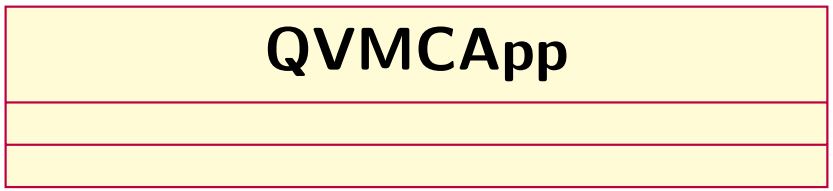
Quantum Variational Monte Carlo



?

Quantum Variational Monte Carlo

Problem statement

- Minimize the functional $E[\Psi_T]$, where

$$E[\Psi_T] = \frac{\int \mathrm{d}\mathbf{R}\Psi_T^*(\mathbf{R})\hat{\mathrm{H}}\Psi_T(\mathbf{R})}{\int \mathrm{d}\mathbf{R}|\Psi_T(\mathbf{R})|^2}. \tag{1}$$

- The variational principle gives us a lower bound on the ground state energy.

$$E_0 \leq E[\Psi_T]$$

Need to calculate the integral in (1).

Quantum Variational Monte Carlo

Problem statement

- Minimize the functional $E[\Psi_T]$, where

$$E[\Psi_T] = \frac{\int \mathrm{d}\mathbf{R}\Psi_T^*(\mathbf{R})\hat{\mathrm{H}}\Psi_T(\mathbf{R})}{\int \mathrm{d}\mathbf{R}|\Psi_T(\mathbf{R})|^2}. \tag{1}$$

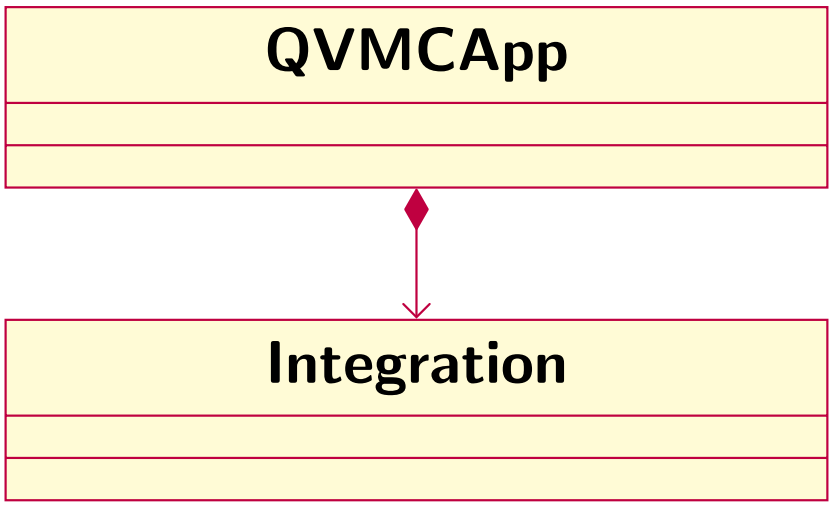
- The variational principle gives us a lower bound on the ground state energy.

$$E_0 \leq E[\Psi_T]$$

Need to calculate the integral in (1).

Quantum Variational Monte Carlo

Integration



?

Quantum Variational Monte Carlo

Monte Carlo integration

- Rewrite the integral using the local energy E_L and its pdf $\rho(\mathbf{R})$

$$E[\Psi_T] = \int \mathrm{d}\mathbf{R} E_L(\mathbf{R})\rho(\mathbf{R})$$

where

$$E_L(\mathbf{R}) = \frac{\hat{\mathrm{H}}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}, \qquad \rho(\mathbf{R}) = \frac{|\Psi_T(\mathbf{R})|^2}{\int \mathrm{d}\mathbf{R}'|\Psi_T(\mathbf{R}')|^2}. \tag{2}$$

Sample the local energy following $\rho(\mathbf{R})$.

Quantum Variational Monte Carlo

Monte Carlo integration

- Rewrite the integral using the local energy E_L and its pdf $\rho(\mathbf{R})$

$$E[\Psi_T] = \int \mathrm{d}\mathbf{R} E_L(\mathbf{R})\rho(\mathbf{R})$$

where

$$E_L(\mathbf{R}) = \frac{\hat{\mathrm{H}}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}, \qquad \rho(\mathbf{R}) = \frac{|\Psi_T(\mathbf{R})|^2}{\int \mathrm{d}\mathbf{R}'|\Psi_T(\mathbf{R}')|^2}. \tag{2}$$

Sample the local energy following $\rho(\mathbf{R})$.

