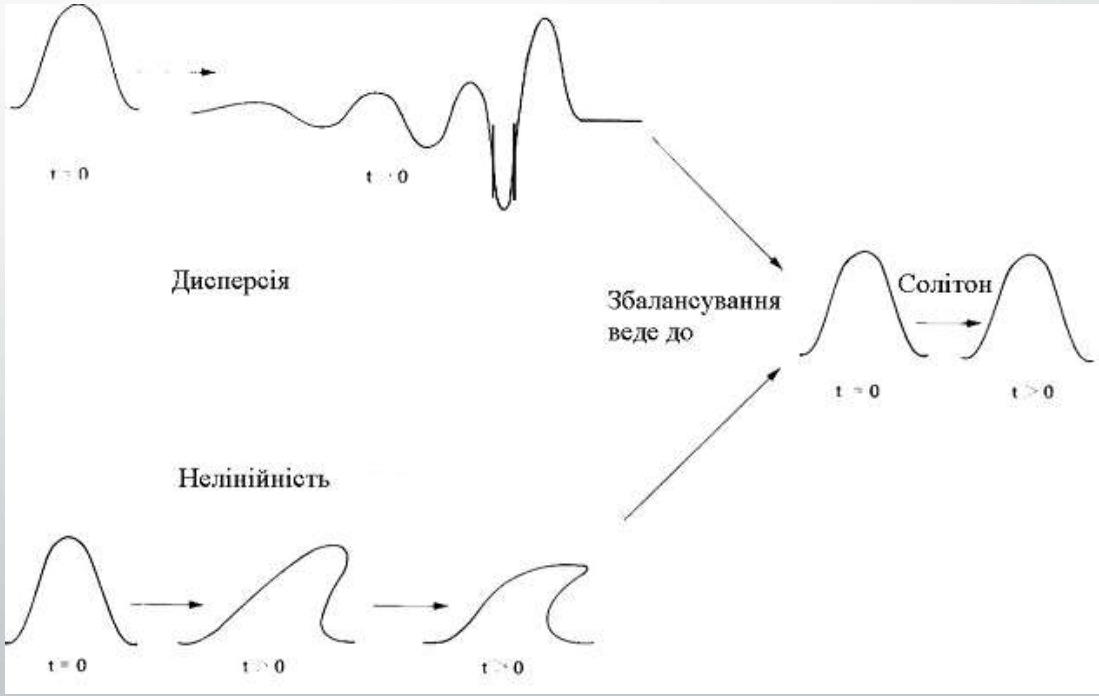
Дослідження засобів моделювання солітонних хвильових пакетів у волоконно-оптичних мережах

Виконав: Сідельников Назар, ДА-62
Науковий керівник: Чкалов О.В

ВСТУП

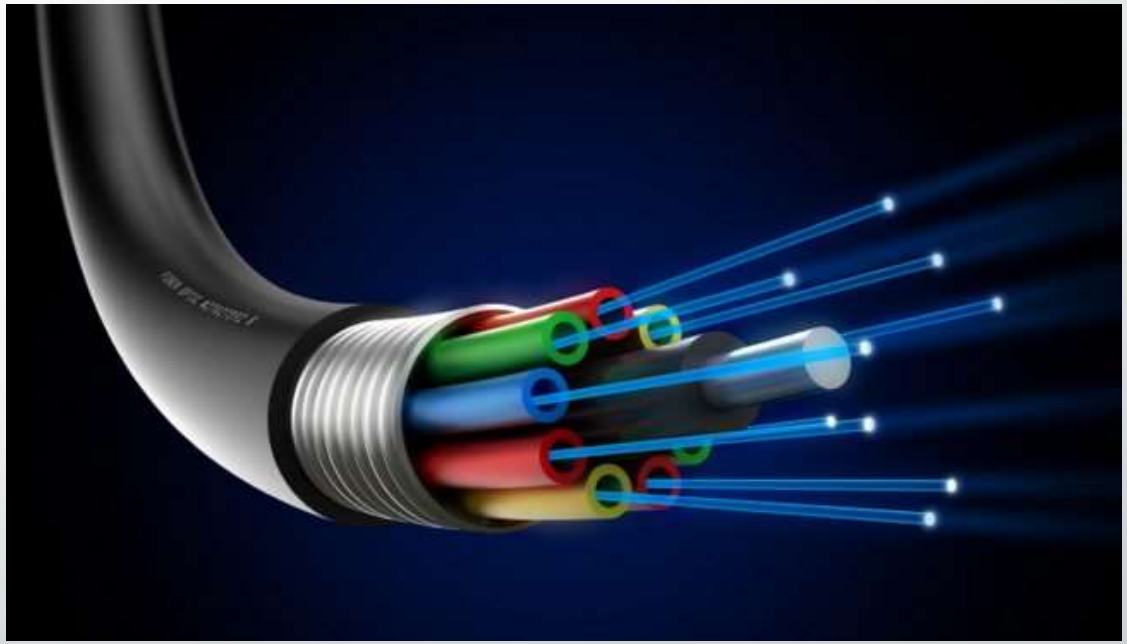
Солітон – це локалізована хвиля, що виникає завдяки балансу між нелінійними та дисперсійними властивостями середовища.

Мета роботи: розробка програмного забезпечення для моделювання розповсюдження солітона у середовищі з дефектом різними програмними засобами і порівняння їх ефективності.



Солітони в лініях зв’язку

Використання солітонів дозволяє досягти великою швидкості передачі даних по оптичному волокну і на великі відстані.



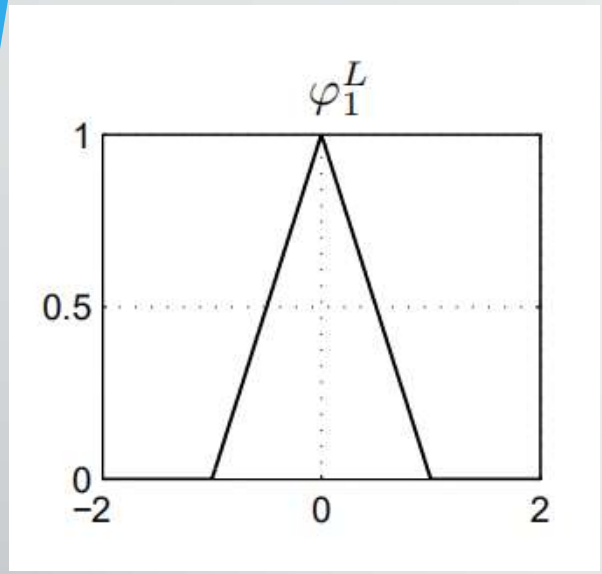
ПОСТАНОВКА ЗАДАЧІ МОДЕЛЮВАННЯ ТА ВИБІР ЧИСЕЛЬНИХ МЕТОДІВ

1) $$i \cdot u_t + \frac{1}{2}u_{xx} + |u|^2 u + \gamma\delta(x)u = 0; \quad (\gamma > 0).$$ (Нелінійне рівняння Шредінгера)

