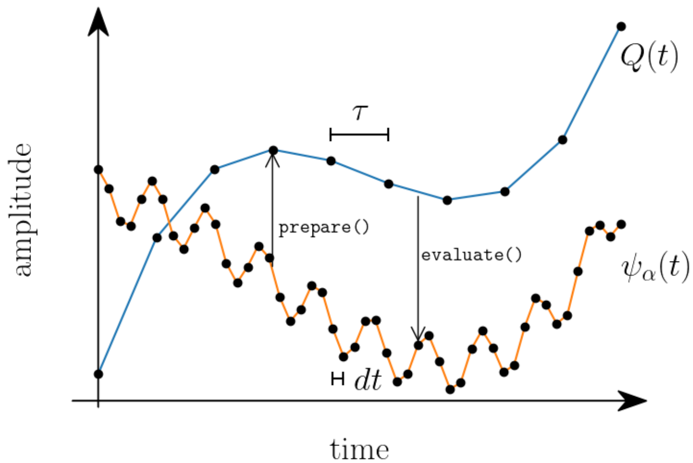
$$d\tau \gg dt$$

Generic solver for self-consistent equations with Tkwant

Efficient algorithm: extrapolate the mean-field potential Q(t)

$$\mathbf{H} = \mathbf{H}_0 + \mathbf{W}(t) + \mathbf{Q}[t, y(t)]$$

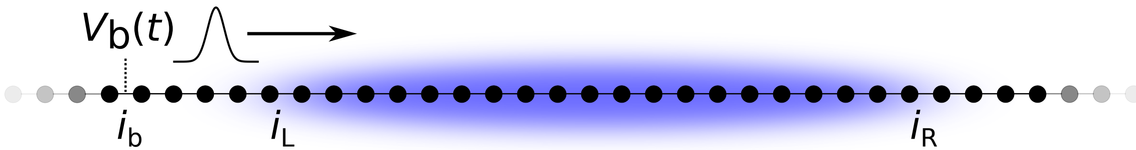
```python
class HartreePotential:
    def __init__(self, interaction_strength, density0):
        self._interaction_strength = interaction_strength
        self._density0 = density0

    def prepare(self, density_func, tmax):
        self._density = density_func

    def evaluate(self, time):
        diag = (self._density(time) - self._density0) * self._interaction_strength
        return scipy.sparse.diags([diag], [0], dtype=complex)

density_operator = kwant.operator.Density(syst)

density0 = wave_function.evaluate(density_operator, root=None)

hartree_potential = HartreePotential(interaction_strength=1, density0=density0)

sc_wavefunc = tkwant.interaction.SelfConsistentState(wave_function,
                                                     density_operator,
                                                     hartree_potential)

for time in times:
    sc_wavefunc.evolve(time)
    density = sc_wavefunc.evaluate(density_operator)
    plt.plot(sites, density)
```



Tkwant performs the extrapolation and error estimate. The efficient algorithm is enforced by the interface.