Quantum Variational Monte Carlo

Random walkers

Metropolis algorithm

for $i = 0, \ldots, ncycles$
  for $j = 1, \ldots, N$
    $X \in [-l, l]$
    $R' \leftarrow R + X$
    $p \leftarrow \frac{|\Psi_T(R')|^2}{|\Psi_T(R)|^2}$
    if $Y \leq p, \quad Y \in [0, 1]$
      $R = R'$
    end if
  end for
  Update statistics
end for

Metropolis Hastings algorithm

for $i = 0, \ldots, ncycles$
  for $j = 1, \ldots, N$
    $F \leftarrow 2\frac{\nabla\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}$
    $X \in gaussian()$
    $\mathbf{R}' = \mathbf{R} + D\Delta tF + X\sqrt{\Delta t}$
    $p \leftarrow \frac{|\Psi_T(R')|^2}{|\Psi_T(R)|^2}$
    $\omega_{RR'} = \exp\left(\frac{-(R'-R-D\Delta tF)^2}{4D\Delta t}\right)$
    $q \leftarrow \frac{\omega_{R'R}}{\omega_{RR'}}p$
    if $Y \leq q, \quad Y \in [0, 1]$
      $R = R'$
    end if
  end for
  Update statistics
end for

Need wavefunction, quantum force and local energy

Quantum Variational Monte Carlo

Random walkers

Metropolis algorithm

for $i = 0, \ldots, ncycles$
  for $j = 1, \ldots, N$
    $X \in [-l, l]$
    $R' \leftarrow R + X$
    $p \leftarrow \frac{|\Psi_T(R')|^2}{|\Psi_T(R)|^2}$
    if $Y \leq p, \quad Y \in [0, 1]$
      $R = R'$
    end if
  end for
  Update statistics
end for

