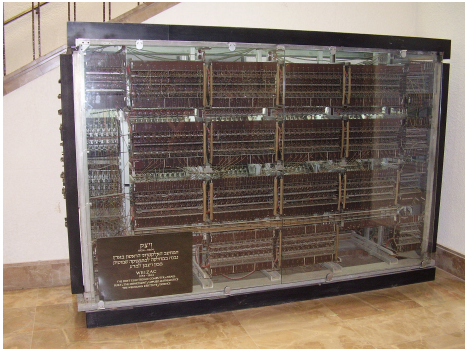
Chemfarm for collective quantum optics
Tutorial 1: Overview and first steps

Nikita Leppenen

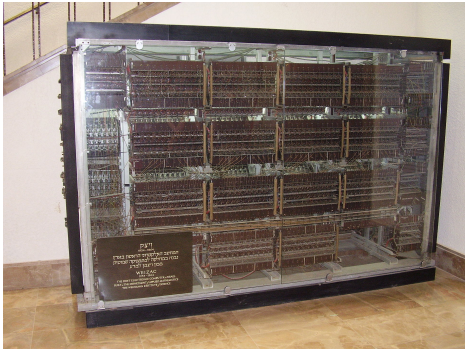
February 2026

Computers in Weizmann Institute



Weizmann Institute's first computer, Weizac, built in 1954.

Computers in Weizmann Institute



Weizmann Institute's first computer, Weizac, built in 1954.

Now

- WEXAC HPC: 34,000 CPU cores, NVidia GPUs, 11000 TB of storage
- Faculty of Math and Computer Science HPC: 2000 CPU cores, NVidia GPUs (with NVidia A 100, 80 GB of memory)
- **ChemFarm** 31648 CPU cores, 169 TB RAM, 19 nodes with NVidia GPU

Useful links: WEXAC HPC, ChemFarm
Only from WIS Network: ChemFarm Wiki

What do people use HPC for?

HPC (High Performance Computing) is used for computationally intensive tasks, such as:

- Simulating complex physical systems (e.g., quantum optics, molecular dynamics)
- Data analysis and machine learning (e.g., large datasets, training neural networks)
- Computational chemistry (e.g., quantum chemistry calculations, drug discovery)

What do people use HPC for?

HPC (High Performance Computing) is used for computationally intensive tasks, such as:

- Simulating complex physical systems (e.g., quantum optics, molecular dynamics)
- Data analysis and machine learning (e.g., large datasets, training neural networks)
- Computational chemistry (e.g., quantum chemistry calculations, drug discovery)

What my friends do:

Olga: computational biology (monte carlo simulation of Ising model for decision making)

Rafail and Narek: machine learning for computer vision (Train large model on many pictures using GPU)

Diana: condensed matter physics (running DFT calculations for topological materials)

Dima: machine learning for particle physics (writing code to analyze data from Large Hadron Collider)

What about collective quantum optics?

Solution of Lindblad equation

$$\dot{\rho} = \mathcal{L}\rho$$

Size of ρ grows exponentially with the number of atoms (for N two-level atoms, ρ is a $2^N \times 2^N$ matrix). For $N = 20$, ρ has $2^{40} \approx 10^{12}$ elements, which is too large to store in memory.
We need to solve to find steady states, dynamics, correlation functions, etc.

What about collective quantum optics?

Solution of Lindblad equation

$$\dot{\rho} = \mathcal{L}\rho$$

Size of ρ grows exponentially with the number of atoms (for N two-level atoms, ρ is a $2^N \times 2^N$ matrix). For $N = 20$, ρ has $2^{40} \approx 10^{12}$ elements, which is too large to store in memory.
We need to solve to find steady states, dynamics, correlation functions, etc.

Mean-field equations

Experiments $N \sim 10^4 - 10^6$ atoms, so we need to solve mean-field equations for N atoms, which is a system of $3N$ coupled nonlinear ODEs. Even at mean-field level computations can be challenging + some new methods (e.g., cluster mean-field, cumulant expansion) require solving even larger systems of equations.

What about collective quantum optics?

