Программа для вычисления простейшей формы коэффициентов Клебша-Гордана
Clebsch2 Автор: А.П.Сапожников

Real*8 function Clebsch2(k,n) вычисляет коэффициент
C(k,n) = k! * (n-k)! / n!
Программа свободна от типичных при вычислении факториалов “в лоб” случаев переполнения при умножении. Программа является коллективной операцией пакета MPI и прямым потомком операции MPI_AllReduce.

А.П.Сапожников

Программа Clebsch вычисления простейшей формы коэффициентов Клебша-Гордана

Википедия сообщает на эту тему следующее:
“Коэффициенты Клебша-Гордана находят применение при описании взаимодействия квантовомеханических моментов импульса.
Они представляют собой коэффициенты разложения собственных функций суммарного момента импульса по базису собственных функций суммируемых моментов импульса. Коэффициенты Клебша-Гордана применяются при вычислении спин-орбитального взаимодействия, а также в формализме изоспина.
Коэффициенты Клебша-Гордана названы в честь Альфреда Клебша (1833—1872) и Пауля Альберта Гордана (1837—1912).”
От себя добавим, что русскоязычному читателю не следует путать Гордана (Gordan) с его современником Жорданом (Jordan, 1838-1922), известным французским алгебраистом.

В простейшей форме коэффициент Клебша-Гордана выглядит так:

C(k,n) = k! * (n-k)! / n!

Здесь n>0, 0<=k<=n - целые числа. Очевидно, что C(k,n)=C(n-k,n).
Также очевидно, что C(k,n) <= 1 при любых k и n, а учитывая, что 0! = 1, видим, что равенство достигается при k=0 или n.

Тем не менее, программа не может вычислять факториалы “в лоб” – очень быстро случится переполнение при операции умножения. С этой проблемой столкнулся студент из ЮАР, проходивший практику в ЛИТ ОИЯИ в сентябре 2011 г. Поэтому было предложено вычислять логарифм коэффициента, а уже потом потенцировать. Программа выглядит так:

```
      function Clebsch(k,n)          / 1 /
! calc Clebsch-Gordan simplifying coeff = k! * (n-k)! / n!
      c1=0.
      c2=0.
      c3=0.
      do i=1,N
        alog=Log(float(i))       !  We use logarithms to avoid overflow!
        if(i.le.k) c1=c1+alog    !  k!
        if(i.le.n-k) c2=c2+alog  !  (n-k)!
        c3=c3-alog               !  n!
      enddo
      Clebsch=Exp(c1+c2+c3)
      return
      end
```

Помнится, эта идея впервые применялась в 70-е годы прошлого века в одной из программ для ЭВМ CDC-6500. Программа вынуждена вычислять N логарифмов, и при больших N время ее работы становится заметным.
На самом деле помянутому выше студенту из ЮАР предлагалось распараллелить эту программу для использования коллективом из P процессов средствами технологии MPI, им только что освоенной. Но об MPI – чуть ниже.

Есть и другой вариант – разумно чередовать операции умножения и деления. Присмотревшись к выражению коэффициента, видим, что он получается равным произведению первых K членов ряда 1,2,3…… N, деленному на произведение K последних членов этого ряда. Поэтому цикл работы программы можно сократить с N до K и учесть симметрию C(k,n)=C(n-k,n). Отказываясь от вынужденного логарифмирования, имеем простейший цикл:

```
      c=1.                          / 2 /
      do i=1, Min(K,N-K)
        c=c*float(i)/float(N-i+1)
      enddo
      Clebsch=c
```

Эта программа делает всего не более N/2 умножений и N/2 делений.
Измерения счетного времени работы процессора показали, что даже в своем наишудшем случае (K=N/2) она работает на порядок быстрее первой.

Вернемся теперь к технологии MPI. Даже (а может быть – именно?) для такой крошечной программы попытка ее распараллеливания представляет некоторый интерес. Приведем полный текст распараллеленной версии /2/:

```
      Real*8 function Clebsch2(k,n)
! calc Clebsch-Gordan simplifying coef = K! * (n-k)! / n!
      implicit real*8 (a-h,o-z)
       Include 'mpif.h'
      common/idents/idcomm,NProc,myProc
      if (idcomm.eq.0) then  ! MPI Initialisation
        idcomm=MPI_Comm_World
        call MPI_Init(ierr)        ! Standard start of any MPI-program
        call MPI_Comm_Size(idcomm,NProc,ierr)! How many processes we have ?
        call MPI_Comm_Rank(idcomm, myProc, ierr)    ! Who am I ?
      endif
      if((n.lt.0).or.(k.lt.0).or.(k.gt.n)) then
        if (myProc.eq.0) write(*,*) ' Clebsch: illegal parameters! ',k,n
        stop
      endif
       c=1.0
      n1=min(k,N-k)  ! slightly reduced loop!

!  The key idea of parallelization :

       do i=1+myProc,N1,NProc  ! each process calculates only its own part!!!
         c=c*float(i)/float(N-i+1)
       enddo
       call MPI_AllReduce(c,prod,1,MPI_Real,MPI_PROD, idcomm,ierr)
       Clebsch2=prod          ! Join all partial results
       return
       end
```

Эта программа является коллективной операцией пакета MPI и прямым потомком операции MPI_AllReduce.
При малых N тест этой программы, будучи распараллеленным на 2 процесса (у автора он работал на 2-ядерном ноутбуке), затратил столько же времени, сколько и его нераспараллеленный оригинал /2/ (около 6 секунд). Зато при больших N (в тесте использовалось N=10000), было достигнуто почти 2-кратное ускорение (22.5 секунды против 40 у оригинала). Поэтому автор с чувством глубокого удовлетворения рекомендует эту программу в нашу библиотеку JINRLIB.

5.10.2011