Exascale Framework for Modeling Quantum Transport through Nanodevices

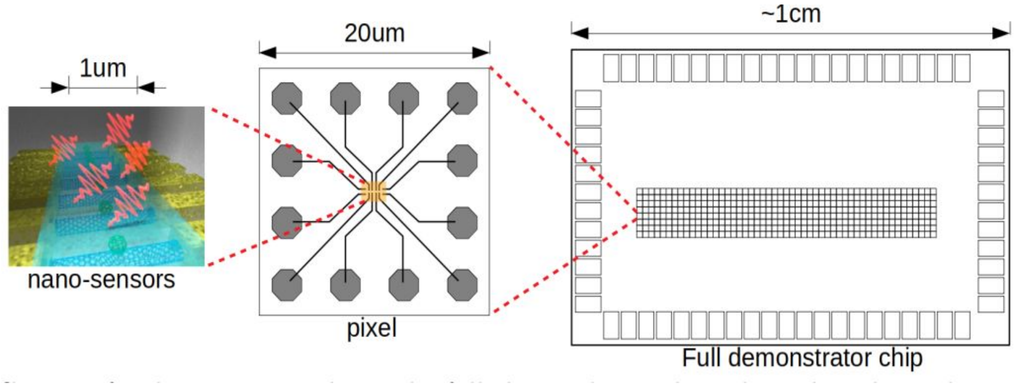
Saurabh S. Sawant

Postdoctoral Scholar, Microelectronics group,
Center for Computational Sciences and Engineering,
AMCRD, LBNL, CA.

Project Collaboration Meeting, Jan 21st, 2025

We need a capability to model charge transport through experimentally relevant 3D nanodevices.

DOE-project to build CMOS chip for photon detection, using nano-sensors made of carbon nanotubes.



Existing tools have limited GPU-support, do not model multiple nanomaterials, and use simplifications.

ELEQTRONeX

https://github.com/ AMReX-Microelectronics/ELEQTRONeX

AMReX

Exascale Computing Project

GPU-capability

Portability

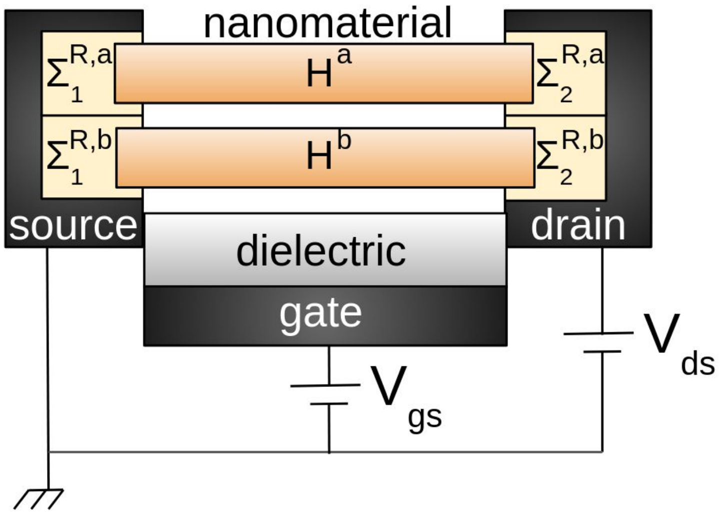
High-scalability

Open-source

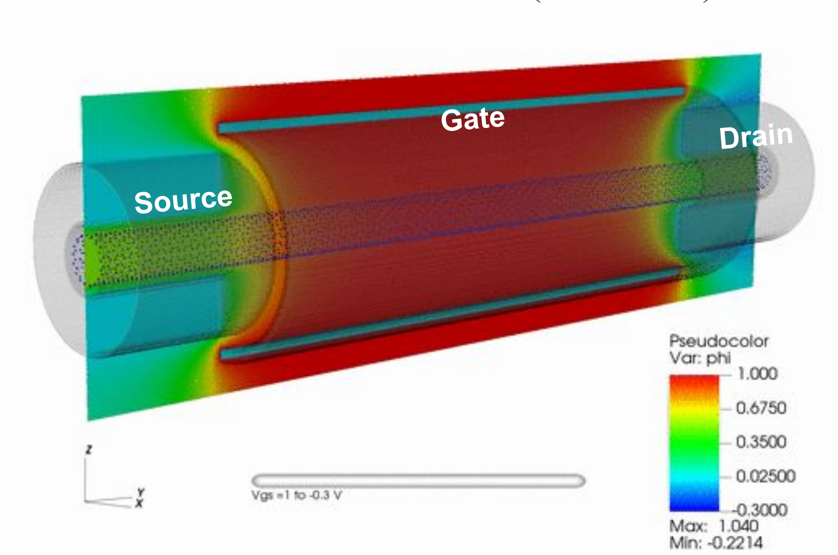
github.com/AMReX-Codes/amrex

ELEQTRONeX can model multi-channel CNTFETs with complex shapes of contact geometries, material representation in real-space.

Schematic of a multi-channel field effect transistor



E.g. a gate-all-around **Carbon nanotube** field effect transistor (CNTFET)



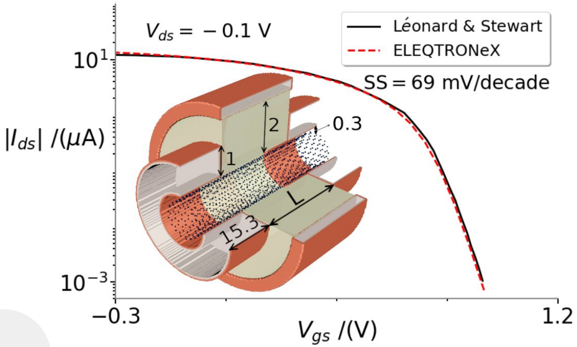
H (Hamiltonian matrix) – describes the material.

Σ^R (self-energy matrices) – boundary conditions for material-metal contacts.

V_{gs} (gate-source voltage) = 1 V to -0.3 V

V_{ds} (source-drain voltage) = -0.1 V

We use a self-consistent approach to model quantum transport.



Set voltages: gate-source V_{gs} , drain-source V_{ds}, **assume surface charge density**, ρ^{in}

Electrostatics

get potential, φ

Nonequilibrium Green's Function (NEGF) method

get charge density, ρ^{out}

Broyden's method

to minimize, ρ^{in}- ρ^{out}

Is $\rho^{in} \approx \rho^{out}$?

No

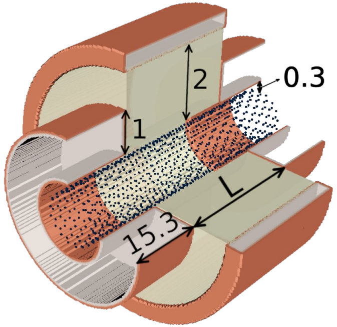
Yes

Compute current, conductance, etc.

Electrostatics module leverages AMReX's multigrid linear solvers & HYPRE by LLNL.

