Rozdział 22

Numeryczne rozwiązywanie cząstkowych równań różniczkowych

22.1. Równanie przepływu ciepła

W rozdziale 19 przedstawiono prosty przykład zastosowania metody Eulera do numerycznego rozwiązywania równań różniczkowych. Opisane tam podejście może być zastosowane do rozwiązywania cząstkowych równań różniczkowych. Rozważmy znane z termodynamiki równanie przepływu ciepła, które dla układu jednowymiarowego przyjmuje postać:

$$\frac{\partial u(t,x)}{\partial t} = c\frac{\partial^2 u(t,x)}{\partial x^2}, \tag{22.1.1}$$

gdzie: x – unormowana współrzędna przestrzenna $0 \leq x \leq 1$,
t – czas $t > 0$.

Przyjęto następujący warunek początkowy oraz warunki brzegowe:

$$\begin{aligned} u(0,x) &= f(x), \\ u(t,0) &= \alpha, \\ u(t,1) &= \beta. \end{aligned} \tag{22.1.2}$$

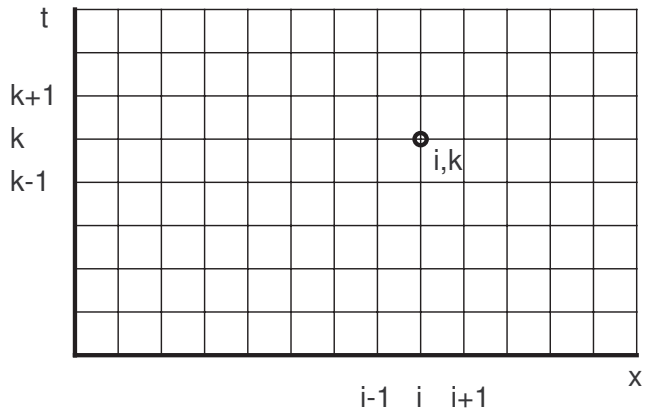
Równanie (22.1.1) należy zastąpić równaniem różnicowym postaci:

$$u_i^{k+1} = u_i^k + \frac{c\Delta t}{\Delta x^2}(u_{i+1}^k - 2u_i^k + u_{i-1}^k), \tag{22.1.3}$$

Po prostym przekształceniu równania (22.1.3) i porównaniu go z (22.1.1) można rozpoznać jak odpowiednie pochodne zostały zastąpione przez odpowiednie ilorazy różnicowe. W równaniu (22.1.3) przyjęto warunki początkowe (22.1.2) zapisane poniżej:

$$\begin{aligned} u_i^0 &= f(x_i), \\ u_0^k &= \alpha, \\ u_n^k &= \beta, \end{aligned} \tag{22.1.4}$$

gdzie: u_i^k – odpowiada aproksymacji rozwiązania dla $u(k\Delta t, i\Delta x)$.
Zasadę numerycznego rozwiązywania równania (22.3.3) najlepiej będzie objaśnić wykorzystując siatkę różnicową przedstawioną na rys. 22.1. Oś pozioma odpowiada współrzędnej przestrzennej x, natomiast oś pio-



Rys. 22.1. Siatka różnicowa

nowa współrzędnej czasowej t. Ponieważ proces rozwiązywania ma charakter dyskretny obie osie podzielono na odcinki o długości Δx i Δt. Węzły tak skonstruowanej siatki można ponumerować przy użyciu pary liczb i, k. Korzystając z warunków początkowych (22.1.4) określamy wartości funkcji $u(t,x) = f(x)$ w pierwszej warstwie siatki, to jest dla współrzędnej czasowej $t = 0$, symbolicznie zapiszemy to w postaci $u_i^1 = f(x_i)$. W jej punktach skrajnych wpisujemy warunki brzegowe $u_1^1 = \alpha$ i $u_n^1 = \beta$. Obliczenie wartości funkcji $u(t,x)$ w następnej warstwie $k = 2$ polega na wykorzystaniu zależności (22.1.3), co jest możliwe ponieważ znamy wszystkie wartości po jej prawej stronie.