Solution of Lindblad equation

$$\dot{\rho} = \mathcal{L}\rho$$

Size of ρ grows exponentially with the number of atoms (for N two-level atoms, ρ is a $2^N \times 2^N$ matrix). For $N = 20$, ρ has $2^{40} \approx 10^{12}$ elements, which is too large to store in memory.
We need to solve to find steady states, dynamics, correlation functions, etc.

Mean-field equations

Experiments $N \sim 10^4 - 10^6$ atoms, so we need to solve mean-field equations for N atoms, which is a system of $3N$ coupled nonlinear ODEs. Even at mean-field level computations can be challenging + some new methods (e.g., cluster mean-field, cumulant expansion) require solving even larger systems of equations.

Large matrix diagonalization

Finding ground states and first excited states of many-body Hamiltonians, which is a large matrix diagonalization problem. $N = 20$ two-level atoms $\Rightarrow 2^{20} = 10^6$
Liouvivillian diagonalization: $4^N \times 4^N$ matrix. For $N = 10$ $\Rightarrow 2^{20} = 10^6$ $4^{10} = 10^6$
GPUs can be used..

I believe there are much more and we should discuss

Plan for today

- Look through the ChemFarm wiki and documentation to understand the available resources and how to access them
- Show examples of HPC usage in Mean-Field with Green Function
- Outline future directions and opportunities for collaboration

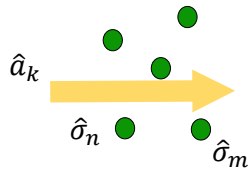
ChemFarm Wiki

- ChemFarm Wiki (Weizmann Network) is the main source of information about ChemFarm, including documentation, tutorials, and user guides.
- It contains information on how to access ChemFarm, how to use the job scheduler, and how to run different types of jobs (e.g., CPU, GPU).
- It also has a section with available software ()

Connect to cluster: ssh terminal command: `ssh <username>@chemfarm.weizmann.ac.il` (need to be on WIS network or use VPN); VS Code Remote SSH extension; JupyterHub; We will try OpenOnDemand, JupyterHub, and VS Code Remote SSH today.

Model: atoms + photon modes

$$H = H_A + H_P + H_{AP}$$



"atoms": two-level emitters $n = 1, \dots, N$ at positions $\mathbf{r}_n$

$$H_A = \hbar\omega_a \sum_{n=1}^{N} \hat{\sigma}_n^\dagger \hat{\sigma}_n \qquad \hat{\sigma}_n = |g\rangle_n\langle e| \qquad = \quad \omega_a \;\; \begin{matrix} \text{——} \; |e\rangle \\ \text{——} \; |g\rangle \end{matrix}$$

"photons" = continuum of photon modes $\hat{a}_k$ (modes defined by geometry) $\{k\}$ = mode index

$$H_P = \sum_k \hbar\omega_k \hat{a}_k^\dagger \hat{a}_k$$

"atom-photon": dipole interaction (+rotating wave approx.)

$$H_{AP} = -\sum_n \hat{E}(\mathbf{r}_n)\hat{\sigma}_n^\dagger d + h.c. \qquad \hat{E}(\mathbf{r}_n) = -i\sum_k \sqrt{\frac{\hbar\omega_k}{2\varepsilon_0}}\, u_k(\mathbf{r}_n)\hat{a}_k$$

mode profile

Heisenberg-Langevin approach

[see Lehmberg PRA 2, 883 (1970) for a full derivation]

$$H = \hbar\omega_a \sum_{n=1}^{N} \hat{\sigma}_n^\dagger \hat{\sigma}_n + \sum_k \hbar\omega_k \hat{a}_k^\dagger \hat{a}_k - \sum_n \left[\hat{E}(\mathbf{r}_n)\hat{\sigma}_n^\dagger d + h.c.\right]$$

$$\hat{E}(\mathbf{r}_n) = -i \sum_k \sqrt{\frac{\hbar\omega_k}{2\varepsilon_0}} u_k(\mathbf{r}_n)\hat{a}_k$$

$$\hat{\sigma}_n = |g\rangle_n\langle e|$$

Heisenberg Eq. for atom n:

$$\partial_t \hat{\sigma}_n = \frac{i}{\hbar}[H, \hat{\sigma}_n]$$

Use: $[\hat{\sigma}_n^\dagger, \hat{\sigma}_n] = \hat{\sigma}_n^z$ $\quad \hat{\sigma}_n^z = |e\rangle_n\langle e| - |g\rangle_n\langle g|$

$$\partial_t \hat{\sigma}_n = -i\omega_a \hat{\sigma}_n - i\frac{d}{\hbar}\hat{\sigma}_n^z \hat{E}(\mathbf{r}_n)$$