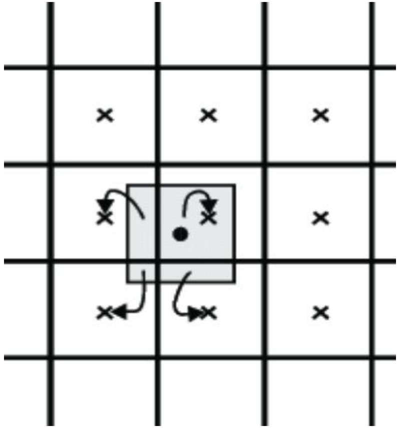
We solve **Poisson equation**:

$$\nabla \cdot (\epsilon \nabla U) = -e(\rho^{in} - \rho_0)$$



AMReX's 3D **embedded boundary** (EB) feature, customized here for multiple EBs with different voltages.

Used AMReX **particle data structure** for atomic representation in real space.



Implemented **cloud-in-cell** algorithm for

Gathering φ from cell-centered mesh to atom locations

Depositing ρ^{out} from atom to cell-centered mesh

Core computations in the nonequilibrium Green’s function method involve:

Computing retarted Green’s function at each E:

$$\boldsymbol{G}^R(E) = \left[(\underset{\text{band energy}}{E} + \underset{\text{small number}}{i\eta})\boldsymbol{I} - \underset{\text{Hamiltonian}}{\boldsymbol{H_0}} - \underset{\text{potential}}{eU} - \sum_{l}^{L\,\text{leads}} \underset{\text{self energy}}{\boldsymbol{\Sigma}_l^R} \right]^{-1}$$

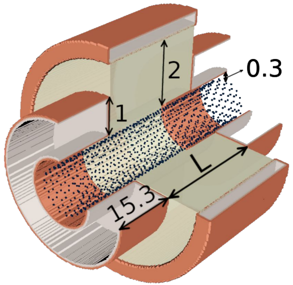
Integrating ***G***

Integrating $\boldsymbol{G}^R(E)$ to compute charge density matrix:

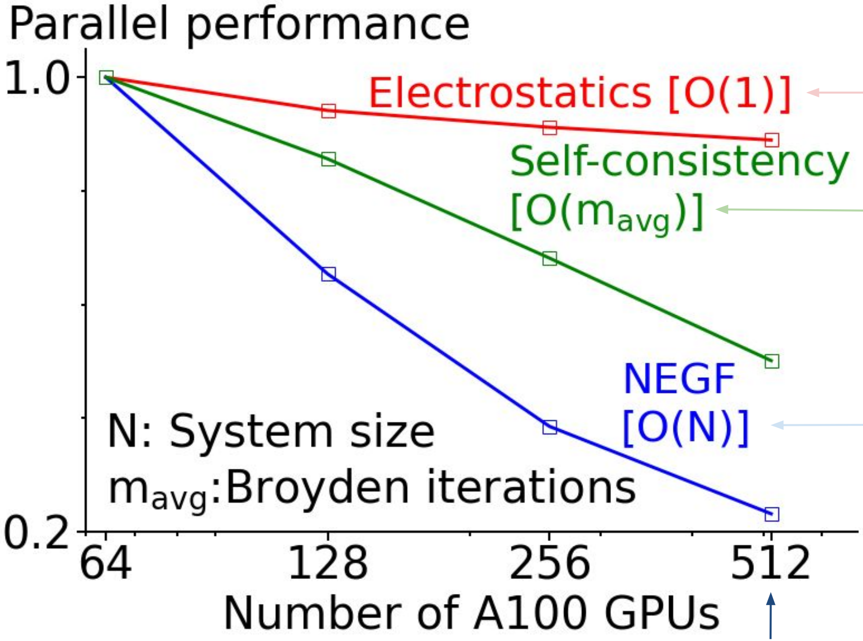
$$\boldsymbol{\rho} = \frac{1}{\pi}\Im\left[\int_{-\infty}^{E_{min}} \boldsymbol{G}^R \overset{\text{Fermi function}}{F_{min}}\, dE\right] - \frac{1}{2\pi}\int_{E_{min}}^{E_{max}} \sum_l \overset{\text{spectral function } \boldsymbol{A}\text{, depends on } \boldsymbol{G}^R}{\boldsymbol{A}_l} F_l \, dE$$

We use tight binding Hamiltonian, semi-infinite contacts (decimation technique for surface Green’s function), object oriented software design to facilitate adding different materials.

Good parallel performance enabled simulation of longer nanotubes.



dimensions in nm



Serial time complexity

O(N)

O(Nm_{avg})

O(N^2)

400 nm channel length

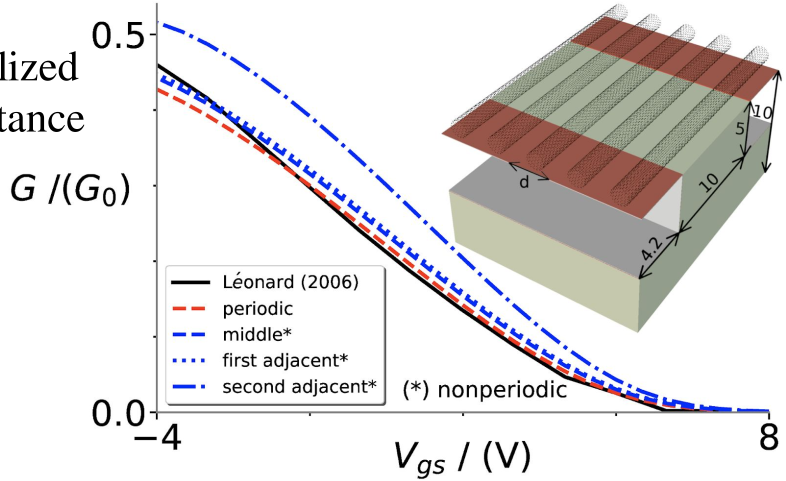
Other approaches parallelize $G^R(E)$ calculations across independent E-points,

requiring serial NEGF algorithms & storage of G^R, A on each processor.

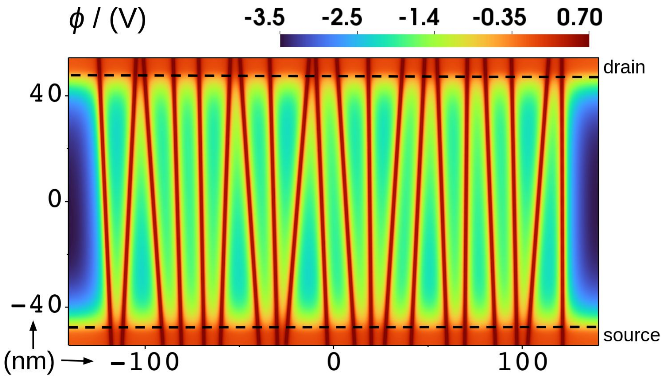
Key finding 1: CNT misalignment and pitch variations have minimal impact on conductance.

