WOLNE OPROGRAMOWANIE W PRAKTYCE INŻYNIERSKIEJ

Wojciech Sobieski
Katedra Mechaniki i Podstaw Konstrukcji Maszyn, Uniwersytet Warmińsko-Mazurski

Streszczenie: Artykuł przedstawia zestaw narzędzi informatycznych wywodzących się z nurtu tzw. Wolnego Oprogramowania oraz propozycję logicznej struktury połączeń między nimi, zapewniających wygodne i efektywne rozwiązywanie dowolnych zadań obliczeniowych współczesnej inżynierii, w tym również inżynierii rolniczej. Celem pracy jest przedstawienie pewnej ideologii programowania, umożliwiającej tworzenie pakietów o luźnej, modułowej strukturze, zawierającej alternatywne rozwiązania i dopuszczającej dowolne konfiguracje połączeń między aplikacjami. Istotne jest też, że w artykule zaproponowano konkretne, oparte na Wolnym Oprogramowaniu, rozwiązania informatyczne. Tłem do przedstawienia proponowanych rozwiązań informatycznych jest równanie ruchu układu drgającego.

Słowa kluczowe: Wolne Oprogramowanie, Fortran, DISLIN, Gnuplot, Lazarus

Wprowadzenie

Wolne Oprogramowanie jest to ogólnoświatowy ruch programistów i użytkowników komputerów zaangażowanych w działania na rzecz wolnego dostępu do oprogramowania przez ogół użytkowników [GNU Operating System 2009]. Ruch powstał w latach 80., gdy zaczęto komercjalizować rynek informatyczny i ograniczać dostęp do bezpłatnego do tej pory oprogramowania. Za twórcę Ruchu Wolnego Oprogramowania uważa się Richarda Stallmana z Massachusetts Institute of Technology. Założył on również Fundację Wolnego Oprogramowania, zajmującą się głównie promocją ruchu oraz rozwijaniem projektu GNU (jest to projekt, którego celem jest stworzenie systemu operacyjnego złożonego wyłącznie z Wolnego Oprogramowania) [GNU Operating System 2009]. Zasady korzystania z Wolnego Oprogramowania regulują liczne liberalne licencje, z których najpopularniejszą jest powszechna licencja publiczna GNU GPL (General Public License) [GNU Operating System 2009]. Celem tej licencji jest przekazanie użytkownikom praw do uruchamiania programu w dowolnym celu (tzw. wolność 1); analizowania działania programu i dostosowywania go do swoich potrzeb (wolność 2); kopiowania (wolność 3); udoskonalania i publikowania własnych poprawek i wersji (wolność 4) [GNU Operating System 2009]. Obecnie projekt GNU nadal dynamicznie się rozwija, chociaż jego podstawowy cel dawno został zrealizowany (systemy Linux [Mandriva Linux 2009, OpenSuse Linux 2009, Ubuntu Linux 2009] UNIX BSD [FreeBSD Project 2009, OpenBSD Project 2009, NetBSD Project 2009], FreeDOS [Free DOS 2009] i inne). Ponieważ wiele z programów utworzonych w ramach projektu GNU uzyskało wysoki stopień dojrzałości, spełniając wymagania

i oczekiwania szerokiego grona użytkowników oraz zapewniając im wygodną i efektywną pracę, programy te migrowały również (dzięki dostępowi do kodu źródłowego) do systemów komercyjnych, takich jak Windows czy MacOS. Przykładami takich aplikacji mogą być OpenOffice [OpenOffice.org 2009], KOffice [The KOffice Project 2009], LaTeX [LaTeX 2009], Mozilla Firefox [Mozilla Firefox 2009] i wiele innych. Liczne zalety Wolnego Oprogramowania oraz zasadność jego stosowania przedstawione zostały szczegółowo w książce [Sobieski 2008] oraz w opublikowanym w Internecie autorskim cyklu wykładów dotyczącym oprogramowania alternatywnego [Sobieski 2009b]. Motywacją do powstania artykułu była chęć zaprezentowania wybranych aplikacji Wolnego Oprogramowania, mogących posłużyć między innymi do realizacji projektów programistycznych typowych dla współczesnej inżynierii, w tym inżynierii rolniczej, oraz promocja rozwiązań bazujących na Wolnym Oprogramowaniu w środowisku akademickim. Celem publikacji jest przekonanie czytelnika do zalet stosowania Wolnego Oprogramowania w codziennej pracy dydaktycznej i naukowej, a także w praktyce inżynierskiej.