Budowę programu rozpoczynamy od ustawienia maksymalnego rozmiaru stosu, jest to niezbędne jeśli dokonamy gęstego podziału zmiennej przestrzennej i czasowej.

```
stacksize(80000000);
```

Kolejne polecenie to określenie gęstości podziału zmiennej przestrzen-

nej x, wynika z niego krok Δx. Jego wartość powiązana z wartością Δt ma istotne znacznie dla stabilności stosowanego algorytmu. Przyjęty sposób obliczania wartości $\Delta t = 1/n^2$ jest wynikiem przeprowadzonych eksperymentów numerycznych.

```
n=50;                // podział zmiennej x
dx = 1/n;            // krok na siatce x
dt=0.5/n^2;          // krok czasowy
```

Znaczenie następnych dwóch poleceń wyjaśni się jeżeli powrócimy do równania (22.1.3).

```
c = 1.0;
lambda =dt/dx^2*c;
```

Kolejne dwie instrukcje tworzą wektory zawierające dyskretne wartości zmiennej przestrzennej oraz zmiennej czasowej:

```
x = (0:dx:1);              // wektor x
t = 0; T=(t:dt:1);         // wektor T
```

Podstawowa tablica wewnątrz której będziemy przechowywać wszystkie wyznaczone wartości funkcji $u(t, x)$ jest tworzona poleceniem `zeros()`, jej wymiary muszą być oczywiście zgodne z wymiarami wektorów `x` i `t`.

```
u = zeros(length(x),length(T));
```

Warunki początkowe i brzegowe definiowane są poleceniami:

```
alfa=0;                     // warunek brzeg. lewy
beta=0;                     // warunek brzeg. prawy
u(:,1)=sin(%pi*x)';         // warunek początkowy
u(1,1)=alfa; u($,1)=beta;
```

Podstawowa pętla realizująca obliczenia wartości funkcji $u(t, x)$, według zależności (22.1.3), wykonywana jest wewnątrz pętli `while`. Zmienna `kol` określa numer „warstwy" na której obliczamy wartości funkcji $u(t, x)$.

```
kol=1;
while t<1 do
  t=t+dt;
  kol=kol+1;
```

```
u(2:$-1,kol) = (1-2*lambda)*u(2:$-1,kol-1) ...
                     + lambda*u(3:$,kol-1) ...
                     + lambda*u(1:$-2,kol-1);
end
```

Ostatnia grupa instrukcji służy do graficznej prezentacji zmian strumienia ciepła w funkcji współrzędnej przestrzennej x i czasowej t.

```
xbasc();
plot3d(x,T,u(:,1:length(T)), ...
   theta=20,alpha=65,leg='x@t@u(t,x)',flag=[32 2 3]);
```

22.2. Równanie struny

Pokazany w poprzednim punkcie sposób konstruowania funkcji do numerycznego rozwiązywania cząstkowych równań różniczkowych może być względnie łatwo przystosowywany do rozwiązywania różnych typów równań różniczkowych. Jego adaptacja zostanie pokazana na przykładzie równania opisującego nietłumione drgania struny. Zjawisko rozchodzenia się fal opisuje się za pomocą równań różniczkowych cząstkowych. Jako przykład weźmy strunę. W każdej chwili czasu $t \geq 0$ kształt struny drgającej przedstawiony jest wykresem pewnej funkcji $u(t, x)$ która mierzy odchylenie struny od położenia ustalonego. Jeśli znamy funkcję $u(t, x)$, to znamy ruch drgający struny. Ruch ten jest spowodowany działaniem sił naprężeniowych na każdy jej element. Struna wydłuża się lub kurczy, przy tym zachowana jest zasada Hooke'a o stałości stosunku wydłużenia do naprężenia. Na podstawie tego można wyprowadzić równanie dla funkcji $u(t, x)$

$$\frac{\partial^2 u(t,x)}{\partial t^2} - c\frac{\partial^2 u(t,x)}{\partial x^2} = 0, \tag{22.2.1}$$

gdzie: x – unormowana współrzędna przestrzenna $0 \leq x \leq 1$,
t – czas $t > 0$.
Przyjęto następujące warunki początkowe i brzegowe:

$$\begin{aligned} u(0,x) &= f(x), \\ \frac{\partial u(0,x)}{\partial t} &= g(x) \\ u(t,0) &= \alpha, \\ u(t,1) &= \beta. \end{aligned} \tag{22.2.2}$$