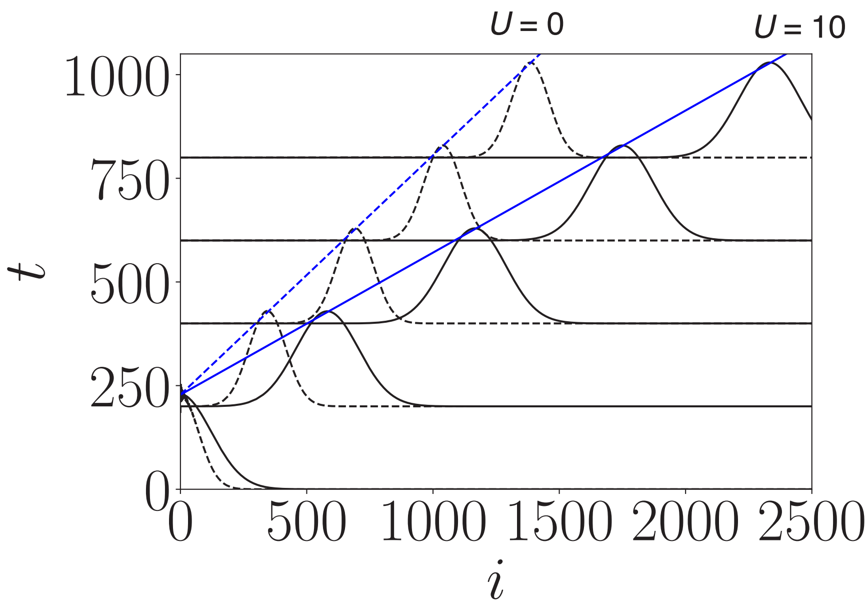
Plasmon velocity simulations

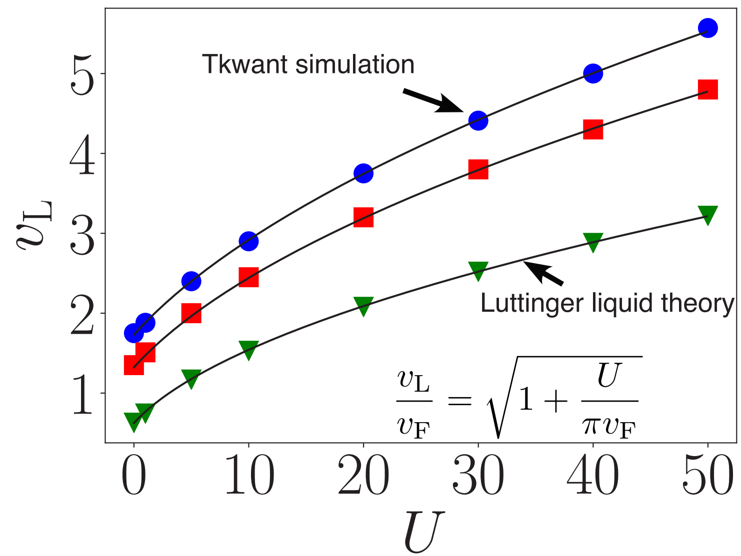
$$\hat{H}(t) = \sum_{<ij>,\sigma} \gamma_{ij} c^\dagger_{i\sigma} c_{j\sigma} + \sum_{i\sigma} V_\mathrm{b}(t)\theta(i_\mathrm{b} - i) c^\dagger_{i\sigma} c_{i\sigma} + U \sum_i s(i)(c^\dagger_{i\uparrow} c_{i\uparrow} - n_0)(c^\dagger_{i\downarrow} c_{i\downarrow} - n_0)$$



Self-consistent Hartree decoupling

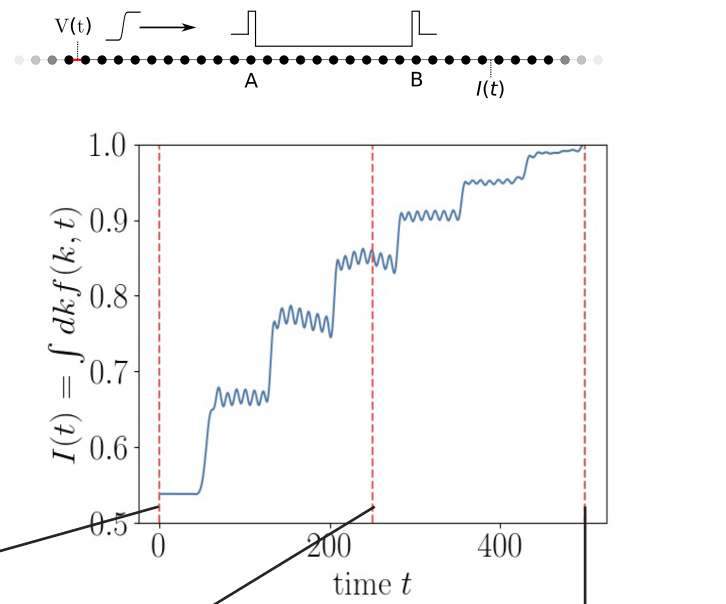
$$\hat{H}_\mathrm{HF} = U \sum_i s(i) c^\dagger_i c_i \left[\langle c^\dagger_i(t) c_i(t) \rangle - n_0 \right]$$





