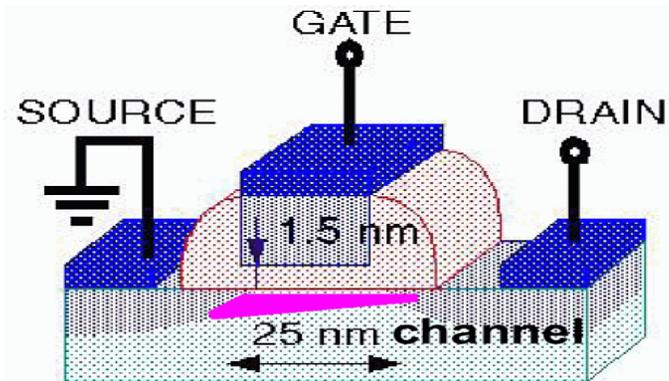


Fast Methods for Computing Certain Elements of the Inverse of a Sparse Matrix

Eric Darve

Institute for Computational Mathematics and Engineering
Stanford University



Application I

Least-square fitting

- Sampled data with experimental uncertainties.
- Least-square fitting by a hyper-plane.
- Variance of fitted parameters is given by the diagonal of the inverse of the normal matrix.

$$\sigma^2(a_j) \approx M_{jj}^{-1}$$

Application II

Eigenvalues of tri-diagonal matrices

- Calculation of eigenvectors with inverse iteration.
- Suppose we start the first iteration with vector e_k

$$(T - \hat{\lambda} I)z_k = e_k$$

- Are all choices of e_k equivalent?
- Wilkinson 1958: no.
 - Accuracy of approximation varies depending on k .
- Often it suffices to choose k such that the (k,k) entry of

$$(T - \hat{\lambda} I)^{-1}$$

has the largest value among all diagonal elements.

Application III

Sensitivity estimation

- How accurate is the solution x of

$$Ax = b$$

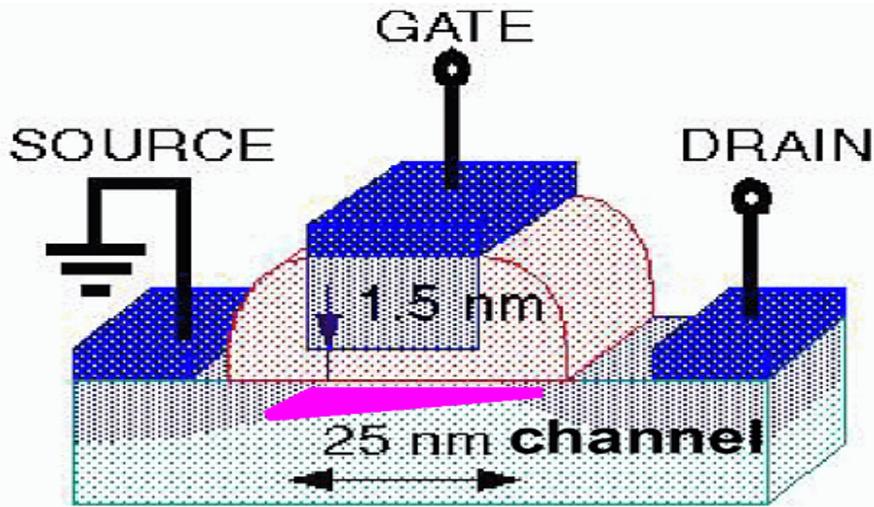
- in the presence of round off errors?
 - uncertainties in A or b ?
- Answer depends on the condition number of A

