Rozdział 1

Fizyka laboratorium 1

1.1. Elementy analizy matematycznej

Funkcje

Zmienna y nazywa się zmienną zależną albo funkcją zmiennej x, jeżeli przyjmuje ona określone wartości dla każdej wartości zmiennej x, w pewnym przedziale zmienności. Zmienna x nazywana jest zmienną niezależną albo argumentem funkcji y. Związek między zmienną zależną y a zmienną niezależną x zapisujemy symbolicznie w postaci:

$$y = f(x)$$

Pochodna funkcji

Niech dwóm wartościom x_1 i x_2 zmiennej niezależnej odpowiadają dwie wartości funkcji y_1 oraz y_2. Oznaczmy:

$$\Delta x = x_2 - x_1$$

$$\Delta y = y_2 - y_1$$

Przez pochodną funkcji y w punkcie x będziemy rozumieli granicę, do której dąży stosunek $\frac{\Delta y}{\Delta x}$, gdy Δx dąży do zera, co zapiszemy symbolicznie

$$\dot{y} = \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

Szereg Maclaurina

Nieskończony szereg potęgowy o n-tym wyrazie równym:

$$a_n = \frac{f^{(n)}(0)}{n!} x^n \tag{1.1}$$

gdzie $f^{(n)}(0)$ – wartość n-tej pochodnej pewnej funkcji $f(x)$ dla $x = 0$.

Można wykazać, że jeśli funkcja $f(x)$ jest różniczkowalna nieskończenie wiele razy w pewnym otoczeniu $x = 0$ oraz:

$$\lim_{n\to\infty} \frac{f^{(n)}(c)}{n!} x^n = 0$$

gdzie c zawarte jest pomiędzy 0 a x, to:

$$f(x) = f(0) + \sum_{n=1}^{\infty} a_n \tag{1.2}$$

Twierdzenie 1. *Jeżeli istnieje n-ta pochodna funkcji $f(x)$ w otoczeniu $x = 0$, istnieje dokładnie jeden wielomian $V(x)$, stopnia n lub niższego spełniający warunek:*

$$V(0) = f(0),\ V^{'}(0) = f^{'}(0),\ V^{''}(0) = f^{''}(0),\ \ldots,\ V^{(n)}(0) = f^{(n)}(0)$$

Dowód 1. *Niech $V(x) = a + bx + cx^2 + \ldots + lx^n$ Wtedy:*

$$\begin{array}{lcccccccc} V^{'}(x) & = & b & + & 2cx & + & \ldots & + & nlx^{n-1} \\ V^{''}(x) & = & & & 2c & + & 6dx & + & \ldots + n(n-1)lx^{n-2} \\ \ldots & & & & & & & & \\ V^{(n)} & = & n!l & & & & & & \end{array}$$

Wtedy:

$$\begin{array}{lcl} V(0) & = & a \\ V^{'}(0) & = & b \\ V^{''}(0) & = & 2c \\ \ldots & & \\ V^{(n)}(0) & = & n!l \end{array}$$

Wymagamy aby spełniony był warunek

$$V(0) = f(0),\ V^{'}(0) = f^{'}(0),\ V^{''}(0) = f^{''}(0),\ \ldots,\ V^{(n)}(0) = f^{(n)}(0)$$

Czyli:

$$\begin{array}{lcl} a & = & f(0) \\ b & = & f^{'}(0) \\ 2c & = & f^{''}(0) \\ \ldots & & \\ n!l & = & f^{(n)}(0) \end{array}$$

Jedyny wielomian stopnia n lub niższego spełniający te warunki ma postać:

$$f(0) + \frac{xf^{'}(0)}{1!} + \frac{x^2 f^{''}(0)}{2!} + \frac{x^3 f^{'''}(0)}{3!} + \ldots + \frac{x^n f^{(n)}(0)}{n!}$$

Szereg Maclaurina jest szczególnym przypadkiem szeregu Taylora:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \tag{1.3}$$

Ponieważ liczymy sumę szeregu nieskończonego musimy w pewnym miejscu dokonać obcięcia, popełnimy więc błąd obliczeniowy, jeżeli dokonamy sumowania k elementów szeregu błąd możemy oszacować w następujący sposób:

$$f(x) = \left(\sum_{n=0}^{k} \frac{f^{(n)}(x_0}{n!}(x - x_0)^n \right) + \frac{f^{(k+1)}(\varepsilon)}{(k+1)!}(x - x_0)^{k+1} \tag{1.4}$$

prawą część wzoru (1.4) nazywamy resztą Lagrange'a i oznaczamy R_n

$$R_n = \frac{f^{(k+1)}(\varepsilon)}{(k+1)!}(x - x_0)^{k+1}$$

Przybliżoną wartość funkcji można znaleźć licząc kilka k pierwszych wartości. Błąd jest wtedy nie większy niż:

$$\max_{\varepsilon \in [x_0, x]} \left((x - x_0) \left| \frac{f^{(k+1)}(x_0)(\varepsilon - x_0)^{k+1}}{(k+1)!} \right| \right) \tag{1.5}$$

