Zürcher Hochschule für Angewandte Wissenschaften

PSP

Programmierung mit FORTRAN

Installation und Austesten der Umgebung

Installieren Sie sich zuerst einen FORTRAN Compiler, z.B. gfortran
gnu-Fortran: http://gcc.gnu.org/wiki/GFortranBinaries
Win32: siehe PSp Seite

Aufgabe

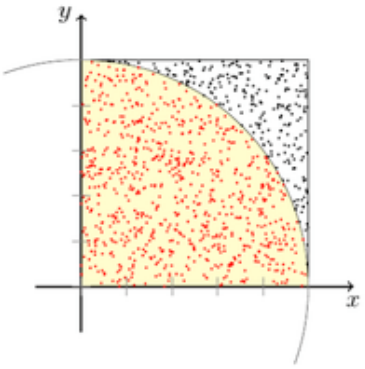
Übersetzen Sie das Hello.f Programm und führen Sie dieses aus.

Hinweise:

- http://gcc.gnu.org/wiki/GFortranGettingStarted
- Aufruf: `gfortran -ffree-form  hello.f -ohello.exe`

1. Berechnung von Pi mittels Montecarlo Verfahren

Die Zahl π lässt sich mittels einer Monte Carlo Simulation bestimmt. Ein einfaches, aber nicht sehr genaues Verfahren funktioniert folgendermassen. Es werden beliebige Punkte innerhalb des Einheitsquadrates zufällig bestimmt. Ist der Abstand zum Ursprung kleiner als 1, dann zählt man ihn zur roten Menge. Die Anzahl der roten Punkte dividiert durch die Gesamtzahl der Versuche ergibt eine Näherung für π/4.



Aufgabe:

Schreiben Sie ein Fortran Programm, das π mittels dem obigen Verfahren bestimmt Verwenden Sie dabei auch die Format Anweisung. Die Anzahl der Versuche soll als Eingabeparameter angegeben werden.

Hinweis:

Die Standardfunktion rand(0) liefert eine Folge von Zufallszahl zwischen [0..1[

2. Erhöhen Sie die Performance mittels Open MP Bibliothek

Mittels der OMP Bibliothek lässt sich die Performance verbessern.

Compilation:

```
gfortran -fopenmp helloomp.f95 -o hello.exe
```

Parallelisieren Sie den Algorithmus entsprechend und stoppen Sie z.B. mit `timeit.exe` die Zeit.

Was stellen Sie fest? Haben Sie eine Erklärung dafür?

3. Neuer Zufallszahlengenerator

Um das Programm zu beschleunigen, kann folgender Zufallszahlengenerator verwendet werden, wobei jedoch der seed (als Variable gespeichert!) unterschiedlich initialisiert sein muss (z.B. mit der ThreadID; siehe HelloOMP). Weiter muss die der Rückgabetyp von ran0 bekannt sein; dafür muss in der Varialblendeklaration noch ran0 angegeben werden.

Hinweis: FORTRAN ist ein ein-Pass Compiler; um den ran0 Funktions-Rückgabetyp bekannt zu machen, muss in der Variablendeklaration des Hauptprogramms noch ran0 als zusätzlicher real Wert deklariert werden.

```
function  ran0(seed)
! Park & Miller Random Generator
integer  seed,ia,im,iq,ir,mask,k
real  ran0,am
parameter  (ia=16807,im=2147483647,am=1./im,
iq=127773,ir=2836,mask=123459876)
seed=ieor(seed,mask)
k=seed/iq
seed=ia*(seed-k*iq)-ir*k
if  (seed.lt.0)  seed=seed+im
ran0=am*seed
seed=ieor(seed,mask)
return
end
```

4. Vergleich mit Bibliotheks rand Funktion (optional)

Obiges Programm stammt übrigens aus Numerical Recipes in FORTRAN 77

FORTRAN stellt eine rand(0) Funktion in der Bibliothek zur Verfügung, aber diese "sträubt" sich i.d.R. gegen die Parallelsierung. Messen Sie die Zeiten mit den rand(0) Funktion und erklären Sie das Verhalten.

Siehe auch: https://www.sciencedirect.com/science/article/pii/S0378475416300829

```
FUNCTION ran0(idum)
INTEGER idum,IA,IM,IQ,IR,MASK
REAL ran0,AM
PARAMETER (IA=16807,IM=2147483647,AM=1./IM,
     IQ=127773,IR=2836,MASK=123459876)
    "Minimal" random number generator of Park and Miller. Returns a uniform random deviate
    between 0.0 and 1.0. Set or reset idum to any integer value (except the unlikely value MASK)
    to initialize the sequence; idum must not be altered between calls for successive deviates
    in a sequence.
INTEGER k
idum=ieor(idum,MASK)              XORing with MASK allows use of zero and other simple
                                      bit patterns for idum.
k=idum/IQ
idum=IA*(idum-k*IQ)-IR*k          Compute idum=mod(IA*idum,IM) without overflows by
if (idum.lt.0) idum=idum+IM           Schrage's method.
ran0=AM*idum                      Convert idum to a floating result.
idum=ieor(idum,MASK)              Unmask before return.
return
END
```

The period of ran0 is $2^{31} - 2 \approx 2.1 \times 10^9$. A peculiarity of generators of the form (7.1.2) is that the value 0 must never be allowed as the initial seed — it perpetuates itself — and it never occurs for any nonzero initial seed. Experience has shown that users always manage to call random number generators with the seed idum=0. That is why ran0 performs its exclusive-or with an arbitrary constant both on entry and exit. If you are the first user in history to be proof against human error, you can remove the two lines with the ieor function.

Park and Miller discuss two other multipliers a that can be used with the same $m = 2^{31} - 1$. These are $a = 48271$ (with $q = 44488$ and $r = 3399$) and $a = 69621$ (with $q = 30845$ and $r = 23902$). These can be substituted in the routine ran0 if desired; they may be slightly superior to Lewis *et al.*'s longer-tested values. No values other than these should be used.

The routine ran0 is a Minimal Standard, satisfactory for the majority of applications, but we do not recommend it as the final word on random number generators.