

FINITE-DIFFERENCE TIME-DOMAIN (FDTD) METHOD

Simulation of the 1d time-dependent Schrodinger equation ('real-time, real-space' method)

$$i\hbar \frac{\partial \Psi(x, t)}{\partial t} = -\frac{\hbar^2}{2m} \Delta \Psi(x, t) + U(x) \Psi(x, t)$$

$$\Psi(x, t) = \Psi(n \cdot \Delta x, m \cdot \Delta t) = \Psi^m(n)$$

Leapfrogging technique между Re и Im членами:

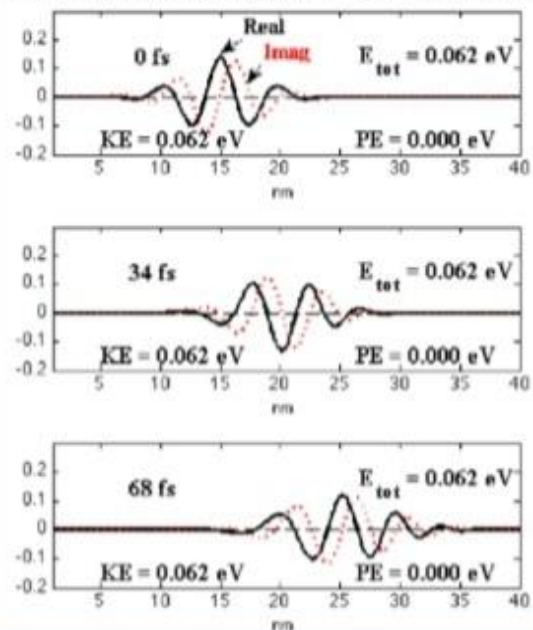
$$\Psi_{Real}^{m+1}(n) = \Psi_{Real}^m(n) - \frac{\hbar}{2m} \frac{\Delta t}{(\Delta x)^2} \left[\Psi_{Im}^{m+\frac{1}{2}}(n-1) - 2\Psi_{Im}^{m+\frac{1}{2}}(n) + \Psi_{Im}^{m+\frac{1}{2}}(n+1) \right] + \frac{\Delta t}{\hbar} U(n) \Psi_{Im}^{m+\frac{1}{2}}(n);$$

$$\Psi_{Im}^{m+\frac{3}{2}}(n) = \Psi_{Im}^{m+\frac{1}{2}}(n) + \frac{\hbar}{2m} \frac{\Delta t}{(\Delta x)^2} \left[\Psi_{Real}^{m+1}(n-1) - 2\Psi_{Real}^{m+1}(n) + \Psi_{Real}^{m+1}(n+1) \right] - \frac{\Delta t}{\hbar} U(n) \Psi_{Real}^{m+1}(n);$$

$$\leq 0,15 \Leftrightarrow \text{Если } \Delta x = 0,1 \text{ нм} \Rightarrow \Delta t = 0,02e-15 = 0,02 \text{ фс.}$$

Упражнение

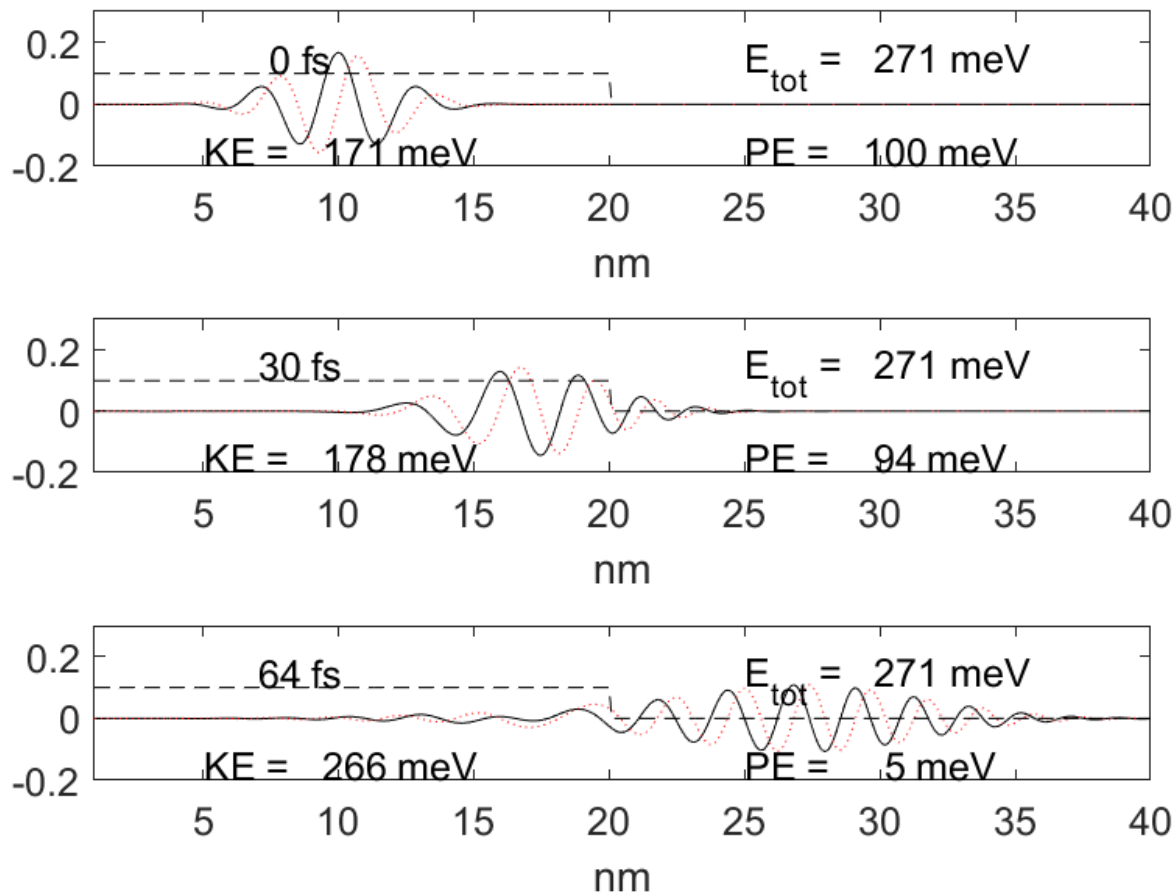
Моделирование движения частицы в свободном пространстве в положительном направлении оси x. Время: $m = [1 \ 1700 \ 3400]$