„Wolne" narzędzia informatyczne

Do rozwiązywania zagadnień typowo obliczeniowych potrzebny jest język programowania o prostej składni oraz maksymalnej szybkości działania. Potrzeba taka ujawniła się już na samym początku wykorzystywania komputerów. Wówczas właśnie – w latach 50. ubiegłego wieku – powstał język Fortran (FORmula TRANslator) [Chrobak 2003, Piechna 2000, Trykozko 1999], wyspecjalizowany do obliczeń zmiennoprzecinkowych. Język ten stosowany jest do dzisiaj, przy czym w przeciągu kilkudziesięciu lat powstało wiele jego specyfikacji i implementacji. Pierwsza „wolna" implementacja, odnosząca się do języka w standardzie Fortranu 77, pojawiła się w roku 1987. Obecnie zrealizowana jest pełna implementacja standardu Fortranu 95, a trwają już zaawansowane prace nad uwzględnieniem ostatniego standardu języka, Fortranu 2003.

Inną kwestią związaną z szeroko pojętymi obliczeniami matematycznymi jest kwestia wizualizacji wyników – tu również znaleźć można liczne „wolne" programy i biblioteki. W zakresie wizualizacji wymienić można narzędzia takie jak ParaView [ParaView 2009], MayaVi [MayaVi 2009], Gnuplot [Gnuplot 2009] czy OpenGL [OpenGL 2009]. W niniejszym artykule wykorzystano dwa, stosunkowo proste narzędzia wizualizacyjne, dobrze sprawdzające się we współpracy z językiem Fortran: bibliotekę DISLIN oraz środowisko wizualizacyjne Gnuplot.

Ponieważ język Fortran służy zasadniczo do przetwarzania liczb, kwestia interfejsu programów została w nim całkowicie pominięta (poza wybranymi implementacjami komercyjnymi). Aplikacje napisane w tym języku uruchamiane są w wierszu poleceń, tam też odbywa się komunikacja z użytkownikiem. Takie podejście jest wystarczające, ale nie najefektywniejsze w praktyce. Wygodniej obsługuje się programy tzw. „okienkowe", poza tym można w nich łatwiej zawrzeć dodatkowe funkcje, przyspieszające i usprawniające proces obliczeń. Aby nie rezygnować z zalet języka Fortran, można skorzystać z innych języków i środowisk programowania, umożliwiających tworzenie graficznych nakładek na fortranowskie programy konsolowe (tzw. solwery, czyli programy realizujące główne zadania obliczeniowe). Autor proponuje w tym zakresie kompilator Free Pascal i środowisko programistyczne Lazarus. Sposób jego wykorzystania przedstawiono w skrócie w dalszej części artykułu.

Poniżej przedstawiono opis wybranych narzędzi informatycznych, wykorzystanych do utworzenia pakietu obliczeniowego rozwiązującego równanie ruchu układu drgającego oraz wizualizującego wyniki tych obliczeń na kilka sposobów. Istotne jest, że wszystkie wymienione tu narzędzia dostępne są dla wielu systemów operacyjnych, w tym systemów Windows, UNIX/Linux i MacOS, nie ograniczając użytkownika w wyborze systemu operacyjnego.

Gfortran [GCC Project 2009] – jeden z podprojektów GNU GCC, będący kompilatorem języka Fortran według standardu Fortranu 95. Kompilator dostępny jest na licencji GNU GPL i może być używany bezpłatnie do wszelkich zastosowań (również komercyjnych), w systemach operacyjnych Windows, UNIX/Linux, MacOS i innych. Charakterystyczną cechą tej implementacji języka jest duży zestaw funkcji wewnętrznych, rozszerzających jego możliwości w zakresie komunikowania się z systemem operacyjnym. Gfortran jest samym kompilatorem i nie zawiera żadnego środowiska programistycznego ani edytora kodu źródłowego. W celu ułatwienia i przyspieszenia pracy można skorzystać z bezpłatnych środowisk programistycznych opartych na tym kompilatorze, np. Edi [Projekt Edi 2009] lub Fortran Force [Force Project 2009] (przeznaczonych dla systemów Windows), albo też samych edytorów wspomagających tworzenie kodu źródłowego, np. Notepad++ [Notepad++ 2009] (dla systemów Windows) czy Kate [The Kate Project 2009] (dla systemów UNIX/Linux).