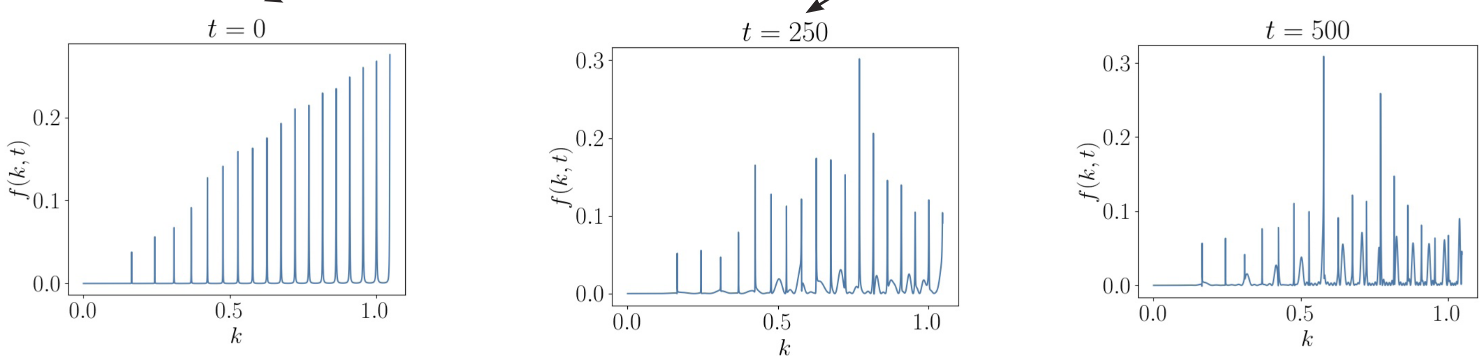
TK, Weston and Waintal, PRB (2017).
Matveev and Glazman, Physica B (1993).

Manybody-integral - ongoing works



Problems

- precision and time limit due to complicated integrand
- integrand change with time
- add new integrand points only at initial time



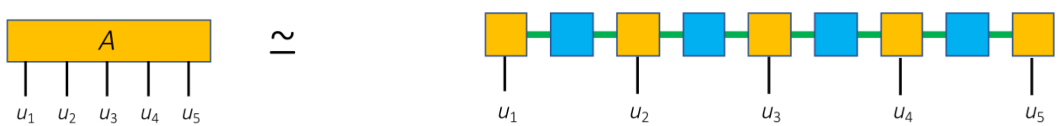
Tensor factorization for functions (Quantics)

Represent a function discretized on n points on a bitstring of length d, where $n = 2^d$

Example $n = 8$: $f(x_i) \longrightarrow f(u_1, u_2, u_3)$

i	$u_1\ u_2\ u_3$
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Factorize the resulting tensor using cross factorization methods



number of evaluations:

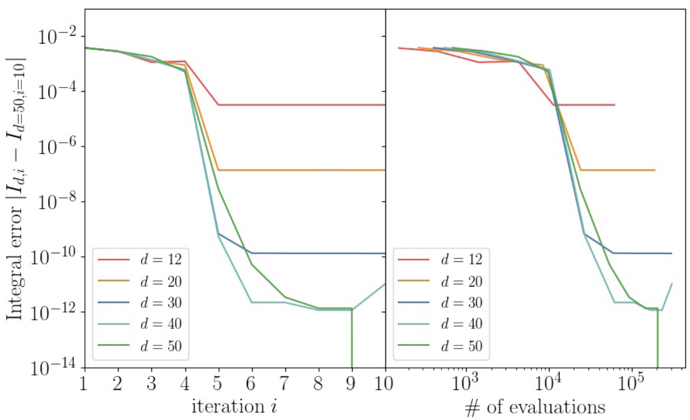
2^d $\qquad$ $2\ d\ \chi 2$

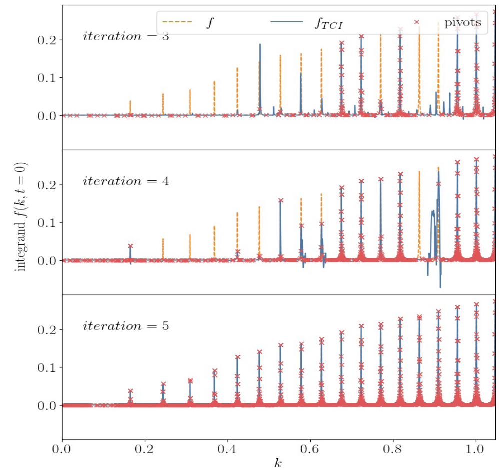
Factorized tensor can be trivially integrated.