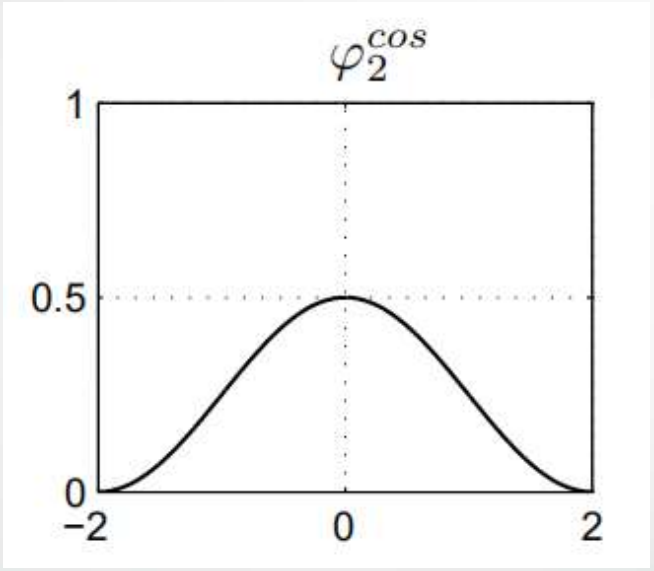
2) $$u_{sol}(x,t) = \eta\,\mathrm{sech}\big(\eta(x - vt)\big)e^{i(vx-\omega t)}$$ (Розв’язок НРШ за відсутністю дефетку)

3) $$u(\pm\infty) = 0$$ (Граничні умови)

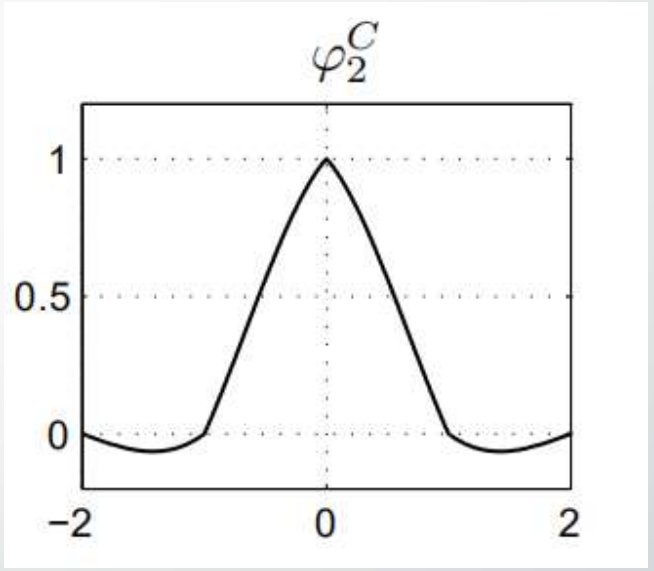
Інтерполяція дельта-функції.



- Трикутна



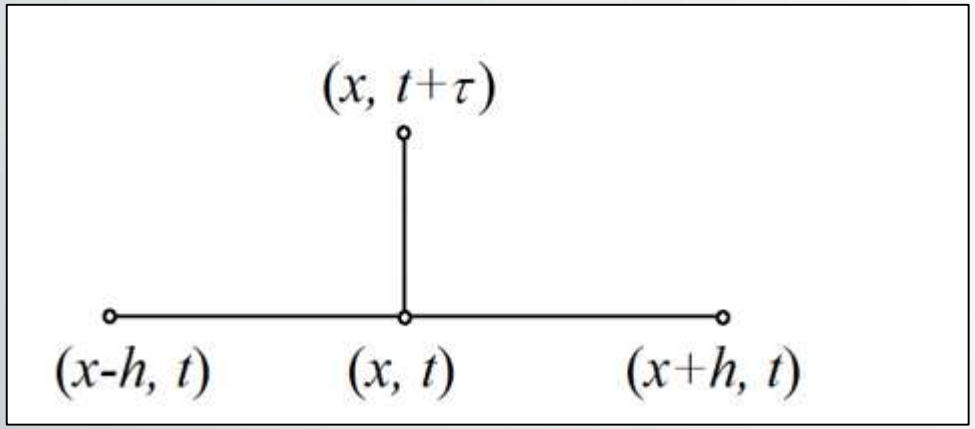
- Косинусна



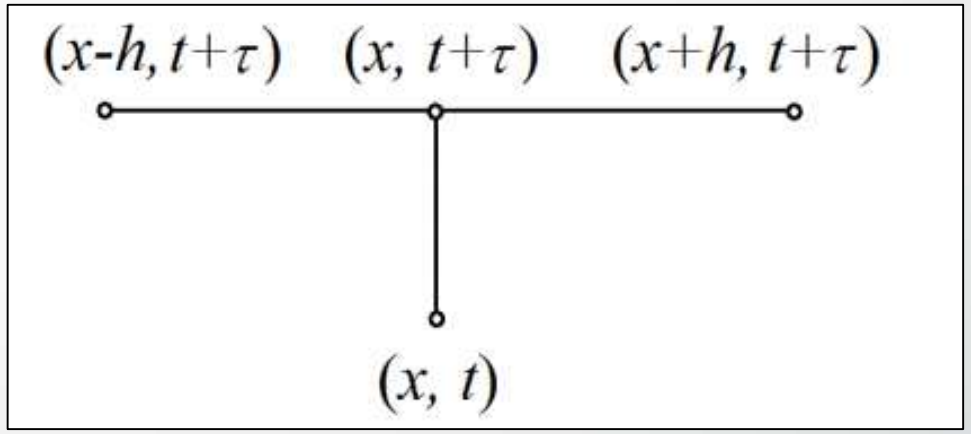
- Кубічна

$$\varphi_2^C(\xi) = \begin{cases} 1 - \frac{1}{2}|\xi| - |\xi|^2 + \frac{1}{2}|\xi|^3 & 0 \le |\xi| \le 1, \\ 1 - \frac{11}{6}|\xi| + |\xi|^2 - \frac{1}{6}|\xi|^3 & 1 < |\xi| \le 2. \end{cases}$$

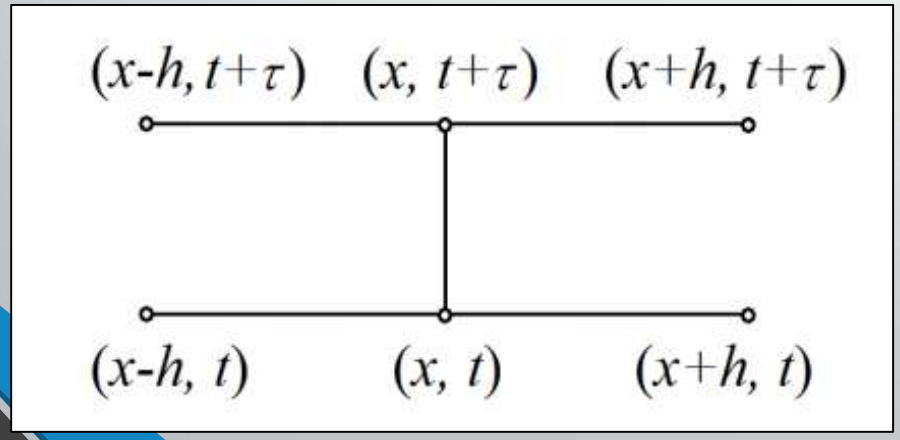
Сітки. Явна, неявна, явно-неявна схема.