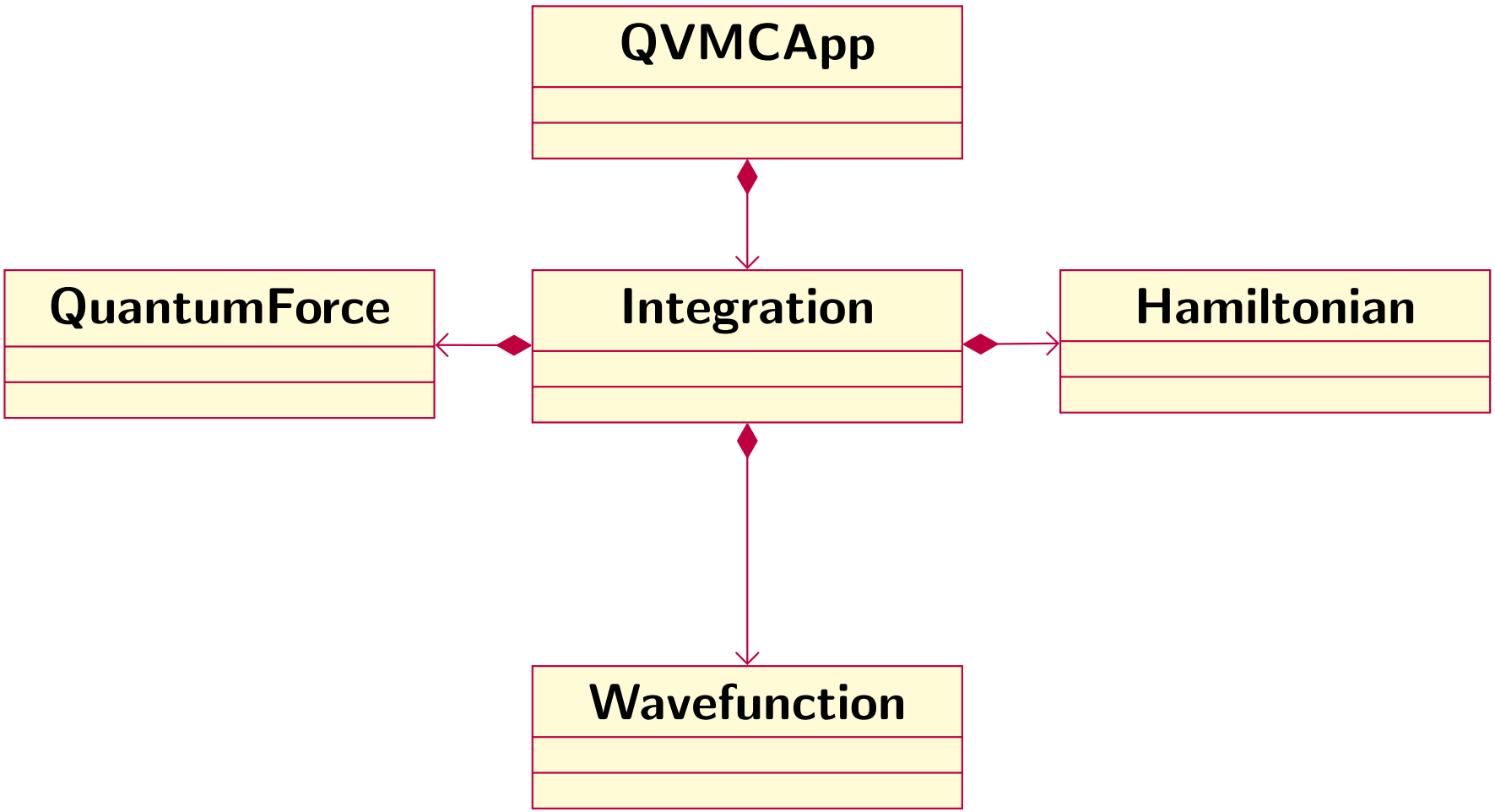
Metropolis Hastings algorithm

for $i = 0, \ldots, ncycles$
  for $j = 1, \ldots, N$
    $F \leftarrow 2\frac{\nabla\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}$
    $X \in gaussian()$
    $\mathbf{R}' = \mathbf{R} + D\Delta tF + X\sqrt{\Delta t}$
    $p \leftarrow \frac{|\Psi_T(R')|^2}{|\Psi_T(R)|^2}$
    $\omega_{RR'} = \exp\left(\frac{-(R'-R-D\Delta tF)^2}{4D\Delta t}\right)$
    $q \leftarrow \frac{\omega_{R'R}}{\omega_{RR'}}p$
    if $Y \leq q, \quad Y \in [0, 1]$
      $R = R'$
    end if
  end for
  Update statistics
end for

Need wavefunction, quantum force and local energy

Quantum Variational Monte Carlo

Ingredients for integration



Quantum Variational Monte Carlo

Sample the local energy

- The local energy depends on the Hamiltonian.

$$\hat{\mathrm{H}} = \sum_{i=1}^{N} \frac{-\nabla_i^2}{2} + \hat{\mathrm{V}} + \hat{\mathrm{H}}_{int}, \tag{3}$$

giving

$$E_L(\mathbf{R}) = \sum_{i=1}^{N} \frac{-\nabla_i^2 \Psi_T(\mathbf{R})}{2\Psi_T(\mathbf{R})} + \frac{\hat{\mathrm{V}}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})} + \frac{\hat{\mathrm{H}}_{int}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}. \tag{4}$$

Need wavefunction with value and laplacian, in additon to a potential and an interaction.

Quantum Variational Monte Carlo

Sample the local energy

- The local energy depends on the Hamiltonian.

$$\hat{\mathrm{H}} = \sum_{i=1}^{N} \frac{-\nabla_i^2}{2} + \hat{\mathrm{V}} + \hat{\mathrm{H}}_{int}, \tag{3}$$

