Shallow-Water Solitons KdeV

Modelowanie komputerowe - Projekt 2

Katarzyna Gajewska
Michał Kreft
Tomasz Biel

Politechnika Krakowska
im. Tadeusza Kościuszki

11 styczeń 2022

Plan prezentacji

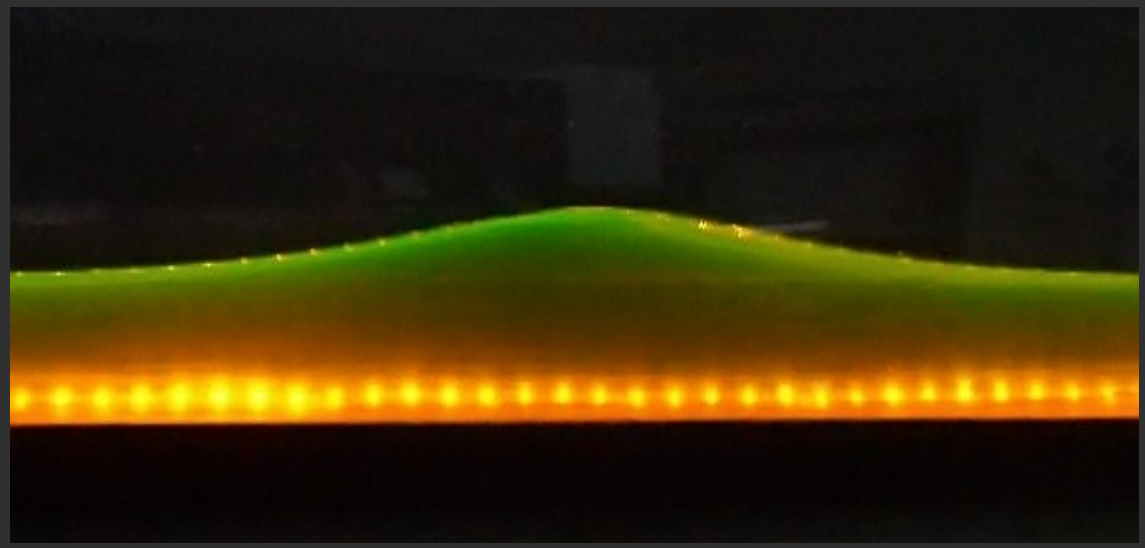
Wstęp teoretyczny

Solitony

W matematyce i fizyce soliton to samopodtrzymująca się odosobniona fala wywołana przez efekty nieliniowe występujące w materiale, w którym fala ta się rozchodzi. Solitony towarzyszą wielu zjawiskom fizycznym; pojawiają się też jako rozwiązania nieliniowych cząstkowych równań różniczkowych. Temat ten, choć zawiera pewne komplikacje, jest fascynujący i jest jedynym dla którego komputer był absolutnie niezbędny do odkrycia i zrozumienia.

Solitony

Zjawisko solitonu zostało po raz pierwszy opisane przez Johna Scotta Russella, który zaobserwował falę solitonu w kanale wodnym, a następnie odtworzył to zjawisko w specjalnie przygotowanym zbiorniku wodnym. Zaobserwowaną falę Russell nazwał „falą przesunięcia” (ang. wave of translation).



Rysunek 1: Soliton wytworzony w wodzie

Zdefiniowanie solitonu

Trudno precyzyjnie zdefiniować czym jest soliton. W 1989 roku zdefiniowano soliton jako rozwiązanie układu nieliniowych równań różniczkowych, które:

- reprezentuje fale o niezmiennym kształcie;
- jest zlokalizowane tak, że zanika lub osiąga stałą wartość w nieskończoności;
- może oddziaływać silnie z innymi solitonami, ale po kolizji zachowuje niezmienioną formę – występuje tylko przesunięcie fazy.

Wielu autorów podkreśla, że solitony mogą zmieniać swój kształt okresowo, a ich wyróżnikiem jest zdolność do kolizji niedestrukcyjnych. Znane są także solitony dwu oraz trójwymiarowe (tzw. pociski świetlne).