$$\|A\| \|A^{-1}\|$$

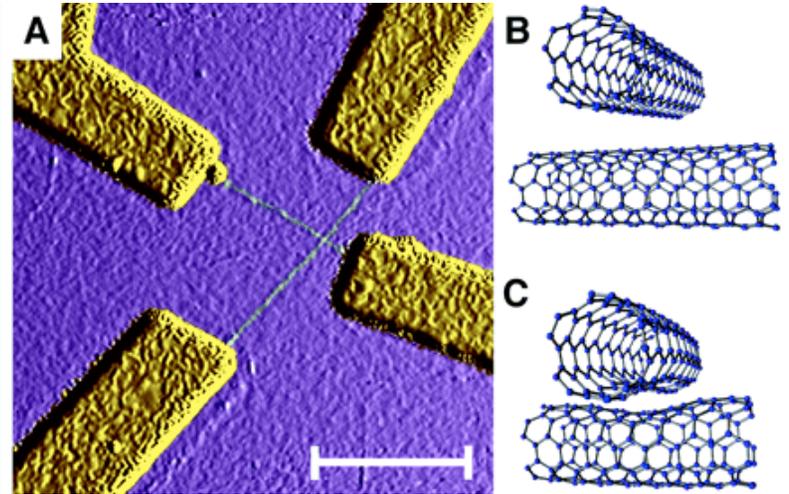
- Condition number can be estimated by computing only part of the inverse.

Application IV

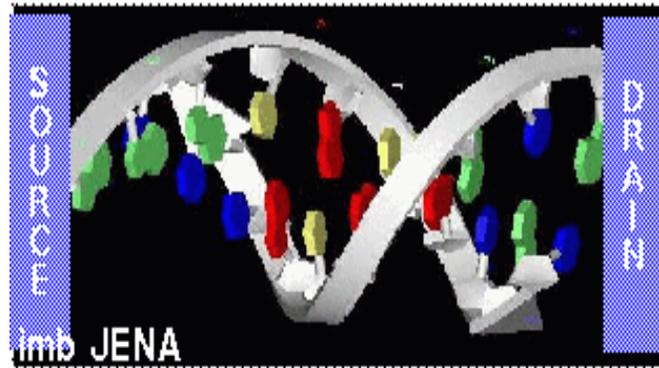
Transport in nano-structures



Nano-transistor

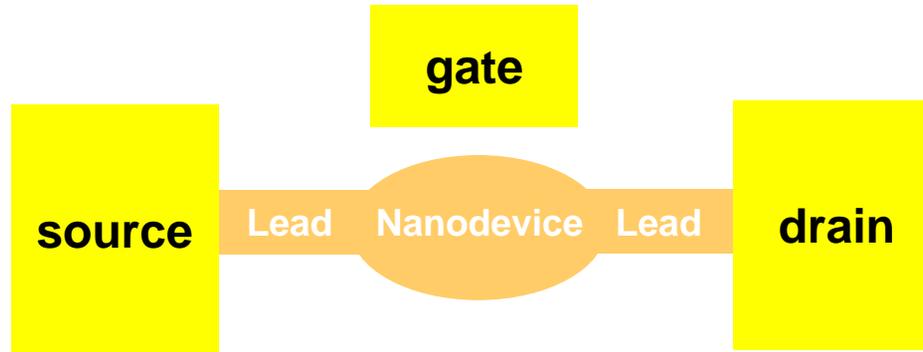


Fuhrer et al, Science (2000)



DNA

Nano-transistor: Non Equilibrium Green's Function Approach



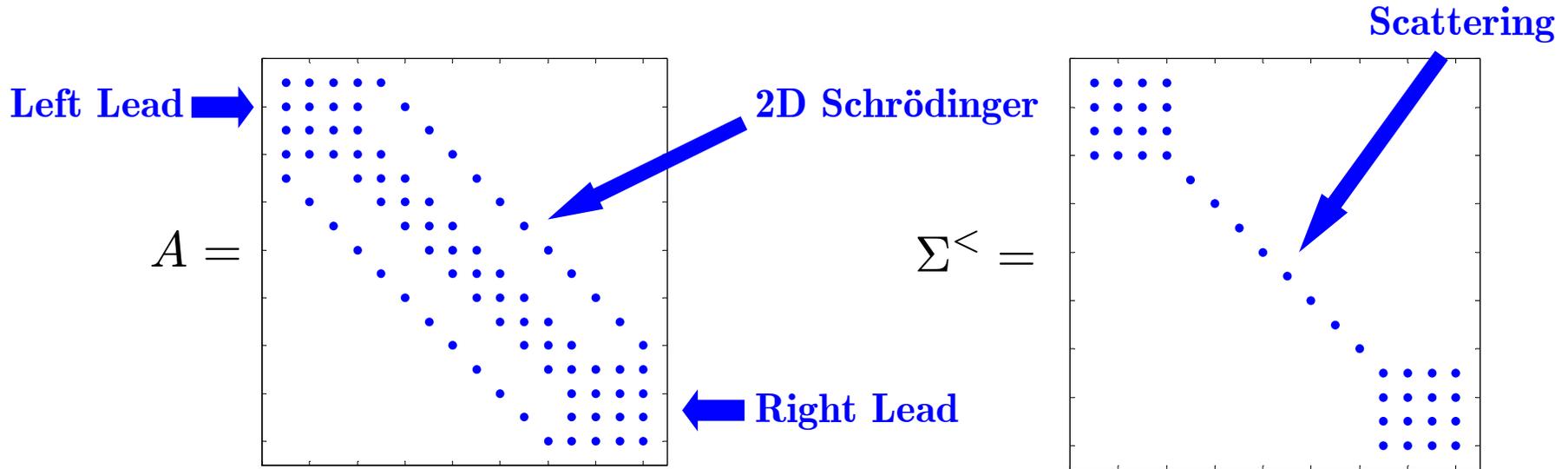
$$\nabla \cdot (\epsilon \nabla U) = q^2 [N_d - n]$$