1.2. Zadania tablicowe

1. policz $\sin(x)$ korzystając z rozwinięcia w szereg Maclaurina
2. policz $\cos(x)$ korzystając z rozwinięcia w szereg Maclaurina
3. policz e^x korzystając z rozwinięcia w szereg Maclaurina
4. policz $\sqrt{10}$ korzystając z rozwinięcia w szereg Maclaurina

$f(x)$	$f^{'}(x)$
$\sin(x)$	$\cos(x)$
$\cos(x)$	$-\sin(x)$

1.2.1. Zadanie 1 $\sin(x)$

$$\sin(x) = \sin(0) + x\cos(0) - \frac{x^2 \sin(0)}{2!} - \frac{x^3 \cos(0)}{3!} + \frac{x^4 \sin(0)}{4!} + \frac{x^5 \cos(0)}{5!} - \dots$$

czyli po uproszczeniu (korzystamy z faktu $\sin(0) = 0$ oraz $\cos(0) = 1$):

$$\sin(x) \approx x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} \tag{1.6}$$

Błąd tego przybliżenia (uznając, że 8-ma pochodna wynosi 0 możemy oszacować:

$$\max_{\varepsilon \in [0, \pi]} \left((\pi - 0) \left| \frac{\varepsilon^9 \cos(\varepsilon)}{9!} \right| \right) = \pi \frac{\pi^9}{9!} = \frac{\pi^{10}}{362880} \approx 0.2580$$

Oczywiście uwzględniając większą liczbę pochodnych otrzymamy dokładniejsze przybliżenie.

1.2.2. Zadanie 2 $\cos(x)$

$$\cos(x) = \cos(0) - x\sin(0) - \frac{x^2\cos(0)}{2!} + \frac{x^3\sin(0)}{3!} + \frac{x^4\cos(0)}{4!} - \frac{x^5\sin(0)}{5!} - \frac{x^6\cos(0)}{6!} + \dots \tag{1.7}$$

czyli po uproszczeniu (korzystamy z faktu $\sin(0) = 0$ oraz $\cos(0) = 1$):

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} \tag{1.8}$$

Błąd tego przybliżenia (uznając, że 9-ma pochodna wynosi 0 możemy oszacować:

$$\max_{\varepsilon\in[0,\pi]} \left((\pi - 0) \left| \frac{\varepsilon^{10}\cos(\varepsilon)}{10!} \right| \right) = \pi\frac{\pi^{10}}{10!} = \frac{\pi^{11}}{3628800} \approx 0.0810$$

Oczywiście uwzględniając większą liczbę pochodnych otrzymamy dokładniejsze przybliżenie.

1.2.3. Zadanie 3 e^x

$$e^x = 1 + e^0 x + \frac{e^0 x^2}{2!} + \frac{e^0 x^3}{3!} + \dots \tag{1.9}$$

czyli

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} \tag{1.10}$$

1.2.4. Zadanie 4. $\sqrt{10}$

Pare faktów:

$$\left(x^n\right)' = nx^{n-1} \tag{1.11}$$

$$\left(\sqrt{x}\right)' = \frac{1}{2\sqrt{x}} \tag{1.12}$$

Wzór (1.12) możemy otrzymać w następujący sposób:

$$\sqrt{x} = x^{\frac{1}{2}} \tag{1.13}$$

czyli:

$$\left(x^{\frac{1}{2}}\right)' = \frac{1}{2}x^{1-\frac{1}{2}} = \frac{1}{2}x^{-\frac{1}{2}} = \frac{1}{2\sqrt{x}} \tag{1.14}$$

Znana jest wartość $\sqrt{9} = 3$ Możemy więc skorzystać z rozwinięcia w szereg i zapisać:

$$\sqrt{10} = \sum_{n=0}^{\infty} \frac{f^{(n)}(9)(10-9)^n}{n!} = \sum_{n=0}^{\infty} \frac{f^{(n)}(9)}{n!} \approx \sum_{n=0}^{k} \frac{f^{(n)}(9)}{n!}$$

czyli licząc kolejne pochodne pierwiastka:

$$\left(\sqrt{x}\right)' = \frac{1}{2\sqrt{x}}$$

$$\left(\sqrt{x}\right)'' = \left(\frac{1}{2\sqrt{x}}\right)' = \left(\frac{1}{2}x^{-\frac{1}{2}}\right)' = -\frac{1}{4}x^{-\frac{1}{2}-1} = -\frac{1}{4}x^{-\frac{3}{2}} = -\frac{1}{4(\sqrt{x})^3}$$

$$\left(\sqrt{x}\right)''' = \left(-\frac{1}{4\sqrt{x}^3}\right)' = \frac{3}{8\sqrt{x}^5}$$

