Precyzja obliczeń w SAS

Zespół Wsparcia Technicznego, SAS Institute Polska marzec 2018

Wstęp

Dokument ten powstał, aby wyjaśnić wątpliwości związane z precyzją obliczeń w SAS zgłaszane przez naszych klientów do wsparcia technicznego. SAS na systemach Windows, Unix i Linux przechowuje liczby w formacie zmiennoprzecinkowym o podwójnej precyzji (binary64), określonym w standardzie IEEE 754. Mechanizmy opisane w tym dokumencie w większości nie są specyficzne dla systemu SAS, ale związane ze sposobem w jakim dane są przechowywane i przetwarzane na komputerach.

Pierwsza część dokumentu dość dokładnie opisuje sposób reprezentacji liczb zmiennoprzecinkowych. Użytkowników nie zainteresowanych takim poziomem szczegółowości zachęcamy do zapoznania się co najmniej z rozdziałami Źródła problemów z precyzją obliczeń i Jak sobie z tym radzić.

Niniejszy dokument i inne informacje związane z tematem są dostępne na naszej stronie internetowej pod adresem http://www.sas.com/poland/services/. W przypadku dodatkowych pytań lub wątpliwości prosimy o kontakt z naszym zespołem. Jesteśmy do Państwa dyspozycji od poniedziałku do piątku, od 8.30 do 16.30, telefonicznie – **(22) 560 46 66**, bądź elektronicznie – support@spl.sas.com.

Prosimy o uwagi i spostrzeżenia, które pomogą nam w przygotowaniu kolejnych wersji tego dokumentu.

Zawsze służąc Państwu pomocą,
Zespół Wsparcia Technicznego
Centrum Technologii SAS Polska

Spis treści

System binarny 2
Reprezentacja liczb w SAS 2
Format binary64 2
Źródła problemów z precyzją obliczeń 4
  Niedokładność reprezentacji 4
  Arytmetyka liczb zmiennoprzecinkowych 4
  Konwersja do formatu dziesiętnego. 5
Jak sobie z tym radzić 5
Polecane materiały 6

System binarny

Wszyscy znamy system dziesiętny, w którym kolejne pozycje oznaczają kolejne potęgi liczby 10:

105 to 1*100 + 0*10 + 5.

W przypadku ułamków dochodzą jeszcze ujemne potęgi 10:

105,25 = 125 to 1*100 + 0*10 + 5 + 2* 1/10 + 5* 1/100

Analogicznie reprezentowane są liczby w systemie binarnym (dwójkowym). Kolejne pozycje oznaczają potęgi liczby 2 (1,2,4,8,…). Liczby zapisane w systemie binarnym i działania na nich będziemy oznaczać $()_2$ dla odróżnienia ich od liczb w systemie dziesiętnym

$(1011)_2$ to 1*8+0*4+1*2+1*1 czyli 11 w systemie dziesiętnym

$(1011.101)_2$ to 1*8+0*4+1*2+*1*1+1*1/2 + 0*1/4 + 1*1/8 czyli 11,625 (11 i 5/8)

Reprezentacja liczb w SAS

SAS stosuje jeden typ numeryczny. Wszystkie liczby przechowywane są w formacie zmiennoprzecinkowym. Domyślną i maksymalną długością liczb jest 8 bajtów czyli 64 bity. Dokładny sposób reprezentacji liczby zależy od systemu operacyjnego, jednak systemy Windows, Unix i Linux zgodnie stosują standard IEEE 754 (binary64). Czyli SAS na systemach Windows, Unix i Linux używa formatu binary64 opisywanego przez standard IEEE 754 (inaczej: liczb zmiennoprzecinkowych o podwójnej precyzji, double-precision binary floating-point).

Format binary64

Liczby w formacie binary64 przechowywane są na 8 bajtach czyli 64 bitach. Są to liczby zmiennoprzecinkowe, to znaczy wartość liczby obliczana jest (poza wyjątkami opisanymi w dalszej części dokumentu) ze wzoru:

$x=S*M*2^E$, gdzie

S to -1 lub +1, w zależności od znaku liczby

M – mantysa

E – wykładnik

Znak, mantysa i wykładnik reprezentowane są oczywiście binarnie.

Pojedyncze bity można podejrzeć w SAS używając formatu binary64. . Zobaczmy jak reprezentowana jest liczba 1,5 :

```
data _null_;
x=1.5;
put x binary64.;
run;
```

Wynik działania kodu to:

0011111111111000000000000000000000000000000000000000000000000000

Pierwszy bit oznacza znak liczby (0 oznacza plus, 1 oznacza minus), kolejnych 11 bitów reprezentuje wykładnik a pozostałe 52 bity mantysę.