- Явна схема



- Неявна схема



- Явно-неявна схема

Різницева апроксимація для НРШ

- Явна схема

$$iu_t - \frac{1}{2}u_{xx} \approx L_{h\tau}^{(0)}u = i\frac{u(x,t+\tau)-u(x,t)}{\tau} - \frac{u(x+h,t)-2u(x,t)+u(x-h,t)}{2h^2}$$

- Неявна схема

$$iu_t - \frac{1}{2}u_{xx} \approx L_{h\tau}^{(1)}u = i\frac{u(x,t+\tau)-u(x,t)}{\tau} - \frac{u(x+h,t+\tau)-2u(x,t+\tau)+u(x-h,t+\tau)}{2h^2}$$

- Явно-неявна схема

$$L_{h\tau}^{(\sigma)}u = \sigma L_{h\tau}^{(1)}u + (1-\sigma)L_{h\tau}^{(0)}u; \quad \sigma \in (0,1).$$

ВИБІР ІНСТРУМЕНТІВ ДЛЯ ОБРОБКИ ОТРИМАНИХ ДАНИХ

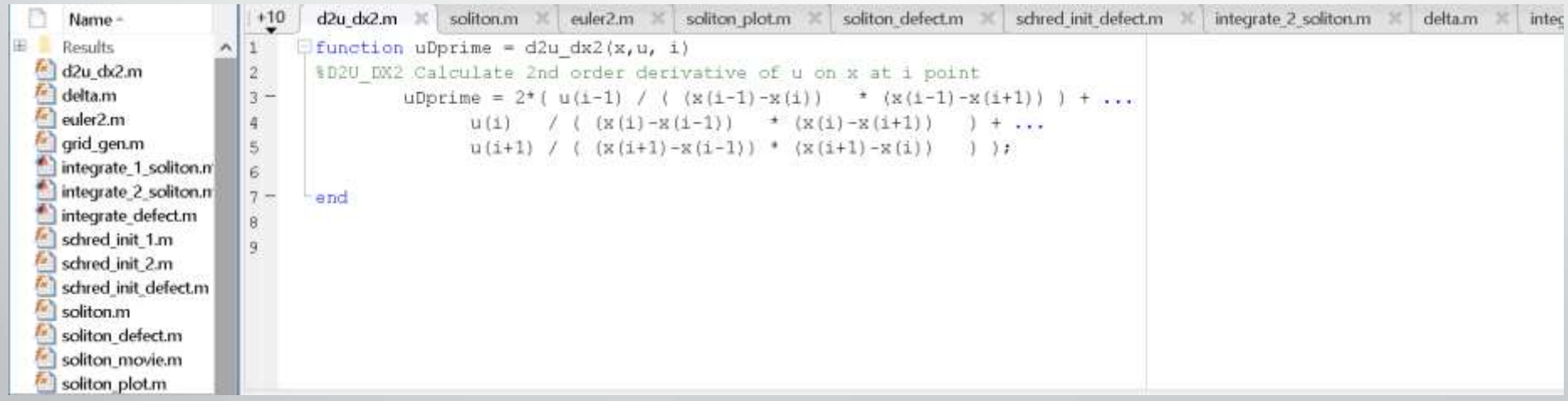
Існуючі програмні засоби:

- Maple
- Fortran
- MATLAB
- Wolfram Mathematica
- Python
- Comsol

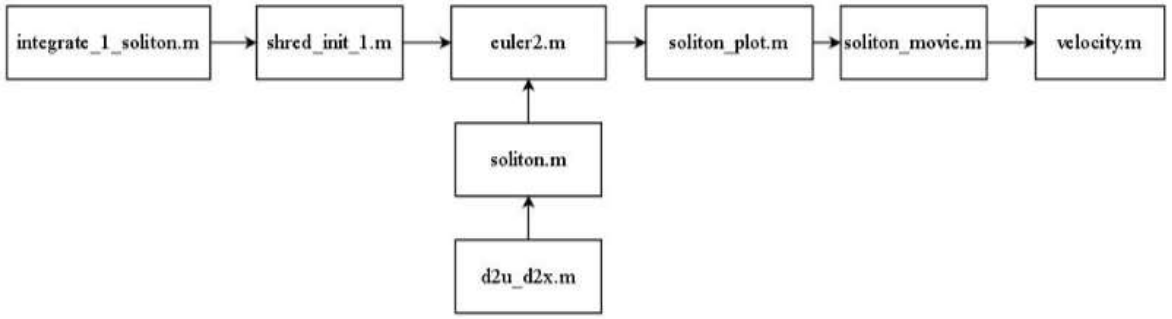
Для реалізації задачі було використано **MATLAB, Wolfram Mathematica, Python**

MATLAB. Побудова архітектури

- Кожний модуль потрібно реалізовувати в окремому файлі.

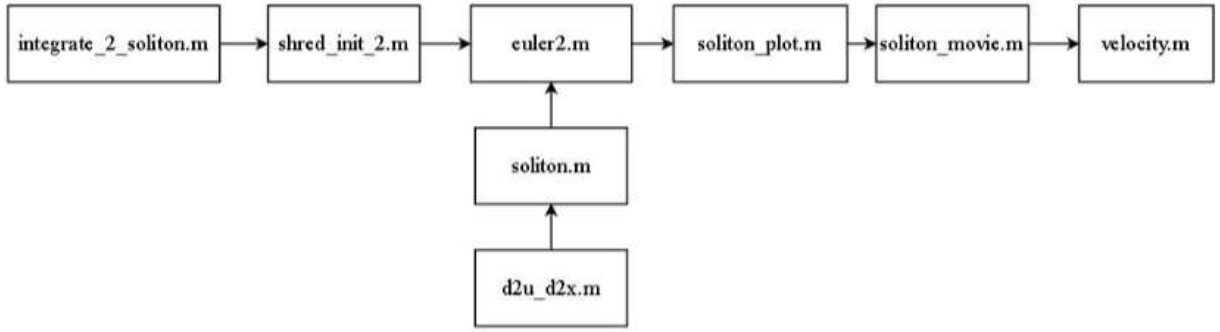


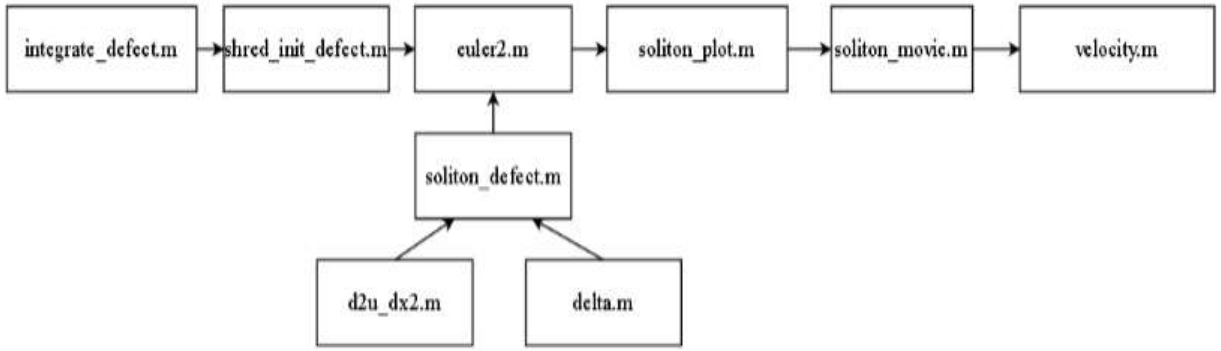
MATLAB. Взаємодія модулів.



