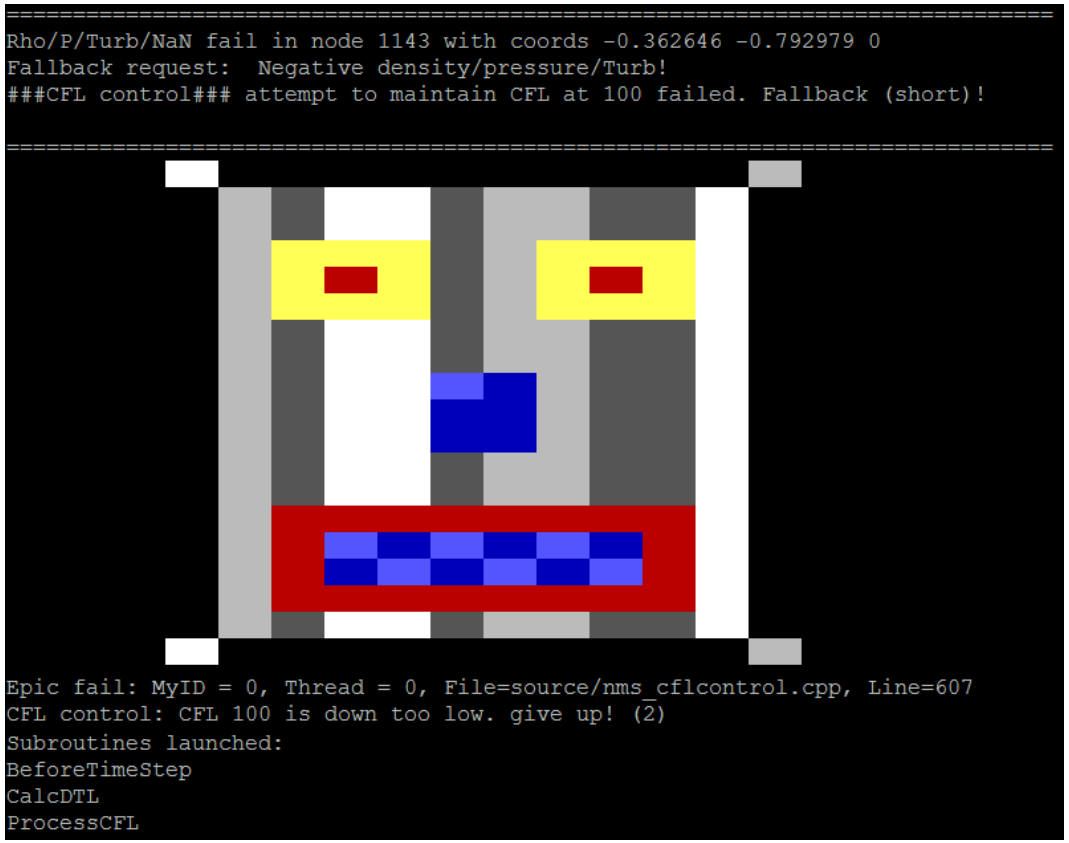
CFD код NOISETTE: прогресс за 2020



NOISETTE:

Вихреразрешающее моделирование

Вычислительная аэродинамика

Вычислительная аэроакустика

Алгоритмы повышенной точности

Неструктурированные сетки

Многоуровневое распараллеливание

MPI + OpenMP + OpenCL

Сектор вычислительной аэродинамики и аэроакустики
ИПМ им. М. В. Келдыша РАН
http://caa.imamod.ru

NOISETTE – CFD код для исследований и для приложений

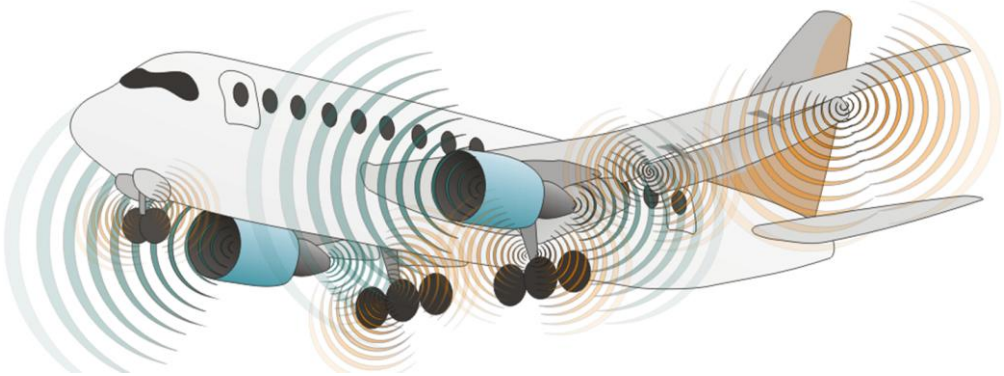
- **Экономичные схемы повышенной точности**
  неструктурированные сетки

