

# **Ручная параметризация**

## Эффективное использование открытых пакетов SALOME, CalculiX, OpenFOAM для создания расчётных сеток в задачах МСС

Калиш С.А. (НИЦ «Курчатовский институт»)  
Крапошин М.В. (НИЦ «Курчатовский институт»)  
Тагиров А.М. (НИЦ «Курчатовский институт»)  
Сибгатуллин И.Н. (НИИ механики МГУ им.  
Ломоносова)  
Стрижак С.В. (МГТУ им. Баумана)



# Ручная параметризация

## Принцип ручной параметризации

Геометрия:

Набор геометрических параметров  
исследуемой геометрии

(Радиусы, длины, высоты и пр.)

Чертёж, словесное описание

Построитель сеток (blockMesh, csgx и пр.)

Описание сетки:

Список вершин, связей между ними,  
контрольных объёмов или  
контрольных элементов

Описание сетки, например для OF:  
`boundary, points, faces`  
`neighbors, owners`

Интерпретатор (m4, bash, perl и др.)

Исходные данные для  
описания сетки:  
Топология и размеры блоков

(Связи между блоками, плотность  
разбиения, координаты вершин и пр.)

Описание блоков в соотв. Формате,  
например - `blockMeshDict`



# Ручная параметризация

## Этапы создания параметризованной сетки

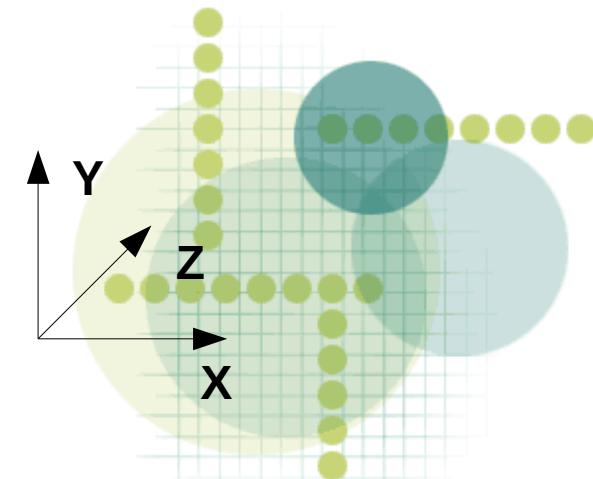
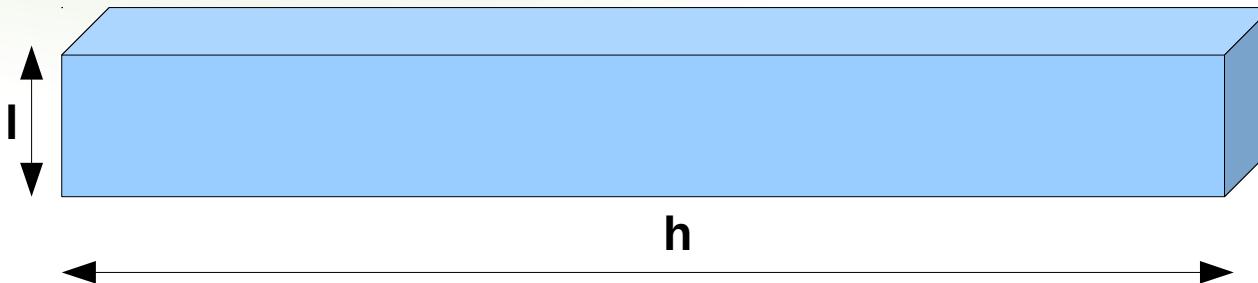
- 1) Формирование списка параметров их отношений, однозначно определяющих геометрию
- 2) Формирование сценария (скрипта), создающего файл описания блочной геометрии
- 3) Тестирование сценария:
  - 1) Запуск сценария для получения варианта описания геометрии для выбранного построителя сетки
  - 2) Построение сетки на основе созданного описания и её тестирование и/или визуализация