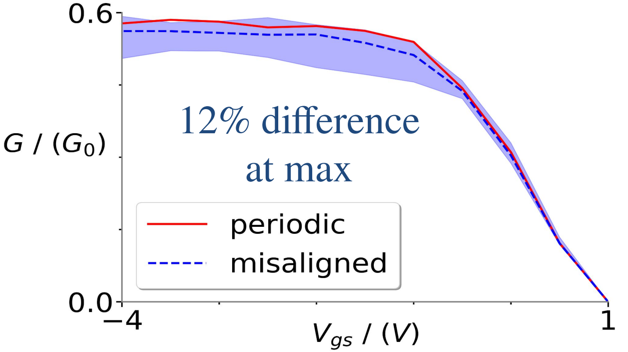
Planar setup[1]
(periodic *vs.* only 5 CNT.)

Normalized conductance



20 CNTs with misalignment[2]
*(**3 mins** per V_{gs} point, 512 GPUs)*



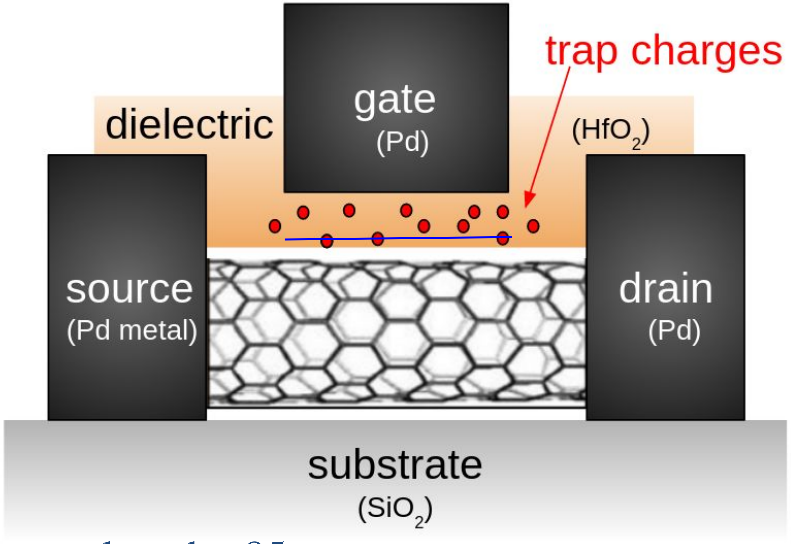


12% difference
at max

[1] F. Léonard, Nanotechnology 17 (9) (2006) 2381.
[2] **S. S. Sawant**, F. Leonard, Z. Yao, A. Nonaka *(under review)* arXiv: https://arxiv.org/abs/2407.14633

Experiments report degraded performance (4x larger SS). Trapped charges are a possible culprit.

Schematic of a top gate configuration



gate length= 85 nm
channel length= 95 nm
gate oxide thickness= 5 nm

Experiment: Lin, Y. et al. (2023). *Nature Electronics*, 6(7), 506-515.

Traps modeled as point charges
with potential-dependent occupation:

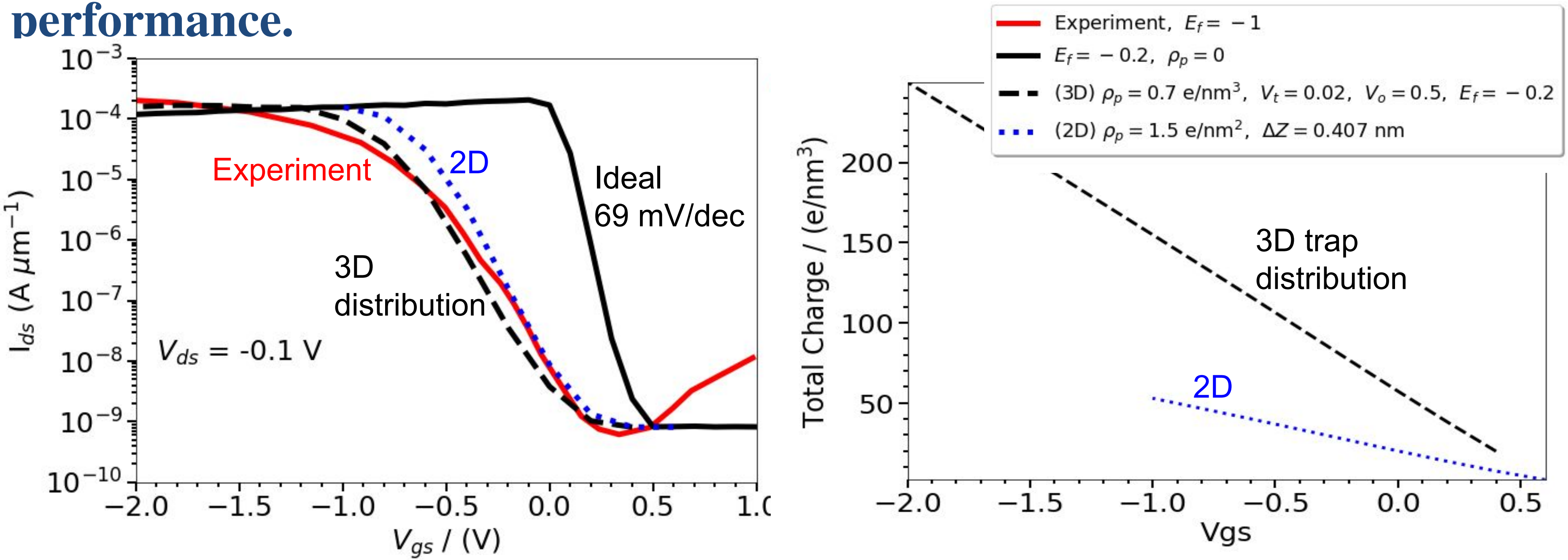
$$Q_i = e\,\text{sigmoid}\left(\frac{eV_o - eV_i}{eV_t}\right)$$

$V_i = V(\mathbf{r}_i)$ Potential at the trap

$eV_o =$ Trap energy level

$V_t =$ Parameter to smoothen transition between empty and filled trap.

Key finding 2: Trap charges indeed detrimentally affect performance.



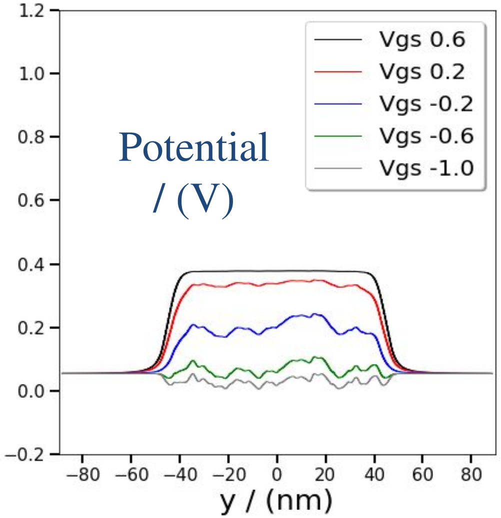
3D trap density of **0.7 e/nm^3 = 750 traps/CNT.** However, many of these traps are inconsequential.