Now wish to find Eq. for field $\hat{E}(\mathbf{r}_n)$ and insert it into the above

Heisenberg Eq. for field = Maxwell's Eqs. (=Hamilton's Eq. for field...)

$$\nabla \times \hat{E}(r,t) = -\partial_t \hat{B}(r,t) \qquad \nabla \times \hat{B}(r,t) = \frac{1}{c^2}\partial_t \left[\hat{E}(r,t) + \frac{1}{\varepsilon_0}\hat{P}(r,t)\right]$$

Polarization density
= dipoles (of atoms) per volume

$$\hat{P}(\mathbf{r},t) = \sum_n \hat{d}_n(t)\delta(\mathbf{r}-\mathbf{r}_n)$$

$$\hat{d}_n = d\hat{\sigma}_n + \text{h.c.}$$

Freq. domain $\partial_t \to -i\,\omega$

$$\nabla \times \hat{E}(r,\omega) = i\,\omega \hat{B}(r,\omega) \qquad \nabla \times \hat{B}(r,\omega) = -i\frac{\omega}{c^2}\left[\hat{E}(r,\omega) + \frac{1}{\varepsilon_0}\hat{P}(r,\omega)\right]$$

Derive wave equation:

$$\nabla \times \nabla \times \hat{E} = i\,\omega \nabla \times \hat{B} = \frac{\omega^2}{c^2}\left[\hat{E} + \frac{1}{\varepsilon_0}\hat{P}\right]$$

$$\nabla \times \nabla \times \hat{E}(\mathbf{r},\omega) - \frac{\omega^2}{c^2}\hat{E}(\mathbf{r},\omega) = \frac{\omega^2}{c^2}\frac{1}{\varepsilon_0}\hat{P}(\mathbf{r},\omega) = \frac{\omega^2}{c^2}\frac{1}{\varepsilon_0}\sum_n \delta(\mathbf{r}-\mathbf{r}_n)d\hat{\sigma}_n(\omega)$$

Heisenberg Eq. for field: formal solution (1)

$$\underbrace{\nabla \times \nabla \times \hat{E}(\mathbf{r},\omega) - \frac{\omega^2}{c^2}\hat{E}(\mathbf{r},\omega)}_{\text{Linear operator (\sim "Helmholtz" Eq. for the field)}} = \underbrace{\frac{\omega^2}{c^2}\frac{1}{\varepsilon_0}\sum_n \delta(\mathbf{r}-\mathbf{r}_n)d\hat{\sigma}_n(\omega)}_{\text{point sources ("dipoles") @ atom positions}}$$

→ Solution to point source (=dipole) @ $\mathbf{r}_n$ – Green's function: $G(\mathbf{r}-\mathbf{r}_n)$

$$\hat{E}(\mathbf{r},\omega) = \hat{E}_0(\mathbf{r},\omega) + \underbrace{\frac{\omega^2}{c^2}\frac{d}{\varepsilon_0}\sum_n G(\mathbf{r}-\mathbf{r}_n,\omega)\hat{\sigma}_n(\omega)}_{\text{Field from all "dipole" sources (=radiating atoms)}}$$

Total field

"Source free" solution
= "free" field (in the absence of atoms)
= vacuum fluctuations + incident laser

field propagator = Green's function $G(\mathbf{r}_1,\mathbf{r}_2) = \mathbf{e}_d{}^{\dagger}\cdot\overline{\overline{\boldsymbol{G}}}(\mathbf{r}_1,\mathbf{r}_2)\cdot\mathbf{e}_d$

$$\mathbf{e}_i{}^{\dagger}\cdot\overline{\overline{\boldsymbol{G}}}(\mathbf{r}_1,\mathbf{r}_2)\cdot\mathbf{e}_j = G_{ij}(k,\mathbf{r}_1,\mathbf{r}_2) = \frac{e^{ikr}}{4\pi r}\left[\left(1+\frac{ikr-1}{k^2r^2}\right)\delta_{ij} + \left(-1+\frac{3-3ikr}{k^2r^2}\right)\frac{r^i r^j}{r^2}\right]$$

with $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$, $r = |\mathbf{r}|$ and $r^i = \mathbf{e}_i \cdot \mathbf{r}$.

