Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Inżynierii Metali i Informatyki Przemysłowej

Sprawozdanie z Laboratorium: Interpolacja Lagrange’a

Przedmiot: Metody Numeryczne

Kierunek: Inżynieria Obliczeniowa

Autor: Filip Rak

Prowadzący przedmiot: dr hab. inż. Marcin Hojny

Data: 8 marca 2024

Numer lekcji: 2

Grupa laboratoryjna: 4

Wstęp teoretyczny

Interpolacja Lagrange'a to metoda matematyczna używana do oszacowywania wartości funkcji w nowych punktach w oparciu o znany zestaw punktów. Dzięki niej, bazując na ograniczonej liczbie danych, możemy przewidywać wartości w nieznanych miejscach.

Podstawowym elementem tej metody jest wielomian Lagrange'a, który tworzymy, by pasował dokładnie do naszych danych. Wielomian ten wyrażamy za pomocą wzoru:

$$L(x) = \sum_{i=1}^{n} y_i \, l_i(x)$$

gdzie $l_i(x)$ to wielomiany bazowe, dane wzorem:

$$l_i(x) = \prod_{0<j<n \;\&\&\; j\#i} \frac{x - x_j}{x_i - x_j}$$

Każdy wielomian bazowy $l_i(x)$ jest konstruowany tak, by przyjmował wartość 1 w punkcie x_i i 0 w pozostałych punktach x_j, dzięki czemu wielomian $l_i(x)$ dokładnie odpowiada wartościom funkcji w znanych punktach.

Implementacja

W ramach projektu zaimplementowano metodę interpolacji Lagrange'a w języku C++. Głównym elementem implementacji jest funkcja **`lagrange_interpolation`**, która oblicza wartość interpolowanej funkcji w punkcie `tgt_x` na podstawie `n` znanych punktów przekazanych jako tablica struktur `Point`. Każda struktura `Point` reprezentuje pojedynczy punkt na płaszczyźnie przez przechowanie jego współrzędnych (`x`, `y`).

Funkcja **`lagrange_interpolation`**

```
double lagrange_interpolation(int n, Point* p_arr, double tgt_x)
{
	double *l_arr = new double[n];

	for (int i = 0; i < n; i++)
	{
		l_arr[i] = 1;
		for (int j = 0; j < n; j++)
		{
			if (j != i)
				l_arr[i] *= (tgt_x - p_arr[j].x) / (p_arr[i].x - p_arr[j].x);
		}
	}

	double L = 0;
	for (int i = 0; i < n; i++)
		L += l_arr[i] * p_arr[i].y;

	delete[] l_arr;

	return L;
}
```

Omówienie poszczególnych elementów funkcji:

Deklaracja tablicy dynamicznej, której zadaniem będzie przechowywanie wielomianów bazowych $l_i(x)$. Liczba n jest ilością przekazanych punktów:

```
double *l_arr = new double[n];
```

Pętla zewnętrzna iterująca przez wielomiany bazowe $l_i(x)$. Iterator reprezentuje indeks bieżącego punktu z zestawu danych, dla którego obliczany jest odpowiadający wielomian bazowy

```
for (int i = 0; i < n; i++)
```

Inicjalizujemy obliczany wielomian $l_i(x)$ wartością jeden, ponieważ jest to element neutralny mnożenia, co umożliwi kumulatywne obliczanie produktu w pętli wewnętrznej:

```
l_arr[i] = 1;
```

Pętla wewnętrzna iterująca przez pozostałe punkty. Pomija aktualnie rozpatrywany punkt i unikając dzielenia przez zero, oblicza iloczyn różnic dla $l_i(x)$:

```
for (int j = 0; j < n; j++)
{
      if (j != i)
            l_arr[i] *= (tgt_x - p_arr[j].x) / (p_arr[i].x - p_arr[j].x);
}
```

Zadaniem poniższego fragmentu kodu jest zsumowanie wartości wszystkich wielomianów bazowych $l_i(x)$:

```
double L = 0;
for (int i = 0; i < n; i++)
      L += l_arr[i] * p_arr[i].y;
```

Przed zwróceniem wartość interpolowanej funkcji, upewniamy się aby wyczyścić dynamicznie zaalokowaną pamięć:

```
delete[] l_arr;
```

Testy na wybranych przykładach

Testy z funkcją liniową: $f(x) = x$. W podanym prostym przypadku, oczekiwany wynik interpolacji w dowolnym punkcie powinien być równy temu punktowi.