Для одного солітону в середовищі без дефектом

Для двох солітонів в середовищі без дефекту





Для солітону в середовищі з дефектом

Mathematica Wolfram і Python. Побудова архітектури

```
File  Edit  Insert  Format  Cell  Graphics  Evaluation  Palettes  Window  Help

schredinit1[x_] :=
 Module[{eta = 0.5, vel = 0.5, x0 = 20}, xr = x - x0; eta * Exp[I * vel * xr] * Sech[eta * xr]]
 программный модуль

schredinitdefect[x_] :=
 Module[{eta = 0.5, vel = 4.0, x0 = -5}, xr = x - x0; eta * Exp[I * vel * xr] * Sech[eta * xr]]
 программный модуль

schredinit2[x_] :=
 Module[{eta1 = 0.5, eta = 0.5, vel1 = 1.0, vel2 = 0.025, x1 = 20, x2 = 45}, xr1 = x - x1; xr2 = x - x2;
 программный модуль
  eta1 * Exp[I * vel1 * xr1] * Sech[Eta1 * xr1] + eta2 * Exp[I * vel2 * xr2] * Sech[eta2 * xr2]]
     по  минимал единица  гиперболический секанс  по  минимал единица  гиперболический секанс

d2udx2[x_, u_, i_] :=
 Module[{}, 2 * (u[i - 1] / ((x[i - 1] - x[i]) * (x[i - 1] - x[i + 1])) + u[i] / ((x[i] - x[i - 1]) * (x[i] - x[i + 1
 программный модуль

delta1[x_, x0_, width_] :=
 Module[{},
 программный модуль
  mag = 1 / width;
  z = abs (x - x0) / width;
  If[z ≤ 1, value = mag * (1 - z / 2 - z^2 + z^3 / 2),
  условный оператор
   If[z ≤ 2, value = mah * (1 - 11 / 6 * z + z^2 - z^3 / 6), value = 0]
   условный оператор
```

Mathematica Wolfram

```
#SCHRED_INIT_1.M
#SCHRED_INIT Specifies the initial condition
# for a PDE in time and space dimension
#
#
# One solution

def schred_init_1(x):
    eta = 0.5
    vel = 0.5
    x0 = 20

    xr = x-x0
    for i in range(len(xr)):
        value = eta*cexp(1j*vel*xr[i])*sech(eta*xr[i])
        xr[i] = complex(value.real, value.imag)

    return xr
```

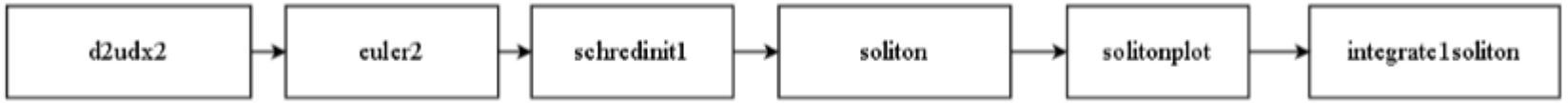
```
#SCHRED_INIT_2.M
#SCHRED_INIT Specifies the initial condition
#for a PDE in time and oe space dimension
#
#
#Two solitons

def schred_init_2(x):
    eta1 = 0.5
    eta2 = 0.5

    vel1 = 1.0
```

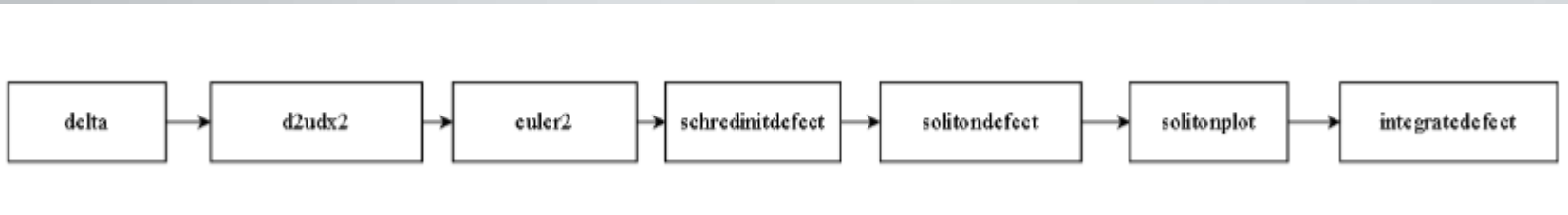
Python

Mathematica Wolfram. Взаємодія модулів.



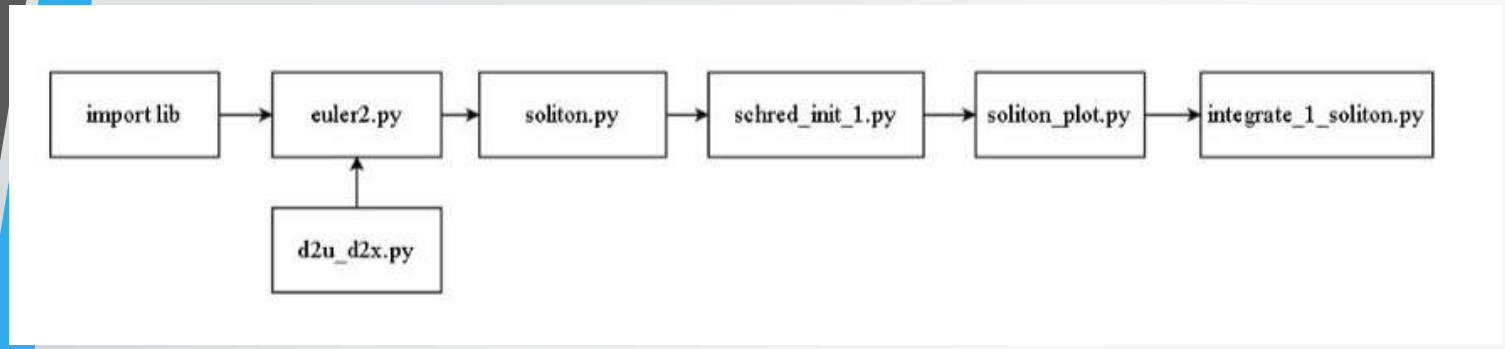
Для одного солітону в середовищі без дефектом

