MOwNiT Lab1 - sprawozdanie

Autor: Michał Flak

Zadanie 1

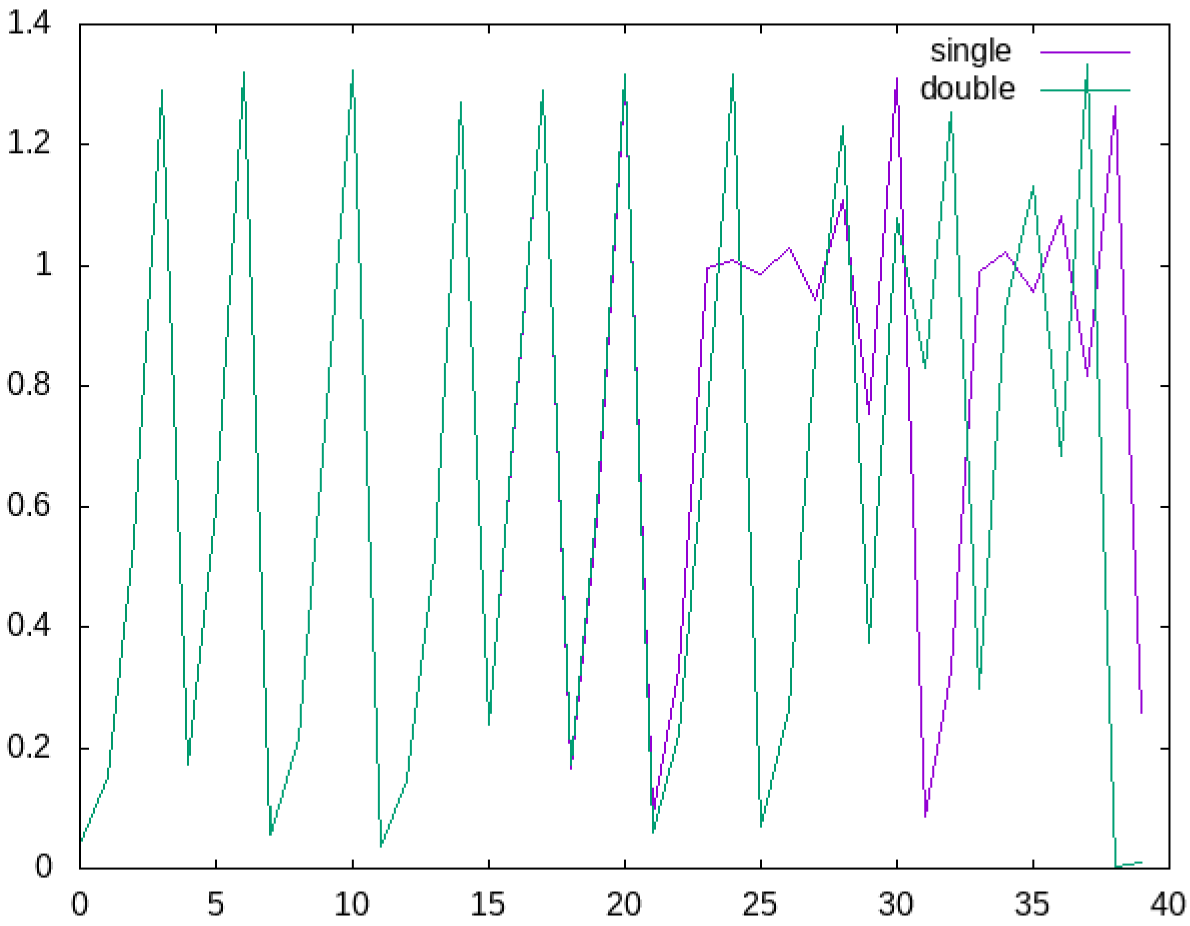
Napisać program liczący kolejne wyrazy ciągu:
*x{n+1}= x{n} + 3.0 * x{n} * (1 - x{n})*
startując z punktu x{0} = 0.01. Wykonać to zadanie dla różnych reprezentacji liczb (float, double). Dlaczego wyniki się rozbiegają?

Napisano funkcję generującą 40 pierwszych wyrazów ciągu:

```
static const int MAX_N = 40;

template<typename T>
void printSeries()
{
  T n = 0, xn = 0.01, xnplus1;
  for(n; n<MAX_N; n++)
  {
    xnplus1 = xn + (T)3.0 * xn * ((T)1.0 - xn);
    std::cout << n << ", " << xnplus1 << std::endl;
    xn=xnplus1;
  }
}
```

Uruchomiono ją podając za `T` typy `float` i `double`, zapisując wyniki do plików tekstowych i rysując na wykresie za pomocą gnuplot:



Zauważamy, że wartości dla X większego od około 22 zaczynają się rozbiegać. Wskazuje to na wyczerpanie precyzji zmiennej pojedynczej prezycji.

W przypadku obliczeń takich jak liczenie kolejnych wyrazów ciągu, błąd wynikający z tego faktu kumuluje się w wyniku korzystania z poprzednich, również obarczonych błędem obliczeń.