# Ручная параметризация

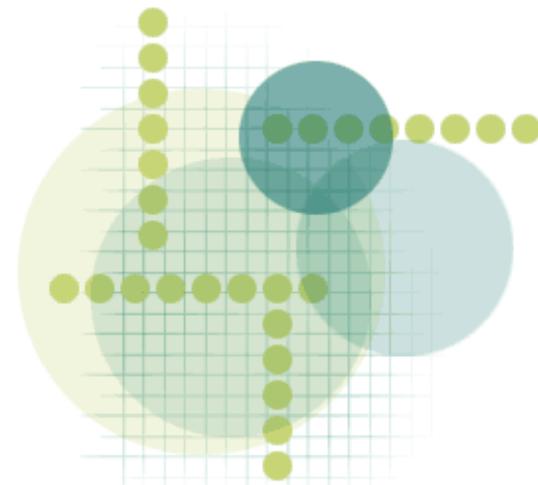
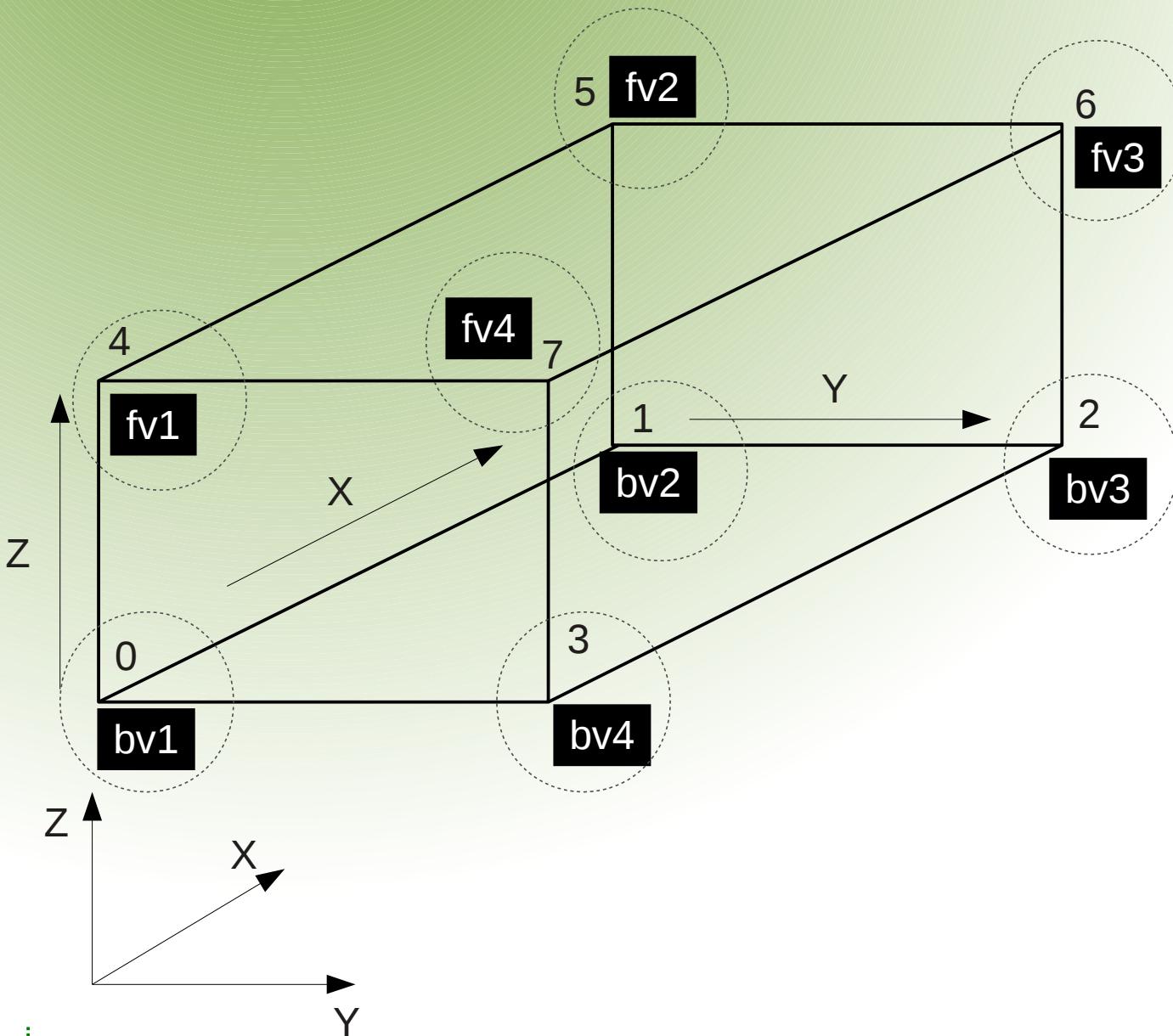
Пример — параллелепипед с квадратным основанием