Równanie (22.2.1) należy zastąpić równaniem różnicowym postaci

$$u_i^{k+1} = 2u_i^k - u_i^{k-1} + \frac{c\Delta t^2}{\Delta x^2}(u_{i+1}^k - u_i^k + u_{i-1}^k). \tag{22.2.3}$$

Po prostym przekształceniu równania (22.2.3) i porównaniu go z (22.2.1) można rozpoznać jak odpowiednie pochodne zostały zastąpione przez odpowiednie ilorazy różnicowe. W równaniu (22.2.3) przyjęto warunki początkowe i brzegowe (22.2.2) zapisane w sposób pokazany poniżej:

$$\begin{aligned} u_i^0 &= f(x_i), \\ u_0^k &= \alpha, \\ u_n^k &= \beta, \end{aligned} \tag{22.2.4}$$

gdzie: u_i^k – odpowiada aproksymacji rozwiązania dla $u(k\Delta t, i\Delta x)$.

Drugi warunek początkowy z (22.2.2) został określony w postaci odpowiednio zadanych wartości w pierwszych trzech „warstwach" naszej siatki. Dla uproszczenia przyjęto, że pochodna funkcji u(t,x) jest wielkością stałą. Program zaczyna się podobnie jak poprzedni od ustawienie stosu i określenia gęstości podziału zmiennej przestrzennej i czasowej. Definiowane są również dwa wektory zawierające wartości dyskretne zmiennej przestrzennej x i czasowej t.

```
stacksize(80000000);
n=100;                          // podział zmiennej x
dt=1/n;                         // krok czasowy
dx = 1/n;                       // krok na siatce x
c = 1.0;                        // stałe z równania
lambda = ((dt^2)/(dx^2))*c;     // stałe z równania

x = (0:dx:1);                   // dyskretyzcja x
t = 0;
T=(t:dt:10);                    // dyskretyzcja t
```

Podstawowa tablica zawierająca położenie poszczególnych punktów drgającej struny to:

```
u = zeros(length(x),length(T));
```

Warunki brzegowe i początkowe to:

```
alfa=0;                       // warunek brzeg. lewy
beta=0;                       // warunek brzeg. prawy
u(:,1)=sin(%pi*x)';           // warunek początkowy
u(1,1)=alfa; u($,1)=beta;
g=1;
```

Drugi warunek początkowym (zależność 22.2.2) został zapisany w postaci ciągu instrukcji warunkowych:

```
if (g==1) then
 u(2:$-1,2) = u(1:$-2,1);
elseif (g==-1) then
 u(2:$-1,2) = u(3:$,1);
else
 u(2:$-1,2) = 0.5*(u(1:$-2,1)+u(3:$,1));
end
u(1,2) = alfa; u($,2) = beta;
u(1,3) = alfa; u($,3) = beta;
```

Kolejna grupa poleceń umożliwia wizualną kontrolę zgodności warunków początkowych i brzegowych:

```
 xbasc(); plot2d(x,u(:,1),rect=[0,-1,1,1]);
 xbasc(); plot2d(x,u(:,2),rect=[0,-1,1,1]);
```

Główna pętla obliczeniowa skonstruowana jest podobnie jak ta dla zagadnienia przepływu ciepła.

```
kol=2;
while t<10 do
 t=t+dt;
 kol=kol+1;
 for j=2:n do
  u(j,kol) = (2-2*lambda)*u(j,kol-1) ...
    - u(j,kol-2) ...
```

