Zakład Systemów Elektronicznych i Telekomunikacyjnych

Metody numeryczne

2 - NARZĘDZIA PROGRAMOWE – CZĘŚĆ 2

I. Cel ćwiczenia

Celem ćwiczenia laboratoryjnego jest poszerzenie wiedzy w zakresie oprogramowania narzędziowego, dedykowanego do wykonywania zaawansowanych obliczeń numerycznych. Część druga tematu dotyczy zagadnień wizualizacji wyników obliczeń numerycznych za pomocą wykresów, a także wykorzystywania typowych konstrukcji programistycznych, które w przyszłości ułatwią sformułowanie i realizację dowolnego problemu w środowisku programu Scilab.

II. Realizacja ćwiczenia laboratoryjnego

Uwaga: w czasie realizacji poszczególnych części ćwiczenia laboratoryjnego należy wspomagać się informacjami, które zawarto w podręcznej pomocy programu Scilab, uruchamianej klawiszem *F1*, niezależnie dla każdego aktywnego okna programu, lub rozkazem `help nazwa_instrukcji`, wprowadzaną (po znaku zachęty `-->`) w oknie głównym konsoli.

1. Wykresy są jednym z podstawowych źródeł informacji o efektach realizowanych obliczeń numerycznych. Program Scilab oferuje szeroką gamę funkcji, za pomocą których można przygotowywać dowolne wizualizacje otrzymanych danych. Wiele użytecznych przykładów jest dostępnych w oknie przewodnika – menu */?/Przykłady*, sekcja *Grafika*. W przygotowanych przykładach zawarty jest kod źródłowy, który pozwala na podgląd skryptu realizującego demonstrowany efekt wizualizacji (opcja *--Zobacz Kod--* dostępna z poziomu menu okna wykresów).

   2.1. Wykresy 2D

   Zadanie do rozwiązania

   Korzystając z podstawowej funkcji do generowania wykresów dwuwymiarowych `plot2d`, należy wykreślić i sformatować (na jednym wykresie) krzywe dwóch funkcji: $y_1 = e^x \cdot \cos(x^2) \cdot \sin\left(\frac{1}{x^2}\right)$ oraz $y_2 = e^x \cdot \sin\left(\frac{1}{x^2}\right)$, dla x z przedziału $(0,2\pi]$.

   Pomoc

   Podział przedziału dla wykresu można zrealizować funkcją: `linspace(x1,x2 [,n])`, gdzie `x1` i `x2` określają odpowiednio początek i koniec przedziału, natomiast opcjonalny parametr `n` oznacza liczbę elementów definiowanego wektora (domyślnie ustawioną na 100). To samo można uzyskać definiując wektor z krokiem (patrz ćwiczenie 1), lecz należy pamiętać o tym, że wartość końcowa przedziału może nie zostać osiągnięta, co wynika z wartości zadeklarowanych przez użytkownika, przykład: `x=x1:krok:x2`, gdzie `krok` oznacza wartość pomiędzy kolejnymi elementami wektora. Przykład wykorzystania funkcji wykresu: `plot2d(x', [y1' y2'])` zakładając, że `x` było wcześniej definiowane jako wektor wierszowy (np. instrukcją `linspace`).

   `plot2d(x,y)` (`x` i `y` – wektory) - wynikiem działania funkcji jest wykres punktów o współrzędnych *(x,y)*, pochodzących kolejno z obu wektorów. Oba wektory muszą mieć jednakowe długości; nie jest istotne czy są one wierszowe czy kolumnowe, przy czym, dla poprawnego kolorowania wykresów, zaleca się by wektory były definiowane jako kolumnowe. Jeśli `x` jest wektorem kolumnowym, a `y` jest macierzą o tej samej - co `x` - liczbie wierszy, to funkcja wykreśli tyle różnych wykresów, ile kolumn jest w macierzy `y` – wszystkie wykresy będą wykreślone względem wartości `x`. Jeśli i `x` i `y` są macierzami o jednakowych rozmiarach, to funkcja wykreśli tyle wykresów ile kolumn jest w macierzach `x` i `y`.