Równanie Kortewega i deVriesa

Chcemy zrozumieć niezwykłe fale wodne, które występują w płytkich, wąskich kanałach. Opis analityczny tego „nagromadzenia wody" podali Korteweg i deVries (KdeV) z równaniem różniczkowym cząstkowym:

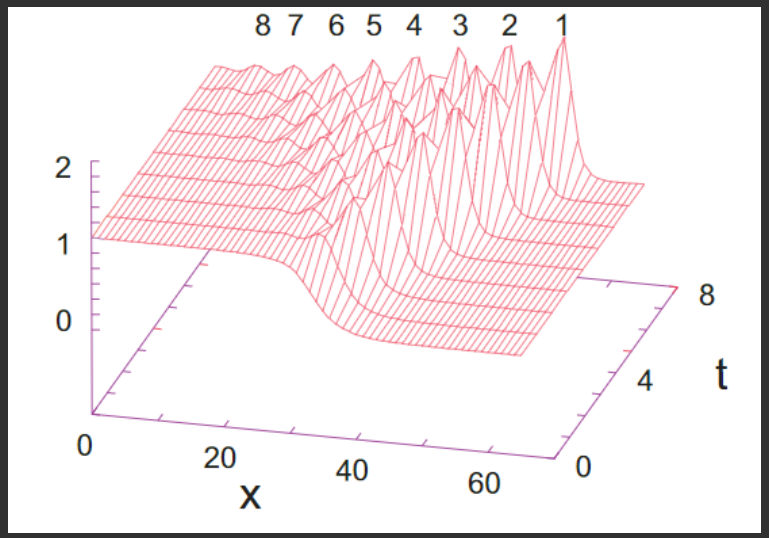
$$\frac{\partial u(x,t)}{\partial t} + \varepsilon u(x,t)\frac{\partial u(x,t)}{\partial x} + \mu\frac{\partial^3 u(x,t)}{\partial x^3} = 0 \tag{1}$$

Analiza teoretyczna

Jak wiadomo z równania Burgersa, nieliniowe wyrażenie $\varepsilon u \frac{\partial u}{\partial t}$ prowadzi do wyostrzenia fali i ostatecznie do fali uderzeniowej. W przeciwieństwie do tego co wiadomo z badania dyspersji, wyrażenie $\frac{\partial^3 u}{\partial x^3}$ powoduje poszerzenie. Te razem z wyrażeniem $\frac{\partial u}{\partial t}$ wytwarzają fale biegnące. Dla odpowiednich parametrów i warunków początkowych dyspersyjne poszerzenie dokładnie równoważy nieliniowe zwężenie i powstaje stabilna fala biegnąca.

Rozwiązanie równania KdeV

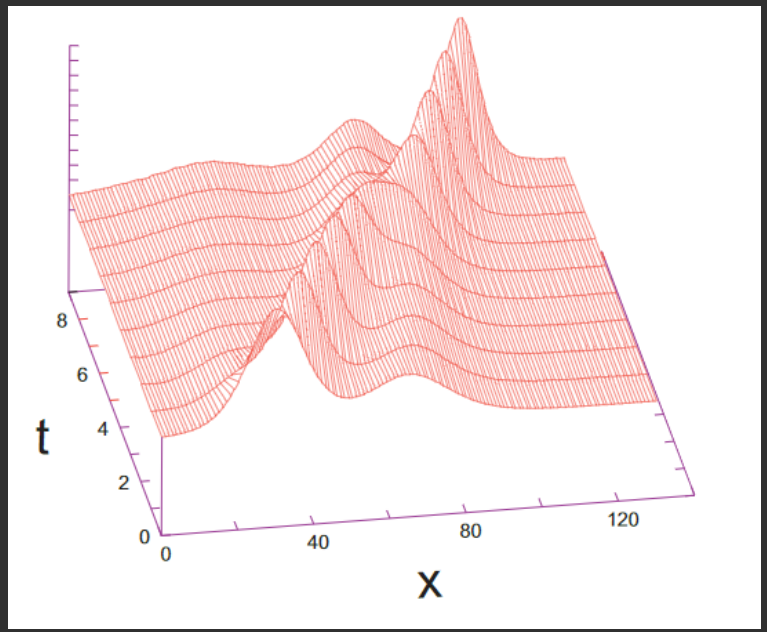
Równanie KdeV rozwiązano analitycznie i udowodniono, że prędkość podana przez Russella jest w rzeczywistości poprawna. Siedemdziesiąt lat po odkryciu równanie KdeV zostało ponownie odkryte przez Zabusky'ego i Kruskala, którzy rozwiązali je numerycznie i odkryli, że warunek początkowy $cos(\frac{x}{L})$ rozpadł się na osiem pojedynczych fal jak na rysunku poniżej.