Khoromskij, Constr. Approx **34**, 257 (2011).
Fernández, Jeannin, Dumitrescu, TK, Kaye, Parcollet, and Waintal,PRX. **80**, 041018 (2022).

Quantics for Tkwant

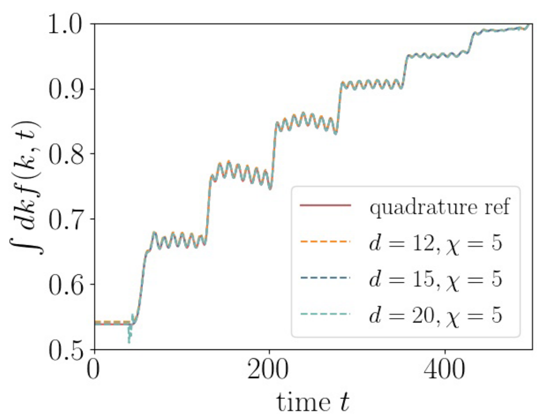
First step: Learn the integrand at the initial time *t*=0





2^{50} (= 10^{15} for direct evaluation) points resolution is at the order of machine precision!

Second step: Reuse lerned evaluation points at later times *t* to calculate the integral

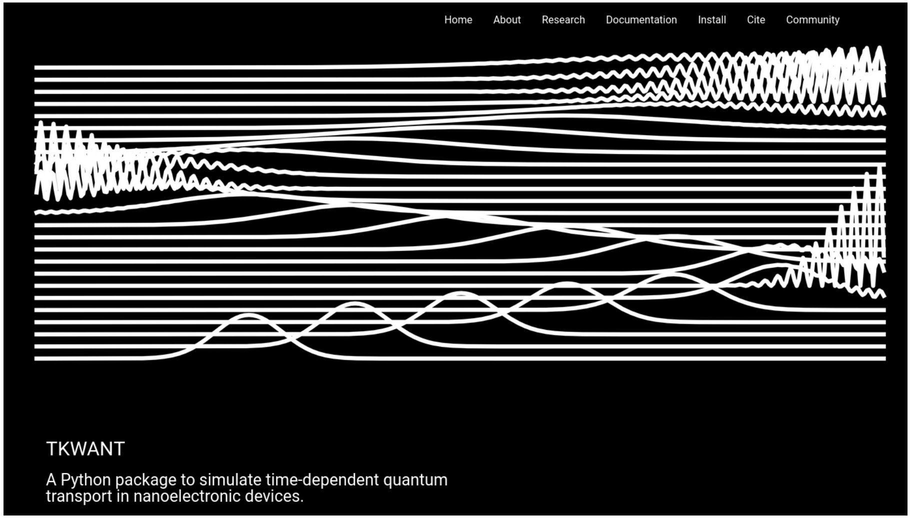


Summary of Tkwant

- simulation many-body observables for generic tight-binding system in a transient regime
- currents, densities, Green functions or arbitrary user defined observables
- generalization for self-consisten problems

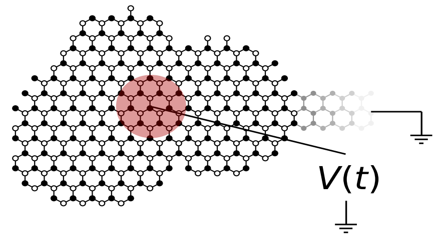
Website with documentation and tutorial:

https://tkwant.kwant-project.org/



Numerical algorithm of Tkwant

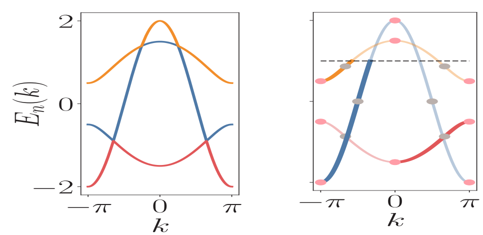
- Define the tight-binding Hamiltonian (arbitrary geometry, dimension, couplings)