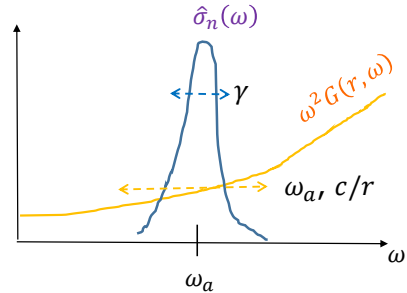
$k = \omega/c$

Heisenberg Eq. for field: formal solution (2)

$$\hat{E}(\mathbf{r},\omega) = \hat{E}_0(\mathbf{r},\omega) + \frac{\omega^2}{c^2}\frac{d}{\varepsilon_0}\sum_n \hat{\sigma}_n(\omega) G(\mathbf{r}-\mathbf{r}_n,\omega)$$

Back to time-domain – perform IFT:

$$\hat{E}(\mathbf{r},t) = \hat{E}_0(\mathbf{r},t) + \frac{1}{c^2}\frac{d}{\varepsilon_0}\sum_n \int_{-\infty}^{\infty}\frac{\mathrm{d}\omega}{2\pi} e^{-i\omega t}\hat{\sigma}_n(\omega)\ \omega^2 G(\mathbf{r}-\mathbf{r}_n,\omega)$$



- Dominant "free" dynamics: oscillations at ω_a
- Slow dynamics due to radiation ~ decay/shift γ

Wide function around ω_a
Width ~ $\omega_a,\ c/r$

Markov approximation: $\omega_a,\ c/r \gg \gamma$

$$\hat{E}(\mathbf{r},t) \approx \hat{E}_0(\mathbf{r},t) + \frac{1}{c^2}\frac{d}{\varepsilon_0}\sum_n {\omega_a}^2 G(\mathbf{r}-\mathbf{r}_n,\omega_a)\int_{-\infty}^{\infty}\frac{\mathrm{d}\omega}{2\pi} e^{-i\omega t}\hat{\sigma}_n(\omega)$$

$$= \hat{E}_0(\mathbf{r},t) + \frac{1}{c^2}\frac{d}{\varepsilon_0}\sum_n {\omega_a}^2 G(\mathbf{r}-\mathbf{r}_n,\omega_a)\hat{\sigma}_n(t)$$

Heisenberg-Langevin Eq. for atoms (1)

$$\partial_t \hat{\sigma}_n = -i\omega_a \hat{\sigma}_n - i\frac{d}{\hbar}\hat{\sigma}_n^z \hat{E}(\mathbf{r}_n)$$

We got for field: $\hat{E}(\mathbf{r}) = \hat{E}_0(\mathbf{r}) + \frac{1}{c^2}\frac{d}{\varepsilon_0}\sum_n {\omega_a}^2 G(\mathbf{r}-\mathbf{r}_n,\omega_a)\hat{\sigma}_n$

→ Field "felt" by **atom** n, $\mathbf{r} = \mathbf{r}_n$:

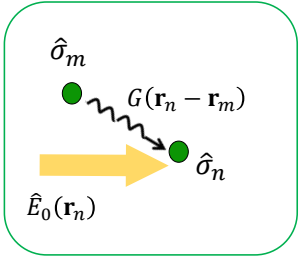
$$\hat{E}(\mathbf{r}_n) = \hat{E}_0(\mathbf{r}_n) + \frac{1}{c^2}\frac{d}{\varepsilon_0}\sum_m {\omega_a}^2 G(\mathbf{r}_n-\mathbf{r}_m)\hat{\sigma}_m$$