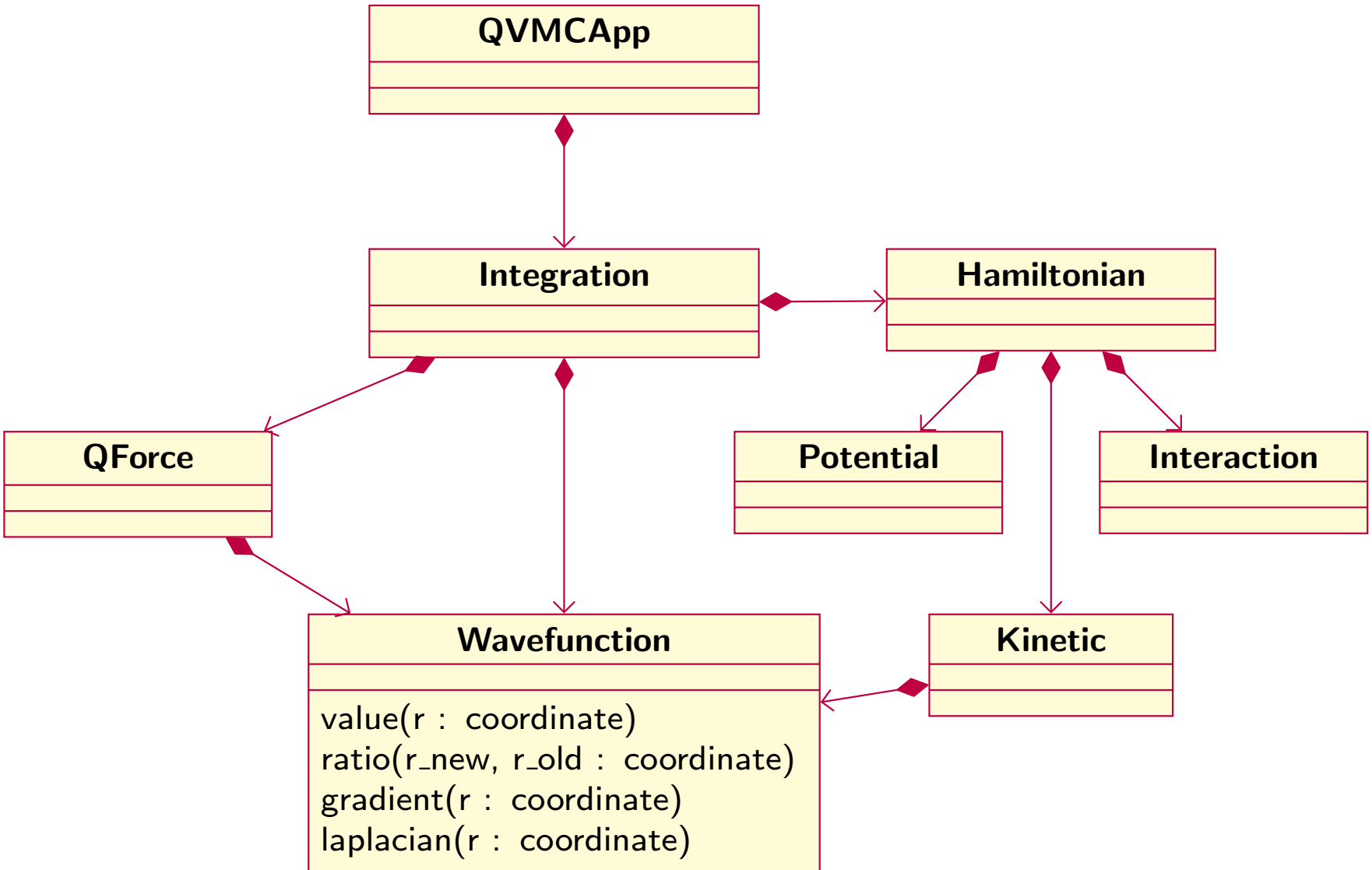
giving

$$E_L(\mathbf{R}) = \sum_{i=1}^{N} \frac{-\nabla_i^2 \Psi_T(\mathbf{R})}{2\Psi_T(\mathbf{R})} + \frac{\hat{\mathrm{V}}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})} + \frac{\hat{\mathrm{H}}_{int}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}. \tag{4}$$

Need wavefunction with value and laplacian, in additon to a potential and an interaction.