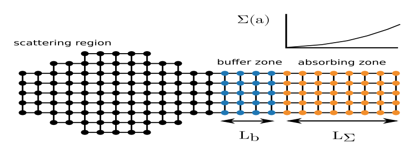
- Calculate time-independent scattering states

$$\mathbf{H}(t=0)\psi_{\alpha E} = E\psi_{\alpha E}.$$

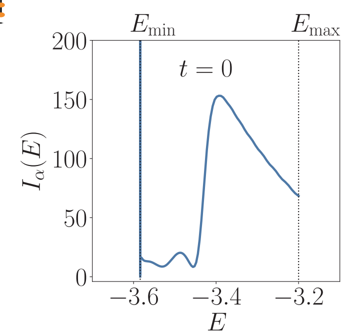
- Band structure analysis



- Devise boundary conditions



- Solve n time-dependent Schrödinger equations

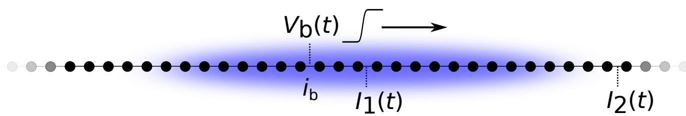


- Evaluate the manybody integral

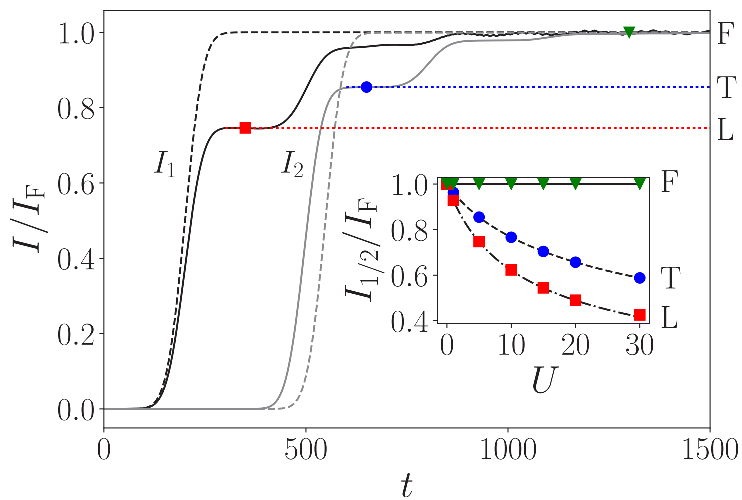
$$n_i(t) \equiv \langle \hat{c}_i^\dagger \hat{c}_i \rangle (t) = \sum_\alpha \int \frac{dE}{2\pi} f_\alpha(E) |\psi_{\alpha E}(t, i)|^2$$

Conductance - transient behaviour

semiclassical Boltzmann theory



$$\partial_t f = -v_k \partial_x f - F(x,t)\partial_k f$$



non-interacting leads $g = g_F$

Safi and Schulz, PRB (1995).

non-interacting/interacting lead

TK, Weston and Waintal, PRB (2017).

$$\frac{1}{g_T} = \frac{1}{2}\left[\frac{1}{g_L} + \frac{1}{g_F}\right]$$

interacting leads $g = g_L$

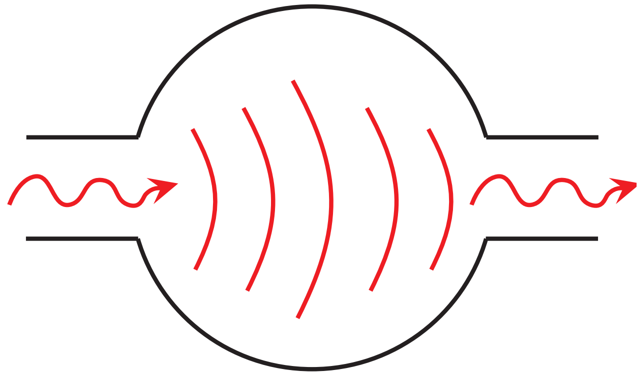
Matveev and Glazman, Physica B (1993).

generalized contact (Sharvin) resistance

$$\frac{1}{g} = \frac{1}{2}\left[\frac{1}{g_{L1}} + \frac{1}{g_{L2}}\right]$$

Table of content

Transient resistance in Luttinger liquids



Luttinger like conductance

$$\frac{g_{\mathrm{L}}}{g_{\mathrm{F}}} = \left(1 + \frac{U}{\pi v_{\mathrm{F}}}\right)^{-1/2}$$

exact bosonization for *N*-channel quasi-1d quantum wire, Matveev and Glazman, Physica B (1993).