= "free" field (in the absence of atoms)
= vacuum fluctuations + incident laser

$$\hat{E}_0(\mathbf{r}_n) = -i\sum_k \sqrt{\frac{\hbar\omega_k}{2\varepsilon_0}} u_k(\mathbf{r}_n)\hat{a}_k(0)e^{-i\omega_k t}$$

Field from all "dipole" sources (=radiating atoms)

field propagator = $G(\mathbf{r}_n - \mathbf{r}_m)$



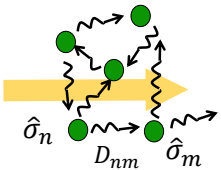
Heisenberg-Langevin Eq. for atoms (2)

$$\partial_t \hat{\sigma}_n = -i\omega_a \hat{\sigma}_n - i\hat{\sigma}_n^z \frac{d}{\hbar}\left[\hat{E}_0(\mathbf{r}_n) + \frac{{\omega_a}^2}{c^2}\frac{d}{\varepsilon_0}\sum_m G(\mathbf{r}_n - \mathbf{r}_m)\hat{\sigma}_m\right]$$

$$D_{nm} = -i\frac{3}{2}\gamma\lambda G(\mathbf{r}_n - \mathbf{r}_m) \qquad \gamma = \frac{{d^2\omega_a}^3}{3\pi\varepsilon_0\hbar c^3}$$

$$\lambda = 2\pi c/\omega_a$$

$$\partial_t \hat{\sigma}_n = -i\omega_a \hat{\sigma}_n + \hat{\sigma}_n^z\left[\hat{F}_n + \sum_m D_{nm}\,\hat{\sigma}_m\right]$$



→ Collective response

(= multiple scattering
= dipole-dipole)

quantum noise (vacuum) + incident laser

$$\hat{F}_n = -i\frac{d}{\hbar}\hat{E}_0(\mathbf{r}_n) = -d\sum_k \sqrt{\frac{\omega_k}{2\varepsilon_0\hbar}}u_k(\mathbf{r}_n)\hat{a}_k(0)e^{-i\omega_k t}$$

Dipole-dipole coupling (with all atomic dipoles)

$$D_{nm} = \gamma_{nm}/2 + i\Delta_{nm}$$

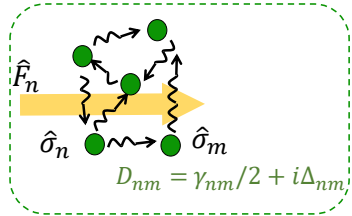
dipole-dipole kernel
(photon Green's function)

$$\partial_t \hat{\sigma}_n = -\left(i\omega_a + \frac{\gamma}{2}\right)\hat{\sigma}_n + \hat{\sigma}_n^z\left[\hat{F}_n + \sum_{m\neq n} D_{nm}\,\hat{\sigma}_m\right]$$

$$\mathrm{Re}\; D_{nn} = \gamma/2$$

$$\mathrm{Im}\; D_{nn} + \omega_a \rightarrow \omega_a$$

Many-body physics of quantum emitters (atoms)



Heisenberg-Langevin Eq. for atoms:

$$\partial_t \hat{\sigma}_n = \frac{i}{\hbar}[H_{\text{eff}}, \hat{\sigma}_n] + \hat{\sigma}_n^z \hat{F}_n$$

Effective Hamiltonian (non-Hermitian): **collectivity**

$$H_{\text{eff}} = \hbar\left(\omega_a - i\frac{\gamma}{2}\right)\sum_{n=1}^{N}\hat{\sigma}_n^\dagger\hat{\sigma}_n + \hbar\sum_{n=1}^{N}\sum_{m\neq n}\left(\Delta_{nm} - i\frac{\gamma_{nm}}{2}\right)\hat{\sigma}_n^\dagger\hat{\sigma}_m$$

Quantum noise
(has to exist due to dissipation)

$$\hat{F}_n = -d\sum_k \sqrt{\frac{\omega_k}{2\varepsilon_0\hbar}}\, u_k(\mathbf{r}_n)\hat{a}_k(0)e^{-i\omega_k t}$$

Dipole-dipole interaction
(reversible excitation exchange)

Collective radiation
(dissipation)