DISLIN [DISLIN 2009] – biblioteka graficzna, służąca do tworzenia wykresów funkcji dwu- i trójwymiarowych oraz konturów i map. Biblioteka przeznaczona jest dla języków C, C++, Fortran 77, Fortran 90/95, Perl oraz Python i jest dostępna dla systemów Windows, UNIX/Linux, MacOS i innych. DISLIN nie należy do Wolnego Oprogramowania, może być jednak stosowana bezpłatnie do celów niekomercyjnych. Biblioteka DISLIN umożliwia wizualizację rysunków na ekranie monitora lub też zapis w kilkunastu formatach graficznych.

Lazarus [Lazarus Project 2009] – zintegrowane wizualne środowisko programistyczne oparte na kompilatorze Free Pascal [Free Pascal 2009], dostępne w systemach operacyjnych Windows, UNIX/Linux, MacOS i innych. Jest ono wzorowane na komercyjnym środowisku programistycznym Delphi firmy Borland i jest z nim w dużej mierze zgodne (umożliwia między innymi import projektów Delphi). Lazarus wykorzystuje bibliotekę Lazarus Component Library (LCL), która jest odpowiednikiem biblioteki VCL używanej w pakiecie Delphi. Program jest udostępniany bezpłatnie na licencji GNU GPL, natomiast biblioteki na zmodyfikowanej licencji LGPL (co oznacza możliwość wykorzystania Lazarusa w projektach o zamkniętym kodzie). Program napisany w środowisku Lazarus można bez żadnych zmian skompilować dla dowolnego obsługiwanego procesora, systemu operacyjnego i interfejsu okienek.

TAChart [TAChart 2009] – komponent wizualny do środowiska Lazarus, dostępny bezpłatnie na licencji LGPL, umożliwiający tworzenie różnego typu wykresów dwuwymiarowych.

Gnuplot [Gnuplot 2009] – środowisko przeznaczone do tworzenia wykresów funkcji jednej i dwóch zmiennych oraz do dopasowywania współczynników funkcji do zbiorów danych. Interpreter Gnuplota dostępny jest na licencji zezwalającej na kopiowanie i modyfikowanie kodu źródłowego i może być używany bezpłatnie do wszelkich zastosowań, również komercyjnych, w systemach operacyjnych Windows, UNIX/Linux, MacOS i in-

nych. Środowisko oferuje dwa tryby pracy: tryb interaktywny, w którym polecenia wprowadza się w wierszu poleceń, oraz wsadowy, w którym polecenia wykonywane są na podstawie skryptu, zapisanego wcześniej do pliku tekstowego.

Równanie ruchu układu drgającego

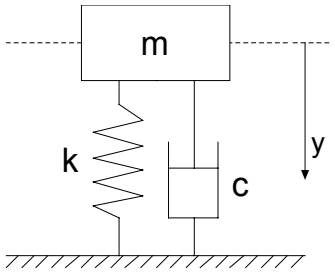
W przypadku układu jak na rys. 1, ruch ciężarka można opisać następującym równaniem różniczkowym [Piechna 2000; Pluta 2009; Wierzbanowski 2009]:

$$m\frac{d^2y}{dt^2} = -mg - k\left(y - w_0\right) - c\frac{dy}{dt}\left|\frac{dy}{dt}\right|, \tag{1}$$

gdzie:

m – masa [kg],
g – przyspieszenie ziemskie [m/s],
k – stała sprężyny [N/m],
w_0 – napięcie wstępne sprężyny [N],
c – stała tłumienia tłumika [kg/m],
y – przemieszczenie pionowe ciężarka [m],
t – czas [s].

Równanie (1) jest to tzw. równanie ruchu harmonicznego tłumionego z siłą wymuszającą [Pluta 2009; Piechna 2000].