Rysunek 2: Formowanie się fali tsunami

Wnioski

Odkryli oni również, że części fali o większych amplitudach poruszały się szybciej niż te o mniejszych amplitudach, dlatego wyższe szczyty mają tendencję do znajdowania się po prawej stronie (jak na Rysunku 2). Jakby cuda nigdy nie ustały, Zabusky i Kruskal, którzy wymyślili nazwę "soliton" dla samotnej fali, również zauważyli, że szybszy szczyt w rzeczywistości przeszedł przez wolniejszy bez szwanku (Rysunek 3 poniżej).



Rysunek 3: Skrzyżowane dwie fale solitonowe w płytkiej wodzie

Analityczne rozwiązanie solitonu

Podejście do zagadnienia

Sztuczka w podejściach analitycznych do tego typu równań nieliniowych polega na zastąpieniu zgadywanego rozwiązania, które ma postać fali biegnącej:

$$u(x,t) = u(\xi = x - ct)$$

Ta forma oznacza, że jeśli poruszamy się ze stałą prędkością c, zobaczymy stałą formę fali (ale teraz prędkość będzie zależeć od wielkości u). Nie ma gwarancji, że taka forma rozwiązania istnieje, ale jest to szczęśliwy traf, ponieważ podstawienie do równania KdeV daje rozwiązywalny ODE i jego rozwiązanie:

$$-c\frac{\partial u}{\partial \xi} + \epsilon u \frac{\partial u}{\partial \xi} + \mu \frac{d^3u}{d\xi^3} = 0,$$

$$u(x,t) = \frac{-c}{2} sech^2[\tfrac{1}{2}\sqrt{c}(x - ct - \xi_0)],$$

Gdzie ξ_0 jest fazą początkową. Widzimy w drugiej linijce amplitudę proporcjonalną do prędkości fali c oraz funkcję $sech^2$ dającą pojedynczą grudkowatą falę. Jest to typowa analityczna forma solitonu.

Algorytm dla solitonów KdeV

Schemat różnic skończonych

Równanie KdeV jest rozwiązywane numerycznie przy użyciu schematu różnic skończonych (finite-difference) z pochodnymi czasowymi i przestrzennymi podanymi przez przybliżenia różnic centralnych:

$$\frac{\partial u}{\partial t} \simeq \frac{u_{i,j+1}-u_{i,j-1}}{2\Delta t}, \frac{\partial u}{\partial x} \simeq \frac{u_{i+1,j}-u_{i-1,j}}{2\Delta x}$$

By przybliżyć $\frac{\partial^3 u(x,t)}{\partial x^3}$, rozszerzamy $u(x,t)$ do $O(\Delta t)^3$ o około 4 punkty $u(x \pm 2\Delta x, t)$ i $u(x \pm \Delta x, t)$,

$$u(x \pm \Delta x, t) \simeq u(x,t) \pm (\Delta x)\frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2}{\partial x^2} \pm \frac{(\Delta x)^3}{3!}\frac{\partial^3 u}{\partial x^3},$$

które rozwiązujemy dla $\frac{\partial^3 u(x,t)}{\partial x^3}$. Wreszcie, czynnik $u(x,t)$ w drugim członie równania (1) jest przyjmowany jako średnia trzech wartości x, z których wszystkie mają to samo t

$$u(x,t) \simeq \frac{u_{i+1,j}+u_{i,j}+u_{i-1,j}}{3}.$$

Omawianie algorytmu cd.

Te podstawienia dają algorytm dla równania KdeV:

$$u_{i,j+1} \simeq u_{i,j-1} - \frac{\epsilon}{3}\frac{\Delta t}{\Delta x}[u_{i+1,j} + u_{i,j} + u_{i-1,j}][u_{i+1,j} - u_{i-1,j}] - \mu\frac{\Delta t}{(\Delta x)^3}[u_{i+2,j} + 2ui - 1, j - 2u_{i+1,j} - u_{i-1,j}].$$

Aby zastosować ten algorytm do przewidywania przyszłych czasów, musimy znać $u(x,t)$ w czasach teraźniejszych i przeszłych. Rozwiązanie czasu początkowego $u_{i,1}$ jest znane dla wszystkich pozycji i przez warunek początkowy. By znaleźć $u_{i,2}$ używamy schematu różnic w przód (forward-difference scheme) w którym rozwijamy $u(x,t)$, zachowując tylko dwa wyrazy dla pochodnej po czasie:

$$u_{i,2} \simeq u_{i,1} - \frac{\epsilon\Delta t}{6\Delta x}[u + i + 1, 1 + u_{i,1} + u_{i-1,1}][u_{i+1,1} - u_{i-1,1}] - \frac{\mu}{2}\frac{\Delta t}{(\Delta x)^3}[u_{i+2,1} + 2u_{i-1,1} - 2u_{i+1,1} - u_{i-2,1}].$$

Omawianie algorytmu cd.