Equivalent description: quantum master equation $\partial_t\hat{\rho} = -\frac{i}{\hbar}\left(H_{\text{eff}}\hat{\rho} - \hat{\rho}H_{\text{eff}}^\dagger\right) + \sum_{n,m}\gamma_{nm}\hat{\sigma}_n\hat{\rho}\hat{\sigma}_m^\dagger$

Output light $\hat{E}(\mathbf{r}) = \hat{E}_0(\mathbf{r}) + \frac{1}{c^2}\frac{d}{\varepsilon_0}\sum_n \omega_a{}^2 G(\mathbf{r}-\mathbf{r}_n,\omega_a)\hat{\sigma}_n$

Quantum noise: "jump term"

Mean-Field in waveguide QED or atomic array/cloud

Denoting $s_n = \langle \sigma_n^- \rangle$ and $s_n^z = \langle \sigma_n^z \rangle$ and neglecting quantum correlations, we get the following system of equations for N atoms:

$$\begin{cases} \dot{s}_n = -\frac{\gamma}{2} s_n + s_n^z \left[\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right] \\ \dot{s}_n^z = -\gamma(s_n^z + 1) - 2\left[s_n^* \left(\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right) + c.c. \right] \end{cases} \tag{1}$$

Geometry information: $D_{nm} = -i\frac{3\gamma}{2} \lambda G(\boldsymbol{r}_n - \boldsymbol{r}_m)$

Mean-Field in waveguide QED or atomic array/cloud

Denoting $s_n = \langle \sigma_n^- \rangle$ and $s_n^z = \langle \sigma_n^z \rangle$ and neglecting quantum correlations, we get the following system of equations for N atoms:

$$\begin{cases} \dot{s}_n = -\frac{\gamma}{2} s_n + s_n^z \left[\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right] \\ \dot{s}_n^z = -\gamma(s_n^z + 1) - 2\left[s_n^* \left(\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right) + c.c. \right] \end{cases} \tag{1}$$

Geometry information: $D_{nm} = -i\frac{3\gamma}{2}\lambda G(\boldsymbol{r}_n - \boldsymbol{r}_m)$

Problem statement: solve the system depending on the geometry. Find $s_n(t)$ and $s_n^z(t)$ for $n = 1, ..., N$ depending on Ω, initial conditions.

Mean-Field in waveguide QED or atomic array/cloud

Denoting $s_n = \langle \sigma_n^- \rangle$ and $s_n^z = \langle \sigma_n^z \rangle$ and neglecting quantum correlations, we get the following system of equations for N atoms:

$$\begin{cases} \dot{s}_n = -\frac{\gamma}{2} s_n + s_n^z \left[\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right] \\ \dot{s}_n^z = -\gamma(s_n^z + 1) - 2\left[s_n^* \left(\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right) + c.c. \right] \end{cases} \tag{1}$$

Geometry information: $D_{nm} = -i\frac{3\gamma}{2} \lambda G(\boldsymbol{r}_n - \boldsymbol{r}_m)$

Problem statement: solve the system depending on the geometry. Find $s_n(t)$ and $s_n^z(t)$ for $n = 1, ..., N$ depending on Ω, initial conditions.

Where HPC comes in: big memory ($N \sim 10^3$), NumPy for Greens Function, parallelization for parameter scans.

Mean-Field in waveguide QED or atomic array/cloud

Denoting $s_n = \langle \sigma_n^- \rangle$ and $s_n^z = \langle \sigma_n^z \rangle$ and neglecting quantum correlations, we get the following system of equations for N atoms:

$$\begin{cases} \dot{s}_n = -\frac{\gamma}{2} s_n + s_n^z \left[\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right] \\ \dot{s}_n^z = -\gamma(s_n^z + 1) - 2\left[s_n^* \left(\frac{\gamma_{1D}}{2} \sum_{m \neq n} D_{nm} s_m + i\Omega e^{ikz_n} \right) + c.c. \right] \end{cases} \tag{1}$$

Geometry information: $D_{nm} = -i\frac{3\gamma}{2} \lambda G(\boldsymbol{r}_n - \boldsymbol{r}_m)$

Problem statement: solve the system depending on the geometry. Find $s_n(t)$ and $s_n^z(t)$ for $n = 1, ..., N$ depending on Ω, initial conditions.