but:

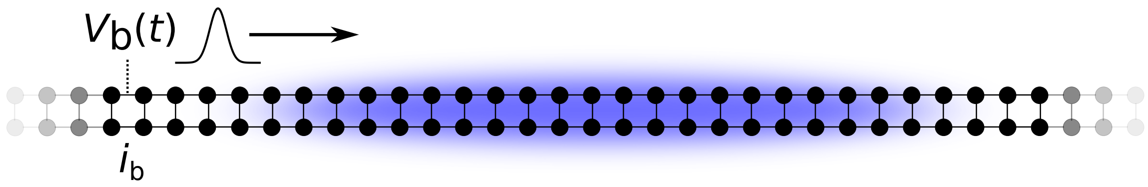
non-interacting leads will wash out the effect (Safi, Schulz, PRB 1995)

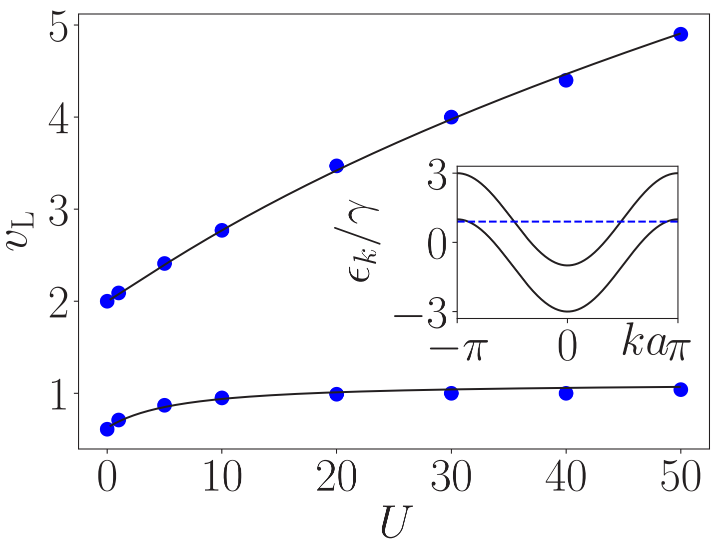
$$g_{\mathrm{F}} = e^2/h$$

Table of content

Plasmon velocities for *N*-channels

pulse propagation in a quasi one-dimensional quantum wire with $N = 2$



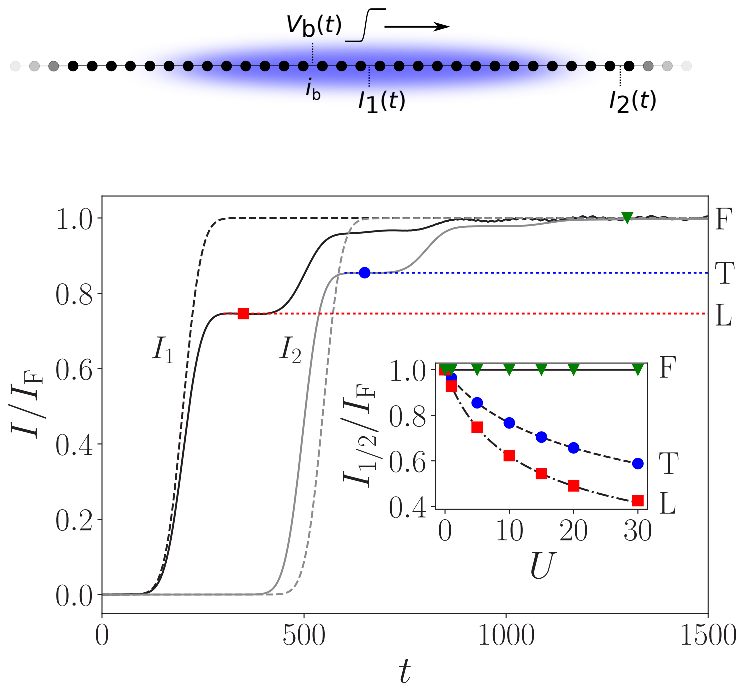


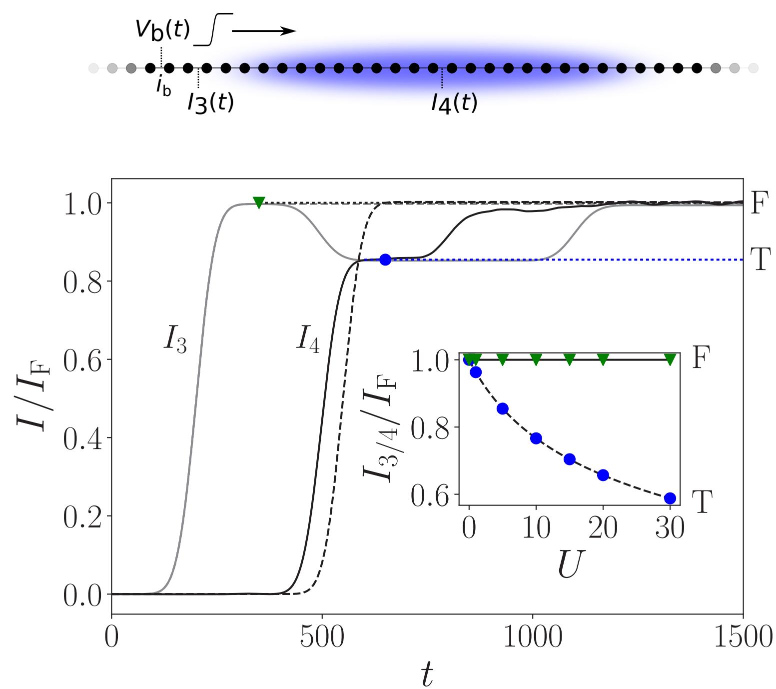
generalized expression for the plasmon velocities in presence of *N*-channels

$$1 = \sum_{\alpha=1}^{N_{ch}} \frac{U}{W\pi} \frac{|v_\alpha|}{v_L^2 - v_\alpha^2}$$

Matveev and Glazman, Physica B (1993).

Conductance - transient behaviour





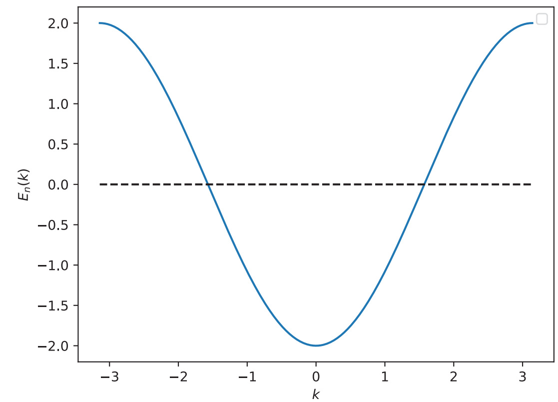