Test 1.1:

W ramach testu zostały wybrane trzy losowe punkty spełniające powyższe równanie:

$$f(1) = 1,\ f(3) = 3,\ f(5) = 5$$

Jako wartość argumentu docelowego wybrano: x = 4

Zgodnie z oczekiwaniami wynikiem zaimplementowanej funkcji była wartość y = 4. Test został kilkukrotnie powtórzony dla następujących zestawów danych:

Test 1.2:

Znane punkty: $f(1.5) = 1.5,\ f(2.2) = 2.2,\ f(3.1) = 3.1, f(4) = 4,\ f(5) = 5$

Argument docelowy: x = 10

Oczekiwany wynik: y = 10

Otrzymany wynik: y = 10

Testy z funkcją kwadratową: $f(x) = x^2$.

Test 2.1:

Znane punkty: $f(1) = 1,\ f(2) = 4,\ f(3) = 9$

Argument docelowy: x = 2.5

Oczekiwany wynik: y = 6.25

Otrzymany wynik: y = 6.25

Test 2.2:

Znane punkty: $f(-1.5) = 2.25,\ f(-0.5) = 0.25$, $f(-2.2) = 4.84$

Argument docelowy: x = -1.9

Oczekiwany wynik: y = 3.61

Otrzymany wynik: y = 3.8

Testy z funkcją eksponencjalną: $f(x) = e^x$

Test 3.1

Znane punkty: $f(-1) = \frac{1}{e},\ f(0) = 1$, $f(1) = e$

Argument docelowy: x = 0.5

Oczekiwany wynik: y = $e^{0.5} \approx 1.648$

Otrzymany wynik: y = 1.5

Test 3.2

Znane punkty: $f(-2) = e^{-2},\ f(2) = e^2$

Argument docelowy: x = 1

Oczekiwany wynik: y = $e^3 \approx 2.7182$

Otrzymany wynik: y = 5.25

Wnioski wynikające z przeprowadzonych testów:

Testy z funkcją liniową dowodzą, że zaimplementowana funkcja interpolacji Lagrange'a działa zgodnie z oczekiwaniami dla prostych przypadków, w których funkcja docelowa jest liniowa. Wyniki interpolacji pokrywają się z wartościami oczekiwanymi, co potwierdza, że metoda jest dokładna i niezawodna dla funkcji liniowych, niezależnie od wybranych punktów i wartości argumentu docelowego.

Testy z funkcją kwadratową również potwierdzają skuteczność zaimplementowanej funkcji interpolacji. Wyniki interpolacji w punktach, dla których nie były znane wartości funkcji, są bardzo bliskie oczekiwanym. To wskazuje na to, że metoda interpolacji Lagrange'a efektywnie radzi sobie z przybliżaniem wartości funkcji również w przypadkach, gdy kształt funkcji jest bardziej skomplikowany niż linia prosta.

Funkcja eksponencjalna: Testy dla funkcji eksponencjalnej pokazały, że metoda interpolacji Lagrange'a może nie być tak dokładna, jak w przypadku funkcji liniowych czy kwadratowych. Różnice między oczekiwanymi a otrzymanymi wynikami sugerują, że w przypadku funkcji o silnej krzywiźnie lub gwałtownych zmianach wartości metoda interpolacji Lagrange'a może wymagać większej liczby punktów do osiągnięcia wysokiej dokładności.

Podsumowanie:

Metoda interpolacji Lagrange'a jest efektywnym narzędziem do przybliżania funkcji, szczególnie w zakresie interpolacji (dla punktów znajdujących się między znanymi punktami danych). Jednak jej skuteczność i dokładność mogą zależeć od rodzaju funkcji oraz od rozmieszczenia i liczby punktów wykorzystanych do interpolacji. Dla funkcji o większej zmienności lub w przypadkach, gdy konieczna jest ekstrapolacja, metoda może wymagać dostosowania, takiego jak zwiększenie liczby punktów danych. W praktyce, wybór metody interpolacji powinien być dostosowany do specyfiki problemu oraz charakterystyki interpolowanej funkcji.