Rys. 1. Schemat układu drgającego
Fig. 1. Vibrating system diagram

Aby obniżyć rząd równania różniczkowego można zastosować następujące podstawienie:

$$\begin{cases} \dfrac{dy}{dt} = u \\ \dfrac{du}{dt} = -g - \dfrac{k}{m}\left(y - w_0\right) - \dfrac{c}{m}u|u| \end{cases}. \tag{2}$$

Podczas rozwiązywania zadania należy wziąć pod uwagę wpływ siły ciężkości. W tym przypadku będzie od dwojaki: spowoduje pewne wstępne napięcie sprężyny – co można obliczyć ze wzoru

$$w_0 = \frac{mg}{k}, \tag{3}$$

a także spowoduje zróżnicowanie wartości współczynnika tłumienia c dla kierunku w górę i w dół (na rys. 1 oś y skierowana jest w dół):

$$\begin{cases} c = c_d & for \quad u > 0 \\ c = c_g & for \quad u \geq 0 \end{cases}. \tag{4}$$

Wzór (2), wraz z uzupełniającymi go wzorami (3) i (4), stanowi podstawę opisanego dalej algorytmu obliczeniowego.

Algorytm obliczeniowy

Numeryczne rozwiązanie równania (2) może być oparte na algorytmie iteracyjnym (pętli) o znanej lub też nieznanej z góry liczbie powtórzeń. W pierwszym przypadku należy założyć maksymalną liczbę iteracji (lub pośrednio, czas obliczeniowy), dla których mają być wykonane obliczenia. W drugim przypadku należy zdefiniować dodatkowy warunek logiczny – np. wartość wychylenia – umożliwiający zakończenie procesu obliczeń. Oczywiście oba te podejścia mogą być zastosowane jednocześnie, wówczas pętla wykona określoną z góry liczbę powtórzeń, chyba że wcześniej wychylenie (lub inna dowolna wielkość) uzyska wartość graniczną i wymusi przerwanie obliczeń. W niniejszym artykule wybrano sposób pierwszy. Liczbę niezbędnych iteracji można wyliczyć na podstawie całkowitego czasu obliczeń oraz kroku czasowego z zależności: $n_{iter} = t_{max} / dt$, gdzie: n_{iter} - całkowita liczba iteracji niezbędna do uzyskania czasu obliczeniowego równego t_{max} [-], t_{max} - całkowity czas obliczeniowy [s], dt - krok czasowy [s].

Algorytm iteracyjny wylicza wartości przemieszczenia pionowego y oraz jego prędkości u w poszczególnych chwilach czasowych wg wzoru:

$$\begin{cases} y^{(n)} = y^{(n-1)} + \dfrac{dy}{dt} dt \\ u^{(n)} = u^{(n-1)} + \dfrac{du}{dt} dt \end{cases}, \tag{5}$$

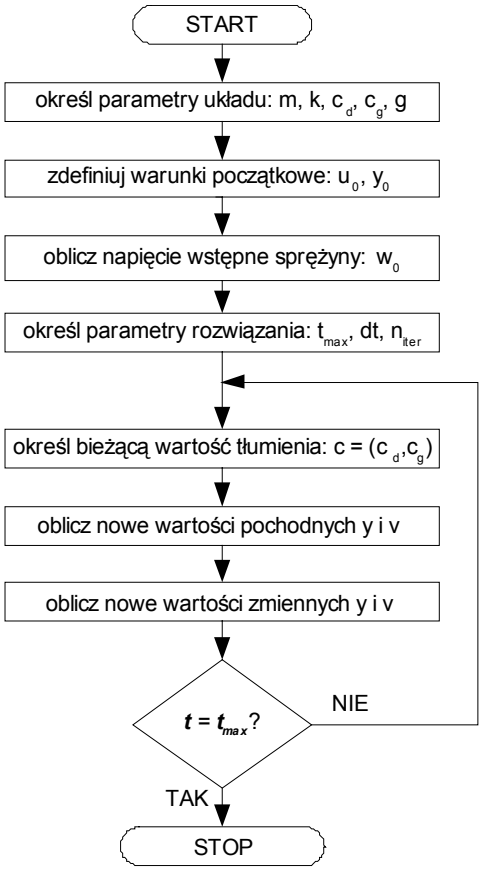
gdzie indeks górny n oznacza poziom czasowy. Przedstawiona tu dyskretyzacja czasowa oparta jest na schemacie Eulera. Jest to schemat I–go rzędu, mimo to jest on wystarczający do realizacji zadania.