Poniżej zawarto wykaz kilku funkcji, przydatnych podczas formatowania wykresów. Należy zapoznać się z ich szczegółową składnią, dostępną w pomocy programu. Więcej informacji o formatowaniu wykresów zawarto także w rozdziale 4 publikacji: B. Pinçon, *Wprowadzenie do Scilaba*, dostępnej na komputerach przedmiotowego laboratorium.

- **`clf()`** - kasowanie aktualnego wykresu;
- **`xtitle()`** – tytuł wykresu;
- **`xgrid()`** – siatka wykresu;
- **`legend()`** – legenda na wykresie;
- **`xset()`** – ustalanie parametrów graficznych wykresu;

W celu obserwacji metod formatowania kolorów i elementów krzywych na wykresach, należy uruchomić i przeanalizować poniższe skrypty:

```
//Program obrazujący kolorystykę krzywych na wykresach
nr=[1:20];
x=[1;10];
y=[1;1];
xset('thickness',4) //grubość linii = 4 pixele
plot2d(x,y*nr,style=nr)
xtitle('Kolory krzywych','To nic nie znaczy','Nr parametru style w f. plot2d')
xset('thickness',1)
xgrid() //siatka wykresu
legend('1','2','3','itd.') //legenda do wykresu

//Program obrazujący elementy krzywych na wykresach
nr=-[1:14]; //UWAGA - istotny jest znak minus dla parametru style w f. plot2d
x=[1:10]';
y=ones(x);
plot2d(x,y*nr,style=nr)
xtitle('Elementy krzywych','To nic nie znaczy','Nr parametru style w f. plot2d')
```

Zaprezentowane metody formatowania wykresów stanowią tylko nieliczną część możliwości programu Scilab w tym zakresie. Niemniej jednak, są one wystarczającą podstawą do realizacji kolejnych zadań w zakresie przedmiotu metody numeryczne.

2.2. Wykresy 3D

Zadanie do rozwiązania

Korzystając z podstawowej funkcji do generowania wykresów trójwymiarowych **`plot3d`**, należy wykreślić i sformatować wykres funkcji: $f(x,y)=\sin(x)\cdot\cos(y)$, dla x i y z przedziału $[0,2\pi]$. W przygotowanym programie zamienić funkcję **`plot3d`** na **`plot3d1`** i scharakteryzować różnicę w funkcjonowaniu obydwu funkcji.

Zadanie do przeanalizowania

Korzystając z funkcji do generowania wykresów trójwymiarowych **plot3d1**, należy wykreślić i sformatować wykres funkcji, który w układzie współrzędnych sferycznych prezentuje unormowaną charakterystykę mocy dla elementarnego źródła promieniowania, jakim jest dipol Herza: $P_n(\theta) = \sin^2(\theta)$, dla θ z przedziału [0, π] oraz Φ z przedziału [0, 2π].

```
theta=linspace(0,%pi,50)';
phi=linspace(0,2*%pi,50)';
//definicja funkcji, reprezentującej charakterystykę mocy w ukł. wsp. sferycznych,
// i jednoczesna zamiana współrzędnych sferycznych na kartezjańskie
deff("[x,y,z]=scp(theta,phi)",["x=(sin(theta)^2).*sin(theta).*sin(phi)";..
                               "y=(sin(theta)^2).*sin(theta).*cos(phi)";..
                               "z=(sin(theta)^2).*cos(theta)"])
//funkcja ułatwiająca konstrukcję powierzchni wykresu, opisanej (w przykładzie)
// definicją zmieniającą współrzędne układu sferycznego na kartezjański
[Xf,Yf,Zf]=eval3dp(scp,theta,phi);
//ustawienie mapy kolorów na wykresie
xset("colormap",jetcolormap(64))
//można jeszcze użyć innych profili kolorów: autumncolormap, bonecolormap,
// coolcolormap, coppercolormap, graycolormap, hotcolormap, hsvcolormap,
// jetcolormap, oceancolormap, pinkcolormap, rainbowcolormap,
//springcolormap, summercolormap, whitecolormap, wintercolormap
plot3d1(Xf,Yf,Zf)
xgrid()
```