Quantum Variational Monte Carlo

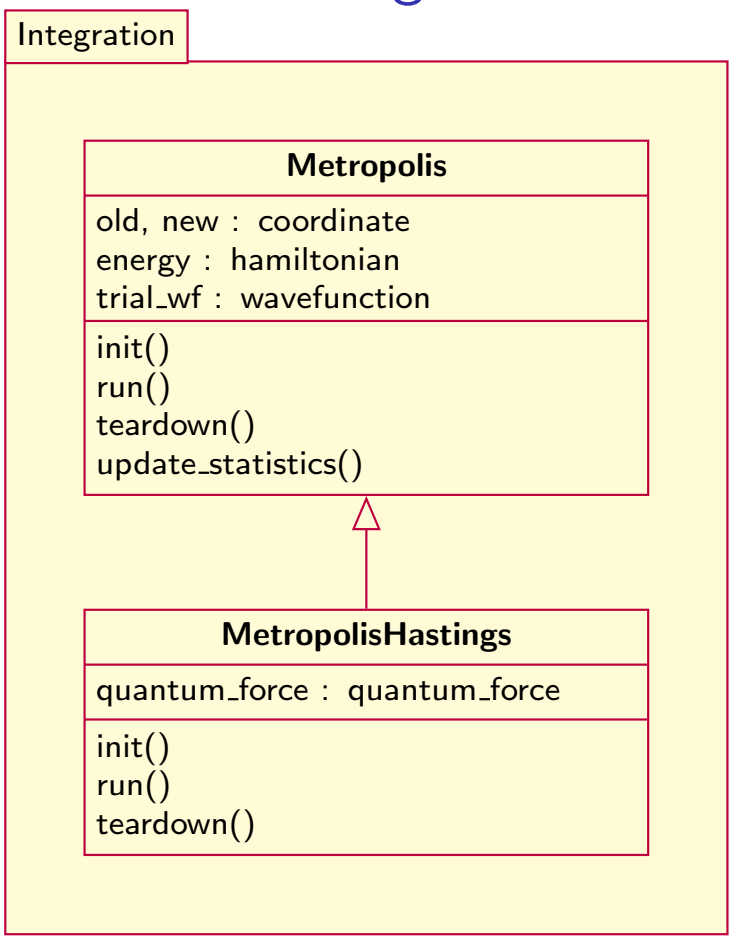
Complete framework



Class definition

Integration

UML diagram



Type declaration

```
1   TYPE, PUBLIC :: metropolis
2       TYPE (coordinate) :: old, new
3       CLASS (hamiltonian), POINTER :: energy => NULL()
4       CLASS (wavefunction), POINTER :: trial_wf => NULL()
5   CONTAINS
6       PROCEDURE :: init => metropolis_init
7       PROCEDURE :: run => metropolis_run
8       PROCEDURE :: teardown => metropolis_teardown
9       PROCEDURE, PRIVATE :: update_statistics
10  END TYPE metropolis
11  TYPE(metropolis) :: a1
12
13  TYPE, PUBLIC, EXTENDS(metropolis) :: metropolis_hastings
14      CLASS (quantum_force), POINTER :: quantum_force => NULL(
15  CONTAINS
16      PROCEDURE :: init => metropolis_hastings_init
17      PROCEDURE :: run => metropolis_hastings_run
18      PROCEDURE :: teardown => metropolis_hastings_teardown
19  END TYPE metropolis_hastings
20  TYPE(metropolis_hastings) :: a3
```

Two dimensional quantum dot

Problem statement

- Hamiltonian

$$\hat{\mathrm{H}} = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 \mathbf{r}_i^2 \right) + \sum_{i<j=1}^{N} \frac{1}{r_{ij}}, \tag{5}$$

- Trial wavefunction

$$\Psi_T = \Psi_D \Psi_J, \tag{6}$$

where

$$\Psi_D = \frac{1}{\sqrt{N!}} \begin{vmatrix} \varphi_1(\mathbf{r}_1) & \dots & \varphi_1(\mathbf{r}_N) \\ \vdots & \ddots & \vdots \\ \varphi_N(\mathbf{r}_1) & \dots & \varphi_N(\mathbf{r}_N) \end{vmatrix} \tag{7}$$

and

$$\Psi_J = \prod_{i<j=1}^{N} \exp\left(\frac{a_{ij} r_{ij}}{1 + \beta r_{ij}} \right). \tag{8}$$

