# Phase 12 – Quantization & Coupling

## Part 2: Canonical Momenta and Hamiltonian

### Setup

I continue from the Part 1 Lagrangian density

Plain text:  
L = 1/2 (∂t ψ)² − 1/2 (∇ψ)² − Vψ(x) ψ²,  
Vψ(x) = −∇²( space(x) + current(x)² ).

### Canonical momentum

The conjugate momentum to is

Plain text:  
π(x,t) = ∂L/∂(∂t ψ) = ∂t ψ.

No primary constraints arise: and are independent canonical variables.

### Hamiltonian density

By definition,

Plain text:  
H = π ∂t ψ − L = 1/2 π² + 1/2 (∇ψ)² + Vψ(x) ψ².

The total Hamiltonian is

Plain text:  
H[ψ,π] = ∫ d³x [ 1/2 π² + 1/2 (∇ψ)² + Vψ(x) ψ² ].

### Hamilton’s equations ⇒ Euler–Lagrange dynamics

Equal-time Poisson brackets (pre-quantization) are

Plain text:  
{ψ(x,t), π(y,t)} = δ³(x−y); {ψ,ψ} = {π,π} = 0.

Hamilton’s equations:

Plain text:  
∂t ψ = δH/δπ = π,  
∂t π = −δH/δψ = ∇² ψ − 2 Vψ(x) ψ.

Combining gives the ψ field equation:

Plain text:  
∂t² ψ = ∇² ψ − 2 Vψ(x) ψ = ∇² ψ + 2 (∇²[ space(x)+current(x)² ]) ψ.

which matches Part 1.

### Canonical stress-energy (Noether form)

For time translations, the conserved energy density equals :

Plain text:  
E-density = 1/2 π² + 1/2 (∇ψ)² + Vψ(x) ψ².

The (symmetric) spatial momentum density can be taken as

Plain text:  
P = π ∇ψ.

When is static (no explicit time dependence), is conserved.

### Boundary conditions and positivity

For fields decaying sufficiently fast (or periodic boundaries), integrations by parts generating terms are well-defined.

Positivity: with sign-indefinite (because it is a Laplacian of a prescribed profile), stability requires the spectral condition that the operator has nonnegative spectrum in the regime of interest (as in Phase 10). This constrains admissible and .

### Discrete 1D Hamiltonian check (energy conservation)

I use a staggered-leapfrog scheme and monitor total energy.

# simulations/phase12\_part2\_hamiltonian\_energy.py  
import numpy as np  
  
# Grid and time  
Nx = 400  
Lx = 20.0  
dx = Lx / Nx  
x = np.linspace(-Lx/2, Lx/2 - dx, Nx)  
dt = 0.005  
steps = 4000  
report\_every = 400  
  
# Profiles: space(x), current(x)  
space = np.exp(-0.2 \* x\*\*2) # smooth bump  
current = 0.6 \* np.sin(0.6 \* x) # smooth wind  
Phi = space + current\*\*2  
  
# Discrete Laplacian (periodic)  
def lap(f):  
 return (np.roll(f, -1) - 2.0\*f + np.roll(f, 1)) / dx\*\*2  
  
# Effective potential Vψ = -∇²(space + current^2)  
Vpsi = -lap(Phi)  
  
# Initial conditions  
psi = np.exp(-0.5 \* x\*\*2)  
pi = np.zeros\_like(x)  
  
# Energy functional: E = ∑ [ 1/2 π^2 + 1/2 (∂x ψ)^2 + Vψ ψ^2 ] dx  
def energy(psi, pi):  
 dpsi = (np.roll(psi, -1) - psi) / dx  
 dens = 0.5 \* pi\*\*2 + 0.5 \* dpsi\*\*2 + Vpsi \* psi\*\*2  
 return np.sum(dens) \* dx  
  
E0 = energy(psi, pi)  
  
# Leapfrog: pi half-step, psi full-step  
pi -= 0.5 \* dt \* ( -lap(psi) + 2.0\*Vpsi\*psi )  
for n in range(1, steps + 1):  
 # ψ update  
 psi += dt \* pi  
 # π update  
 pi += dt \* ( -lap(psi) + 2.0\*Vpsi\*psi )  
  
 if n % report\_every == 0:  
 En = energy(psi, pi)  
 rel = (En - E0) / (abs(E0) + 1e-14)  
 print(f"step={n:5d} E={En:+.8e} ΔE/E0={rel:+.3e}")