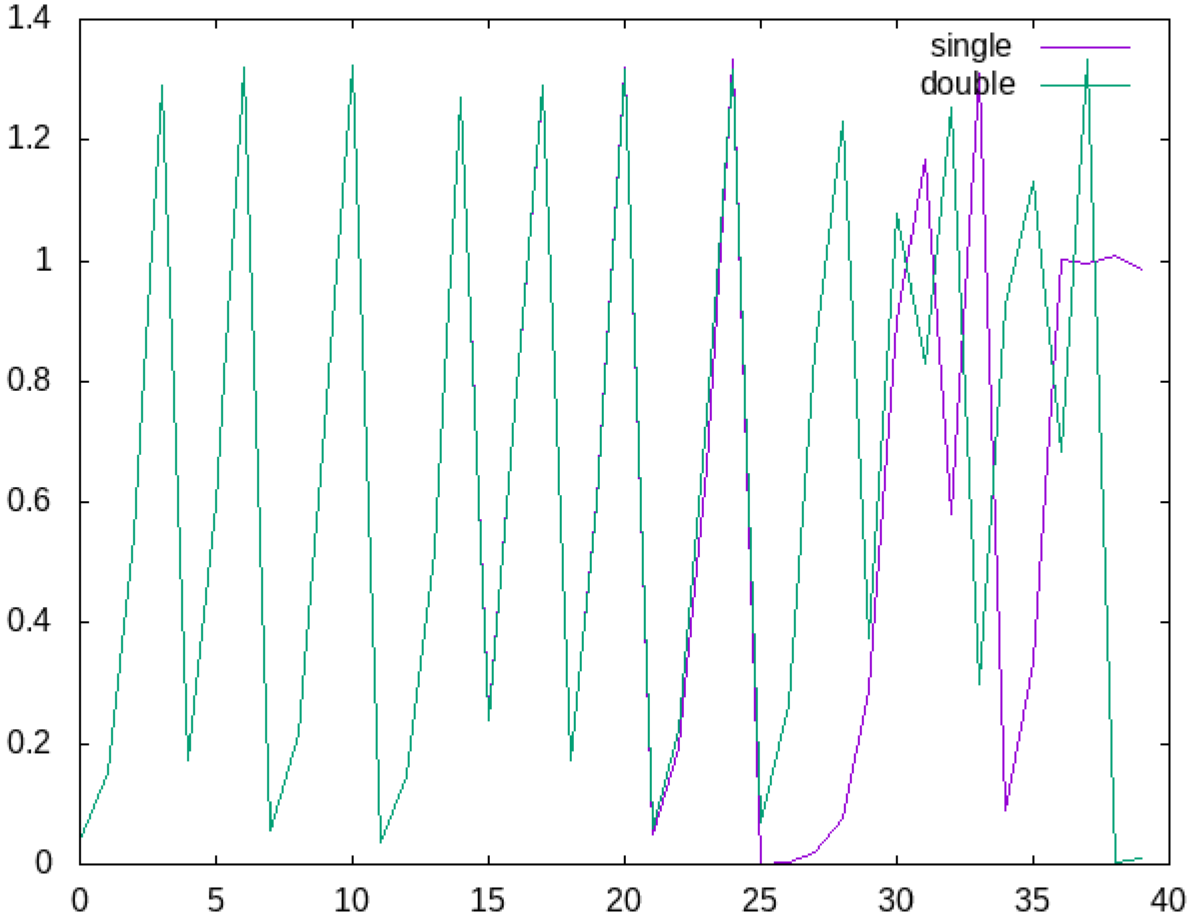
Zadanie 2

*Napisać program liczący ciąg z wcześniejszego zadania, ale wg wzoru x{n+1} = 4.0 * x{n} - 3.0 * x{n} * x{n} porównać z wynikami z wcześniejszego zadania.*

Zmodyfikowano funkcję z poprzedniego zadania i wykonano analogiczne kroki:

```
template<typename T>
void printSeries()
{
  T n = 0, xn = 0.01, xnplus1;
  for(n; n<MAX_N; n++)
  {
    xnplus1 = xn * (T)4.0 - (T)3.0 * xn * xn;
    std::cout << n << ", " << xnplus1 << std::endl;
    xn=xnplus1;
  }
}
```

Uzyskany plik `zad2.png`:



Pierwsza rozbieżność jest widoczna jak poprzednio w okolicach X=22, natomiast wyniki znacznie rozbiegają się od X=25 wzwyż.

Zadanie 3

Znaleźć "maszynowe epsilon", czyli najmniejszą liczbę a, taką, że a+1>1.

Epsilon maszynowy typu `T` możemy odczytać z `std::numeric_limits<T>::epsilon()`. Użyjemy przeszukiwania binarnego liczb w zakresie [0.0..1.0] tak długo aż epsilon przestanie się zmieniać. Kod:

```
template<typename T>
void findEpsilon()
{
  T reflimit = std::numeric_limits<T>::epsilon();
  std::cout << std::setprecision(16) << "Reference epsilon is: " <<
reflimit << std::endl;

  T epsilon = (T)0.0;
  T cur = (T)1.0;

  while ((T)1.0 + cur > (T)1.0)
  {
    cur /= (T)2.0;
  }
```

```
    epsilon = cur*(T)2.0;
    std::cout << std::setprecision(16) << "Calculated epsilon is: " <<
  epsilon << std::endl;
  }
```

Uruchamiając dla single-precision float:

```
Reference epsilon is: 1.192092895507812e-07
Calculated epsilon is: 1.192092895507812e-07
```

Uruchamiając dla double-precision float:

```
Reference epsilon is: 2.220446049250313e-16
Calculated epsilon is: 2.220446049250313e-16
```

Widzimy zatem, że rząd wielkości epsilonu dla SP jest mniej więcej dwukrotnie mniejszy niż rząd wielkości epsilonu DP.