Uważny obserwator zauważy, że wciąż istnieją pewne niezdefiniowane kolumny punktów, a mianowicie $u_{1,j}, u_{2,j}, u_{Nmax1,j}$ i $u_{Nmax,j}$, gdzie $Nmax$ to całkowita liczba punktów siatki. Prostą techniką wyznaczania ich wartości jest założenie, że $u_{1,2} = 1$ i $u_{Nmax,2} = 0$. Aby otrzymać $u_{2,2}$ i $u_{Nmax1,2}$ załóżmy, że $u_{i+2,2} = u_{i+1,2}$ i $u_{i2,2} = u_{i1,2}$ (unikamy $u_{i+2,2}$ dla $i = Nmax - 1$, oraz $u_{i2,2}$ za $i = 2$). Aby wykonać te kroki, przybliżamy powyższy wzór tak, że:

$$u_{i+2,2} + 2u_{i1,2}2u_{i+1,2}u_{i2,2} \rightarrow u_{i1,2}u_{i+1,2}$$

Omawianie algorytmu cd.

Błąd obcięcia i warunek stabilności naszego algorytmu są powiązane:

$$\varepsilon(u) = O[(\Delta t)^3] + O[\Delta t(\Delta x)^2]$$
$$\frac{1}{(\Delta x/\Delta t)}\ [\epsilon|u| + 4\frac{\mu}{(\Delta x)^2}] \leq 1$$

Pierwsze równanie pokazuje, że mniejsze kroki czasowe i przestrzenne prowadzą do mniejszego błędu aproksymacji (przybliżeń), ale ponieważ błąd zaokrąglenia wzrasta wraz z liczbą kroków, całkowity błąd niekoniecznie maleje. Jednak jesteśmy również ograniczeni w tym, jak małe kroki mogą być wykonane przez warunek stabilności, co wskazuje, że zbyt małe Δx zawsze prowadzi do niestabilności.