Pomoc

Wynik działania polecenia **plot3d** jest analogiczny do przypadku funkcji **plot2d**. Polecenie to musi posiadać co najmniej trzy argumenty, np.: **plot3d(x,y,z)** - wartości **x** i **y** muszą być wektorami (o różnych lub równych długościach, natomiast wartości **z** muszą zostać podane jako macierz o rozmiarze (*x* x *y*) (gdzie *x* i *y* są długościami wektorów **x** i **y**). UWAGA: należy pamiętać, że macierz **z** musi zawierać wartości dla wszystkich kombinacji współrzędnych *x* i *y*. Istnieje wiele sposobów zbudowania macierzy macierz **z**. Można do tego celu wykorzystać różne metody programowania (np. pętle **for**) lub wbudowane funkcje Scilab'a (np. **z=feval(x,y,f)**, gdzie **x** i **y** to wektory, natomiast **f** jest funkcją dwóch zmiennych pobieranych z **x** i **y**).

Polecenie **plot3d1** działa analogicznie do **plot3d** z tą różnicą, ze kolor danego elementu powierzchni zależy od wartości *z* jaka odpowiada temu elementowi. UWAGA: w przypadku braku kolorowania wykresu należy zamienić miejscami definicje dwóch osi układu współrzędnych.

W każdym oknie wykresu 3D znajduje się przycisk umożliwiający obracanie wykresu. Podczas jego obracania, u dołu okna pojawiają się wartości kątów (nazwane: *alpha*, *theta*) odpowiadających aktualnie wybranemu widokowi. Warto wiedzieć, że wartości te można podać jako argumenty polecenia **plot3d**.

Użytecznym narzędziem podczas formatowania wykresów jest użycie funkcji **get** i **set**, które jako argument przyjmują łańcuch określający obiekt graficzny, jakiego własności można poznać instrukcją **get** lub zmienić instrukcją **set**. Dla sporządzonego wykresu można np. sprawdzić (i zapamiętać pod zmienną **f**) jego parametry: **f=get("current_figure")**. Tak samo należy postępować w przypadku innych obiektów, przykładowo osi wykresu: **f=get("current_axes")**. Jednym ze sposobów zmiany tych parametrów jest użycie instrukcji set, dla przykładu: **set(f,"font_color",1)**. Szczegółowy opis własności obiektów graficznych znajduje się w pomocy pod hasłem **graphics_entities**.

3. Zapoznać się z metodami programowania w środowisku Scilab.

Zadania do rozwiązania

3.1. Wykorzystując operacje na macierzach, napisać 4 przykładowe skrypty demonstrujące działanie pętli programowych: `for` i `while` oraz instrukcji warunkowych: `if` oraz `select case`.

3.2. Napisać skrypt, w którym należy:

- utworzyć macierz elementów losowych o rozmiarze 5x5 i wartościach z przedziału [0,10);
- wyliczyć liczbę elementów: a) większych od 5; b) mniejszych lub równych 1;
- wyliczyć procent, jaki stanowią elementy: a) większe od 5; b) mniejszych lub równych 1.

Pomoc

Scilab, oprócz instrukcji matematycznych, dysponuje prostymi funkcjami języka programowania. Polecenia te są bardzo podobne do funkcji stosowanych w językach wysokiego poziomu, takich jak C, C++, Pascal itp. Podczas wykonywania różnego rodzaju pętli lub instrukcji warunkowych, użyteczne stają się operatory relacji, które w programie Scilab przyjmują postać:

- równości: `==`;
- nierówności: `~=`;
- większości, mniejszości itp.: `<`, `>`, `<=`, `>=`.

Wynikiem działania operatorów relacji jest wartość logiczna prawdy: `%T` lub fałszu: `%F`. Operatory te mogą także działać na macierze liczbowe. Jeżeli np. poleceniem `A=rand(5,6)` zostanie wygenerowana macierz liczb losowych, to polecenie `B=A<0.5` wygeneruje macierz o takim samym rozmiarze jak `A`, zawierającą wartości: `%T` w miejscach gdzie relacja jest spełniona, `%F` – w pozostałych miejscach macierzy `B`.

- Pętla `for`.

  Pętla `for` służy do powtarzania pewnego ciągu instrukcji. Zazwyczaj zmienna sterująca pętli przyjmuje kolejne wartości wektora liniowego, przykład: `for k=1:10, y=k/2, end`. Wektor iteracji może być także nieliniowy, np.: `for k=[1:10,13,18], y=k/2, end`. Liczba iteracji określona jest przez liczbę składników wektora. Jako wektora można użyć wyrażenia typu „start, krok, stop”, np: `for k=[1:0.2:2], y=k/2, end`.