2D trap density of **1.5 e/nm^2 = 320 traps/CNT.** Only **~50 effective traps** are close enough to matter.

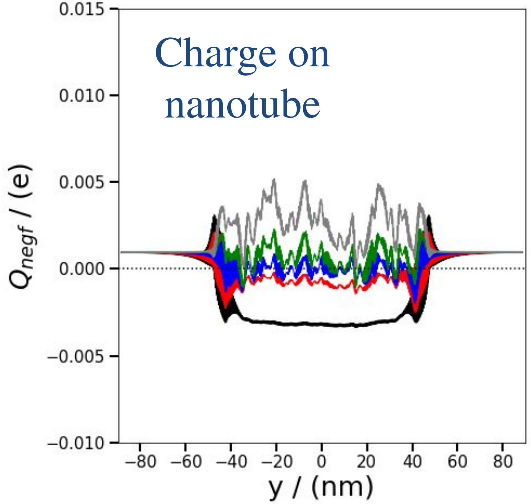
Assuming trap states distributed over 1 eV, the density of trap states from 2D distribution would be ~O(10^{13} /cm^2/eV). Experimental estimates ~10^{12} /cm^2/eV using capacitance measurement.

Variations in band structure caused by variations in the surface potential due to traps.

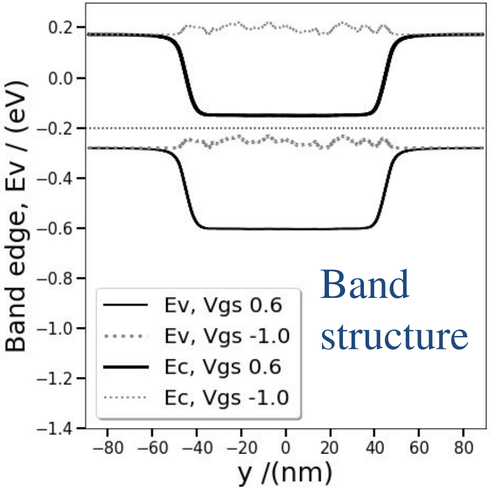
Data for 2D trap charge distribution (1.5 e/nm^2).



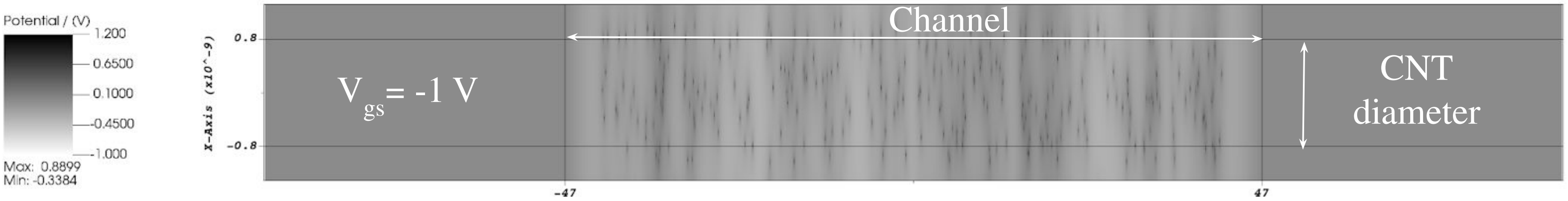
Potential / (V)



Charge on nanotube



Band structure



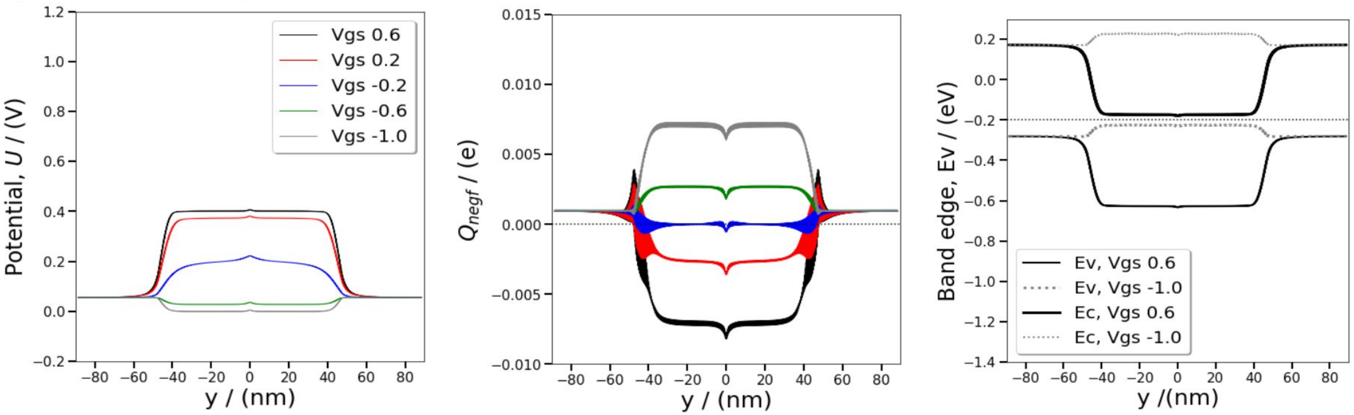
Channel

V_{gs} = -1 V

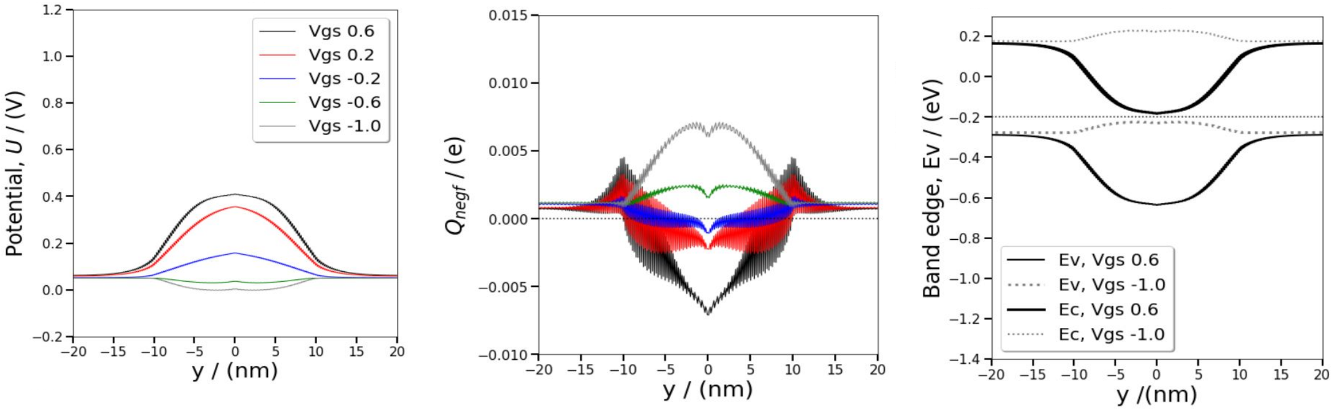
CNT diameter

Single trap charge doesn’t affect SS.