Two dimensional quantum dot

Problem statement

- Spin independent Hamiltonian means we can split the Slater determinant in a spin up part and a spin down part

$$\Psi_D \approx \Psi_\uparrow \cdot \Psi_\downarrow, \tag{9}$$

where

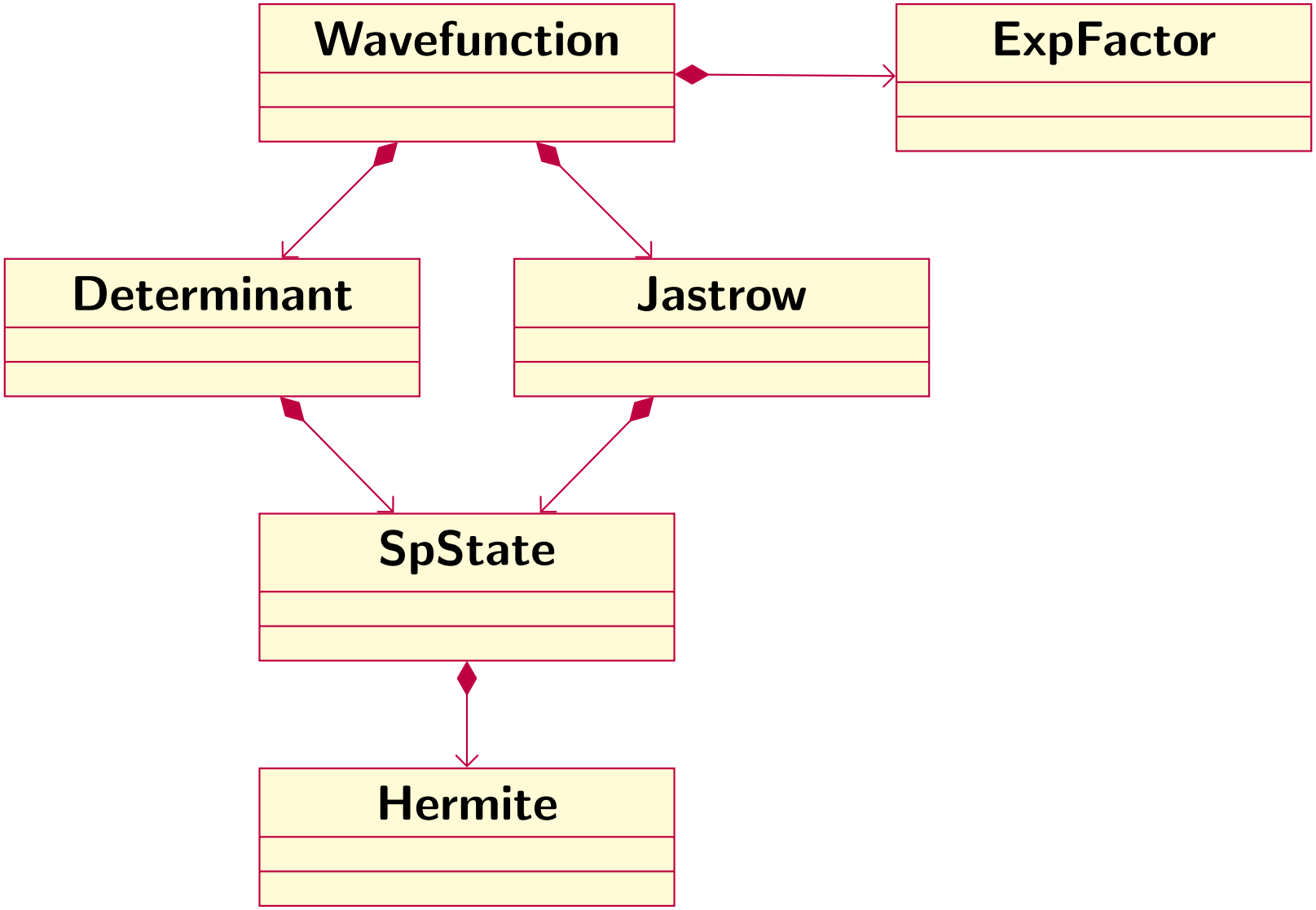
$$\Psi_\uparrow = \frac{1}{\sqrt{(N/2)!}} \begin{vmatrix} \varphi_2(\mathbf{r}_2) & \varphi_2(\mathbf{r}_4) & \dots & \varphi_2(\mathbf{r}_N) \\ \varphi_4(\mathbf{r}_2) & \varphi_4(\mathbf{r}_4) & \dots & \varphi_4(\mathbf{r}_N) \\ \vdots & \ddots & \vdots & \\ \varphi_N(\mathbf{r}_2) & \varphi_N(\mathbf{r}_4) & \dots & \varphi_N(\mathbf{r}_N) \end{vmatrix} \tag{10}$$