F: Fermi (non-interacting)
T: transient
L: Luttinger

Wave function formalism (equivalent to Keldysh)

Evolve all eigenstates below the Fermi energy with the time-dependent Schödinger equation

$$i\partial_t \psi_{\alpha E}(t,i) = \sum_j \mathbf{H}_{ij}(t)\psi_{\alpha E}(t,j),$$

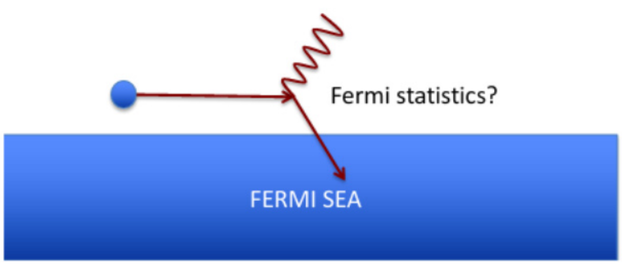
$$\psi_{\alpha E}(t < t_0, i) = \psi_{\alpha E}(i)e^{-iEt}$$



Calculate observables

$$n_i(t) \equiv \langle \hat{c}_i^\dagger \hat{c}_i \rangle(t) = \sum_\alpha \int \frac{dE}{2\pi} f_\alpha(E)|\psi_{\alpha E}(t,i)|^2$$

$$f_\alpha(E) = \frac{1}{e^{(E-\mu_\alpha)/k_B T_\alpha} + 1}$$



Gaury, Weston, Santin, Houzet, Groth and Waintal, Phys. Rep. 534, 1 (2014).