Data for
L=95 nm,
Trap **0.3 nm** above CNT,
CNT-GO gap=0.3 nm
Ef=-0.2 eV,
Vds=-0.1 V



Data for
L=10 nm,
Trap **0.1 nm** above CNT,
CNT-GO gap=0.1 nm
Ef=-0.2 eV,
Vds=-0.1 V



Future Work

- **Enhance accuracy of electron transport code.**
  - Extend the model to include electron scattering due to e-ph interactions, specifically addressing both acoustic and optical phonons.
  - Parallel effort underway to actively model phonon transport (under Alp's project) Perhaps we may be able to do detailed coupling of electron-phonon transport.

- **Model electron-photon interactions to maximize light absorption in 2D materials.**
  - Expand the code to model 2D materials and electron-photon interactions using the tight-binding approximation. Some references for guidance:

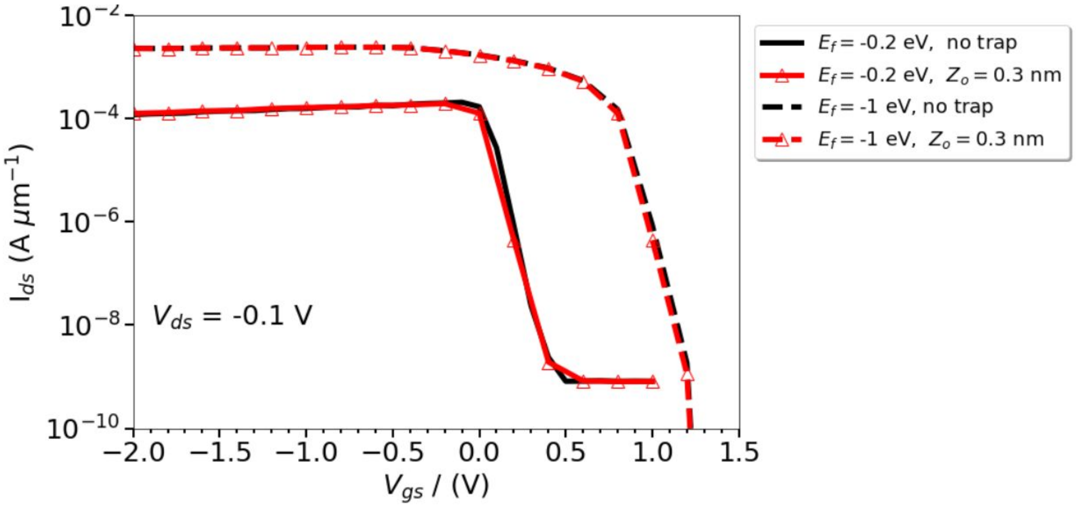
    [1] Hussein et al 2019 **Modeling electron-photon interaction in monolayers of graphene and TMDs in tight binding approximation.** J Phys.: Conf. Series 1368 022012. (Didn't use NEGF)

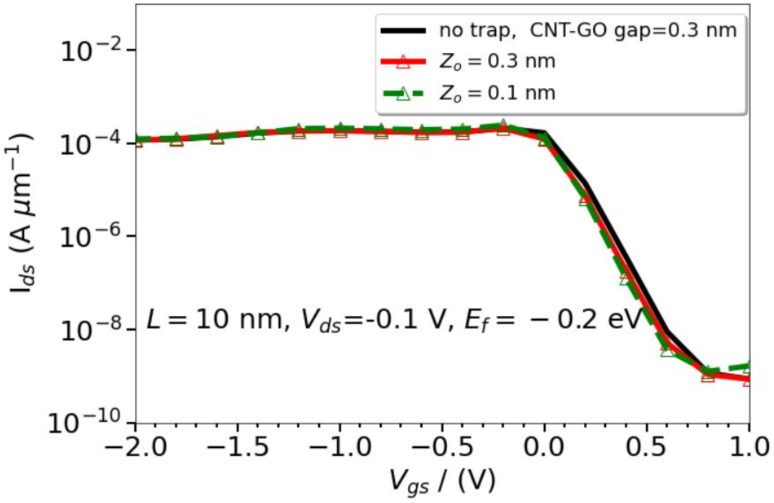
    [2] Zhang et al. (2014) **Generation and transport of valley-polarized current in transition-metal dichalcogenides.** (2014) Phys. Rev. B. 90, 195428. (They used NEGF-DFT for obtaining Hamiltonian)

- **Machine learning for predicting trap charge density in gate oxides**
  - Construct a predictive model for trap charge density in gate oxide based on geometric parameters and subthreshold swing. Beneficial to community.
  - Compact model.

Backup Slides

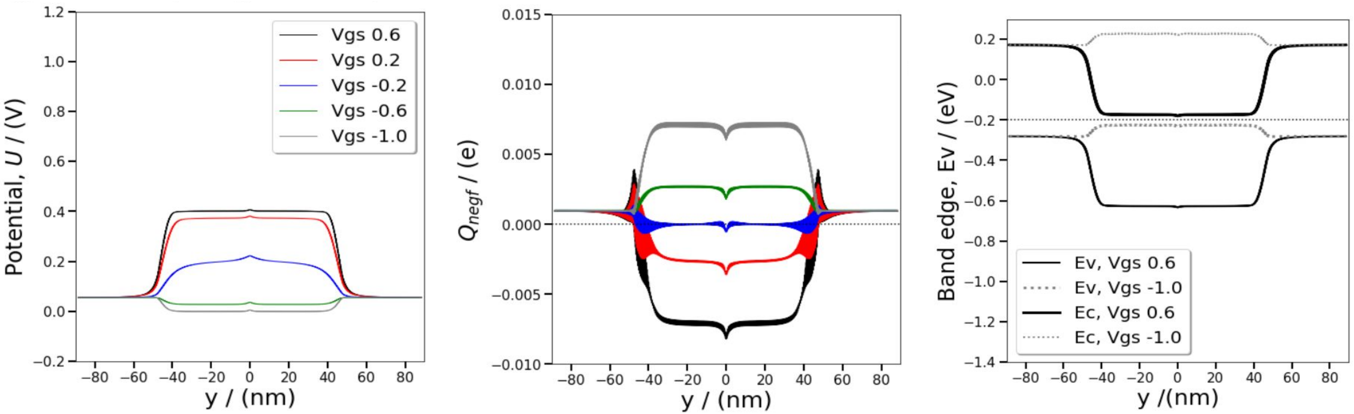
I-V Characteristics Corresponding to Cases for Scaling Studies