- **Вихреразрешающие расчеты**
  сжимаемые течения, дозвук, сверзвук,
  предсказание ~~на кофейной гуще~~
  аэродинамических и акустических характеристик

- **C++, MPI, OpenMP, OpenCL**

- **HPC параллельная реализация**
  масштабируемые алгоритмы
  гетерогенные вычисления

- **Технологии моделирования**
  турбулентность – RANS, LES, DES
  IBC, акустика в дальнем поле FW/H,
  численный beamforming, ускорение,
  динамически адаптивные сетки,
  параллельный постпроцесс, …

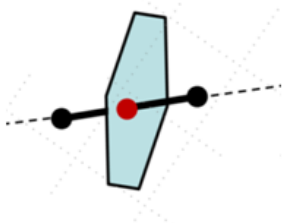


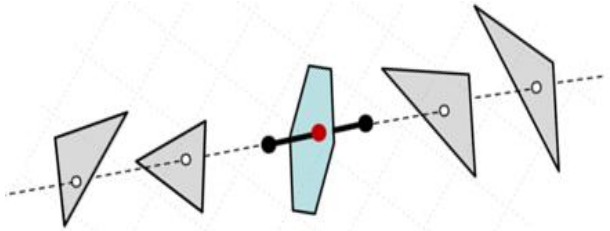
http://caa.imamod.ru/noisette

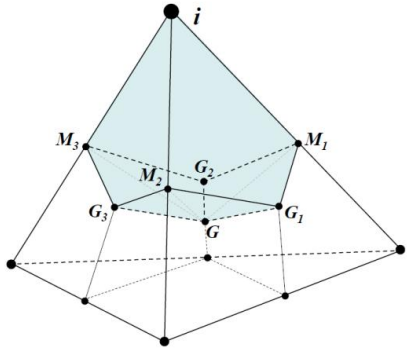
Технологии вычислительного эксперимента

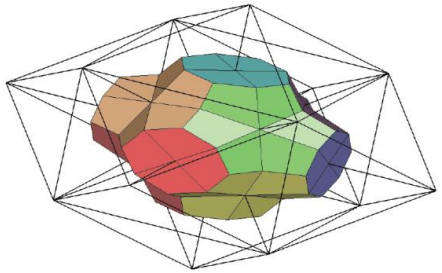
Оригинальные схемы EBR

повышенная точность на неструктурированных смешанных сетках
cell-centered & vertex-centered

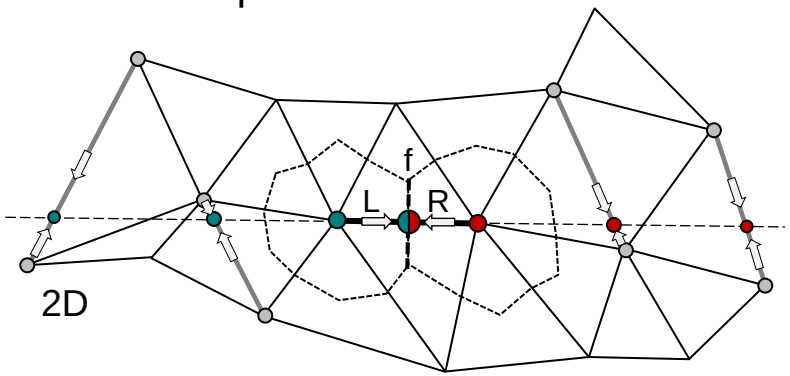


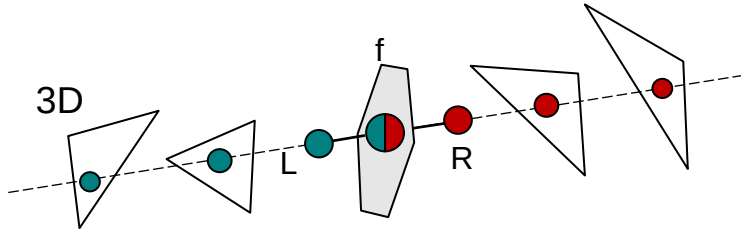






Контрольные объемы





Интерполяционные конструкции

Bakhvalov Pavel, Abalakin Ilya, Kozubskaya Tatiana. Edge-based reconstruction schemes for unstructured tetrahedral meshes. International Journal for Numerical Methods in Fluids. 2016. Vol.81(6). P. 331–356. https://doi.org/10.1002/fld.4187

Bakhvalov Pavel, Kozubskaya Tatiana. EBR-WENO scheme for solving gas dynamics problems with discontinuities on unstructured meshes. Computers and Fluids. 2017. Vol. 157, p. 312-324. https://doi.org/10.1016/j.compfluid.2017.09.004

Технологии вычислительного эксперимента

- **Схемы интегрирования по времени**

  Неявные схемы – линеаризация по Ньютону
  решение больших СЛАУ с блочной разреженной матрицей