- Pętla `while`.

  Pętla `while` pozwala na powtarzanie ciągu instrukcji tak długo, dopóki prawdziwy będzie zdefiniowany warunek, na którego wartość mogą wpływać wykonywane operacje, przykład: `x=1; while x<14,x=2*x, end`. Zadając warunek można wykorzystać wcześniej opisane operatory relacji. Możliwe jest także poszerzenie działania instrukcji warunkowej o słowo kluczowe `else` wg składni:

```
while wyrażenie_logiczne,
      polecenia_A
   else
      polecenia_B
end
```

- Funkcja `break`.

  W celu przerwania wykonywania pętli należy posłużyć się poleceniem `break`. Po jej użyciu wykonywane będą instrukcje, które są umieszczone poza pętlą.

- Instrukcja warunkowa **`if`**.

  Podobnie, jak w językach wysokiego poziomu, program Scilab oferuje instrukcje warunkowe, które można wykorzystać zgodnie ze składnią:

```
if warunek1 then
        ciag_instrukcji_wykonywanych_gdy_spelniony_jest_warunek1
    elseif warunek2 then
        ciag_instrukcji_wykonywanych_gdy_spelniony_jest_warunek2
        ...
    elseif warunekN then
        ciag_instrukcji_wykonywanych_gdy_spelniony_jest_warunekN
    else
        ciag_instrukcji_wykonywanych_gdy_nie_jest_spelniony_zaden_z_w/w
end
```

- Instrukcja warunkowa **`select case`**.

  Uproszczeniem w stosowaniu wielu instrukcji **`if`** jest konstrukcja warunkowa postaci:

```
select zmienna
    case wyrazenie_1
        ciag_instrukcji_wykonywanych_gdy_zmienna_równa_jest_wyrazeniu_1
        ...
    case wyrazenieN
        ciag_instrukcji_wykonywanych_gdy_zmienna_rowna_jest_wyrazeniu_N
    else
        ciag_instrukcji_wykonywanych_gdy_zmienna_nie_jest_równa_w/w
end
```

III. Sprawozdanie z ćwiczenia laboratoryjnego

Uwaga: sprawozdanie – przygotowane w pliku *.docx (*.doc) na podstawie wszystkich wytycznych formatki *mn_formatka.doc* – należy przesłać na adres pjanko@prz.edu.pl, **najpóźniej w terminie 1 tygodnia od dnia zakończenia ćwiczenia laboratoryjnego**. Odbiór każdego sprawozdania zostanie potwierdzony wiadomością zwrotną przez prowadzącego zajęcia. Sprawozdania powinny być opisane: nazwa roku, datą wykonania ćwiczenia, numerem grupy laboratoryjnej, numerem zespołu ćwiczącego (1-4) i jego składem osobowym. W przypadku odrabiania ćwiczenia, w/w informacje powinny zostać zawarte przy nazwisku osoby odrabiającej laboratorium. Odpowiedzialność za sprawozdanie jest zbiorowa, co oznacza, że wszyscy członkowie danego zespołu ćwiczącego (1-4) otrzymują tą samą ocenę za złożone sprawozdanie.

Sprawozdanie z drugiego ćwiczenia laboratoryjnego powinno zawierać kompleksowe rozwiązanie wszystkich zadań (także tych do przeanalizowania), które wyszczególniono w instrukcji. W sprawozdaniu należy zawrzeć uzyskiwane wyniki oraz obszerny komentarz do każdego etapu postępowania. W dokumencie należy zawrzeć przygotowane skrypty z komentarzami do każdej linii programu. Sprawozdanie należy podsumować wnioskami, wyciągniętymi z realizacji procesu obliczeń numerycznych w programie Scilab.

IV. Przygotowanie do następnych zajęć

1. Wiedza teoretyczna z zakresu rachunku macierzowego.
2. Umiejętność posługiwania się oprogramowaniem narzędziowym w zakresie zrealizowanym podczas pierwszego i drugiego ćwiczenia laboratoryjnego.