$$[H + U] \Psi_\alpha(r) = \epsilon_\alpha \Psi_\alpha(r)$$

Poisson Equation
 $n \rightarrow U$

Equilibrium Statistical
Mechanics
 $H, U \rightarrow n$

Numerical Linear Algebra Problem



$$G^r = A^{-1}$$

$$G^< = A^{-1} \Sigma^< A^{-\dagger}$$

From the **diagonal** of G^r and $G^<$, we can calculate all the quantities of interest:

- electron density n ,
- current, I - V characteristics, density of states...

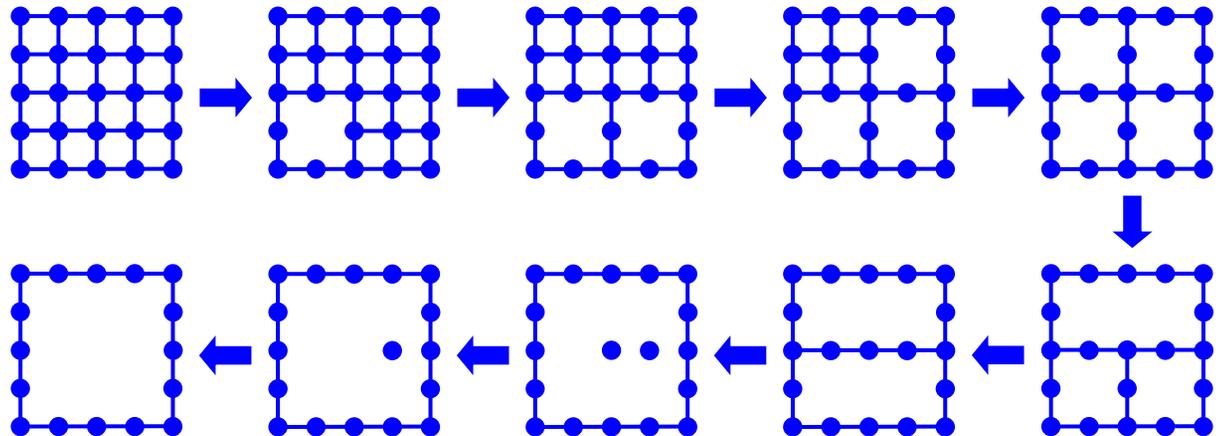
Nested Dissection

- We want to solve: $Ax = b$
- Introducing a permutation matrix P we could equivalently solve:

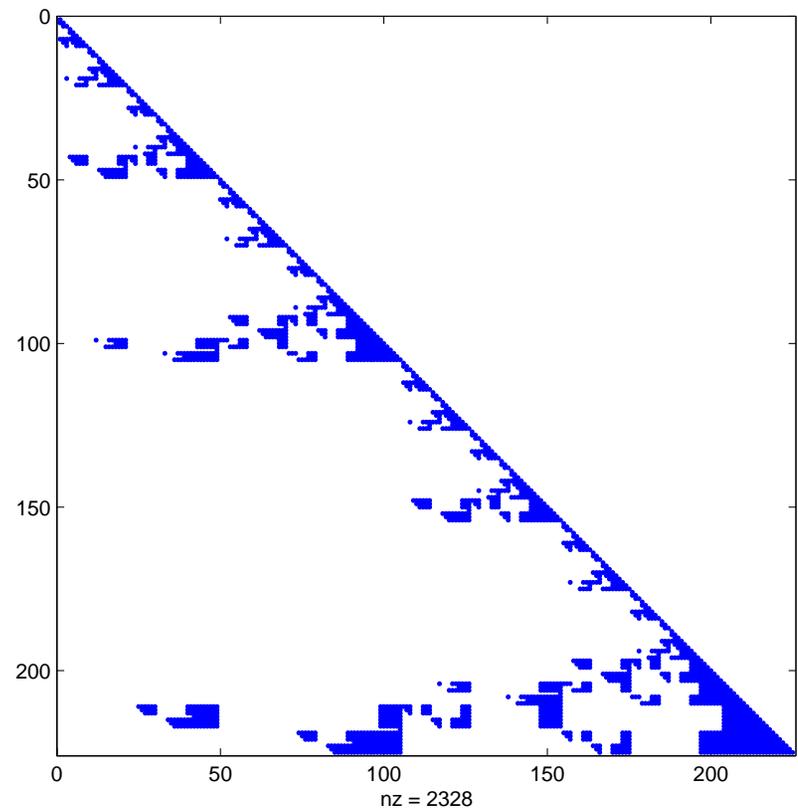
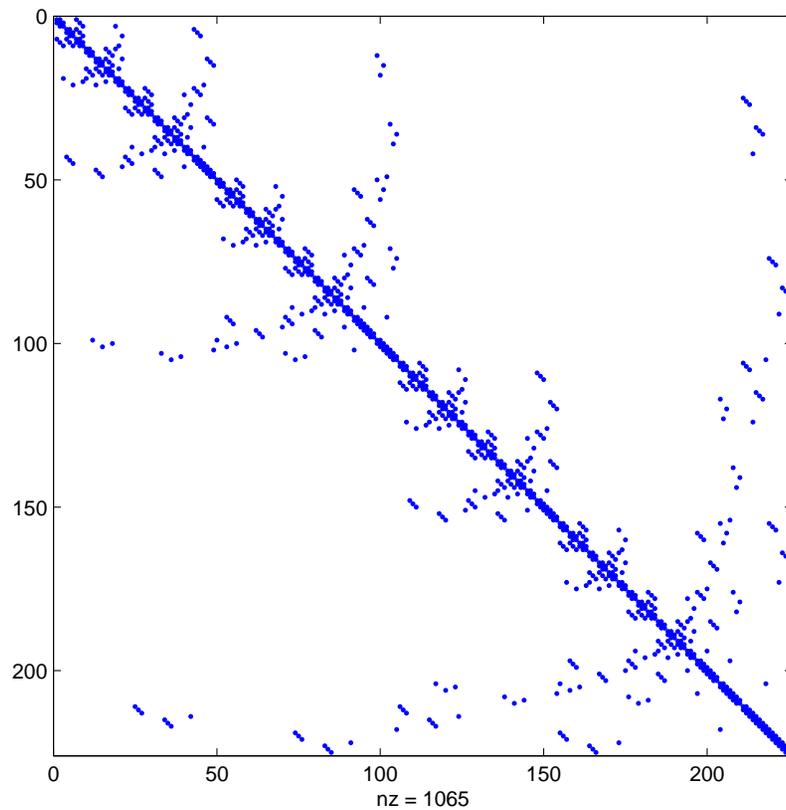
$$(PAP^T)(Px) = Pb$$

- A clever choice of P can often reduce fill.
- Using *nested dissection*:
 - Computational cost is reduced to $O(s^2 r)$ [$N=sr$]
 - Memory requirement to $O(sr \log(sr))$.