0 01111111111 1000000000000000000000000000000000000000000000000000

Wykładnik jest kodowany z nadmiarem $(1111111111)_2$ (dziesiętne 1023). Czyli trzeba od binarnej liczby $(01111111111)_2$ odjąć binarne $(1111111111)_2$, aby otrzymać rzeczywisty wykładnik. W tym przypadku rzeczywisty wykładnik to po prostu 0.

Bity reprezentujące mantysę to binarny ułamek, do którego dodać trzeba jedynkę, aby otrzymać rzeczywistą mantysę. W naszym tym przypadku rzeczywista mantysa to

$(1)_2 + (0,1000000000000000000000000000000000000000000000000000)_2$ czyli $(1,1)_2$

$(1,1* 10^0 = 1,1 * 1 = 1,1)_2$ czyli w zapisie dziesiętnym 1,5

(10^0 w powyższej linii to zapis binarny czyli 2^0 w zapisie dziesiętnym)

Dziesiętna liczba 3,5 reprezentowana jest następująco:

0 10000000000 1100000000000000000000000000000000000000000000000000

$(1,11 * 10^{10000000000 - 1111111111} = 1,11 * 10^1 = 11,1)_2$ czyli dziesiętnie 3,5

(10^1 w powyższej linii to zapis binarny czyli 2^1 w zapisie dziesiętnym)

Dziesiętna liczba 4 reprezentowana jest następująco:

0 10000000001 0000000000000000000000000000000000000000000000000000

$(1,0 * 10^{10000000001-1111111111} = 1 * 10^{10} = 100)_2$ czyli dziesiętnie 4

(10^{10} w powyższej linii to zapis binarny czyli 2^2 w zapisie dziesiętnym)

Jeśli przez „s" oznaczymy bit reprezentujący znak, przez „e" bity reprezentujący wykładnik a bity reprezentujące mantysę przez m_1, m_2,…, m_{52} to ogólny wzór możemy zapisać następująco:

$x=S*M*2^E= (-1)^s * (1 + \sum_{i=1}^{52} \frac{m_i}{2^i}) * 2^{e-1023}$

Wyjątki:

- Liczba zero reprezentowana jest gdy bity wykładnika i mantysy ustawione są na same zera

- Gdy bity wykładnika ustawione są na zero a co najmniej jeden bit mantysy jest jedynką to wartość liczby wyliczamy ze wzoru
  x=S*M*2^E= $(-1)^s * (0 + \sum_{i=1}^{52} \frac{m_i}{2^i}) * 2^{-1022}$
- Wykładniki z wszystkimi bitami ustawionymi na jeden służą do reprezentacji nieskończoności (gdy bity mantysy są zerowe) i różnych wartości specjalnych.

Źródła problemów z precyzją obliczeń

Niedokładność reprezentacji

Pierwszym źródłem jest niedokładność binarnej reprezentacji ułamków. Znanym faktem jest, że nie każdy ułamek można przedstawić jako ułamek dziesiętny ze skończonym rozwinięciem. Na przykład 1/3 to 0,333333333333…., zapisywane jako 0,(3). Skończone reprezentacje dziesiętne mają tylko te ułamki, których jedynymi dzielnikami mianownika są iloczyny liczb 2, 5 i ich potęg.

Analogiczna sytuacja występuje dla liczb w systemie dwójkowym, tylko że w tym przypadku skończone reprezentacje dwójkowe mają tylko ułamki, których jedynymi dzielnikami mianownika są potęgi liczby 2. Czyli dokładną reprezentację na komputerze mają tylko ułamki 1/2, 1/4, 1/8, 1/16 …. 1/2^n i ich sumy. Wszystkie pozostałe ułamki mogą być zapisane tylko w pewnym skończonym przybliżeniu, którego dokładność zależy od liczby cyfr (bitów) w których zapisujemy tę liczbę.

Nie możemy więc dokładnie zapisać ułamka 1/10, bo jego binarne rozwinięcie jest nieskończone: 0,0(0011). Nie jest więc tak, że przy odpowiednio dużej precyzji obliczeń dokładnie reprezentowane są ułamki do pewnej ilości miejsc po przecinku. Niezależnie od precyzji, ułamek 1/10 nie może być dokładnie reprezentowany przez komputer.