- Исходные данные:
  - а) координаты центра основания ( $cx$ ,  $cy$ ,  $cz$ )
  - б) ширина основания -  $l$
  - в) высота параллелипипеда -  $h$
  - г) основание строится в плоскости YoZ
  - д) блок вытягивается вдоль оси X
- Блочная геометрия с помощью утилиты `blockMesh`



# Ручная параметризация

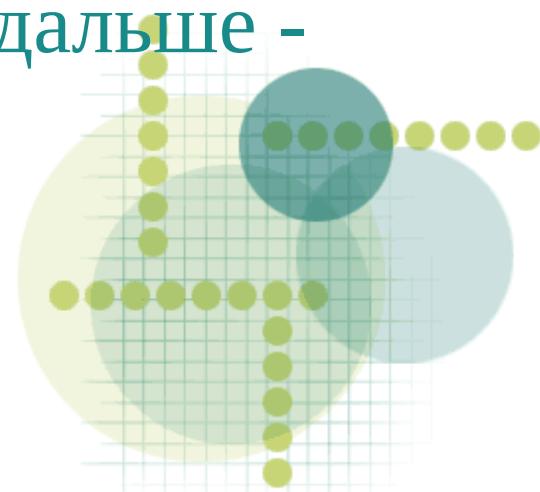
## Схема блочной дискретизации геометрии



# Ручная параметризация

## Основы интерпретатора bash

- Переменные
- Ветвление
- Циклы
- Арифметические вычисления
- «Документ-здесь»
- Комментарии - '#'. Всё, что следует дальше - комментарий
- Процедуры



# Ручная параметризация

## BASH: Переменные, Ветвление

- Переменные

```
var=1.0 #переменная var1 содержит значение 1.0  
echo $var1 #вывести на экран значение в var1  
var2=$var1 + 1 #увеличить значение var1 на 1
```

- Ветвление

```
var1=1  
if [ $var1 -eq 1 ]; then # если var1=1  
    echo "true"  
else  
    echo "false"  
fi
```

# если var1 не равно 1



# Ручная параметризация

## Циклы, Арифметические вычисления

- Циклы (while, until, for-in)

```
var1=2; j=0
```

```
while [ $j -le $var1 ]; do
```

```
    echo $j
```

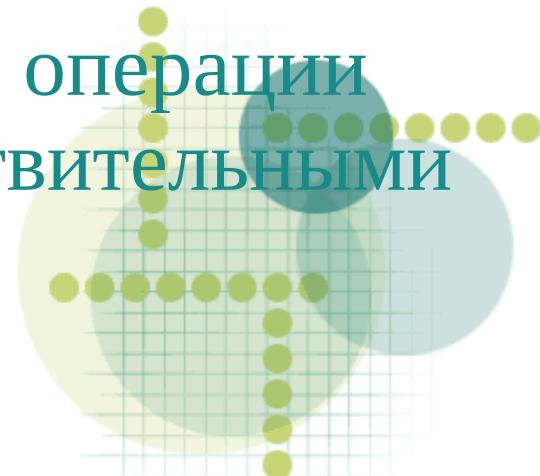
```
    j=`expr $j + 1`
```

```
done
```

