Modelowanie systemów dynamicznych

Krzysztof Wesołowski,
grupa, 11:00, s. 317c
Prowadzący: dr inż. Adam Piłat
Data odbycia zajęć: 12 X 2009

Sprawozdanie numer 2

Część I – podsumowanie zajęć.

Zajęcia służyły głównie zapoznaniu się z Simulinkiem, sposobem jego działania oraz różnymi sposobami jego obsługi. Aby nie powtarzać w sprawozdaniu prostych układów bloczków wraz z Scopem zmienimy trochę podejście. Ręczne przełączanie się do workspace w celu analizy wyników również nie należy do najefektywniejszych sposobów pracy. W związku z tym poza zachowaniem narzuconej zawartości mocno skupiłem się na integracji Simulinka i Matlaba za pomocą komend wywoływanych z konsoli/m-pliku.

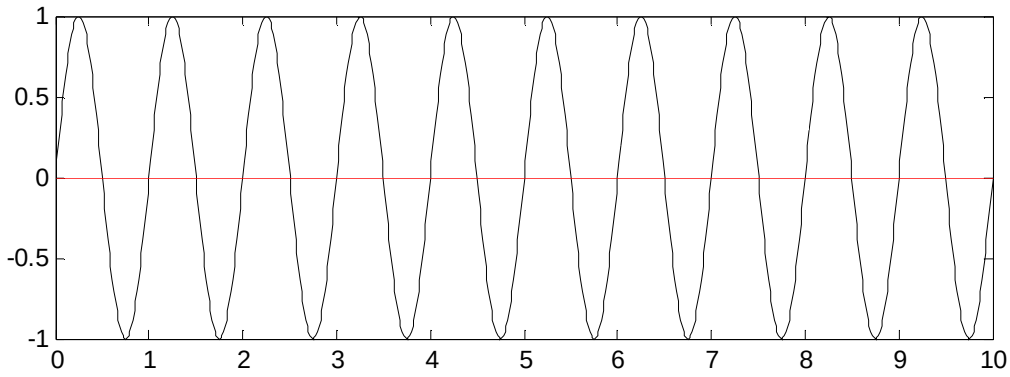
Generowanie funkcji i porównanie metod rozwiązywania.

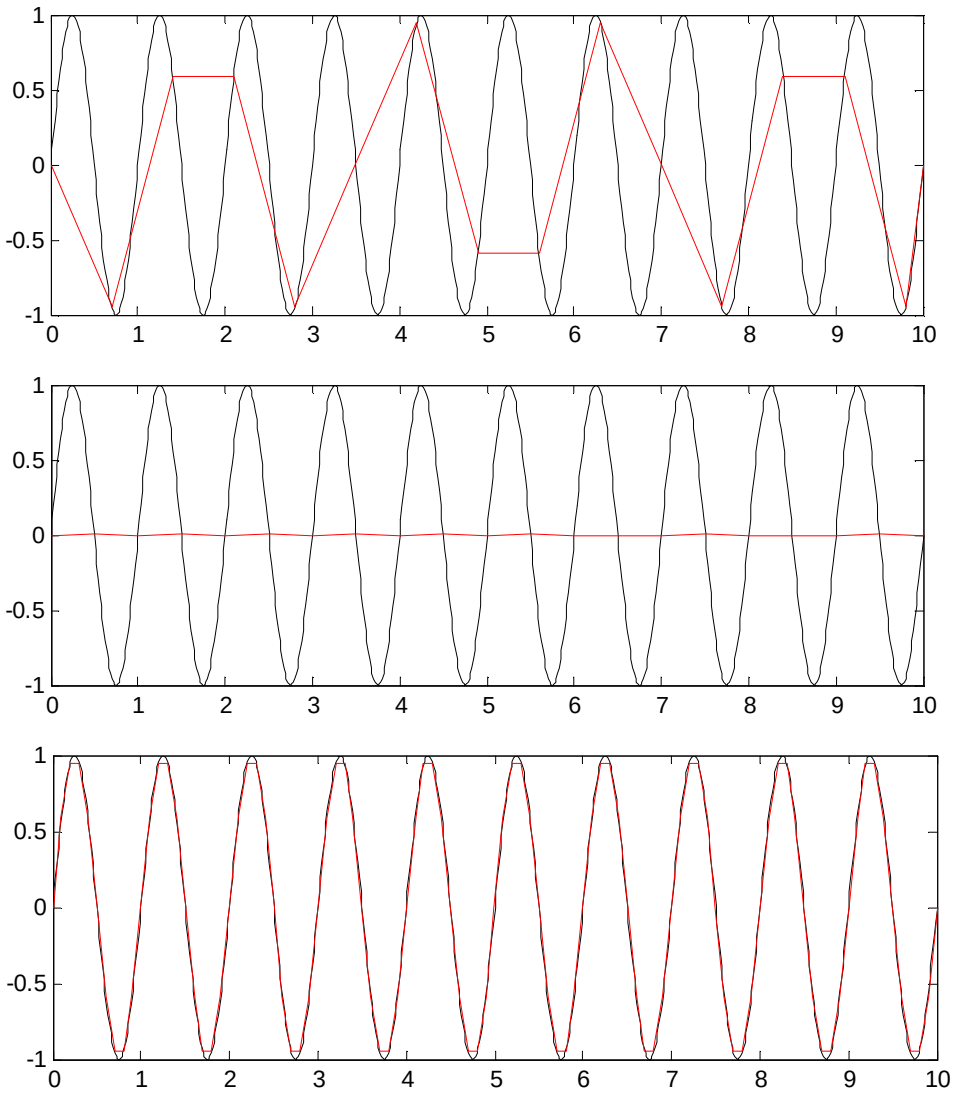
Simulink 1 – Porównywanie rozwiązań w zależności od step size.


```
function [] = porownanie(step)

[idealny_t,null,idealny_x]=sim('model_01',[],SIMSET('MaxStep',1/64));
[symulowany_t,null,symulowany_x]=sim('model_01',[],SIMSET('MaxStep',step));
plot(idealny_t,idealny_x,'k',symulowany_t,symulowany_x,'r');

end
```