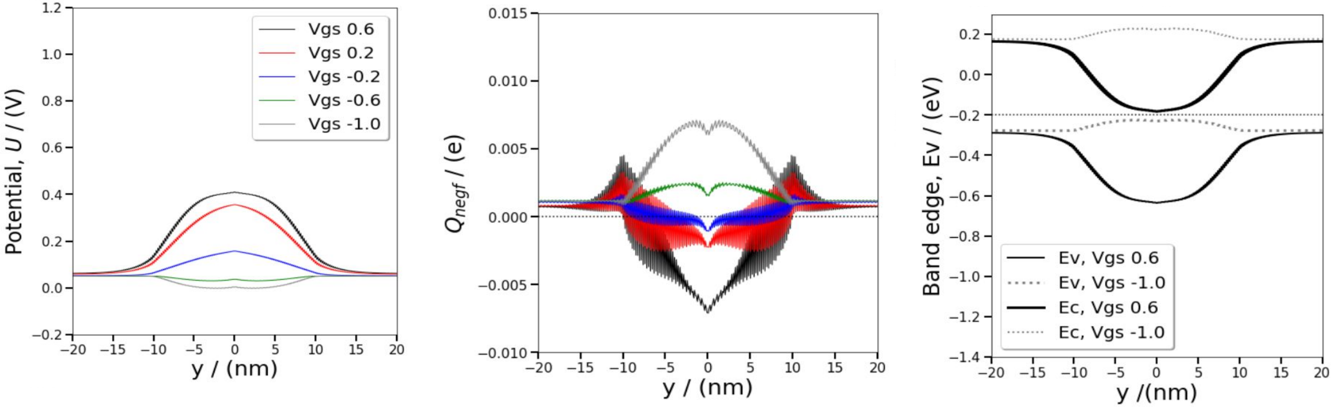


Single trap charge doesn’t affect SS.

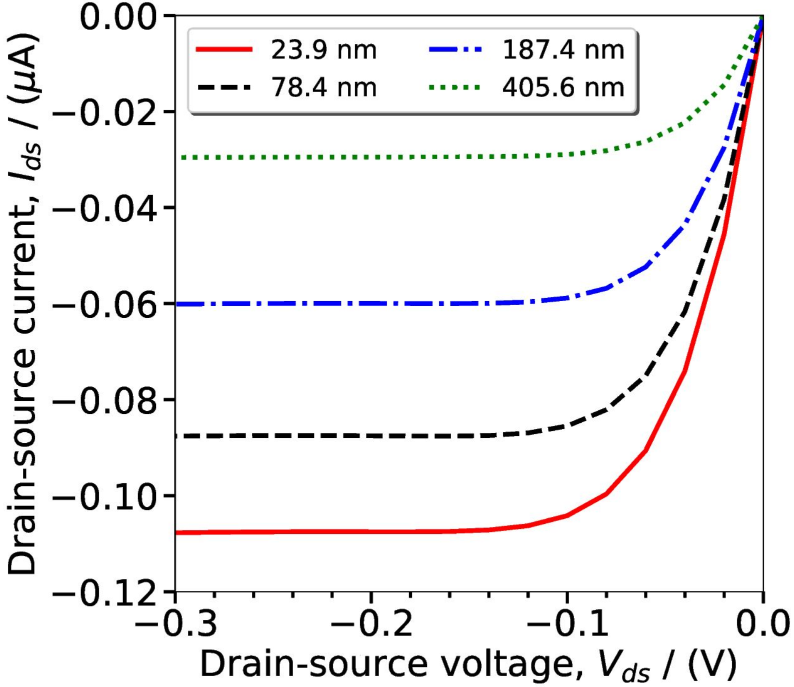
Data for
L=95 nm,
Trap **0.3 nm** above CNT,
CNT-GO gap=0.3 nm
Ef=-0.2 eV,
Vds=-0.1 V



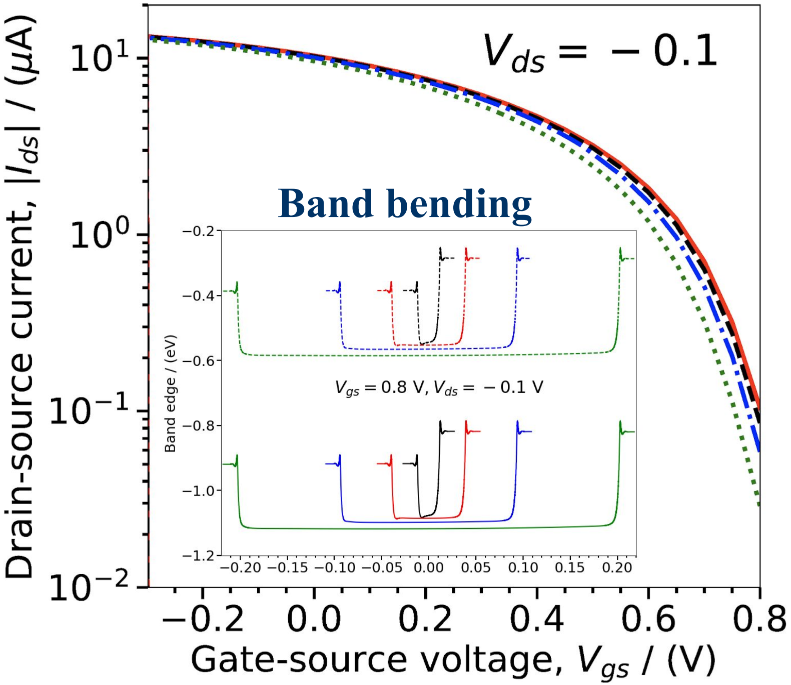
Data for
L=10 nm,
Trap **0.1 nm** above CNT,
CNT-GO gap=0.1 nm
Ef=-0.2 eV,
Vds=-0.1 V



I-V Characteristics Corresponding to Cases for Scaling Studies



I_{ds} versus V_{ds}



$|I_{ds}|$ versus V_{gs}

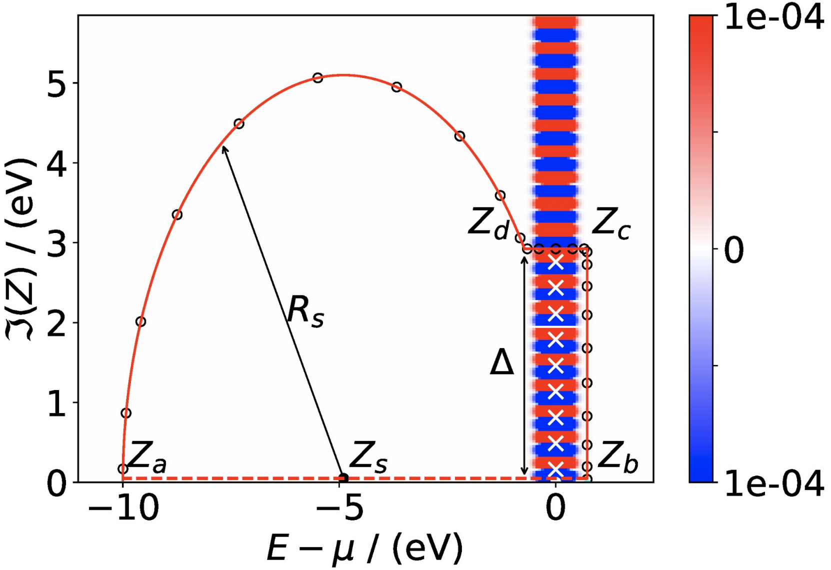
Band bending

Note: These results may change with inclusion of electron-phonon coupling, a factor not considered in this study.