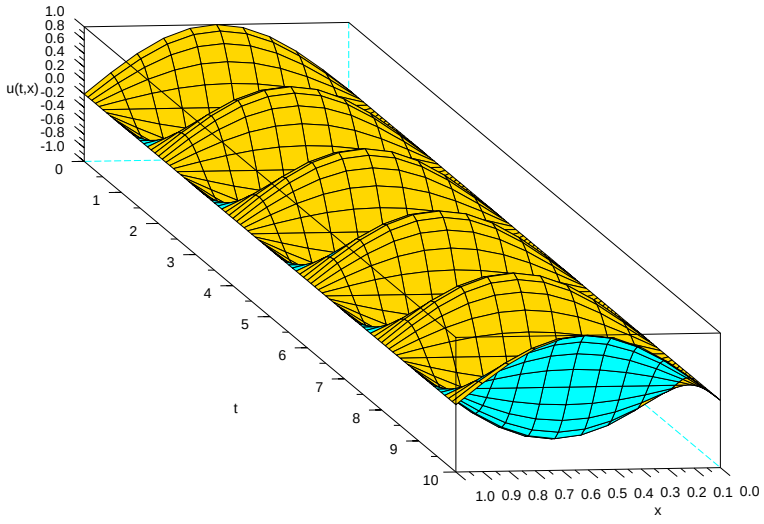
```
    + lambda*u(j+1,kol-1) ...
    + lambda*u(j-1,kol-1);
 end
```

Polecenie `plot2d()` wewnątrz pętli `while` umożliwia obserwację drgającej struny na ekranie monitora.

```
 xbasc(); plot2d(x,u(:,kol),rect=[0,-1,1,1]);
 xpause(5000);
end
```

Program kończy się kreśleniem wychyleń struny (rys. 22.2) dla wszystkich analizowanych chwil czasowych.

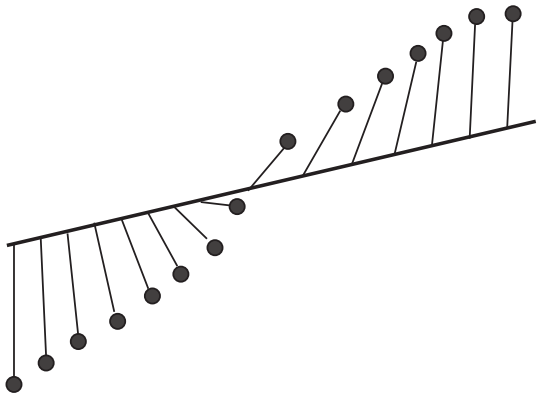
```
xbasc();
plot3d(x,T,u(:,1:length(T)),...
 theta=20,alpha=65,leg='x@t@u(t,x)',flag=[32 2 3]);
```



Rys. 22.2. Drgania struny

22.3. Nieliniowe cząstkowe równanie różniczkowe sinus-Gordona

Własności wahadła matematycznego są powszechnie znane. W niniejszym punkcie chcemy skupić naszą uwagę na numerycznym rozwiązaniu równania opisującego bardzo złożony układu wielu sprzężonych ze sobą wahadeł matematycznych. Uzyskane rozwiązanie ma ogromne znaczenie teoretyczne ale i praktyczne, jest wykorzystywane w wielu działach fizyki i jej zastosowań [6]. Nie bez znaczenia jest również matematyczne piękno uzyskiwanych rozwiązań. Na rysunku 22.3 pokazano prosty przykład takiego układu. Składa się on z wału skrętnego oraz osadzonych w stałych odległościach od siebie wahadeł matematycznych. Rozważmy dalej równowagę układu trzech wahadeł (rys. 22.4)

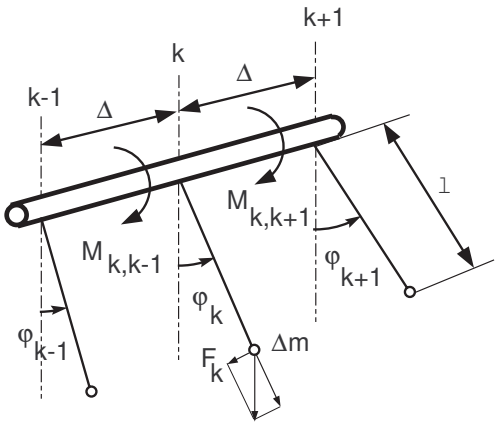


Rys. 22.3. Łańcuch wahadeł matematycznych

[26]. Równanie równowagi k-tego wahadła przyjmuje postać:

$$l(\Delta ml\varepsilon_k) = -lF_k + M_{k,k+1} + M_{k,k-1}, \tag{22.3.1}$$

Pamiętając, że oddziaływanie pomiędzy dowolnymi trzema sąsiednimi wahadłami jest proporcjonalne do różnic ich wychyleń kątowych, tj. $M_{k,k\pm1} = \alpha(\varphi_{k\pm1} - \varphi_k)/\Delta$, gdzie współczynnik α charakteryzuje sprężyste własności materiału (dla drgań skrętnych) oraz, że składowa siły ciężkości działająca na k-te wahadło, to $F_k = \Delta mgsin(\varphi_k)$, gdzie Δm



Rys. 22.4. Trzy sąsiednie wahadła

jest masą punktu materialnego na końcu wahadła, m gęstością liniową , równanie (22.3.1) przyjmuje postać:

$$\Delta m l^2 \varepsilon_k = -\Delta m g l sin(\varphi_k) + \alpha(\varphi_{k+1} + \varphi_{k-1} - 2\varphi_k)/\Delta, \quad (22.3.2)$$
$$dla\ k = 1, 2, \ldots.$$

Ponieważ przyspieszenie kątowe ε_k jest drugą pochodną wychylenia względem czasu, ten układ równań można zapisać jeszcze inaczej:

$$\frac{l}{g}\frac{d^2\varphi_k(t)}{dt^2} = -sin(\varphi_k(t)) + \frac{\alpha}{mgl}\frac{\varphi_{k+1}(t) + \varphi_{k-1}(t) - 2\varphi_k(t)}{\Delta^2}. \quad (22.3.3)$$

Równanie (22.3.3) to układ równań różniczkowo-różnicowych, opisujący własności ciągu wahadeł matematycznych oddziaływujących ze sobą według najprostszego sposobu: oddziaływanie dotyczy jedynie sąsiednich wahadeł i jest liniowe względem różnicy wychyleń. Jeżeli współczynnik α będzie zmierzał do zera, równanie (22.3.3) przyjmuje znaną postać równania opisującego ruch swobodny oscylatora harmonicznego. Jeśli natomiast będziemy zmniejszać masę wahadeł i równocześnie zmniejszać odległość pomiędzy nimi, to ostatni człon równania (22.3.3) zmierza do $\frac{\partial^2\varphi}{\partial x^2}$, a całe równanie przyjmuje postać

$$\frac{l}{g}\frac{d^2\varphi_k(t)}{dt^2} = -sin(\varphi_k(t)) + \frac{\alpha}{mgl}\frac{\partial^2\varphi(x,t)}{\partial x^2}, \quad (22.3.4)$$

gdzie poszukiwana funkcja $\varphi(x,t)$ zależy teraz od współrzędnej x mierzonej wzdłuż łańcucha wahadeł a nie od numeru wahadła. Zależy oczywiście od czasu *t*.

Dokonując podstawień:

$$\tau = \sqrt{\frac{g}{l}}t, \quad \xi = \sqrt{\frac{mgl}{\alpha}}x,$$

otrzymujemy ostatecznie

$$\frac{\partial^2\varphi(\xi,\tau)}{\partial\xi^2} - \frac{\partial^2\varphi(\xi,\tau)}{\partial\tau^2} = sin(\varphi(\xi,t)). \tag{22.3.5}$$

Otrzymane nieliniowe równanie różniczkowe cząstkowe nosi nazwę równania sinus-Gordona. Równanie to ma nieskończenie wiele rozwiązań. Jednak ich konstrukcja analityczna jest często bardzo trudna. Połowa z nich daje się jednak przedstawić przez funkcje elementarne. Jedno z tych rozwiązań nosi nazwę solitonu i ma bardzo ciekawe własności.