Where HPC comes in: big memory ($N \sim 10^3$), NumPy for Greens Function, parallelization for parameter scans.

Solution step: calculate Green function using vectorized code, solve ODEs using `scipy.integrate.solve_ivp` (parallelization for parameter scans), run on chemfarm

Parallel solver function

```python
def solve_system_wrapper(args):
    """Wrapper for parallelization."""
    return solve_system_prop(*args)


def parallel_solve_omega(N, gamma, Dnm, Gnm, Omega_n_values, t_span, y0, n_workers=4):
    """Parallelized execution for solving ODEs across different Omega values."""
    with mp.Pool(processes=n_workers) as pool:
        results = list(tqdm(
            pool.imap(
                solve_system_wrapper,
                [(N, gamma, Dnm, Gnm, Omega_n, t_span, y0) for Omega_n in Omega_n_values],
                chunksize=2
            ),
            total=len(Omega_n_values),
            desc="Solving ODE sweep"
        ))
    return results
```

n_workers - number of parallel processes to use

Vectorized 1D Greens function

```python
def create_1d_gnm_matrix_vectorized(positions, gamma1D=1):
    """Creates the Green's function matrix for 1D lattice using vectorized operations."""
    N = len(positions)

    # Compute all pairwise displacement vectors in 1D
    r_vec = positions[:, np.newaxis] - positions[np.newaxis, :]
    # Compute Euclidean distances
    R = np.abs(r_vec)
    # Set diagonal elements to avoid division by zero
    np.fill_diagonal(R, 1.0)
    # Compute kr
    kr = 2 * np.pi * R
    # Compute the 1D Green's function
    g1 = gamma1D / 2 * np.exp(1j * kr)
    np.fill_diagonal(g1, 0)

    gnm = 2 * np.real(g1)
    dnm = np.imag(g1)

    return gnm, dnm
```

3D lattice with numpy

```python
#create an 3D array with x,y,z lattice with lattice spacing a_x, a_y, a_z
def create_lattice(nx, ny, nz, a_x, a_y, a_z):
    """Creates a 3D lattice of points with given dimensions and spacings."""
    x = np.arange(-nx/2, nx/2) * a_x
    y = np.arange(-ny/2, ny/2) * a_y
    z = np.arange(-nz/2, nz/2) * a_z
    xv, yv, zv = np.meshgrid(x, y, z, indexing='ij')
    lattice = np.stack((xv, yv, zv), axis=-1).reshape(-1, 3)
    return lattice
```

3D Greens function with numpy

```python
def create_gnm_matrix_vectorized(positions):
    """Creates the Green's function matrix using vectorized operations."""
    N = len(positions)

    # Compute all pairwise displacement vectors using broadcasting
    # Shape: (N, N, 3)
    r_vec = positions[:, np.newaxis, :] - positions[np.newaxis, :, :]

    # Compute Euclidean distances
    # Shape: (N, N)
    R = np.linalg.norm(r_vec, axis=2)

    # Set diagonal elements to a non-zero value to avoid division by zero
    np.fill_diagonal(R, 1.0)
    kr = 2 * np.pi * R

    # Create e_0 vector (polarization)
    e_0 = (1 / np.sqrt(2)) * np.array([1, 1j, 0])
    #e_0 = np.array([0, 0, 1])
    # Compute dot products between e_0 and all displacement vectors
    # Shape: (N, N)
```

```
    # Shape: (N, N)
    dot_products = np.sum(e_0 * r_vec, axis=2)

    # Compute cos^2(theta)
    # Shape: (N, N)
    cos2_theta = np.abs(dot_products / R) ** 2

    # Compute the Green's function
    g1 = np.exp(1j * kr) / kr * (
        (1 + (1j * kr - 1) / kr**2) + cos2_theta * (-1 + (3 - 3 * 1j * kr) / kr**2))

    # Set diagonal elements to zero (no self-interaction)
    np.fill_diagonal(g1, 0)

    # Compute gnm and dnm
    dnm = -3/4 * np.real(g1)
    gnm = 3/2 * np.imag(g1)
    #dnm = np.zeros((N,N))

    return gnm, dnm
```