```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulation [
<!ENTITY Npts     "64">
<!ENTITY Nsamples    "64">
<!ENTITY L  "3e-5">
]>
<simulation xmds-version="2">
  <name>GPE_1D_spinor_course</name>

  <author> Sebastian Wuester </author>
  <description>
    Gross-Pitaevskii-equation in 1D, SI units
  </description>

  <geometry>
    <propagation_dimension> t </propagation_dimension>
    <transverse_dimensions>
      <dimension name="x" lattice="&Npts;"  domain="(-&L;, &L;)" />
      <dimension name="n" type="integer" lattice="3"  domain="(0,2)" aliases="k"/>
      <!-- Assignment of components
             0: mF= -1
             1: mF=  0
             2: mF= +1  -->
     </transverse_dimensions>
  </geometry>

  <features>
    <benchmark />
    <auto_vectorise />
    <fftw />
    <globals>
      <![CDATA[
      const double hbar = 1.05457266e-34;
      const double omega = 10.0*(2.0*M_PI);
      const double omega_perp = 200.0*(2.0*M_PI);

      // Rb 87
      const double mass = 1.4432e-25;
      const double a0 = 5.5e-9;
      const double a2 = 5.5e-9/101.8*100.4;

      //dervied quantities
      const double Natoms = 100.0;

      const double Us = 4.0*M_PI*hbar*hbar*(a0 + 2.0*a2)/3.0/mass;
      const double Ua = 4.0*M_PI*hbar*hbar*(a2 - a0)/3.0/mass;

      const double sigmascale = 2.3;
      const double sigma = sigmascale*sqrt(hbar/mass/omega);
      const double sigma_perp = sqrt(hbar/mass/omega_perp);
      const double U1ds = Us/(2.0*M_PI*sigma_perp*sigma_perp);
      const double U1da = Ua/(2.0*M_PI*sigma_perp*sigma_perp);

      const double normfact = pow(M_PI*sigma*sigma,-0.25);

      ]]>
    </globals>
  </features>

  <vector name="wavefunction" initial_space="x n" type="complex">
    <components>psi</components>
    <initialisation>
      <![CDATA[
         double coeff = 0.0;

         if(n == 0)
            coeff = sqrt(0.1);
         if(n == 1)
            coeff = sqrt(0.8);
         if(n == 2)
            coeff = sqrt(0.1);

         psi = normfact*coeff*sqrt(Natoms)*exp(-0.5*x*x/sigma/sigma);
      ]]>
    </initialisation>
  </vector>

  <vector name="potentials" initial_space="x n" type="real">
    <components>trap</components>
    <initialisation>
      <![CDATA[
        trap=0.5*mass*omega*omega*x*x;
      ]]>
    </initialisation>
  </vector>

  <computed_vector name="auxiliary" dimensions="x" type="real">
    <components> totdens </components>
      <evaluation>
      <dependencies basis="x n"> wavefunction </dependencies>
      <![CDATA[
          totdens = mod2(psi);
      ]]>
    </evaluation>
  </computed_vector>


  <vector name="coupling" type="real" dimensions="n k">
    <components> F1 </components>
    <initialisation>
      <![CDATA[
          F1 = 0.0;

        // dpsi(-1)/dt
          if(n==0 && (k==0 || k==1))
            F1 = 1.0;
          if(n==0 && k ==2)
            F1 = -1.0;

        // dpsi(0)/dt
          if(n==1 && (k==0 || k==2))
            F1 = 1.0;

        // dpsi(+1)/dt
          if(n==2 && (k==1 || k==2))
            F1 = 1.0;
          if(n==2 && k ==0)
            F1 = -1.0;

      ]]>
    </initialisation>
  </vector>
  <computed_vector name="auxiliary2" dimensions="x n" type="real">
    <components> asymmdens </components>
      <evaluation>
      <dependencies basis="x n k"> wavefunction coupling</dependencies>
      <![CDATA[
          asymmdens = F1*mod2(psi( n=>k ));
      ]]>
    </evaluation>
  </computed_vector>


  <sequence>
    <integrate algorithm="ARK89" interval="4" tolerance="1e-8">
      <samples>200 200 200</samples>
      <operators>
        <operator kind="ip">
          <operator_names>L</operator_names>
          <![CDATA[
           L = -i*0.5*hbar*kx*kx/mass;
           ]]>
        </operator>
        <integration_vectors>wavefunction</integration_vectors>
        <dependencies>potentials auxiliary auxiliary2</dependencies>
        <![CDATA[
         complex conversion = 0.0;

         if(n == 0)
            conversion = conj(psi(n=>2))*psi(n=>1)*psi(n=>1);
         if(n == 1)
            conversion = 2.0*conj(psi)*psi(n=>n+1)*psi(n=>n-1);
         if(n == 2)
            conversion = conj(psi(n=>0))*psi(n=>1)*psi(n=>1);

         dpsi_dt =  L[psi] - i*( (U1ds*totdens + trap )*psi
                        + U1da*(asymmdens*psi   + conversion)    )/hbar;

         ]]>
      </operators>
    </integrate>
  </sequence>

  <output format="hdf5">
    <group>
      <sampling basis="x(&Nsamples;) n" initial_sample="yes">
        <moments>density psire psiim  </moments>
        <dependencies>wavefunction </dependencies>
        <![CDATA[
          density = mod2(psi);
          psire = psi.Re();
          psiim = psi.Im();
        ]]>
      </sampling>
    </group>
    <group>
      <sampling basis="kx(&Npts;) n" initial_sample="yes">
        <moments>fspec</moments>
        <dependencies>wavefunction</dependencies>
        <![CDATA[
          fspec = mod2(psi);
        ]]>
      </sampling>
    </group>
    <group>
      <sampling basis="kx(0) n" initial_sample="yes">
        <moments>population</moments>
        <dependencies>wavefunction</dependencies>
        <![CDATA[
          population = mod2(psi);
        ]]>
      </sampling>
    </group>
  </output>
</simulation>
```