Для двох солітонів в середовищі без дефекту



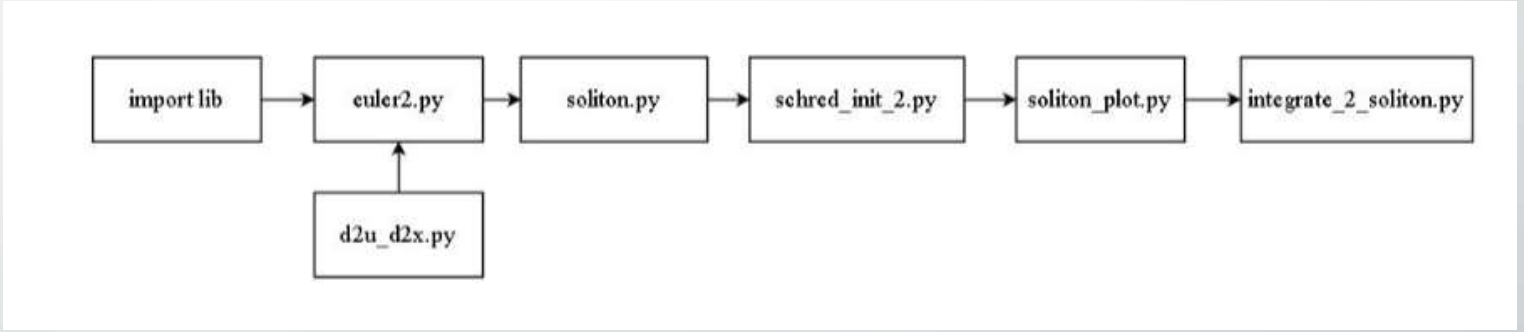
Для солітону в середовищі з дефектом

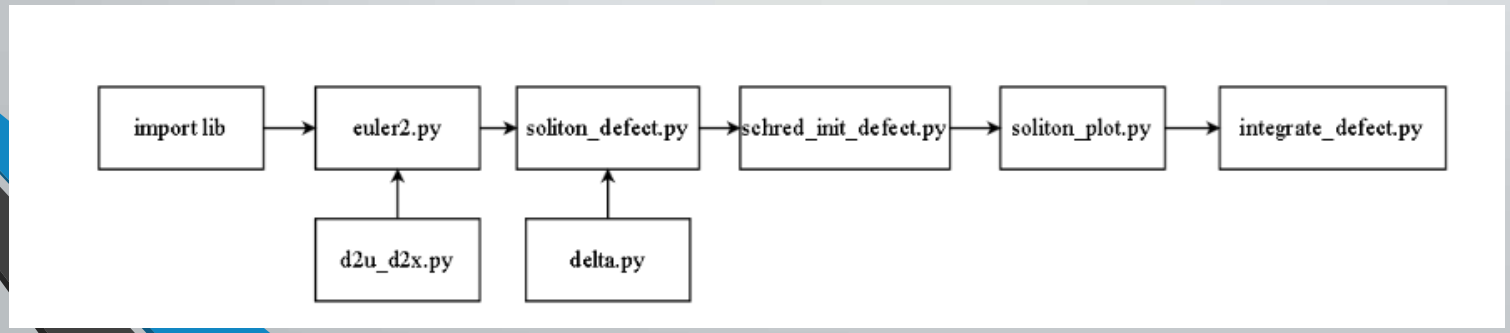
Python. Взаємодія модулів.



Для одного солітону в середовищі без дефектом

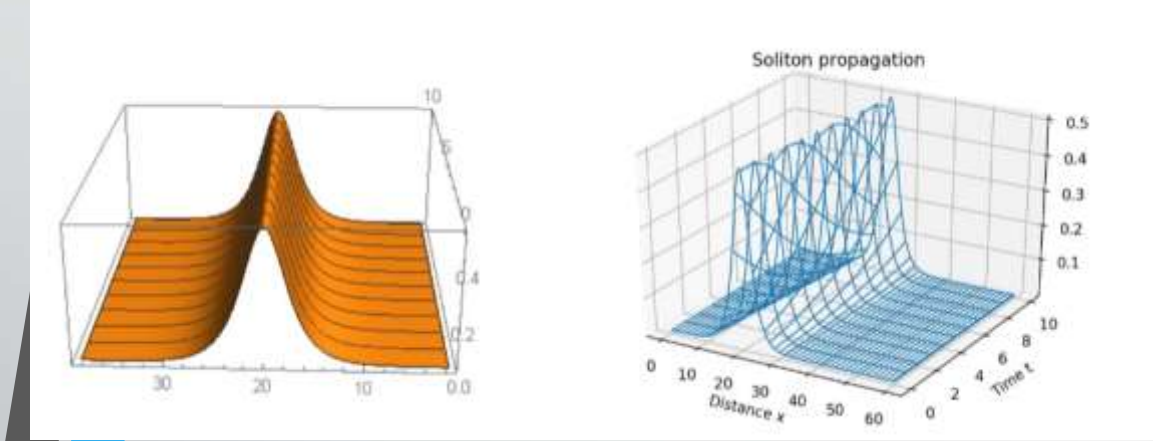
Для двох солітонів в середовищі без дефекту

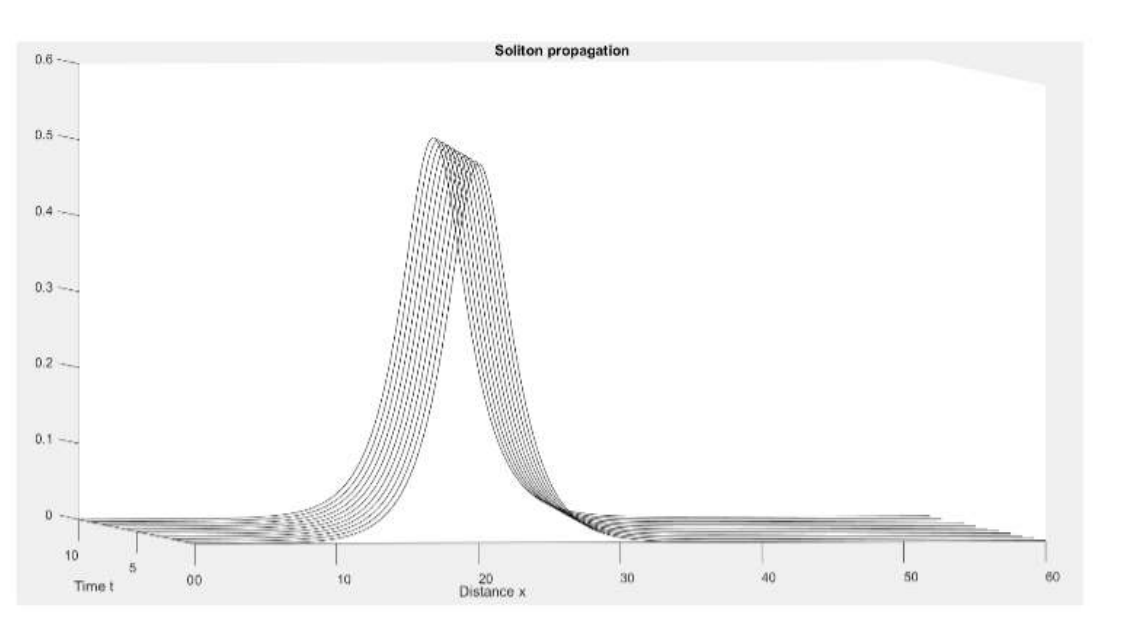




Для двох солітонів в середовищі без дефекту

Тестування. Розповсюдження одного солітона у зоні без дефекту.