Poniżej kilka wywołań funkcji porowanie, dla wartości step= odpowiednio 1, 0.7, 0.5, 0.1

Za idealny przebieg przyjęto step=1/64.





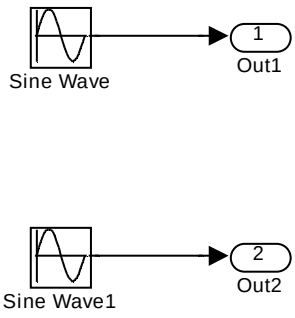
Widać jak ważne jest dobranie kroku do dynamiki obiektu. Za stosunkowo empiryczna wartość wymaganego step'a przyjmuje się 1/32 lub 1/64 wartości okresu.

Wymiana danych simulink<->Matlab

Matlab->Simulink	Simulink->Matlab
1. Zmienne globalne bieżącego workspace – można po prostu używać ich nazw w modelu, musza być ustawione przed symulacja. 2. Parametry symulacji ustawiane za pomocą polecenia simset podanego, jako argument do sim (przy programowym wywołaniu symulacji). 3. Każdy model w wielu sytuacjach (np. on load, on run etc.) może uruchamiać zadany kod matlaba przygotowując sobie zmienne z pkt 1.	1. Zapis poprzez obiekt Scope (w ustawieniach) 2. Zapis poprzez bloczek To Workspace (tylko nowsze wersje, bardziej intuicyjny) 3. Zapis poprzez porty wyjściowe i programowa realizacje ([t,x,y]=sim(…)).

Krzywe Lissajous

Simulink 2 - Lissajous




```
function [] = Lissajous(freq1, phase1, freq2, phase2)

assignin('base','first_freq',freq1);
assignin('base','first_phase',phase1);
assignin('base','second_freq',freq2);
assignin('base','second_phase',phase2);

[t,null,x1,x2]=sim('model_02');

plot(x1,x2);
end
```

```
function [] = plotLs(max_size)

fazy=[0     ,0  ,0      ,0;
      pi/2  ,0  ,pi/2   ,pi/3];
freq=[1     ,1  ,1      ,1;
      1     ,2  ,3      ,1];
posx=[100,100,100+max_size,100+max_size];
posy=[100,100+max_size,100,100+max_size];

for i=1:4
    hf=figure(i);
    Lissajous(freq(1,i),fazy(1,i),freq(2,i),fazy(2,i))
    set(hf,'Position',[posx(i),posy(i),max_size,max_size]);
end

end
```

Powyższy prosty model (który można było całkowicie zrealizować w m-file) uzupełniony skryptem rysującym na bieżącym figure zadana krzywa, oraz skrypt zmieniajacy i rozmieszczający figure tworzą prezentacje krzywych Lissajous.