Arytmetyka liczb zmiennoprzecinkowych

Podstawowe prawa dotyczące działań na liczbach to :

Łączność dodawania: (x+y)+z=x+(y+z)

Łączność mnożenia: (x*y)*z=x*(y*z)

Rozdzielność mnożenia względem dodawania: a*(x+y)=a*x+a*y

Okazuje się, że w arytmetyce liczb zmiennoprzecinkowych nie zawsze zachodzą nawet takie podstawowe prawa. Czyli wynik może być zależny od kolejności obliczeń. W szczególności: jeśli sumowane są wszystkie liczby w kolumnie to suma może zależeć od posortowania zbioru. Dokładne wyjaśnienie tego zjawiska można znaleźć na przykład tutaj:
http://wazniak.mimuw.edu.pl/index.php?title=WDP_Reprezentacja_liczb

Dodatkowo sprawę komplikuje fakt, że przetwarzanie może odbywać się w kilku wątkach. Liczba wątków, w których wykonywane są obliczenia może zależeć od wielu czynników, na przykład aktualnego obciążenia systemu. Oglądając wyniki na poziomie pojedynczych bitów można więc odnieść wrażenie, że działanie oprogramowania jest niedeterministycznie.

Konwersja do formatu dziesiętnego.

Pisząc kod i oglądając dane używamy najczęściej formatu dziesiętnego podczas gdy obliczenia wykonywane są na liczbach w formacie binarnym. Dlatego za każdym razem dokonywana jest konwersja między formatem dziesiętnym i binarnym. Często konieczne jest zaokrąglanie wartości. Reguły zaokrągleń używane przez formaty SAS mogą się nieznacznie różnić od reguł stosowanych przy obliczeniach. Formaty powstały wcześniej niż aktualna wersja standardu IEEE 754, która pochodzi z 2008 roku.

Przykład: wykonanie poniższego kodu:

```
data a;
x=282.90*12+17.10*12;
put x 30.20;
if x=3600 then put "równe"; else put "nierówne";
run;
```

da następujący wynik:

```
1    data a;
2    x=282.90*12+17.10*12;
3    put x 30.20;
4    if x=3600 then put "równe"; else put "nierówne";
5    run;

     3600.00000000000000000000
nierówne
```

Jak sobie z tym radzić

Nie ma jednego prostego i uniwersalnego sposobu na radzenie sobie z niedokładnością obliczeń. Zagadnienie to stanowi samodzielną dyscyplinę matematyczną – analizę numeryczną. Na pewno warto przestrzegać następujących zasad:

- Przy wszelkich porównaniach liczb, np. przy wybieraniu wierszy na podstawie wartości numerycznej należy zaokrąglać liczby do precyzji, która w danym przypadku jest dla nas istotna.
- Im mniej działań tym mniej błędów. Każde działanie na liczbach może być obarczone błędem. I oczywiście konsumuje zasoby. Dlatego w zapytaniu SQL lepiej użyć select sum(x)*0.1 niż select sum(x*0.1)
- Należy unikać odejmowania liczb bliskich sobie (zjawisko redukcji cyfr)

Przykład użycia zaokrąglania przy porównywaniu liczb z użyciem funkcji round

```
data a;
x=282.90*12+17.10*12;
put x 30.20;
if round(x,0.01)=3600 then put "równe"; else put "nierówne";
run;
```

```
7    data a;
8    x=282.90*12+17.10*12;
9    put x 30.20;
```

```
10    if round(x,0.01)=3600 then put "równe"; else put "nierówne";
11    run;

      3600.00000000000000000000
równe
```

Przy braku zaokrąglenia do odpowiedniej precyzji na wynik mogą wpłynąć różnice na poziomie pojedynczego bitu, które z kolei mogą zależeć od kolejności obliczeń, posortowania danych a nawet liczby wątków.

Polecane materiały

Dokument ten nie wyczerpuje oczywiście tematu. Polecamy także dokumentację SAS:

Numerical Accuracy in SAS Software

oraz inne artykuły o precyzji obliczeń w SAS'ie:

Dealing with Numeric Representation Error in SAS® Applications
Why .1 + .1 Might Not Equal .2 and Other Pitfalls of Floating-Point Arithmetic
Precision in SAS numbers

Warto też zapoznać się z artykułem firmy Microsoft dotyczącą analogicznych problemów w programie Excel:

http://support.microsoft.com/kb/78113

Dla dociekliwych polecamy tez wykłady na stronie wazniak.mimuw.edu.pl:

Reprezentacja liczb
Arytmetyka zmiennoprzecinkowa