Zadanie znalezienia analitycznego rozwiązania równania (22.3.5) przy użyciu standardowych poleceń dostępnych w Maximie jest skazane na niepowodzenie. Pozostaje więc próba znalezienia rozwiązania numerycznego. Pierwszym krokiem jest oczywiście zamiana cząstkowego równania sinus Gordona na równanie różnicowe. W literaturze można znaleźć wiele różnych schematów różnicowych [6] pozwalających rozwiązać równanie sinus Gordona, do dalszej analizy wybierzemy jeden z nich

$$u_i^{k+1} = -u_i^{k-1} + \left(\frac{\Delta x}{\Delta t}\right)^2 (u_{i+1}^k + u_{i-1}^k) + 2(1 - \left(\frac{\Delta x}{\Delta t}\right)^2)u_i^k - \left(\frac{\Delta x}{\Delta t}\right) sin(u_i^k), \tag{22.3.6}$$

gdzie $\left(\frac{\Delta x}{\Delta t}\right)^2 \approx 0.95$. Rozwiązywanie równania różnicowego (22.3.6) wymaga zdefiniowania warunków początkowych i brzegowych w naszym przypadku połączonych ze sobą:

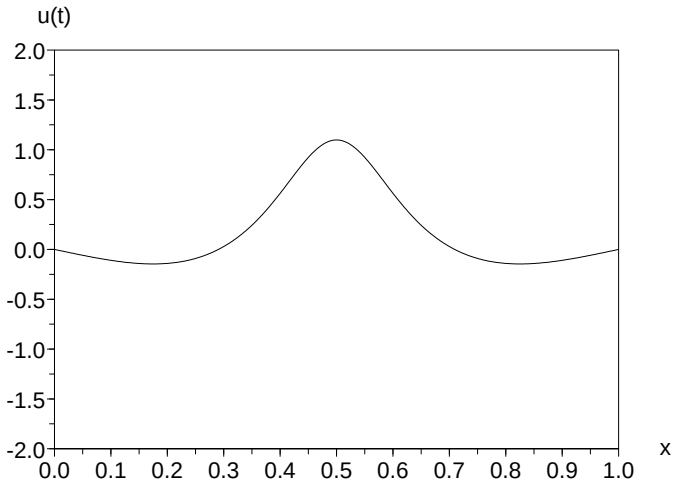
$$\begin{aligned} u(0,x) &= sin(x), \\ \frac{\partial u(0,x)}{\partial t} &= g(x). \end{aligned} \tag{22.3.7}$$

Przyjęty sposób implementacji równania różniczkowego identyczny z tym opisanym w punkcie 22.2.

```
stacksize(80000000);
n=500;                // podział zmiennej x
dt=100/(n*90);              // krok czasowy
dx = 1/n;             // krok na siatce x
r = dx/dt;            // stale z równania
x = (0:dx:1);         // dyskretyzcja x
t = 0;
T=(t:dt:2);           // dyskretyzcja t
u = zeros(length(x),length(T));
alfa=0.;              // warunek brzeg. lewy
beta=0;               // warunek brzeg. prawy
u(:,1)=sin(%pi*x)';   // warunek początkowy
u(1,1)=alfa;
u($,1)=beta;
g=0;
if (g==1) then
   u(2:$-1,2) = u(1:$-2,1);
 elseif (g==-1) then
   u(2:$-1,2) = u(3:$,1);
else
 u(2:$-1,2) = 0.5*(u(1:$-2,1)+u(3:$,1));
end
u(1,2) = alfa; u($,2) = beta;
u(1,3) = alfa; u($,3) = beta;
kol=2;
while t<2
  t=t+dt;
  kol=kol+1;
  for j=2:n do
    u(j,kol)=-u(j,kol-2)+...
    r^2*(u(j+1,kol-1)+u(j-1,kol-1))..
    +2*(1-r^2)*u(j,kol-1)-dx*sin(u(j,kol-1));
  end
  xbasc(); plot2d(x,u(:,kol),rect=[0,-2,1,2]);
  xpause(5000);
end
xbasc();
plot3d(x,T,u(:,1:length(T)),theta=20,alpha=65,...
```

```
leg='x@t@u(t,x)',flag=[32 2 3]);
```

Po uruchomieniu powyższego programu możemy obserwować jedno z rozwiązań naszego równania, zwane breather („oddychacz") odpowiada ono parze rozwiązań soliton i antysoliton (rys. 22.5), które w naszym przypadku pulsują nie zmieniając swojego położenia.



