Wolfram *Mathematica*
Интерактивные вычисления и визуализация
Таранчук Валерий Борисович

Учебные материалы, рекомендации пользователям СКА *Mathematica*, обучающие примеры и упражнения.
Оригинал документа создан и демонстрируется в NB, конспект предоставляется студентам в формате PDF,
учебные материалы размещены на соответствующей странице курса в LMS Moodle.

Уважаемые читатели. В сгруппированных секциях ниже (и везде далее в подобных) размещён материал для демонстраций и обсуждения на лекциях. Такой материал каждый может подготовить сам (например, по записям на лекции) или, найдя аналогичный в других электронных ресурсах, книгах.

▶ Содержание учебных материалов по темам лекций, заданиям практических занятий, имена файлов

Основные конструкции и операции языка Wolfram Language.

Контроль продолжительности вычислений. Инструменты распараллеливания

▼ План лекции

Получение даты и времени, расчёты продолжительности. Задачи для самостоятельного выполнения

Время расчета. Подсчет коэффициентов ускорения. Рекомендации ...

Распараллеливание. Подсчет коэффициентов ускорения. Задачи для самостоятельного выполнения.

▼ Получение даты и времени, расчёты продолжительности. Задачи для самостоятельного выполнения

▼ Варианты вывода даты

Разные варианты вывода даты (DateString - строковая дата):

```
DateString[]
```

Примеры задания форматов вывода:

```
DateString[{"DayName", "  ", "Month", "/", "YearShort"}]
DateString[{1948, 5, 25, 0, 0, 0}, "DayName"]
```

Вывести только день недели:

```
DateString[{1948, 5, 25, 0, 0, 0}, "DayName"]
```

Время в секундах с 1 января 1900 г.:

```
AbsoluteTime[{2017, 10, 30}]
```

Получение даты по абсолютному времени:

```
DateList[%]
```

Задачи для самостоятельного выполнения (освоить не менее 5: *Name, *Short, ISO*)

▶ ✓ Примеры разных форматов вывода

Разница дат в днях:

```
DateDifference["May 25, 1948", "Oct 30, 2017", "Day"]
```

▼ Контроль времени расчёта

```
Quit[]
```

AbsoluteTime – абсолютное значение времени

```
timeLaunchingKernels = AbsoluteTime[];
Sum[N[i^3 * Pi], {i, 1000000}]
timeLaunchingCalcul = AbsoluteTime[];
N[(timeLaunchingCalcul - timeLaunchingKernels), 5]
```

Timing – затраченное время (AbsoluteTiming)

```
Timing[Sum[N[i^3 * Pi], {i, 1000000}]]
```

```
Sum[N[i^3 * π], {i, 1000000}] // Timing
```

```
Timing[Sum[N[i^3 * π], {i, 1000000}]][[1]]
```

```
{Sum[N[i^3 * π], {i, 1000000}] // Timing, Sum[N[i^3 * π], {i, 1000000}] // AbsoluteTiming}
```

Задачи для самостоятельного выполнения:

▼ ✓ Расчёты (3 раза) суммы факториалов (до 300) с подсчетом и выводом времени выполнения

```
f1[n_] := Block[{nCount = n},
   t1 = AbsoluteTiming[Sum[Factorial[i] / (i!! + 1), {i, nCount}]]; Return[t1[[1]]]];
For[i = 0, i < 3, i++, Print[f1[300]]]
```

Расчеты на fpmi602-nout
0.105246
0.0969909
0.0969316

Относительная погрешность – 8.9%:

```
(0.105 - 0.096) / ((0.105 + 0.096) / 2) * 100
```

▶ ✓ Расчёты (3 раза) суммы факториалов (до 500) с подсчетом и выводом времени выполнения

▶ ✓ Расчёты (3 раза) суммы факториалов (до 1000) с подсчетом и выводом времени выполнения

▼ ✓ Расчёты (3 раза) суммы факториалов (до 1200) с подсчетом и выводом времени выполнения

Расчеты на fpmi602-nout

```
f1[n_] := Block[{nCount = n},
   t1 = AbsoluteTiming[Sum[Factorial[i] / (i!! + 1), {i, nCount}]]; Return[t1[[1]]]];
For[i = 0, i < 3, i++, Print[f1[1200]]]
```

Расчеты на fpmi602-nout

18.7806
18.764
18.8006

Относительная погрешность – 0.21%:

```
(18.80 - 18.76) / ((18.80 + 18.76) / 2) * 100
```

Обратить внимание – есть различие в затраченном времени.