Example of *mesh transformation rules*



Sparsity pattern of A and L with nested dissection numbering



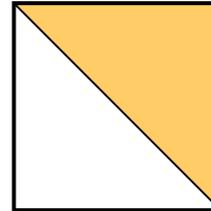
Takahashi's relationships

- Takahashi, Fagan and Chin (1973):

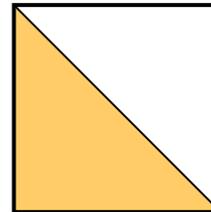
$$\begin{aligned}A &= LDU & G^r &= A^{-1} \\ G^r &= D^{-1}L^{-1} + (I - U)G^r \\ G^r &= U^{-1}D^{-1} + G^r(I - L)\end{aligned}$$

- From which:

$$g_{ij}^r = - \sum_{k>i} u_{ik}z_{kj}, \quad i < j$$



$$g_{ij}^r = - \sum_{k>i} z_{ik}l_{kj}, \quad i > j$$



Graph of $(L \setminus U)^T$

$$z_{ij} = - \sum_{k>i} u_{ik} z_{kj}, \quad i < j$$

- Not all entries z_{ij} need to be computed.
- Only those that belong to the graph of $(L \setminus U)^T$.
- **Proof:** F graph of $(L \setminus U)$

$$\begin{cases} j > i, (j, i) \in F \\ k > i, (i, k) \in F \end{cases} \Rightarrow (j, k) \in F$$

- Produces all z_{ij} with $(i, j) \in (L \setminus U)^T$
- Cost of procedure is the same as nested dissection:

$$O(s^2 r) [N=sr]$$

$$G^< = A^{-1} A^{-\dagger}$$

Part I

- Takahashi's relationships do not lead to an efficient method.
- $G^<(n,n)$ can be computed efficiently with nested dissection!
- Consider step k of Gaussian elimination applied to: $AG^<A^\dagger = I$



$$A_{22}^{(k)} = A_{22} - \frac{1}{D_{kk}} L(k+1:n, k) U(k, k+1:n)$$

Update of right-hand side $R^{(k)}$

$$R^{(k)} = \begin{array}{c} \text{orange square} \\ \text{light orange square} \end{array}$$

⇓

$$R_{22}^{(k)} = R_{22} - L(k+1:n, k) R_{12}(k, :) - R_{21}(:, k) L(k+1:n, k)^\dagger + R_{11}(k, k) L(k+1:n, k) L(k+1:n, k)^\dagger$$

- Theorem: if $F(L^T) \subset F(U)$, i.e.

$$U(k, j) = 0 \quad \Rightarrow \quad L(j, k) = 0$$

Then: $F(R_{22}^{(k)}) \subset F(A_{22}^{(k)})$

- Corollary: elimination process takes $O(s^2 r)$ operations.

Calculation of $G^<(n, n)$



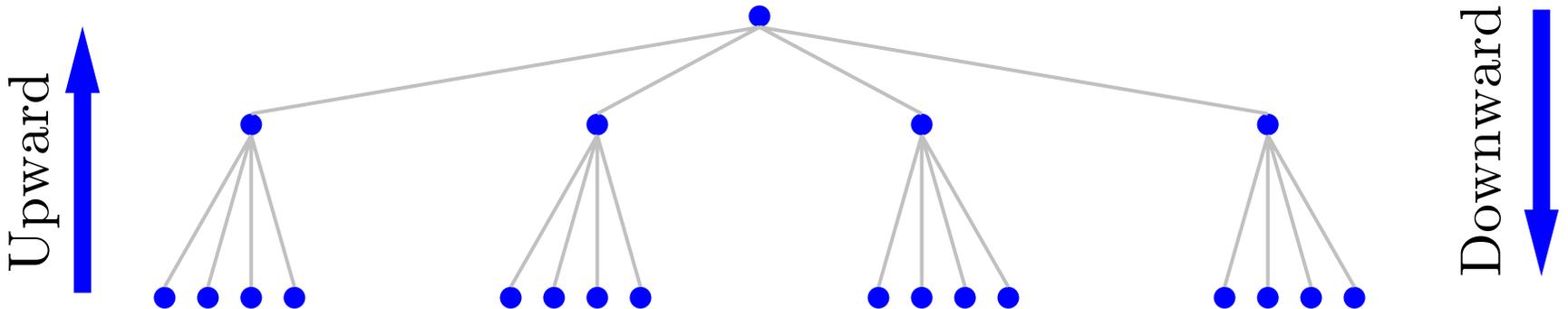
$$G^<(n, n) = R(n, n)$$

Can we compute $G^<(i, i)$ for any i ?