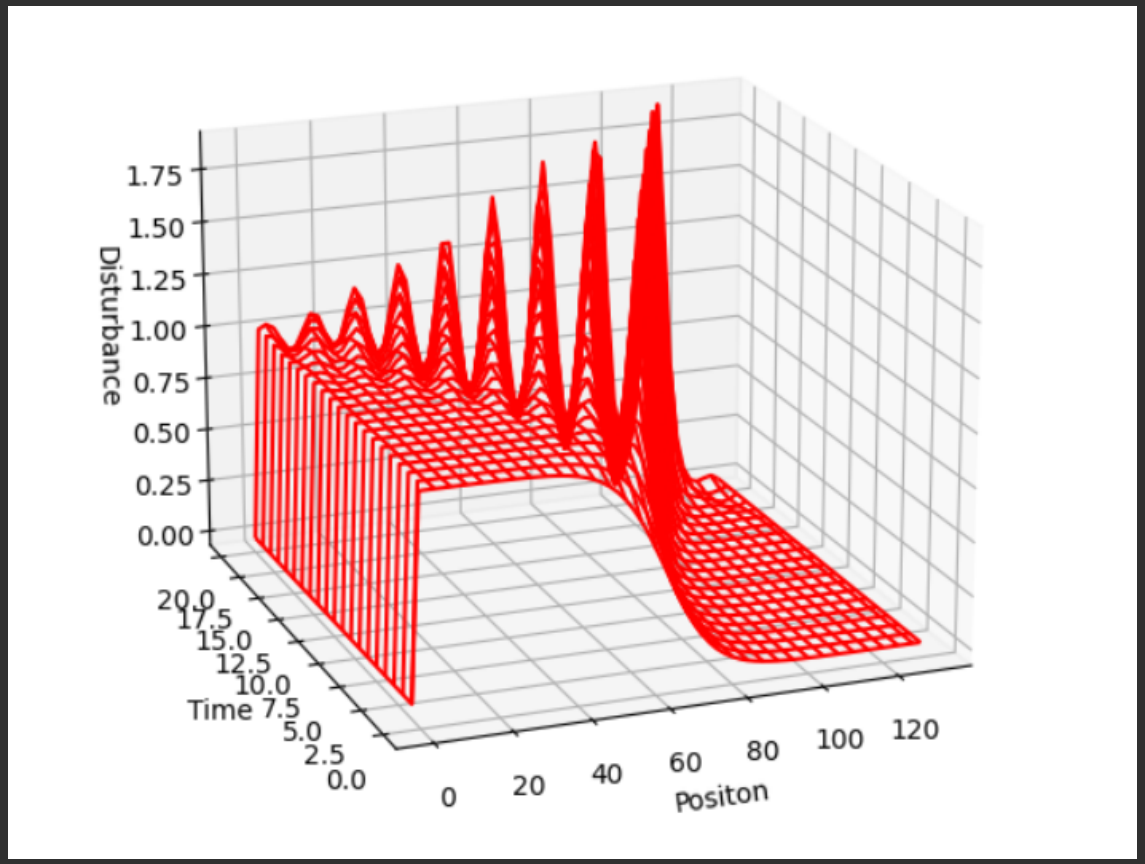
Implementacja rozwiązania

Kod w języku Python

```python
# Soliton.py:      Korteweg de Vries equation for a soliton

import matplotlib.pylab as p;
from mpl_toolkits.mplot3d import Axes3D ;
import numpy as np
import math

ds = 0.4;    dt = 0.1;    max = 2000
mu = 0.1;  eps = 0.2;    mx = 131
u   = np.zeros( (mx, 3), float); spl = np.zeros( (mx, 21), float); m = 1

for  i in range(0, 131):                # Initial wave
    u[i, 0] = 0.5*(1 -((math.exp(2*(0.2*ds*i-5.))-1)/
            (math.exp(2*(0.2*ds*i-5.))+1)))
u[0,1] = 1.; u[0,2] = 1.; u[130,1] = 0.; u[130,2] = 0.  # Ends

for i in range (0, 131, 2): spl[i, 0] = u[i, 0]
fac = mu*dt/(ds**3)
print("Working. Please hold breath and wait while I count to 20")
for  i in range (1, mx-1):                    # First time step
    a1 = eps*dt*(u[i + 1, 0] + u[i, 0] + u[i - 1, 0])/(ds*6.)
    if i > 1 and  i < 129:
        a2 = u[i+2,0]+2.*u[i-1,0]-2.*u[i+1,0]-u[i-2,0]
    else:  a2 = u[i-1, 0] - u[i+1, 0]
    a3 = u[i+1, 0] - u[i-1, 0]
    u[i, 1] = u[i, 0] - a1*a3 - fac*a2/3.
for j in range (1, max+1):                    # Next time steps
    for i in range(1, mx-2):
        a1 = eps*dt*(u[i+1,1] + u[i, 1] + u[i-1,1])/(3.*ds)
        if i > 1 and i < mx-2:
            a2 = u[i+2,1] + 2.*u[i-1,1]- 2.*u[i+1,1]-u[i-2,1]
        else:  a2 = u[i-1, 1] - u[i+1, 1]
        a3        = u[i+1, 1] - u[i-1, 1]
        u[i, 2] = u[i,0] - a1*a3 - 2.*fac*a2/3.
    if j%100 ==  0:                 # Plot every 100 time steps
        for i in range (1, mx - 2): spl[i, m] = u[i, 2]
        print(m)
        m = m + 1
    for k in range(0, mx):     # Recycle array saves memory
        u[k, 0] = u[k, 1]
        u[k, 1] = u[k, 2]

x = list(range(0, mx, 2))           # Plot every other point
y = list(range(0, 21))      # Plot 21 lines every 100 t steps
X, Y = p.meshgrid(x, y)

def functz(spl):
    z = spl[X, Y]
    return z

fig  = p.figure()                          # create figure
ax = Axes3D(fig)                             # plot axes
ax.plot_wireframe(X, Y, spl[X, Y], color = 'r')   # red wireframe
ax.set_xlabel('Positon')                       # label axes
ax.set_ylabel('Time')
ax.set_zlabel('Disturbance')
p.show()                  # Show figure, close Python shell
print("That's all folks!")
```

Rysunek 4: Implementacja rozwiązania równania w języku Python

Otrzymany wykres



Rysunek 5: Efekt zaimplementowanego rozwiązania w postaci wykresu solitonów

Bibliografia

Bibliografia

https://pl.wikipedia.org/wiki/Soliton
https://www.sciencedirect.com/topics/mathematics/soliton-solution

Dziękujemy za uwagę