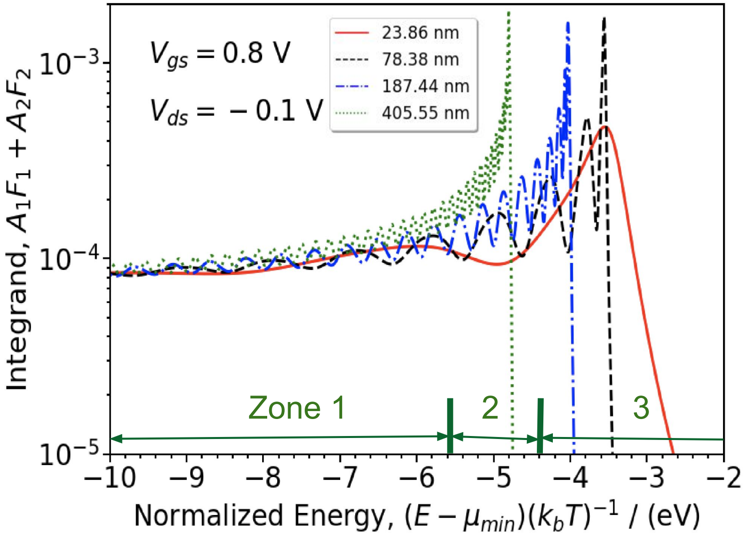
First essential optimization: reduce the number of E-points

$$\boldsymbol{\rho} = \frac{1}{\pi}\Im\left[\int_{-\infty}^{E_{min}} \boldsymbol{G}^R F_{min}\, dE\right] - \frac{1}{2\pi}\int_{E_{min}}^{E_{max}} \sum_l \boldsymbol{A}_l F_l\, dE$$

Residue theorem for 1st integral
& Gauss-Legendre quadrature.



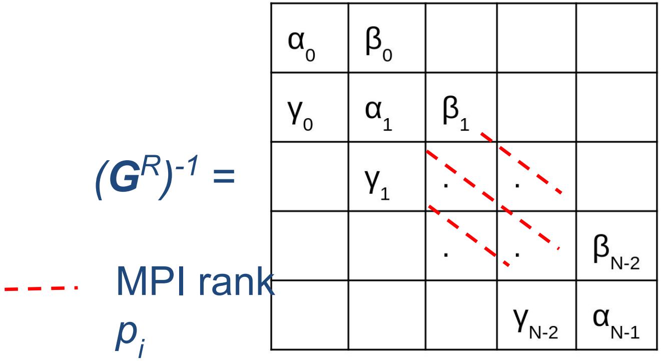
A Semi-adaptive scheme for 2nd integral



Code is customizable to handle different materials.

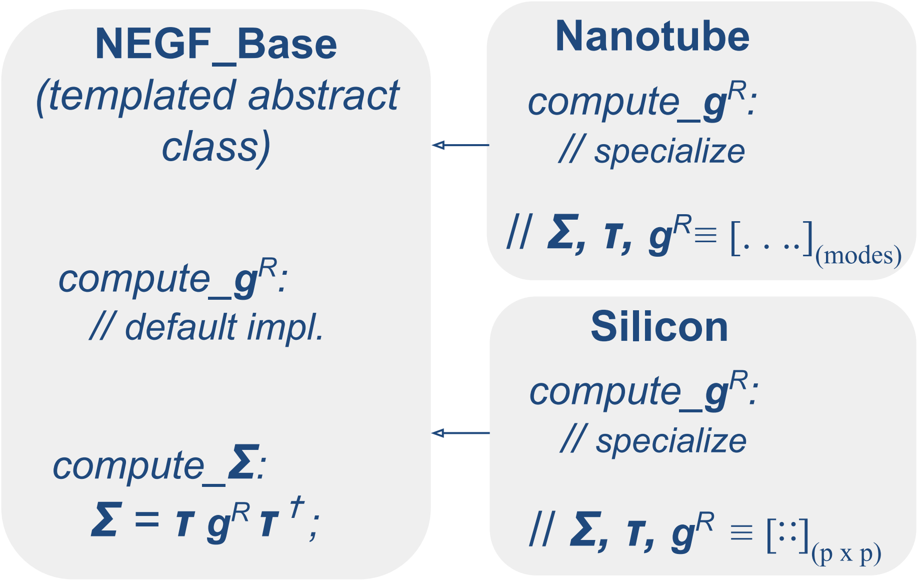
$$\boldsymbol{G}^R(E) = \left[(E + i\eta)\boldsymbol{I} - \boldsymbol{H_0} - eU - \sum_{l}^{L} \boldsymbol{\Sigma}_l^R \right]^{-1}$$

Block tridiagonal form



$$(G^R)^{-1} = \begin{array}{|c|c|c|c|c|} \hline \alpha_0 & \beta_0 & & & \\ \hline \gamma_0 & \alpha_1 & \beta_1 & & \\ \hline & \gamma_1 & . & . & \\ \hline & & . & . & \beta_{N-2} \\ \hline & & & \gamma_{N-2} & \alpha_{N-1} \\ \hline \end{array}$$

- - - MPI rank p_i



NEGF_Base
(templated abstract class)

compute_g^R:
// default impl.

compute_Σ:
$\Sigma = \tau\, g^R \tau^{\dagger}$;

Nanotube

compute_g^R:
// specialize

$// \ \Sigma, \tau, g^R \equiv [\,.\ .\ .\ .]_{(\mathrm{modes})}$

Silicon

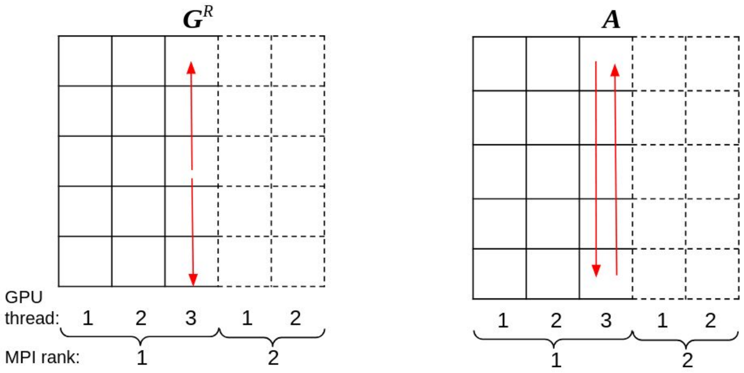
compute_g^R:
// specialize

$// \ \Sigma, \tau, g^R \equiv [::]_{(\mathrm{p\ x\ p})}$

α, β, γ can be a number, array,
or a submatrix
achieved using templates, virtual functions, operator overloading.