Rys. 22.5. Para soliton i antysoliton, nazywana czasami oddychacz (breather)

Bibliografia

[1] ASTM. *Standard Practice for Statistical Analysis of Linear or Linearized Stress-Life (S-N) and Strain-Life (ε-N) Fatigue Data*, volume 03.01. Annual Book of ASTM Standards, 2003.

[2] S. Banach. *Mechanika*. PWN, Warszawa 1956.

[3] J. S. Bendat and A. G. Piersol. *Random Data, Analysis and Mesurement Procedures*. John Wiley and Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1989.

[4] R. Beniak and P. Wach. *Zadania z dynamiki układów elektromechanicznych przy zastosowaniu Maple V*. Oficyna Wydawnicza Politechniki Opolskiej, 1999.

[5] I. N. Bronsztejn and K. A. Siemiendiajew. *Matematyka. Poradnik encyklopedyczny*. PWN Warszawa, 1998.

[6] R. K. Dodd et al. *Solitons and nonlinear wave equations*. Academic Press Inc., 1988.

[7] A. Ekberg. Rolling contact fatigue of railway wheels - computer modelling and in-field data. Division of Solid Mechanics, Chalmers University of Technology, Göteborg, Sweden, anek@solid.chalmers.se, 1997.

[8] J. J. Girardot. Une introduction a la programmation par scilab. École des Mines de Saint-Etienne, Internet, 2000.

[9] C. Gomez. Simulation de system dynamiques. *Linux Magazin France*, (5), 2000.

[10] J. Grębosz. *Pasja C++*. Oficyna Kallimach Kraków, 1999.

[11] J. Grębosz. *Symfonia C++*. Oficyna Kallimach Kraków, 1999.

[12] Calfem Group. *CALFEM, a finite element toolbox to MATLAB*. Department of Mechanics and Material, LUND University, 3.3 edition, 1999.

[13] Scilab Group. *Introduction to Scilab*. Scilab Group Inc, 2004.

[14] M. T. Huber. *Steromechanika techniczna. Wytrzymałość materiałów*. PWN, Warszawa 1958.

[15] A. Jakubowicz and Z. Orłoś. *Wytrzymałość materiałów*. WNT, 1970.

[16] G.A. Korn and T.M. Korn. *Matematyka dla pracowników naukowych i inżynierów*, volume I i II. PWN, 1983.

[17] E. Majchrzak and B. Mochnacki. *Metody numeryczne, Podstawy teo-*

retyczne, aspekty praktyczne i algorytmy. Wydawnictwo Politechniki Śląskiej, 2004.
[18] N. M. Matwiejew. *Metody całkownia równań różniczkowych zwyczajnych*. PWN, 1986.
[19] J. Ombach. *Wykłady z równań różniczkowych wspomagane komputerowo Maple*. Wydawnictwo Uniwersytetu Jagiellońskiego, 1999.
[20] B. Pinçon. Une introduction á scilab. Technical report, Institut Elie Cartan Nancy, Université Henri Poincaré, 2004.
[21] Palej R. *Algebra komputerowa w mechanice. Rozwiązywanie zagadnień z mechaniki za pomocą programu Maple V*. Politechnika Krakowska, 2000.
[22] G. Sallet. Ordinary differential equations with scilab, wats lectures, provisional notes. Technical report, Université de Saint-Louis, Internet, 2004.
[23] F. Socie, D. and B. Marquis, G. *Multiaxial Fatigue*. Society of Automotive Engineers Inc., Warrendale. Pa., 2000.
[24] T. Williams et al. *Gnuplot, An interactive ploting program*. http://www.gnuplot.info, 2004.
[25] J. Wolska et al. *Zarys teorii równań całkowych i równań różniczkowych cząstkowych*. PWN Warszawa, 1981.
[26] J. Zagrodziński. Solitony w domu. *Delta*, 9(129), 1984.