- Арифметические вычисления

```
var1=`expr $var1 * 2` #целочисленные операции
```

```
var3=`echo "$var1 * 3.14" | bc` #с действительными  
числами
```



# Ручная параметризация

## Документ-здесь

- Документ-здесь

```
cat << LABEL
```

```
vertices (
```

```
(0 0 0)
```

```
(1 0 0) );
```

```
LABEL
```

```
# всё, что находится между парой LABEL-LABEL выводится на экран (передаётся на вход команды cat, которая выводит содержимое файла в консоль)
```



# Ручная параметризация

## Процедуры

- Процедуры

```
procedure1()  
{
```

```
    echo "$1 $2"
```

```
}
```

```
procedure1 "var1" "var2"
```

```
#вызов процедуры procedure1 с параметрами  
"var1" и "var2"
```



# Ручная параметризация

## Пример параметризации с помощью BASH

- Вычисление координат вершин

```
minx=$cx  
maxx=`echo "$cx+$h" | bc`  
miny=`echo "$cy-$l*0.5" | bc`
```

- Вывод в файл

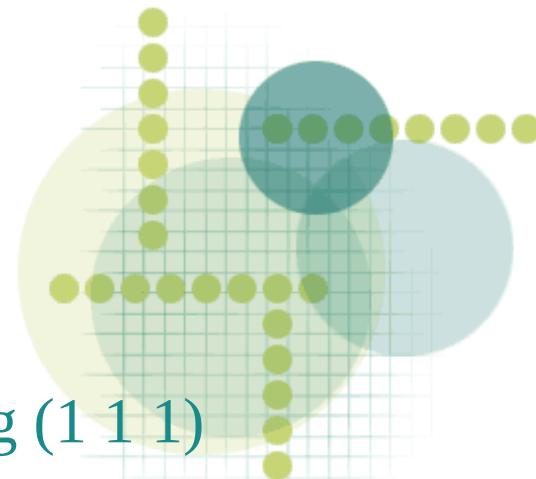
```
cat << BLOCKMESHHEADER  
/*-----*- C++-*-----*/
```

- Формирование списка вершин

```
vertices  
(  
    ($minx $miny $minz)
```

- Формирование списка блоков

```
blocks  
(  
    hex ($block1) ($xdivs $ydivs $zdivs) simpleGrading (1 1 1)  
);
```



# Ручная параметризация

## Основы интерпретатора m4

- Ключевая идея — считать текстовый файл, содержащий основные инструкции (например blockMeshDict) и в определённых частях совершить подстановку согласно заданным правилам
- Объявление имён — define(имя, значение)
- Объявление макроподстановок — define(имя, макроподстановка)
- Отложенная подстановка — `имя'
- Встроенные макросы — changeom, changequote, ifdef, incr, include
- Удаление всех символов строки — dnl>
- Пример макроса — вызов сторонней функции для вычисления алгебраических выражений:  
changequote([,]) dnl> Смена символов отмены подстановки `имя' на [имя]  
define(calc, [esyscmd(perl -e 'print (\$1)')]) dnl> Вычислить значение строки, переданной в качестве аргумента (\$1) средствами Perl



# Ручная параметризация

## Пример параметризации с помощью m4

- Пример создания вершины в blockMesh средствами m4  
vertices  
(  
    vert(minx, miny, minz, bv1) dnl> Создание точки с  
координатами minx, miny, minz и меткой bv1
- Макрос создания метки для точки  
define(vlabel, [ // ]point VCOUNT (\$1)  
define(\$1,VCOUNT)define([VCOUNT], incr(VCOUNT))) dnl>
- Макрос создания точки (вершины)  
define(vert,  
    [(\$1 \$2 \$3)] dnl>  
    [vlabel(\$4)] dnl>  
    ) dnl>