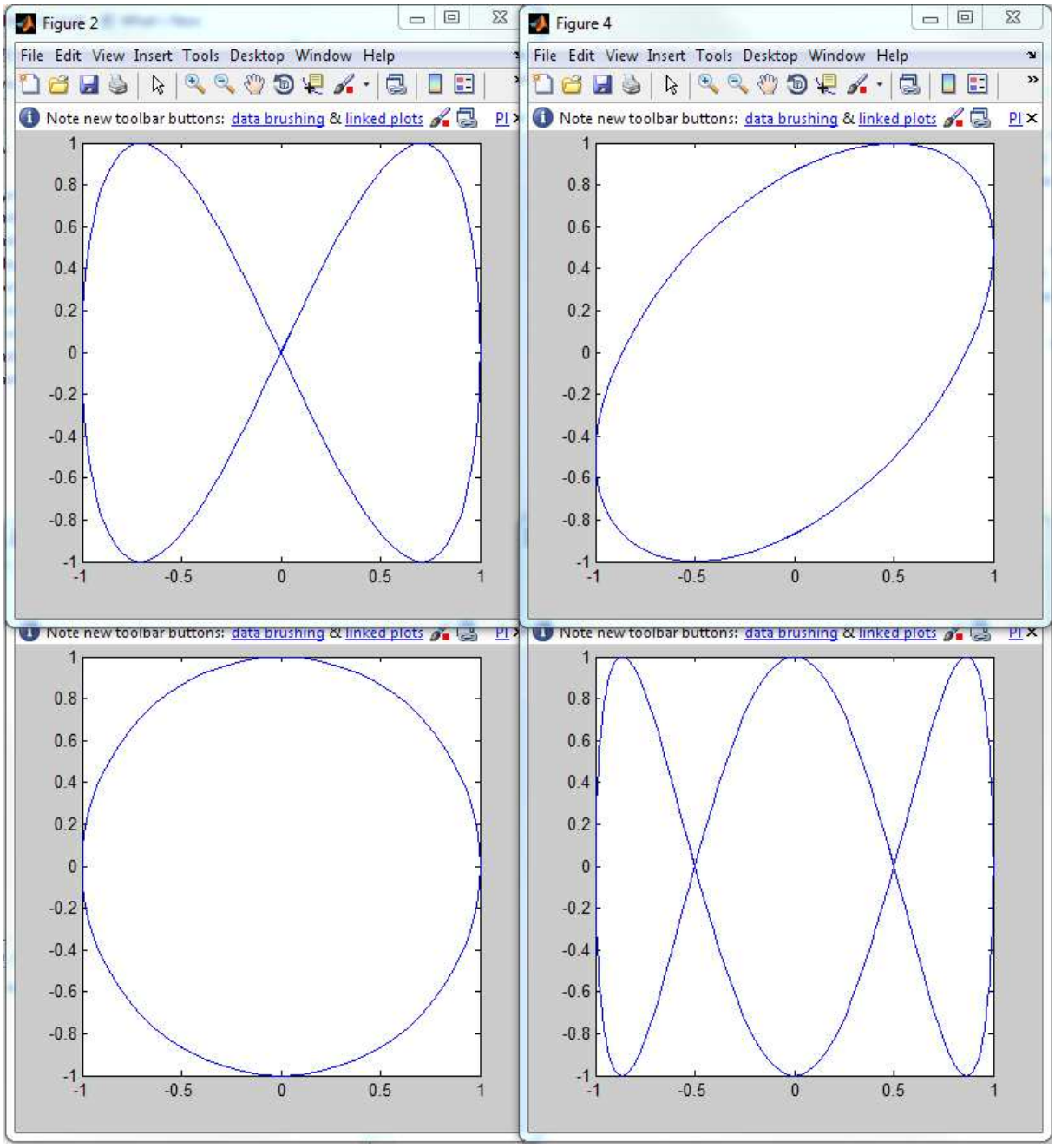
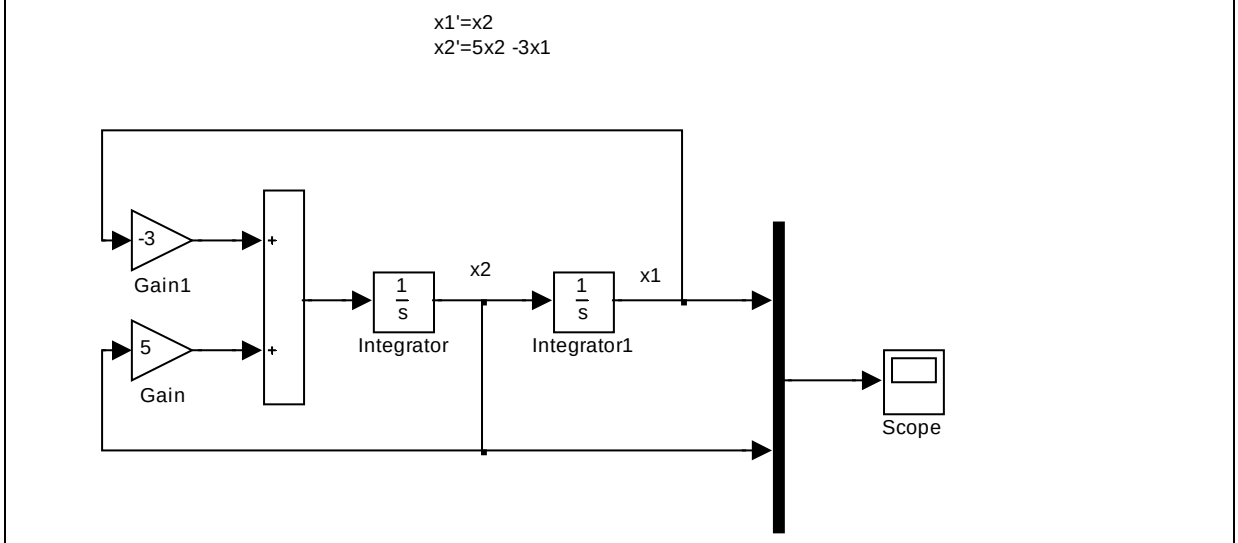