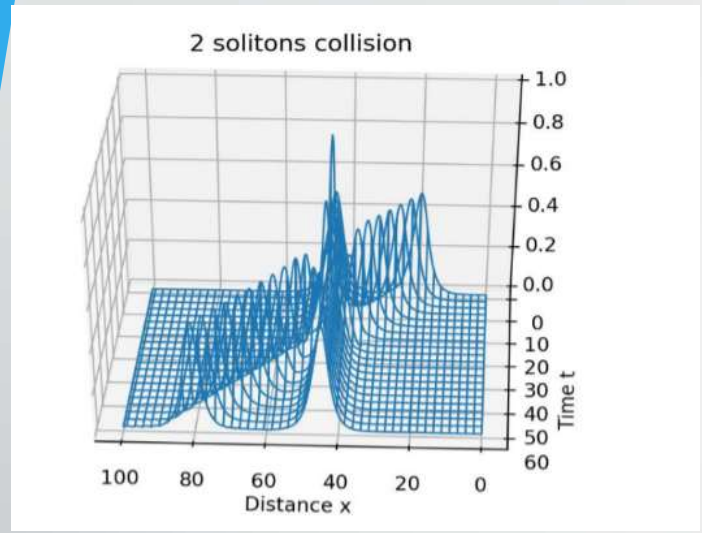


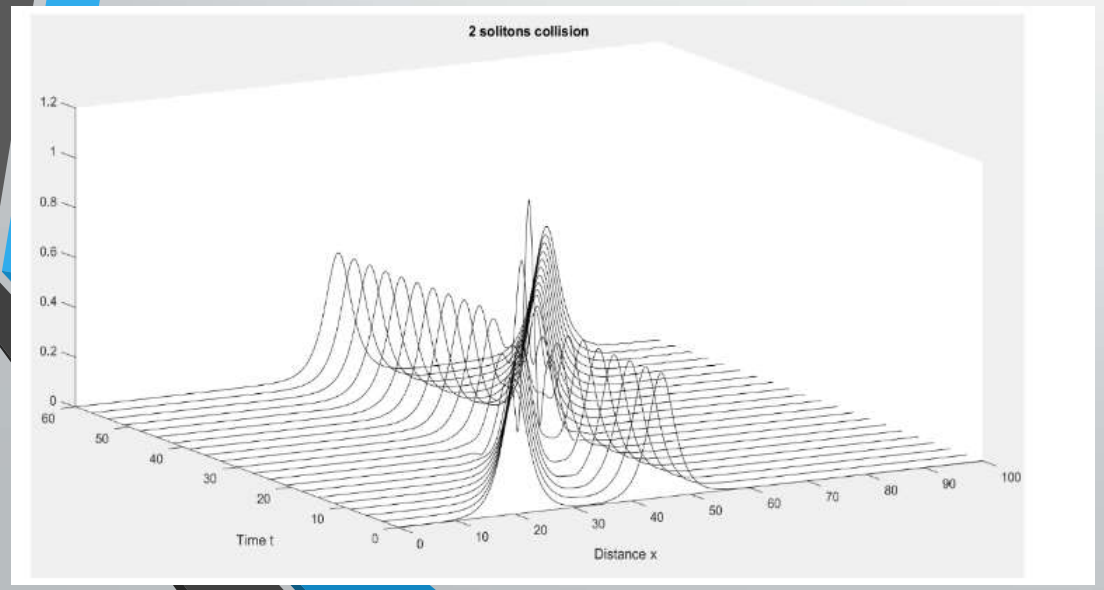
Integrate_1_soliton (MATLAB)		
Модулі	Виклики	Час
d2u_dx2	7980000	0 хв 30 с
euler2	100	1 хв 03 с
soliton	20000	0 хв 58 с
integrate_1_soliton	1	1 хв 10 с
Загально		1 хв 11 с

Integrate_1_soliton (Python)		
Модулі	Виклики	Час
d2u_dx2	7980000	1 хв 34 с
euler2	100	3 хв 04 с
soliton	20000	3 хв 02 с
integrate_1_soliton	1	3 хв 03 с
Загально		3 хв 04 с

Integrate_1_soliton (Wolfram)		
Модулі	Виклики	Час(хв,сек)
d2u_dx2	266734700	0 хв 44 сек
euler2	100	3 хв 40 сек
soliton	222650	3 хв 42 сек
integrate_1_soliton	1	3 хв 44 сек
Загально		3 хв 49 сек

Тестування. Розповсюдження двох солітонів у зоні без дефекту.



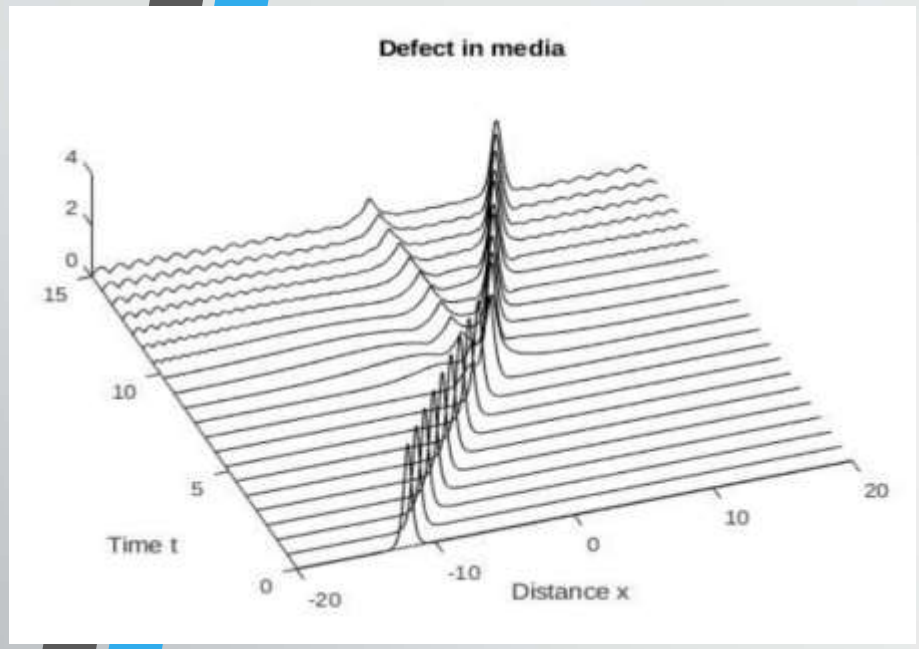


Integrate_2_soliton (MATLAB)		
Модулі	**Виклики**	**Час**
d2u_dx2	95760000	7 хв 42 с
euler2	200	15 хв 39 с
soliton	240000	14 хв 41 с
integrate_2_soliton	1	16 хв 10 с
Загально		16 хв 38 с

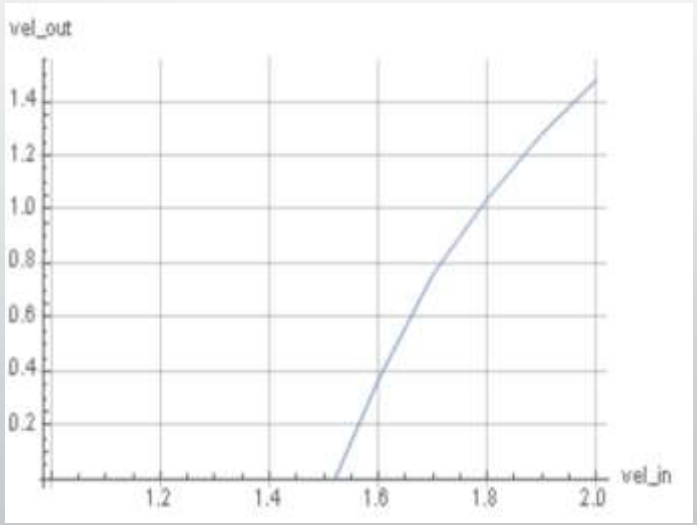
Integrate_2_soliton (Python)		
Модулі	**Виклики**	**Час**
d2u_dx2	47880000	08 хв 43 с
euler2	100	17 хв 26 с
soliton	120000	17 хв 18 с
integrate_2_soliton	1	17 хв 20 с
Загально		17 хв 27 с

Integrate_2_soliton (Wolfram)		
Модулі	**Виклики**	**Час**
d2u_dx2	56780000	09 хв 00 с
euler2	100	18 хв 11 с
soliton	120000	18 хв 10 с
integrate_defect	1	18 хв 23 с
Загально		18 хв 30 с