Решение:

Задание 1:

Тестовое задание:



Код Matlab:

```
cla reset;
hold on;
load("constans.mat");
```

• Константы и значения

```
i = sqrt(-1);
Np = 400; % Количество точек в проблемном пространстве Np = 400;
%5Np = 250;
T = 0;
Nstep=1;
time = [1 1500 1700]; % 1 1699 1700
meff = 1; % эффективная масс: Si is 1.08, Ge is 0.067, GaAs is 0.55
m = meff * m0; % Масса электрона
dx = 0.1e-9; % Размер ячейки dx = 0.1e-9;
%dx = 0.2e-9;
dt = 2e-17; % Временные шаги dt = 0.2e-17;
%dt = 1e-16;
```

$$ra = \frac{\hbar}{2m} * \frac{\Delta t}{(\Delta x)^2}$$

```
ra = (hbar / (2 * m)) * (dt / (dx)^2) % Коэффициент номер 1, должен быть меньше < 0.15
```

```
ra = 0.1158
```

ra < 0.15 -> возможна связь

```
DX = dx * 1e9; % Целое количество нм
XX = (1:Np) * DX; % Длина в нм для построения графика
Dsquared = (diag(ones(1,Np-1),1) - 2 * diag(ones(1,Np)) + diag(ones(1,Np-1),-1)); % диагональ
```

• Потенциал

```
V = zeros(1,Np)'; % 0..- => ' => 0 |.. вертикальной сделает
V = [zeros(1,Np/2) + 0.1 zeros(1,Np/2)]'*eV2J; % половина нули половина eV2J
```

• Инициализируем синусоидальную волну в гауссовой огибающей

```
lambda = 30; % Длина волны импульса
lamda_test = lambda * dx * 1e9
```

```
lamda_test = 3
```

```
sigma = 20; % Ширина импульса
nc = 100; % Исходное положение 150
n=(1:Np)';
```

$$P = e^{-\frac{(n - \text{исходное положение})^2}{2 * \text{ширина импульса}^2}} * \left(\cos\left(2\pi \frac{n - \text{исходное положение}}{\text{длина волны}}\right) + i * \sin\left(2\pi \frac{n - \text{исходное положение}}{\text{длина волны}}\right) \right)$$

```
Pulse = exp(-0.5*((n-nc)/sigma).^2).*(cos(2*pi*(n-nc)/lambda) + i*sin(2*pi*(n-nc)/lambda));
PulseReal = real(Pulse); % Реальная часть импульса
PulseImag = imag(Pulse); % Мнимая часть импульса
PulseZ = PulseReal + i*PulseImag; % И мнимая и реальная часть импульса
```

• Нормируем её и проверяем

$$N = \frac{\vec{n}}{|\vec{n}|}$$

```
PulseNormal = sqrt(sum(PulseZ.*conj(PulseZ))); % Константа нормализации
PulseReal = PulseReal / PulseNormal;
PulseImag = PulseImag / PulseNormal;
PulseZ = PulseReal + i*PulseImag;
control = sqrt(sum(PulseZ.*conj(PulseZ)));
PulseNormal = sqrt(sum(PulseZ.*conj(PulseZ))) % Константа нормализации
```