Figure 2
File Edit View Insert Tools Desktop Window Help
Note new toolbar buttons: data brushing & linked plots
Figure 4
File Edit View Insert Tools Desktop Window Help
Note new toolbar buttons: data brushing & linked plots
Note new toolbar buttons: data brushing & linked plots
Note new toolbar buttons: data brushing & linked plots

Modelowanie prostego równania różniczkowego.

Simulink 3



`Podstawowy schemat realizujący zapisane na nim r.r.`
`W całkach można podać warunki początkowe, ale układ ten nie osiąga nieskończoności tylko dla 0,0`

Korzystając z polecenia eig można łatwo obliczyć wartości właśnie takiego układu równań(`eig([0 1;-3 5])`) i tym samym wyjaśnić niestabilność: obie wartości własne są dodatnie, co daje nam rozwiązanie wykładnicze – dla niezerowych warunków początkowych biegnące do nieskończoności.

Aby układ ustabilizować wystarczy zamiana 5 na -5, co da nam wartości własne:
-0.6972
-4.3028

Warto zwrócić uwagę, że dla zerowego tłumienia otrzymujemy:
> eig([0 1;-3 0])

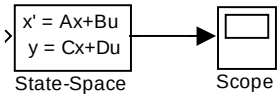
ans =

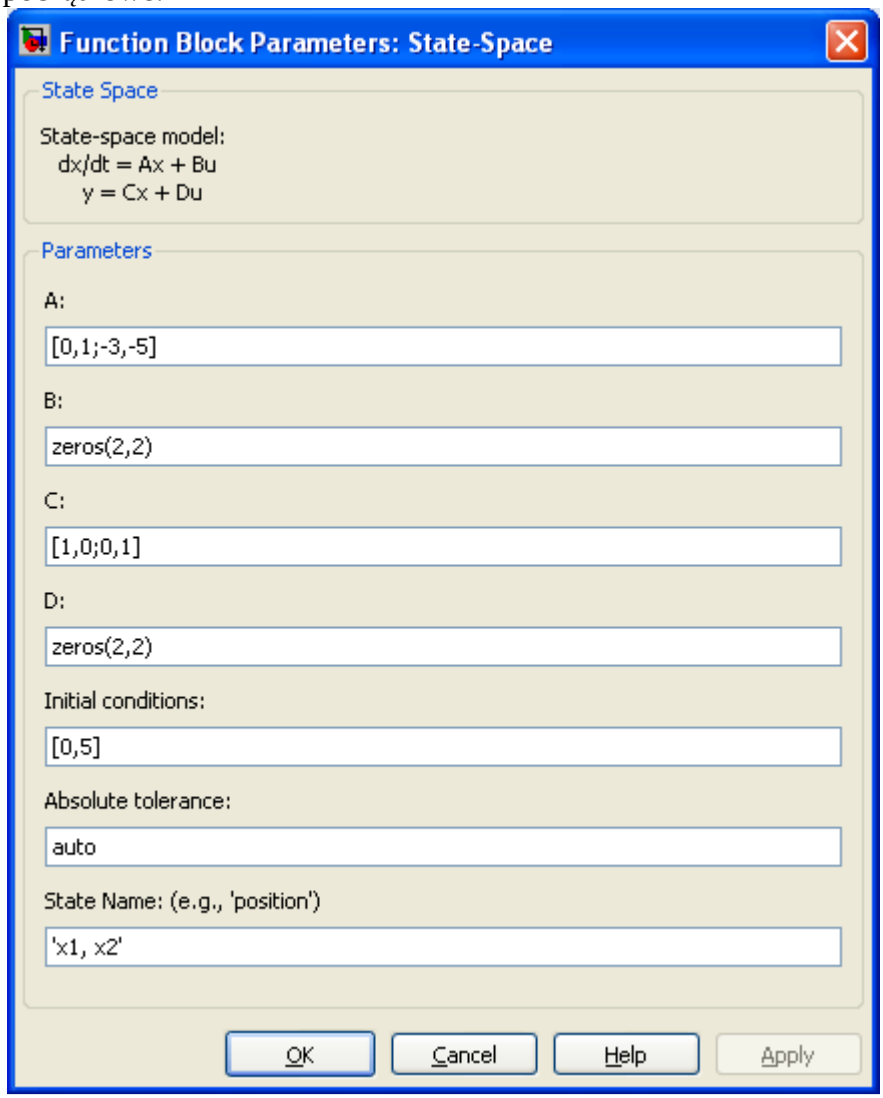
0 + 1.7321i
0 - 1.7321i

Czyli oscylacje o okresie 2*pi/sqrt(3)= 3.6276 s

Można więc podsumować, iż Simulink modeluje rzeczywisty obiekt poprzez rozwiązywanie opisujących go równań różniczkowych.