Do wyznaczenia $y^{(1)}$ oraz $u^{(1)}$, czyli wartości w pierwszym przebiegu pętli obliczeniowej, potrzebne są informacje o wartości $y^{(0)}$ oraz $u^{(0)}$ - wielkości te stanowią tzw. warunki

początkowe, definiujące stan układu w chwili $t = 0$. Istotne jest przy tym, aby początkowe wychylenie było różne od zera ($y^{(0)} \neq 0$), w przeciwnym razie układ znajdować się będzie w stanie równowagi i drgania w ogóle się nie rozpoczną.

Do wyznaczania nowych wartości y oraz u na kolejnych poziomach czasowych potrzebne są jeszcze wartości pochodnych $\mathrm{dy/dt}$ oraz $\mathrm{du/dt}$. Pochodne te należy obliczać ze wzoru (2), przy czym przy pierwszym powtórzeniu pętli y oraz u przyjmą wartości początkowe $y^{(0)}$ i $u^{(0)}$.

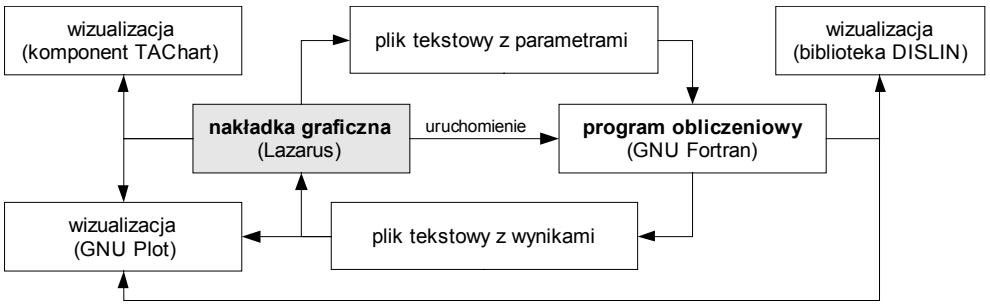
Z przedstawionego powyżej opisu wynika kolejność wykonywania poszczególnych obliczeń w pętli: najpierw należy wyznaczyć bieżącą wartość stałej tłumienia, zależnej od kierunku ruchu, następnie wartości pochodnych, a na koniec nowe wartości y oraz u. Pełny algorytm obliczeniowy przedstawiony jest na rysunku 2.



Rys. 2. Algorytm programu obliczeniowego
Fig. 2. Computational programme algorithm

Implementacja algorytmu obliczeniowego

Implementację przedstawionego wyżej algorytmu obliczeniowego można wykonać na wiele odmiennych sposobów i przy użyciu różnych języków programowania. Na rys. 3 przedstawiono schemat przykładowego pakietu obliczeniowego, zrealizowanego w całości za pomoca opisanych wcześniej narzędzi z nurtu Wolnego Oprogramowania. Pakiet ten dostępny jest na stronie autora, w dziale poświęconym przykładom zastosowania Fortranu i technik wizualizacyjnych [Sobieski 2009a].



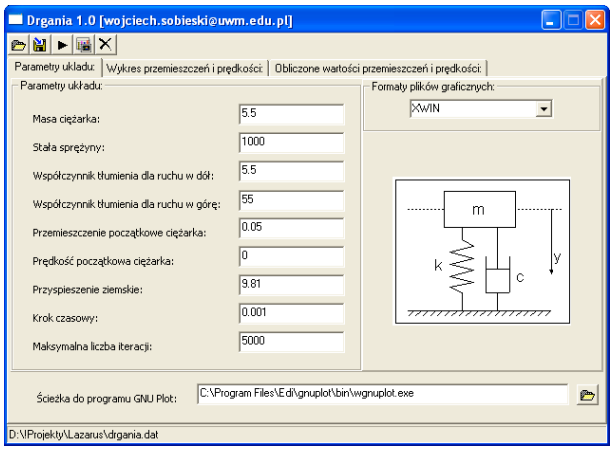
Rys. 3. Struktura logiczna pakietu obliczeniowego
Fig. 3. Logical structure of computational package

Działanie przedstawionego na rys. 3 pakietu obliczeniowego jest następujące:

– Nakładka graficzna wczytuje plik z parametrami układu lub tworzy nowy na podstawie wartości podanych przez użytkownika (rys. 4). Oprócz podstawowych danych wymaganych przez algorytm obliczeniowy, w nakładce definiuje się również wyjście dla biblioteki graficznej DISLIN. W ogólniejszym przypadku, w nakładce graficznej można dołączyć tablice materiałowe, fizyczne, matematyczne, termodynamiczne bądź też inne elementy ułatwiające definiowanie modelowanych zagadnień i usprawniające proces przygotowywania modelu komputerowego. Ponieważ plik z parametrami jest zwykłym plikiem tekstowym, można go również edytować bezpośrednio, bez pomocy nakładki, lub też korzystać z nakładek alternatywnych (lub różnych programów go generujących).
– Po przygotowaniu pliku z danymi niezbędnymi do wykonania obliczeń, z poziomu nakładki (lub w inny sposób) uruchamiany jest program obliczeniowy. Program obliczeniowy odczytuje dane z pliku wygenerowanego wcześniej przez nakładkę (lub powstałego w inny sposób), wykonuje obliczenia, a wyniki wizualizuje. W rozwiązaniu przedstawionym na rys. 3, z poziomu programu obliczeniowego dostępne są dwie metody wizualizacji. Pierwsza polega na wykorzystaniu biblioteki graficznej DISLIN, druga na wywołaniu polecenia systemowego uruchamiającego zewnętrzny skrypt Gnuplota. Po uruchomieniu tego skryptu rysunek przedstawiany jest na ekranie monitora lub też zapisywany do pliku graficznego w jednym z kilkunastu dostępnych formatów. Istotne jest, że w przypadku korzystania ze skryptów Gnuplota wymagany jest wcześniejszy zapis wyników obliczeń do pliku (lub plików) tekstowego o odpowiedniej strukturze. W przypadku korzystania z biblioteki DISLIN dane do obliczeń zapisać na-

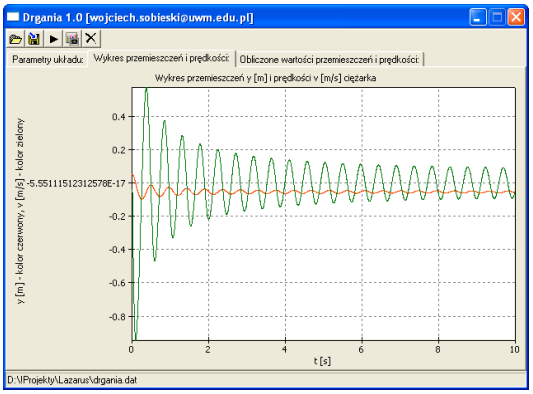
leży w dodatkowych, jednowymiarowych zmiennych tablicowych. Podobnie jak Gnuplot, biblioteka DISLIN umożliwia wizualizację na ekranie monitora lub zapis do plików graficznych w kilkunastu formatach. Więcej informacji o bibliotece DISLIN oraz środowisku Gnuplot znaleźć można w pracy [Sobieski 2008].

– Po zakończeniu obliczeń można powrócić do nakładki graficznej i wykonywać dalsze operacje. W przykładzie zilustrowanym na rys. 3, nakładka graficzna umożliwia działania dwojakiego typu. Po pierwsze, z poziomu interfejsu programu można ponownie uruchomić skrypt Gnuplota. W ogólnym przypadku można zdefiniować dowolną liczbę takich skryptów, bazujących na tym samym pliku wynikowym (lub wielu plikach zapisanych wcześniej z poziomu programu obliczeniowego), lub też wywoływać dowolne inne środowiska wizualizacyjne, obliczeniowe itp. Po drugie, w nakładce graficznej można ponownie wizualizować wyniki lub też poddawać je dalszej obróbce. W przedstawionym przykładzie implementacji rozwiązania zadania możliwe jest wyświetlenie wyników w postaci kolumn liczb oraz sporządzenie wykresu przemieszczeń i prędkości ciężarka jako funkcji czasu (rys. 5). Do tworzenia wykresu zastosowano komponent wizualny do środowiska Lazarus o nazwie TAChart.