Время расчета. Подсчет коэффициентов ускорения. Рекомендации ...

Код не оптимальный, а рабочий, максимально простой и понятный

Пояснения и выполнение кода для очень простого примера:

Пример расчета суммы вычисляемых с машинной точностью значений $\frac{i^3-5}{3\,i^2+7}$.

Есть сложение, вычитание, умножение, деление, возведение в степень. Почему акценты о содержимом формулы, будет понятно далее.

```
nSum = 75 000;
timeLaunchingKernels = AbsoluteTime[];
rSum = Sum[(i^3 - 5) / (3 * i^2 + 7), {i, 1, nSum}];
timeLaunchingCalcul = AbsoluteTime[];
{nSum, N[timeLaunchingCalcul - timeLaunchingKernels, 3], N[rSum]}
```

HP Pavilion 10/11/18 Sum[(i^3-5)/(3*i^2+7), {i, 1, nSum}]

{nSum, time, rSum} *	{nSum, time, rSum} *	{nSum, time, rSum} *
$\{25000, 4.76, 1.04171 \times 10^8\}$	$\{50000, 22.2, 4.16675 \times 10^8\}$	$\{75000, 60.1, 9.37512 \times 10^8\}$
$\{25000, 4.74, 1.04171 \times 10^8\}$	$\{50000, 22.2, 4.16675 \times 10^8\}$	$\{75000, 59.6, 9.37512 \times 10^8\}$
$\{25000, 4.76, 1.04171 \times 10^8\}$	$\{50000, 22.2, 4.16675 \times 10^8\}$	
$\{25000, 4.76, 1.04171 \times 10^8\}$		
$\{25000, 4.82, 1.04171 \times 10^8\}$		

▼ *Попутно*. Цена вопроса “точно” или “с машинной точностью”:

“точно” или “с машинной точностью”:

```
nSum = 75000;
timeLaunchingKernels = AbsoluteTime[];
rSum = Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}];
timeLaunchingCalcul = AbsoluteTime[];
{nSum, N[timeLaunchingCalcul - timeLaunchingKernels, 3], N[rSum]}
```

HP Pavilion 10/11/18

{nSum, time, rSum} *	{nSum, time, rSum} *
Sum[(i^3 - 5) / (3 * i^2 + 7), {i, 1, nSum}]	Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}]
$\{75000, 60.1, 9.37512 \times 10^8\}$	$\{75000, 1.15, 9.37512 \times 10^8\}$
$\{75000, 59.6, 9.37512 \times 10^8\}$	$\{75000, 1.15, 9.37512 \times 10^8\}$

▼ *Попутно*. Цена вопроса ...

“в целой степени” или просто умножение:

```
nSum = 2 000 000;
timeLaunchingKernels = AbsoluteTime[];
rSum = Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}];
timeLaunchingCalcul = AbsoluteTime[];
{nSum, N[timeLaunchingCalcul - timeLaunchingKernels, 3], N[rSum]}
```

```
nSum = 2 000 000;
timeLaunchingKernels = AbsoluteTime[];
rSum = Sum[N[(i * i * i - 5) / (3 * i * i + 7)], {i, 1, nSum}];
timeLaunchingCalcul = AbsoluteTime[];
{nSum, N[timeLaunchingCalcul - timeLaunchingKernels, 3], N[rSum]}
```

HP Pavilion 10/11/18

{nSum, time, rSum} *	{nSum, time, rSum} *
Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}]	Sum[N[(i * i * i - 5) / (3 * i * i + 7)], {i, 1, nSum}]
{500 000, 8.30, 4.16667×10^{10}}	{500 000, 6.36, 4.16667×10^{10}}
{1 000 000, 16.7, 1.66667×10^{11}}	{1 000 000, 12.5, 1.66667×10^{11}}
{2 000 000, 33.5, 6.66667×10^{11}}	{2 000 000, 25.2, 6.66667×10^{11}}

```
(33.5 - 25.2) / 25.2 * 100
```

```
32.9365
```

Второй вариант начальной оценки времени расчета

```
nSum = 1 000 000;
timeLaunchingKernels = AbsoluteTime[];
Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}]
timeLaunchingCalcul = AbsoluteTime[];
t1 = timeLaunchingCalcul - timeLaunchingKernels

t2 = Timing[Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}]];
Part[t2, 2]
Part[t2, 1]
```

HP Pavilion 10/11/18: nSum=1000000 1.66667×10^{11} 16.7544000 1.66667×10^{11} 16.7233