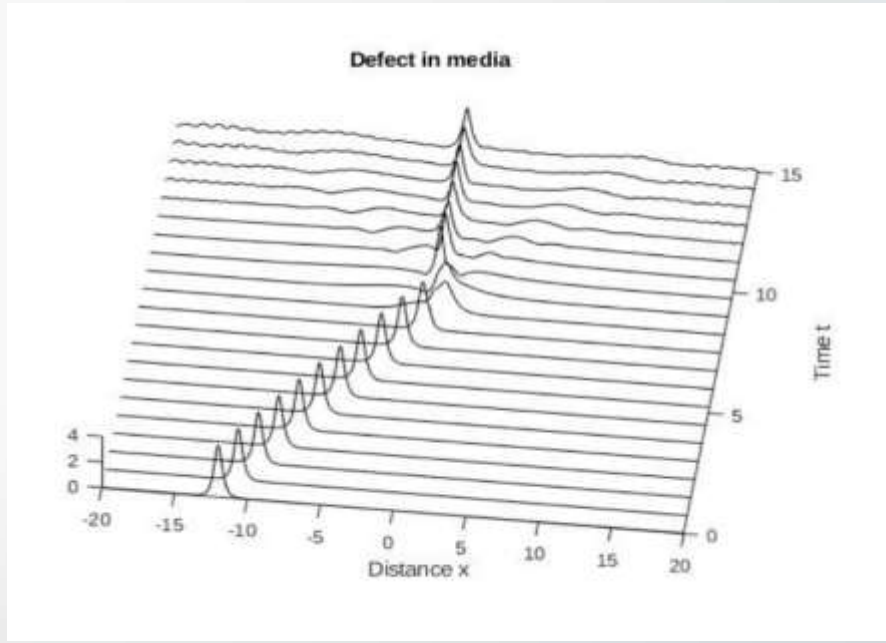
Проходження солітона через дефект.



- v =1.7, eta =4

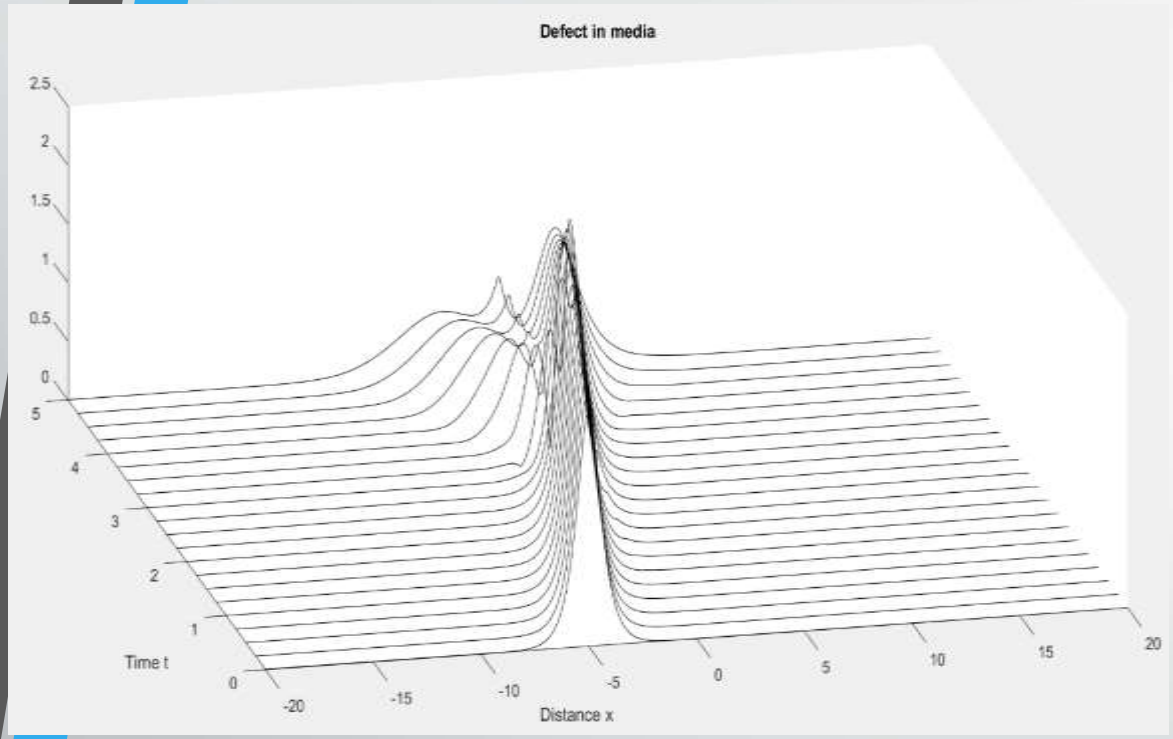


Вхідна та вихідна швидкість v

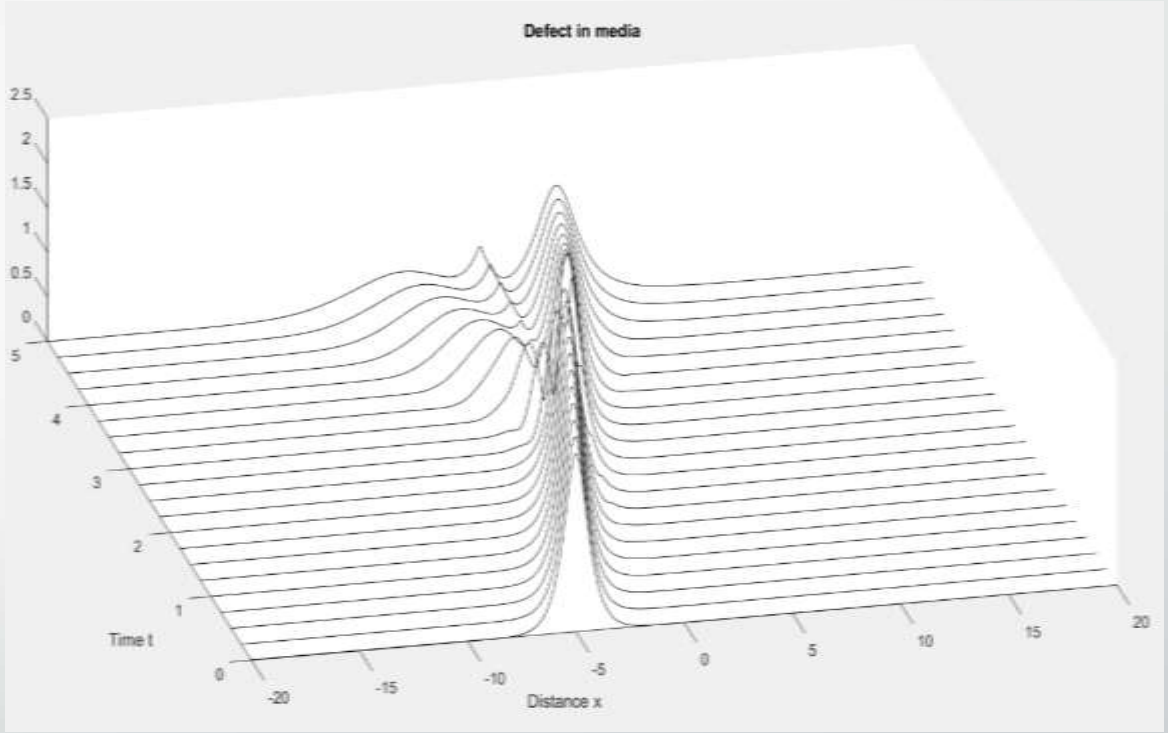


- v =1.5, eta =4.

Проходження солітона через дефект. Менша амплітуда.