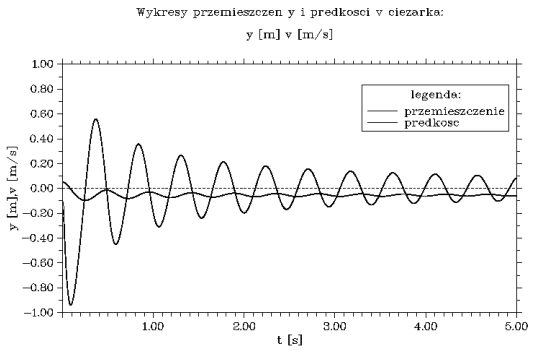


Rys. 4. Interfejs nakładki graficznej (zakładka z definicją parametrów modelu)
Fig. 4. Graphical overlay interface (overlap with definition of model parameters)

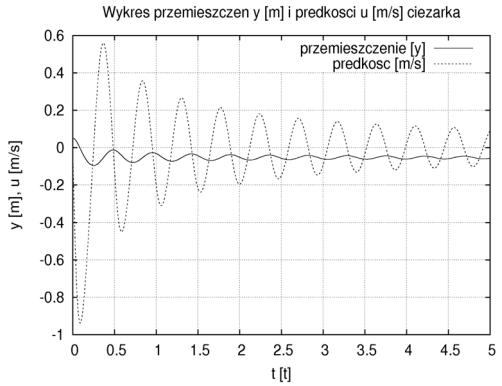
Na rysunkach 6 i 7 przedstawiono wykresy uzyskane za pomocą biblioteki DISLIN oraz środowiska Gnuplot. Parametry użyte do wykonania przykładowych obliczeń były następujące: masa ciężarka $m = 5{,}5$ [kg], stała sprężyny $\mathrm{k} = 1000$ [N·m^{-1}], współczynnik tłumienia w górę $\mathrm{c_g} = 55$, współczynnik tłumienia w dół $\mathrm{c_g} = 5{,}5$, początkowe wychylenie ciężarka $y^{(0)} = 0{,}05$ [m], początkowa prędkość ciężarka $u^{(0)} = 0$ [m·s^{-1}], przyspieszenie ziemskie $g = 9{,}81$ [m·s^{-2}], krok czasowy $\mathrm{dt} = 0{,}001$ [s], maksymalna liczba iteracji $n_{iter} = 2000$ [-].



Rys. 5. Interfejs nakładki graficznej (zakładka z wykresem)
Fig. 5. Graphical overlay interface (overlap with a diagram)



Rys. 6. Wykres uzyskany przy zastosowaniu biblioteki graficznej DISLIN
Fig. 6. Graphical overlay interface (overlap with a diagram)



Rys. 7. Wykres uzyskany przy zastosowaniu środowiska graficznego Gnuplot
Fig. 7. Diagram obtained using the Gnuplot graphical environment

Podsumowanie

Przedstawiona w artykule ideologia programowania może być zastosowana do tworzenia pakietów obliczeniowych o dowolnym stopniu komplikacji i w dowolnym obszarze zastosowań. W prezentowanym podejściu istotne jest to, że wszystkie użyte do realizacji zadania narzędzia informatyczne są bezpłatne i ogólnie dostępne oraz fakt, że można ich używać w kilku najpopularniejszych systemach operacyjnych. Jak pokazuje przedstawiony w artykule przykład, a także inne podobne przykłady pakietów obliczeniowych umieszczone na stronie autora, rozwiązania bazujące na Wolnym Oprogramowaniu są równie efektywne i wygodne w obsłudze jak analogiczne rozwiązania utworzone przy zastosowaniu narzędzi komercyjnych. Mimo, że z pewnością istnieją pewne grupy zagadnień, w których Wolne Oprogramowanie nie będzie wystarczające, to w większości zastosowań typowych dla inżynierii rolniczej, i nie tylko dla niej, może ono być z powodzeniem stosowane. Warto zwrócić uwagę Czytelnika na jeszcze jeden aspekt. Otóż opierając swoje rozwiązania informatyczne na Wolnym Oprogramowaniu, użytkownik zawsze ma możliwość bezpłatnej jego aktualizacji. W przypadku programów komercyjnych mogą pojawić się pewne trudności z pozyskaniem środków finansowych na ten cel – szczególnie w okresach recesji. Wolne Oprogramowanie jest na ten aspekt gospodarczy bardzo odporne. Zdarza się również, że oprogramowanie komercyjne, ale nie aktualizowane na bieżąco, staje się szybko przestarzałe w stosunku do ciągle rozwijających się projektów Wolnego Oprogramowania: poprzez „wolne" rozwiązania można zatem uwolnić się od charakterystycznej dla aplikacji komercyjnych spirali kosztów. W praktyce inżynierskiej liczy się głównie jakość efektu końcowego, a jeśli została ona osiągnięta bez uszczerbku mniejszym nakładem kosztów, tym lepiej dla jego realizatora.