PulseNormal = 1

- PDTD

```
for n_step = 1:3
```

- Главные вычисления

```
for mm=1:time(n_step)
    T = T + 1;
```

$$\Psi_{\text{Real}}(n) = \Psi_{\text{Real}}^m(n) - \frac{\hbar}{2m} * \frac{\Delta t}{(\Delta x)^2} * \Psi_{\text{Imag}}[(n-1) - 2 * (n) + (n+1)] + \frac{\Delta t}{\hbar} U(n) \Psi_{\text{Imag}}$$

```
PulseReal = PulseReal - ra*Dsqared*PulseImag + (dt/hbar)*V.*PulseImag;
```

$$\Psi_{\text{Imag}}(n) = \Psi_{\text{Imag}}^m(n) + \frac{\hbar}{2m} * \frac{\Delta t}{(\Delta x)^2} * \Psi_{\text{Real}}[(n-1) - 2 * (n) + (n+1)] - \frac{\Delta t}{\hbar} U(n) \Psi_{\text{Real}}$$

```
    PulseImag = PulseImag + ra*Dsqared*PulseReal - (dt/hbar)*V.*PulseReal;
end
subplot(3,1,Nstep)
plot(XX,PulseReal,'k','Color','black')
hold on
plot(XX,PulseImag,':k','Color','red')
plot(XX,J2eV*V,'--k') % потенциальная энерги
hold off
axis( [ 1 DX*Np -0.2 0.3 ])
xlabel('nm')
set(gca,'fontsize',12)
Nstep=Nstep+1;
```

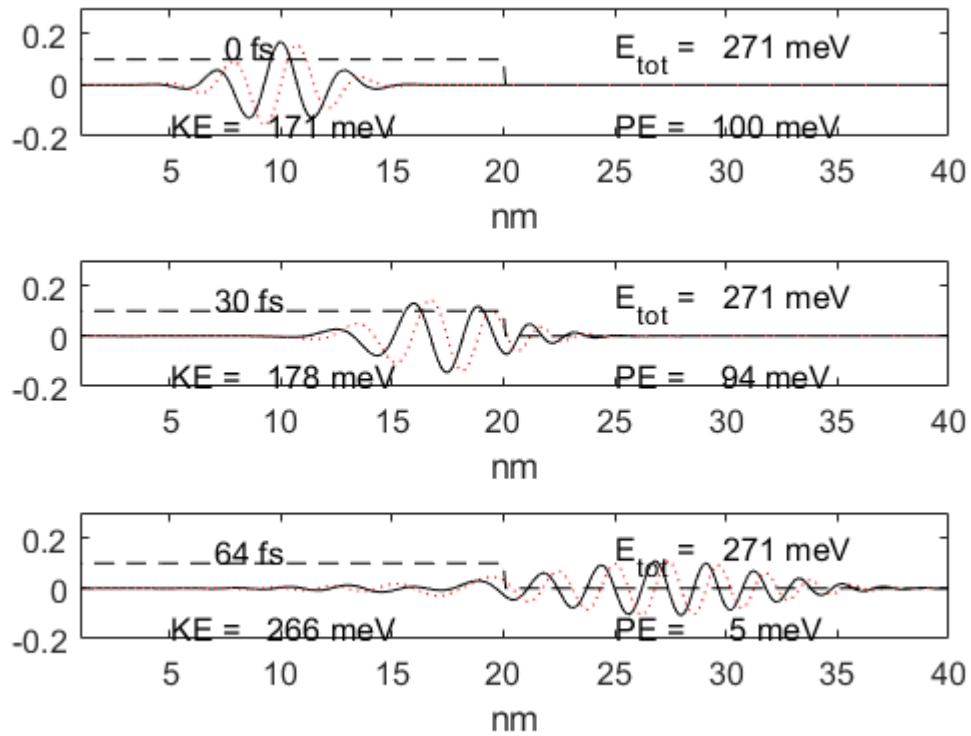
- Подписываем график

```
% ----- Проверяем нормализацию -----
PulseZ = PulseReal+i*PulseImag;
PulseZ'*PulseZ
PulseNormal = sqrt(sum(PulseZ.*conj(PulseZ)));
% ----- Вычислите ожидаемые значения -----
KE = J2eV * real(-(hbar/dx)^2/(2*m)*PulseZ'*(Dsqared*PulseZ)) *1e3; % Кинетическая энергия
PE = J2eV * PulseZ' * (V.*PulseZ)*1e3; % Потенциальная энергия
TT = text(5,.15,sprintf('%7.0f fs',T*dt*1e15));
set(TT,'fontsize',12)
TT = text(5,-.15,sprintf('KE = %5.0f meV',KE));
set(TT,'fontsize',12)
TT = text(25,-.15,sprintf('PE = %5.0f meV',PE));
set(TT,'fontsize',12)
TT = text(25,.13,sprintf('E_t_o_t = %5.0f meV',KE+PE));
set(TT,'fontsize',12)
```

- Сохраняем график

```
% saveas(gcf,'test.png')    % This saves the picture to a file
end
```

```
ans = 1.0000
ans = 1.0000
ans = 1.0000
```



- Время:

```
datetime('now')
```

```
ans = datetime
      16-Dec-2021 11:05:23
```