and

$$\Psi_\downarrow = \frac{1}{\sqrt{(N/2)!}} \begin{vmatrix} \varphi_1(\mathbf{r}_1) & \varphi_1(\mathbf{r}_3) & \dots & \varphi_1(\mathbf{r}_{N-1}) \\ \varphi_3(\mathbf{r}_1) & \varphi_3(\mathbf{r}_3) & \dots & \varphi_3(\mathbf{r}_{N-1}) \\ \vdots & \ddots & \vdots & \\ \varphi_{N-1}(\mathbf{r}_1) & \varphi_{N-1}(\mathbf{r}_3) & \dots & \varphi_{N-1}(\mathbf{r}_{N-1}) \end{vmatrix}. \tag{11}$$

Two dimensional quantum dot

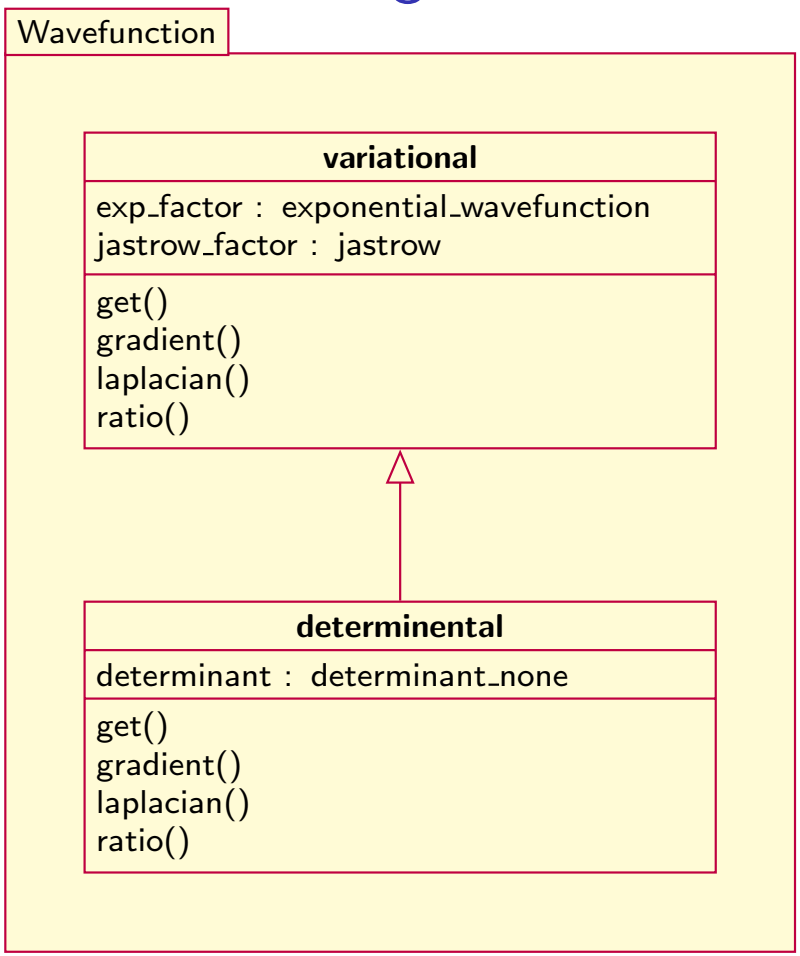
Complete wavefunction framework



Class definition

Wavefunction

UML diagram



Type declaration

```
TYPE, PUBLIC, EXTENDS(wavefunction) :: variational
    CLASS (exponential_wavefunction), POINTER :: exp_factor
    CLASS (jastrow), POINTER :: jastrow_factor => NULL()
CONTAINS
    PROCEDURE get => variational_get
    PROCEDURE gradient => variational_gradient
    PROCEDURE laplacian => variational_laplacian
END TYPE variational
```

```
TYPE, PUBLIC, EXTENDS(variational) :: determinental
    CLASS (determinant_none), POINTER :: determinant => NULL
CONTAINS
    PROCEDURE get => determinental_get
    PROCEDURE gradient => determinental_gradient
    PROCEDURE laplacian => determinental_laplacian
    PROCEDURE ratio => determinental_ratio
END TYPE determinental
```

Profiling with grof

- Compile your code with the -pg option.
- Run your code.
- Run gprof: gprof ./exeutable ¿ outputfile
- Study **outputfile**