# Ручная параметризация

## Основы интерпретатора perl

- Переменные
- Ветвление
- Циклы
- Арифметические вычисления
- «Документ-здесь»
- Комментарии - #. Всё, что после # не читается
- Процедуры
- Массивы и хэш-таблицы
- <http://www.tutorialspoint.com/perl/>



# Ручная параметризация

## Переменные, Ветвление

- Создание переменных. Все переменные начинаются с символа \$

```
$var1 = 1.0;
```

```
$var2 = $var1 * 1.1;
```

```
$var3 += $var;
```

- Ветвление. Организуется аналогично языку C

```
if ($var1 > 0.0)
```

```
{
```

```
    $var2 = 0.0;
```

```
}
```

```
else
```

```
{}
```



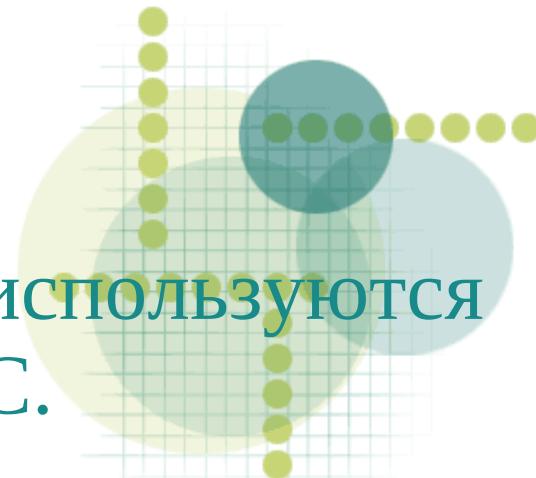
# Ручная параметризация

## Циклы, Арифметические вычисления

- Циклы (while, for, for-in):

```
for $quad (@quads)
{
    print "\t\t\t".$quad."\n";
}
$i=0;
while ($i++ < 10) #вывести числа от 1 до 10
{
    print "$i\n";
}
```

- Для арифметических операций в Perl используются в основном те же, операторы, что и в С.



# Ручная параметризация

## Документ-здесь

- Документ-здесь

```
$blockMeshHeader = <<BLOCKMESHHEADER;  
/*-----*- C++-----*\\"  
BLOCKMESHHEADER  
#перенаправляет строки между BLOCKMESHHEADER-  
BLOCKMESHHEADER в переменную $blockMeshHeader
```

- Процедуры

```
sub procedure1()  
{  
    print "$_[0] $_[1]\n";  
    return $_[0] + $_[1];  
}
```

```
&procedure1(2,3);
```

```
#выводит на экран значения первых двух аргументов и возвращает их  
сумму
```



# Ручная параметризация

## Массивы и хэш-таблицы

- Массив:

```
@array1 = (1, 2, 'val1')  
print $array[0];
```

Массив — это пронумерованный (начиная с 0) список значений

- Хэш-таблица:

```
%hash1 = ('var1', 1, 'var2', 2);  
print $hash1{'var2'};
```

Хэш-таблица — список пар «ключ-значение», в которой как ключ, так и значение могут быть произвольным типом:

```
'var1' => 1  
'var2' => 2
```



# Ручная параметризация

## Пример параметризации с помощью Perl

- &definePoint(\$minx,\$miny,\$minz,'bv1');  
#выводит на экран строку, описывающую вершину с координатами  
(\$minx, \$miny, \$minz) и присваивает ей имя 'bv1'
- sub definePoint()  
{  
    \$x = \$\_[0];  
    \$y = \$\_[1];  
    \$z = \$\_[2];  
    \$pn= \$\_[3];  
    @ks = keys %points;  
    \$pidx = @ks;  
    if (exists(\$points{\$pn}))  
    {  
        print "Found point \$pn with duplicating name at index \$pidx\n";  
        exit ("Found point \$pn with duplicating name at index \$pidx");  
    }  
    \$points{\$pn} = \$pidx;  
    print "\t( \$x \$y \$z ) // point \$pidx (\$pn)\n";  
}



# Ручная параметризация

Вопросы?