Czyli rozwinięcie możemy zapisać:

$$\sqrt{10} \approx \sqrt{9} + \frac{1}{1! \cdot 2\sqrt{9}} - \frac{1}{2! \cdot 4(\sqrt{9})^3} + \frac{3}{3! \cdot 8(\sqrt{9})^5} = 3 + \frac{1}{6} - \frac{1}{216} + \frac{3}{3988} \approx 3.16229$$

błąd jest nie większy niż:

$$\max_{\varepsilon \in [9,10]} \left((10-9) \left| \frac{15}{4! \cdot 16(\sqrt{9})^7} \right| \right) = \frac{15}{839808} \approx 0.000017861$$

1.3. Zadania programistyczne

1. Napisać program liczący funkcję $\sin(x)$, $\cos(x)$, e^x z parametrem określającym długość rozwinięcia szeregu.
2. Porównać wyniki otrzymane z obliczenia funkcji poprzez rozwinięcie w szereg z wynikami funkcji z biblioteki matematycznej c++
3. Zapisać dane z poprzedniego punktu do pliku i sporządzić wykresy funkcji oraz oszacowania błędu.

1.4. Środowisko programistyczne

Programy w trakcie ćwiczeń pisane będą w języku C++ z wykorzystaniem dostarczonych przez prowadzących szkieletów, szkieletów realizujących podstawowe zadania związane z graficzną prezentacją danych. Zadaniem studenta będzie dobranie odpowiednich równań zgodnie z prezentowanym zagadnieniem, rozwiązanie tych równań stosując metody numeryczne, oraz uzupełnienie dostarczonego kodu odpowiednimi instrukcjami. Instrukcjami. Istnieje możliwość wyboru innego języka programowania jednak w tym przypadku Student musi napisać całość we własnym zakresie.

Do celów graficznej prezentacji stosowana będzie biblioteka OpenGL, w zasadzie nie jest wymagana od studentów znajomość tej biblioteki (podstawowe zagadnienia związane z OpenGL zostaną przedstawione w trakcie zajęć)

Studenci PJWSTK mają dostęp do kompilatora C++ firmy Microsoft Visual Studio 2005, można również skorzystać z darmowych środowisk programistycznych:

- MinGW Developer Studio 2.05
- Dev-C++

Dostarczone kody wykorzystują dodatkową bibliotekę o nazwie GLUT The OpenGL Utility Toolkit. Do poprawnej kompilacji potrzebny jest plik nagłówkowy glut.h, który należy umieścić najlepiej w folderze include/GL wybranego kompilatora, oraz w przypadku kompilatorów opartych na gcc biblioteka glutlib32.a do umieszczenia w folderze lib kompilatora, natomiast w przypadku środowiska Visual Studio biblioteka glut32.lib. Skompilowana wersja dla systemu windows jest do pobrania z następującego adresu: http://www.xmission.com/ nate/glut/glut-3.7.6-bin.zip

W przypadku środowiska windows do poprawnego wykonania programu potrzebna jest biblioteka dynamiczna glut32.dll, która powinna być umieszczona w katalogu widocznym w zmiennej środowiskowej PATH lub bezpośrednio w miejscu gdzie znajduje się skompilowany plik exe. Biblioteka GLUT jest do pobrania w postaci kodów źródłowych http://www.xmission.com/ nate/glut/glut-3.7.6-src.zip do samodzielnej kompilacji w przypadku innego systemu operacyjnego.

Do poprawnej konsolidacji programu niezbędne jest dołączenie następujących bibliotek:

- glut32,
- opengl32,
- glu32.

1.4.1. Korzystanie z biblioteki

Proste środowisko, które umożliwi graficzną prezentacje wyników, animację zostanie dostarczone w celu usprawnienia pisania kodu. Dla osoby piszącej program ważna jest klasa **SceneObject** z której powinny być wyprowadzane wszystkie klasy na podstawie, których budowane będą obiekty wizualne:

```
class SceneObject{
public:
   SceneObject();
   virtual ~SceneObject() {};
   virtual void draw()=0;
   virtual void drawShadow() shadow=1; draw(); shadow=0;
   virtual void doStep() {};
   virtual void drawControl(int id, int w, int h) {}
   virtual bool getCastsShadows(void) const { return false; }
protected:
   int shadow;
};
```

Klasa pochodna musi przedefiniować wirtualną metodę **draw**() oraz **doStep**(). Metoda **draw**() wywoływana jest przez środowisko, kiedy zachodzi potrzeba namalowania obiektu, natomiast **doStep**() kiedy dokonywane są obliczenia dla kolejnego kroku czasowego.

Oczywiście w celu umieszczenia obiektów statycznych, które mają mieć tylko wizualną reprezentację metoda **doStep**() może być pusta.

Program wykorzystujący bibliotekę wygląda następująco:

```
#include "soleng.h"

int main(int argc, char *argv[]){
    CWorld *world;
    world = CWorld::getWorldInstance(argc, argv);
    world->sceneManager->addSceneObject(new Pendulum());
    world->mainLoop();
    return 0;
}
```

Dodatkową przydaną klasą jest **DisplayControl** realizując okienko powiązane z obiektem sceny w momencie kiedy zachodzi konieczność namalowania okienka wysyła ono komunikat do swojego właściciela **drawControl**(int id, int w, int h).