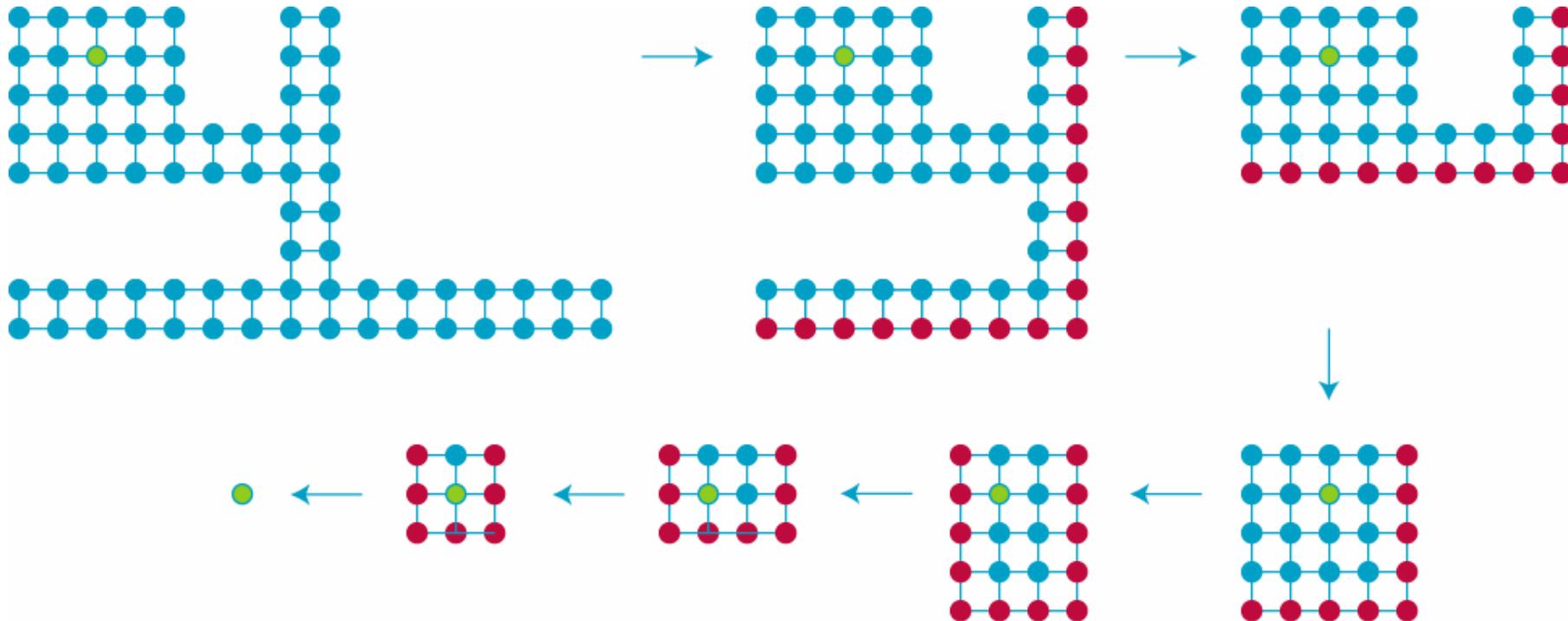
Part II

Elimination tree

- Step 1: upward pass. “Inner” nodes are eliminated.
- Step 2: downward pass. “Outer” nodes are eliminated.
- Based on a tree structure.



Step 2: eliminate “Outer” nodes



- Final phase: we are left with rs 1×1 linear systems.
- Total cost of algorithm: $O(s^2 r)$.

Conclusion

- Non-Equilibrium Green's Function approach: powerful and flexible approach to modeling nano-transistors.
- Fast numerical methods developed in 2D.
Cost $O(s^2 r)$.
- Algorithm is parallel.
- Cost in 3D is larger: $O(s^4 r q)$.
- Other possible strategies:
 - Lanczos procedure? (shown to work for $(A^T A)^{-1}$)
 - Total reduction? (shown to work for Laplace)
 - Multigrid?