Примеры ниже поясняют - нет “криминала” в различии t1 и t2

Рекомендации, как получать оценки времени расчета

```
f1[] := Block[{nSum = 1 500 000},
   t3 = Timing[Sum[N[(i^3 - 5) / (3 * i^2 + 7)], {i, 1, nSum}]];
   (*Return[{Part[t3,2],Part[t3,1]}]];*)
   Return[Part[t3, 1]]];
For[i = 0, i < 5, i++, Print[f1[]]]
```

HP Pavilion 10/11/18

nSum = 10 000;	nSum = 200 000;	nSum = 500 000;	nSum = 1 000 000;	nSum = 1 500 000;
0.156001	3.26042	8.36165	16.8169	25.319
0.140401	3.26042	8.37725	16.8949	25.2878
0.140401	3.24482	8.36165	16.8793	25.2722
0.124801	3.27602	8.36605	16.8793	25.2722
0.140401	3.27602	8.37725	16.8793	25.319

```
(0.156001 - 0.140401) / 0.140401
(25.2722 - 25.319) / 25.2722
```

Распараллеливание. Подсчет коэффициентов ускорения. Задачи для самостоятельного выполнения.

В тестах в вопросах типа “**Команды поиска функций**” было:

$MachineType - тип используемого компьютера
$ProcessorType - тип процессора
$ProcessorCount - число ядер процессора
$KernelCount - количество доступных вычислительных ядер
Kernels - список ядер
LaunchKernels - запустить ядра
CloseKernels - закрыть параллельные ядра
AbortKernels - прекратить вычисления в ядрах

The proper way to measure the timing of parallelized calculations is ***AbsoluteTiming***, which measures wall time. ***Timing*** measures CPU time on the main kernel only and won’t give a correct result when used with parallel calculations.

... Do not parallelize code that is already parallelized internally, as this is likely to reduce performance. Certain functions such as ***LinearSolve*** use multi-threaded code internally. Some others, such as ***NIntegrate***, will make use of multiple cores in certain cases only. Use a process monitor to check whether your code already makes use of multiple cores without explicit parallelization.

Внимание! Обязательно учитывать “In this kind of simple example, the communication time of parallization is much larger than the time it is trying to save”.

HP Pavilion 10/11/18
{$MachineType,$ProcessorType,$ProcessorCount,$KernelCount}
{PC,x86,2,0}

```
{$MachineType, $ProcessorType, $ProcessorCount, $KernelCount}
```

```
{PC, x86, 2, 0}
```

Parallelize. На примере Sum[N[(i^3-5)/(3*i^2+7)]

```
ClearAll["Global`*"]
nCnt = 50 000;
f[n_] := Block[{nCount = n}, CloseKernels[];
   LaunchKernels[4];
   t1 = AbsoluteTiming[Sum[(i^3 - 5) / (3 * i^2 + 7), {i, nCount}]];
   t2 = AbsoluteTiming[Parallelize[Sum[(i^3 - 5) / (3 * i^2 + 7), {i, nCount}]]];
   CloseKernels[];
   Return[{N[t1[[2]], 4], t1[[1]], t1[[1]] / t2[[1]]}]];
For[i = 0, i < 3, i++, Print[f[nCnt]]]
```

HP Pavilion 10/11/18

Kernels[2]	Kernels[2]	Kernels[2]
nSum = 5000 *	nSum = 15 000 *	nSum = 25 000 *
{4.167×10^6, 0.209434, 0.892488}	{3.750×10^7, 1.67571, 3.70287}	{1.042×10^8, 4.76304, 6.07559}
{4.167×10^6, 0.209232, 0.807517}	{3.750×10^7, 1.66812, 3.19493}	{1.042×10^8, 4.76078, 6.22442}
{4.167×10^6, 0.207021, 0.88689}	{3.750×10^7, 1.66073, 3.64148}	{1.042×10^8, 4.77864, 6.35504}

Kernels[2]	Kernels[2]
nSum = 50 000 *	nSum = 75 000 *
{4.167×10^8, 22.1408, 10.5457}	{9.375×10^8, 60.7177, 16.1957}
{4.167×10^8, 22.4307, 10.8201}	{9.375×10^8, 60.5571, 16.162}
{4.167×10^8, 22.478, 10.7911}	{9.375×10^8, 59.9631, 16.0955}