Bibliografia

Chrobak D. 2003. Fortran, praktyka programowania. Wydawnictwo Mikom, Warszawa. ISBN: 83-7279-361-1

Piechna J. R. 2000. Programowanie w języku Fortran 90 i 95. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa. ISBN 83-7207-225-6

Pluta J. Ruch harmoniczny wymuszony [on line]. [dostęp 20.01.2009]. Dostępny w Internecie: http://www.if.pw.edu.pl/~pluta/pl/dyd/am/am2/w5/segment4/main.htm

Sobieski W. 2008. GNU Fortran z elementami wizualizacji danych. Wydawnictwo UWM, Olsztyn. ISBN 978-83-7299-553-7

Sobieski Wojciech – strona informacyjna [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://moskit.uwm.edu.pl/~wojsob/

Wierzbanowski K. Materiały dydaktyczne z fizyki ogólnej [on line]. [dostęp 20.01.2009]. Dostępny w Internecie: http://www.ftj.agh.edu.pl/~wierzbanowski/R_Harm%287%29.pdf

Trykozko A. 1999. Ćwiczenia z języka Fortran. Wydawnictwo Mikom, Warszawa. ISBN: 83-87102-66-0.

DISLIN [on line]. [dostęp 14.01.2009]. Dostępny w Internecie: http://www.mps.mpg.de/dislin/

GCC Project [on line]. The GNU Compiler Collection. [dostęp 14.01.2009]. Dostępny w Internecie: http://gcc.gnu.org/

Free BSD [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.freebsd.org/

Free DOS [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.freedos.org/

Free Pascal [on line]. [dostęp 14.01.2009]. Dostępny w Internecie: http://www.freepascal.org/

Force Project [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://force.lepsch.com/

GNU Operating System [on line]. [dostęp 14.01.2009]. Dostępny w Internecie: http://www.gnu.org/

Gnuplot [on line]. [dostęp 14.01.2009]. Dostępny w Internecie: http://www.gnuplot.info/

Lazarus Project [on line]. [dostęp 14.01.2009]. Dostępny w Internecie: http://www.lazarus.freepascal.org/

LaTeX [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.latex-project.org/

Mandriva Linux [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.mandriva.com/

MayaVi [on line]. [dostęp 15.01.2009]. Dostępny w Internecie: http://mayavi.sourceforge.net/

Mozilla Firefox [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.mozilla-europe.org/pl/

NetBSD [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.netbsd.org/

Notepad++ [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://notepad-plus.sourceforge.net/

OpenBSD [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.openbsd.org/

OpenGL [on line]. [dostęp 15.01.2009]. Dostępny w Internecie: http://www.opengl.org/

OpenOffice.org [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.openoffice.org/

OpenSuse Linux [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://en.opensuse.org/

ParaView [on line]. [dostęp 15.01.2009]. Dostępny w Internecie: http://www.paraview.org/

Projekt Edi [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://moskit.uwm.edu.pl/~wojsob/

TAChart [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://wiki.lazarus.freepascal.org/TAChart

The Kate Project [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.kate-editor.org/

The KOffice Project [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.koffice.org/

Ubuntu Linux [on line]. [dostęp 16.01.2009]. Dostępny w Internecie: http://www.ubuntu.com/

FREE SOFTWARE IN ENGINEERING PRACTICE

Abstract. The article presents a set of informatic tools derived from the stream of the so-called Free Software, and a proposal of logical structure for connections between them, ensuring comfortable and efficient solving of any computational tasks occurring in contemporary engineering, including agricultural engineering. The purpose of the work is to present certain programming ideology, which allows to create packages characterised by loose, modular structure, containing alternative solutions and allowing to make any configurations of connections between applications. It is also important that the article proposes specific informatic solutions based on the Free Software. Vibrating system motion equation makes the background for presenting the proposed informatic solutions.

Key words: Free Software, Fortran, DISLIN, Gnuplot, Lazarus

Adres do korespondencji:
Wojciech Sobieski: e-mail: wojciech.sobieski@uwm.edu.pl
Katedra Mechaniki i Podstaw Konstrukcji Maszyn
Uniwersytet Warmińsko-Mazurski
ul. M. Oczapowskiego 11
10-957 Olsztyn