Warto w tym miejscu zwrócić uwagę że gdy nasze umiejętności się rozwiną można zamiast misternie układać model z klocków skorzystać z obiektu StateSpace, którego ustawienia to świetnie znane z TS macierze i warunki początkowe:

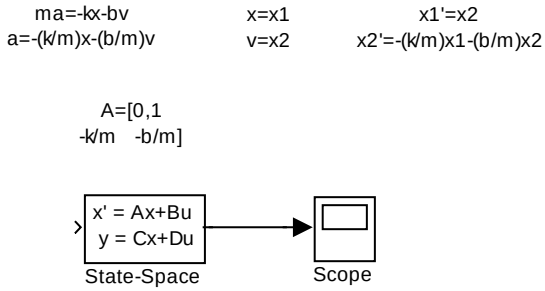


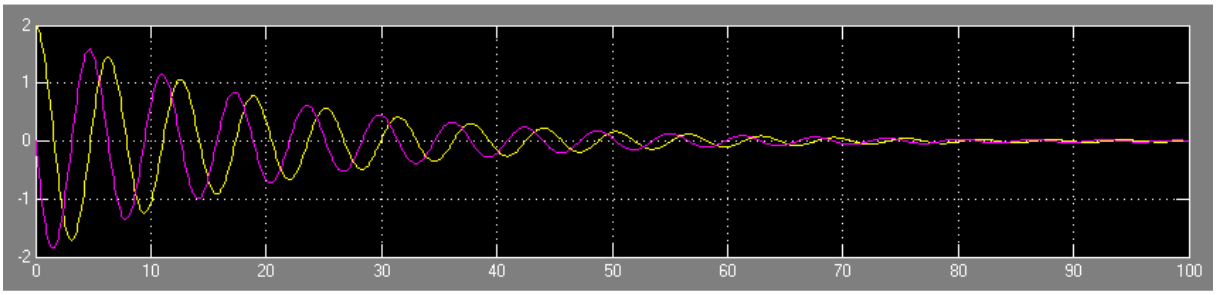


Macierz C służy tu tylko przepisaniu naszego stanu na wyjście (na potrzeby wykresów).

Część II – zadanie

Najprostsza implementacja, wymaga zdefiniowania w workspace zmiennych sprężystości – k, masy – m, i tłumienia – b.





Przebieg dla m=1, k=1, b=0.1

Model

Model jest to twór przybliżający zachowanie się rzeczywistego systemu. Model zwykle wprowadza pewne uproszczenia, często też nie jesteśmy w stanie zmierzyć potrzebnych do modelu parametrów.
Celem tworzenia modelu jest uzyskanie możliwości przeprowadzania różnego rodzaju analiz dotyczących zachowania się modelu w danych warunkach, możliwości jego sterowania bez oddziaływania na rzeczywisty obiekt.

Modele dzielą się na fizyczne – będące tylko przeskalowanymi obiektami z rzeczywistości, lub matematyczne, opisujące rzeczywistość za pomocą równań.
Dzięki wzrastającej mocy obliczeniowej komputerów największą rolę zyskały metody matematyczne – a dokładniej metody modelowania komputerowego, polegające na numerycznym rozwiązywaniu równań różniczkowych opisujących model.