Kernels[4]	Kernels[4]
nSum = 50 000 *	nSum = 75 000 *
{4.167×10^8, 22.2928, 11.5118}	{9.375×10^8, 60.0063, 17.4362}
{4.167×10^8, 22.2932, 11.5735}	{9.375×10^8, 60.289, 17.7172}
{4.167×10^8, 22.3328, 11.6065}	{9.375×10^8, 60.1344, 17.6604}

* -- выгружены все программы кроме WM

```
(0.156001 - 0.140401) / 0.140401
(25.6622 - 26.255) / 25.6622
```

```
0.11111
```

```
-0.0231001
```

Внимание! Не всё так ... Примеры ниже дают совсем другое ускорение

Parallelize. На примере Sum[Factorial[i]/(i + 1)^5]

```
ClearAll["Global`*"]
CloseKernels[];
nCnt = 5000;
f[n_] := Block[{nCount = n}, CloseKernels[];
   LaunchKernels[2];
   t1 = AbsoluteTiming[Sum[Factorial[i] / (i + 1) ^5, {i, nCount}]];
   t2 = AbsoluteTiming[Parallelize[Sum[Factorial[i] / (i + 1) ^5, {i, nCount}]]];
   CloseKernels[];
   Return[{N[t1[[2]], 4], t1[[1]], t1[[1]] / t2[[1]]}]];
For[i = 0, i < 3, i++, Print[f[nCnt]]]
```

HP Pavilion 10/11/18

Kernels[2] nSum = 5000*	Kernels[2] nSum = 10000*	Kernels[2] nSum = 15000*
$\{1.352\times10^{16307}, 2.80737, 2.29457\}$	$\{2.845\times10^{35639}, 16.7227, 2.75184\}$	$\{3.616\times10^{56108}, 46.7566, 2.83904\}$
$\{1.352\times10^{16307}, 2.80169, 1.96858\}$	$\{2.845\times10^{35639}, 16.6974, 2.73888\}$	$\{3.616\times10^{56108}, 46.7145, 2.86094\}$
$\{1.352\times10^{16307}, 2.82483, 2.37657\}$	$\{2.845\times10^{35639}, 16.7481, 2.8135\}$	$\{3.616\times10^{56108}, 46.7372, 2.81954\}$

Kernels[4] nSum = 5000*	Kernels[4] nSum = 10000*	Kernels[4] nSum = 15000*
$\{1.352\times10^{16307}, 2.80789, 2.45576\}$	$\{2.845\times10^{35639}, 16.7564, 2.88382\}$	$\{3.616\times10^{56108}, 46.7177, 2.89765\}$
$\{1.352\times10^{16307}, 2.80242, 2.52185\}$	$\{2.845\times10^{35639}, 16.7586, 2.9237\}$	$\{3.616\times10^{56108}, 46.8171, 2.95925\}$
$\{1.352\times10^{16307}, 2.80282, 2.52042\}$	$\{2.845\times10^{35639}, 16.7151, 2.89393\}$	$\{3.616\times10^{56108}, 46.6793, 2.95622\}$

* -- выгружены все программы кроме WM

Parallelize. На примере Sum[Factorial[i]/(i!! + 1)^5]

```
ClearAll["Global`*"]
nCnt = 750;
f[n_] := Block[{nCount = n}, CloseKernels[];
   LaunchKernels[2];
   t1 = AbsoluteTiming[Sum[Factorial[i] / (i!! + 1)^5, {i, nCount}]];
   t2 = AbsoluteTiming[Parallelize[Sum[Factorial[i] / (i!! + 1)^5, {i, nCount}]]];
   CloseKernels[];
   Return[{N[t1[[2]], 4], t1[[1]], t1[[1]] / t2[[1]]}]];
For[i = 0, i < 3, i++, Print[f[nCnt]]]
```

```
{0.04586, 40.1832, 3.43618}
{0.04586, 40.1958, 3.42972}
{0.04586, 40.1675, 3.43249}
```

HP Pavilion 10/11/18

Kernels[2]	Kernels[2]	Kernels[2]
nSum = 100*	nSum = 500*	nSum = 750*
{0.04586, 0.0215807, 0.316801}	{0.04586, 8.93141, 2.56238}	{0.04586, 40.1832, 3.43618}
{0.04586, 0.0238418, 0.361829}	{0.04586, 8.93352, 2.56583}	{0.04586, 40.1958, 3.42972}
{0.04586, 0.020395, 0.280375}	{0.04586, 8.91871, 2.54102}	{0.04586, 40.1675, 3.43249}

* -- выгружены все программы кроме WM

Внимание. А иногда и не ускоряет, а замедляет

Рекомендуемая литература