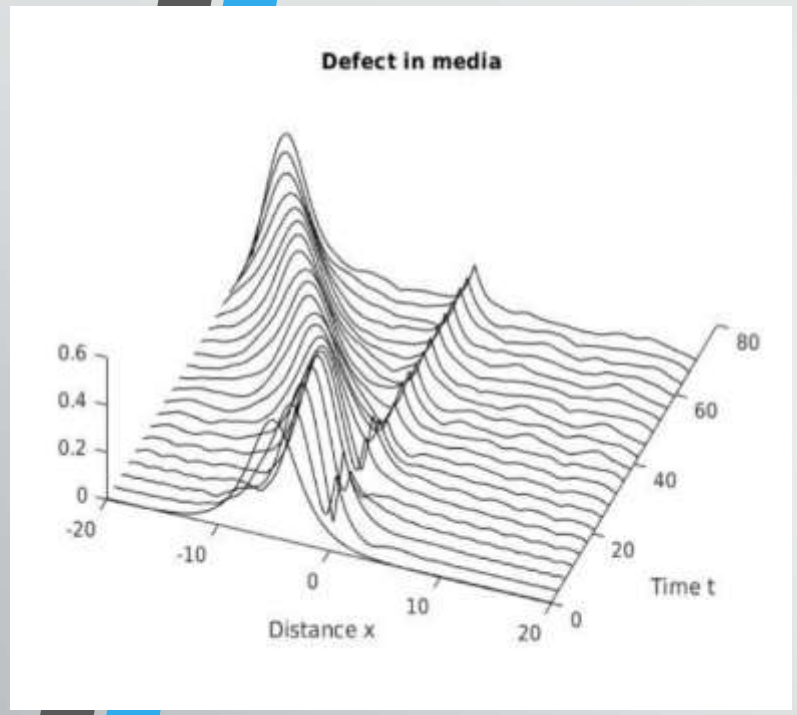


- *v =1.5, eta =2*

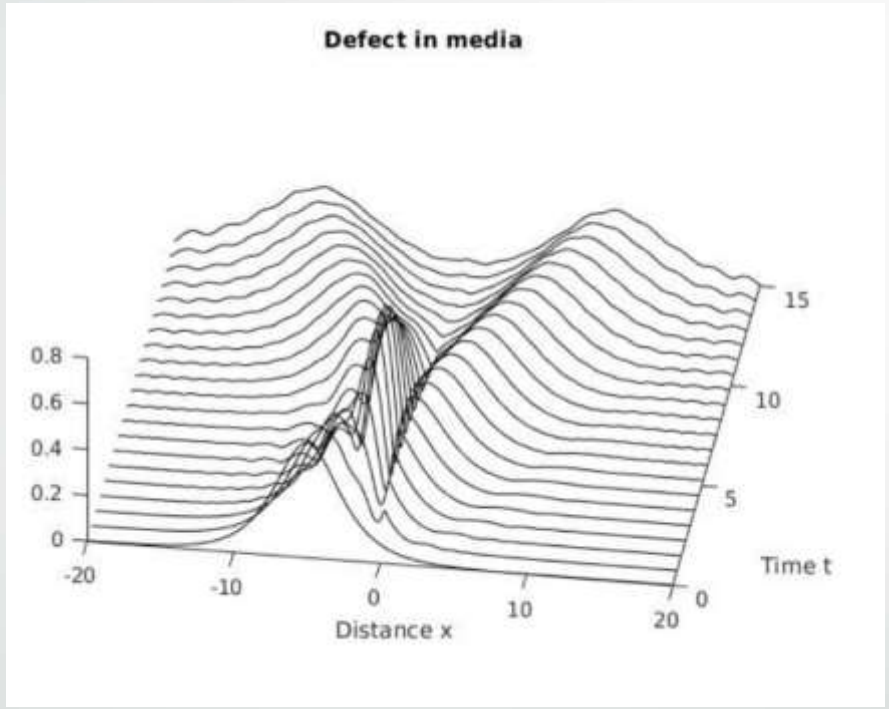


- *v =1.7, eta =2*

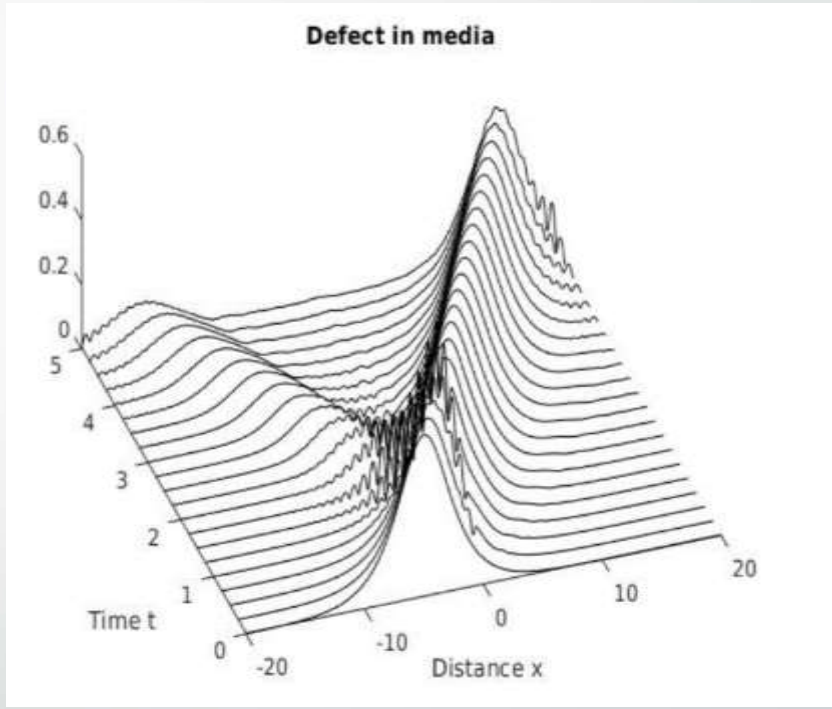
Проходження солітона через дефект. Мала амплітуда.



- *v =0.5, eta =0.5*



- *v =1, eta =0.5*



- *v =4, eta =0.5*

Результати моделювання

Integrate_defect (MATLAB)		
Модулі	Виклики	Час
d2u_dx2	438971160	26 хв 21 с
euler2	100	1 год 32 хв 32 с
soliton_defect	466420	1 год 21 хв 22 с
delta	348971160	22 хв 08 с
integrate_defect	1	1 год 32 хв 32 с
Загально		1 год 32 хв 43 с

Integrate_defect (Python)		
Модулі	Виклики	Час
d2u_dx2	266734700	51 хв 39 с
euler2	100	2 год 25 хв 04 с
soliton_defect	222650	2 год 24 хв 37 с
delta	266734700	22 хв 38 с
integrate_defect	1	2 год 24 хв 50 с
Загально		2 год 25 хв 4 с

Integrate_defect (Wolfram)		
Модулі	Виклики	Час
d2u_dx2	266734700	58 хв 39 с
euler2	100	2 год 35 хв 04 с
soliton	222650	2 год 35 хв 37 с
integrate_defect	266734700	2 год 35 хв 21 с
Загально	1	2 год 36 хв 30 с

Напрямки вдосконалення програмного забезпечення

1. Підвищення порядку апроксимації чи перехід від методу скінченних різниць до методу скінченних елементів.
2. Розглянути використання більш ефективних стандартних бібліотек для математичних розрахунків від сторонніх виробників.

Висновки по результатах роботи

1. Створено програмне забезпечення для моделювання поведінки солітона у середовищі з дефектом на основі запропонованої та з використанням обраних чисельних методів.
2. Виконання тестування та апробацію розробленого програмного забезпечення на тестових задачах.
3. Виконано обчислювальні експерименти з моделювання розповсюдження солітона в середовищі з дефектом.
4. Виконано порівняльний аналіз ефективності програмного забезпечення, розробленого в різних програмних середовищах.
5. Виконано функціонально-вартісний аналіз розробки.

Дякую за увагу!