G^R and A is obtained using a MPI/GPU parallelized block-tridiagonal matrix inversion algorithm. *(a two step process)*

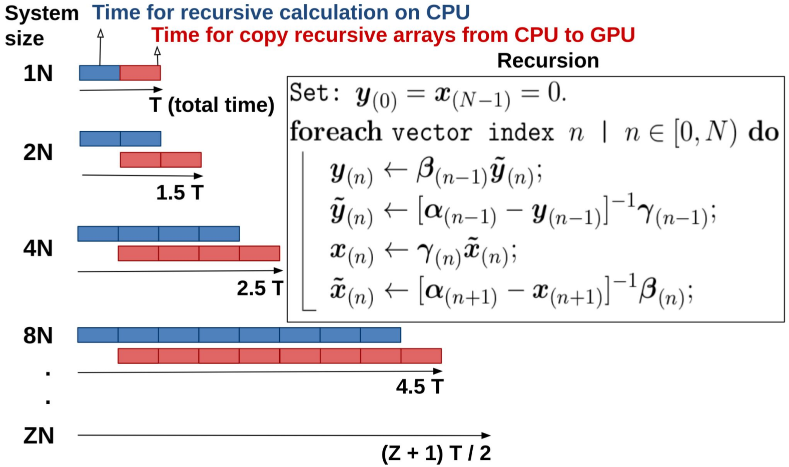
Parallel part: computing each column independently



```
foreach GPU thread $t$ | $t \in [0, N)$ do
    $\boldsymbol{G}^R_{(t,t)} \leftarrow [\boldsymbol{\alpha}_{(t)} - \boldsymbol{x}_{(t)} - \boldsymbol{y}_{(t)}]^{-1}$;
    $\boldsymbol{G}^R_{(n+1,t)} \leftarrow -\tilde{\boldsymbol{x}}_{(n)} \boldsymbol{G}^R_{(n,t)}$    for $n \geq t$;
    $\boldsymbol{G}^R_{(n-1,t)} \leftarrow -\tilde{\boldsymbol{y}}_{(n)} \boldsymbol{G}^R_{(n,t)}$    for $n \leq t$;
```

```
foreach GPU thread $t$ | $t \in [0, N)$ do
    $k \leftarrow$ block index for lead $l$;
    $\boldsymbol{A}_{l,(k,t)} \leftarrow \boldsymbol{G}^R_{(k,k)} \Gamma_k \boldsymbol{G}^{R\dagger}_{(t,k)}$;
    $\boldsymbol{A}_{l,(n+1,t)} \leftarrow -\tilde{\boldsymbol{x}}_{(n)} \boldsymbol{A}_{l,(n,t)}$    for $n \geq t$;
    $\boldsymbol{A}_{l,(n-1,t)} \leftarrow -\tilde{\boldsymbol{y}}_{(n)} \boldsymbol{A}_{l,(n,t)}$    for $n \leq t$;
```

Serial part: computing *x, y* block vectors using recursion



Recursion

```
Set: $\boldsymbol{y}_{(0)} = \boldsymbol{x}_{(N-1)} = 0$.
foreach vector index $n$ | $n \in [0, N)$ do
    $\boldsymbol{y}_{(n)} \leftarrow \boldsymbol{\beta}_{(n-1)} \tilde{\boldsymbol{y}}_{(n)}$;
    $\tilde{\boldsymbol{y}}_{(n)} \leftarrow [\boldsymbol{\alpha}_{(n-1)} - \boldsymbol{y}_{(n-1)}]^{-1} \boldsymbol{\gamma}_{(n-1)}$;
    $\boldsymbol{x}_{(n)} \leftarrow \boldsymbol{\gamma}_{(n)} \tilde{\boldsymbol{x}}_{(n)}$;
    $\tilde{\boldsymbol{x}}_{(n)} \leftarrow [\boldsymbol{\alpha}_{(n+1)} - \boldsymbol{x}_{(n+1)}]^{-1} \boldsymbol{\beta}_{(n)}$;
```

Overlap CPU computation with CPU-to-GPU asynchronous copy.

MPI/GPU parallelized Broyden's modified second method is used for minimizing $F=\rho^{in} - \rho^{out}$.

Broyden's Second Algorithm[1]

$$\rho_{n+1}^{in} = \rho_n^{in} - J_n^{-1} F_n$$

Sherman-Morrison formula for Inverse Jacobian:

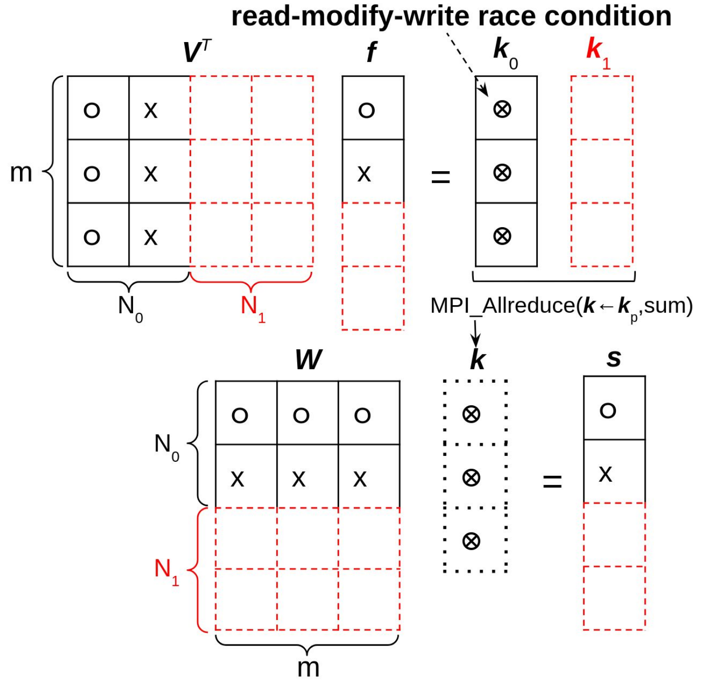
$$J_n^{-1} = J_{n-1}^{-1} + \frac{(\rho_n^{in} - \rho_{n-1}^{in}) - J_{n-1}^{-1}\Delta F_n}{||F_n - F_{n-1}||^2}(F_n - F_{n-1})^T$$

$$F_n = \rho_n^{in} - \rho_n^{out} \quad \texttt{(Minimize)}$$

We use a modified version of this algorithm[2], which **does not** require storage of inverse jacobian (N^2 elements).

Critical Step:

$$\boldsymbol{s}_{(N\times 1)} = \boldsymbol{W}_{(N\times M)} \boldsymbol{V}^T_{(M\times N)} \boldsymbol{f}_{(N\times 1)}$$



read-modify-write race condition

MPI_Allreduce(**k**←**k**$_p$,sum)

[1] Broyden, C. G. (1965) *Mathematics of Computation*, 19(92), 577-593.

[2] Srivastava, G. P. (1984) *Journal of Physics A: Mathematical and General*, 17(6), L317.