  Многосеточные методы – FAS MG

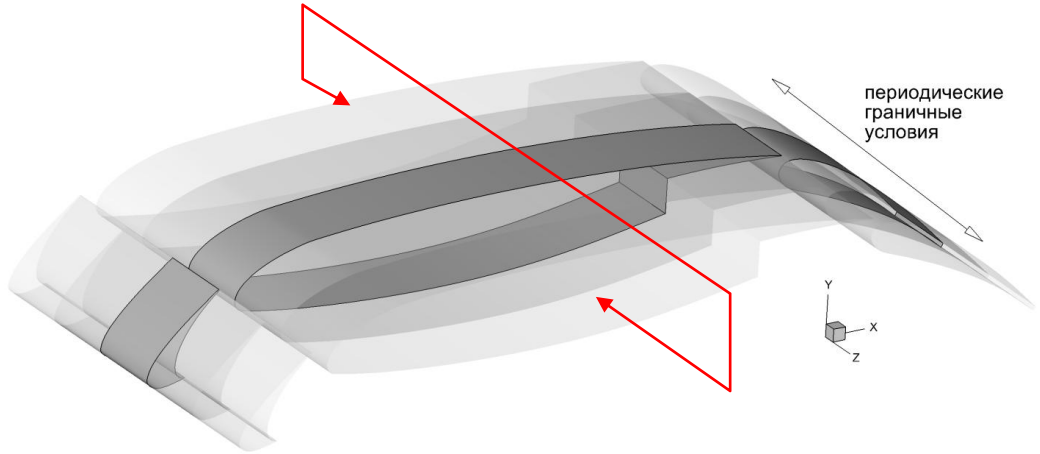
  Методы на основе LU-SGS

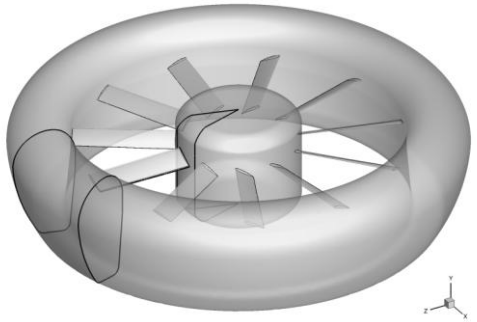
- **Динамическая конфигурация метода**
  управление расчетом, автовыбор параметров численного метода

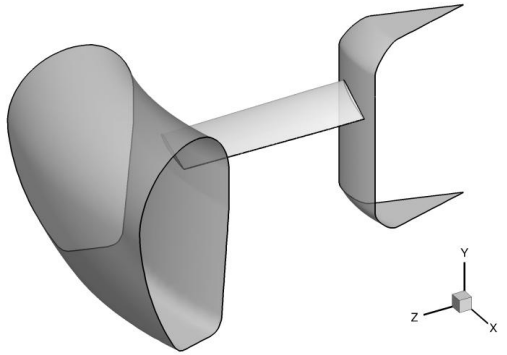
- **Runtime диагностика и анти-авост**
  автоконтроль корректности вычислений, коррекция без авоста

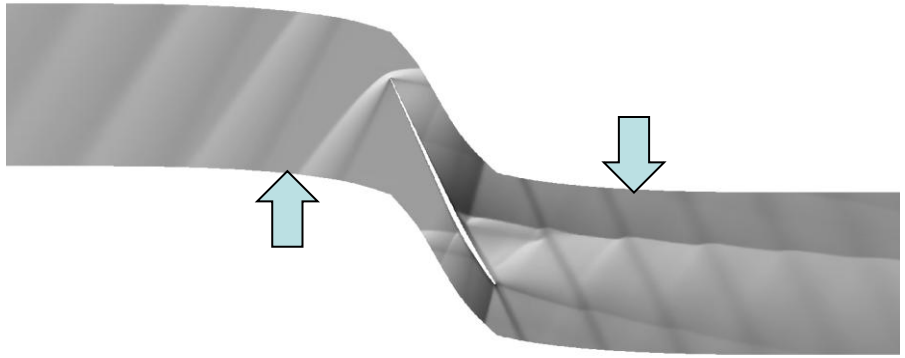
Технологии вычислительного эксперимента

- **Периодические граничные условия**

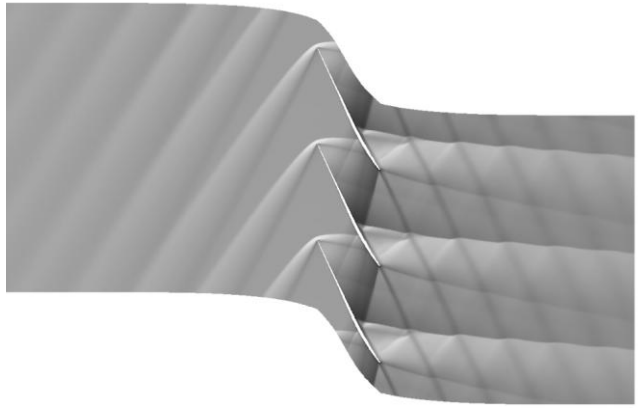








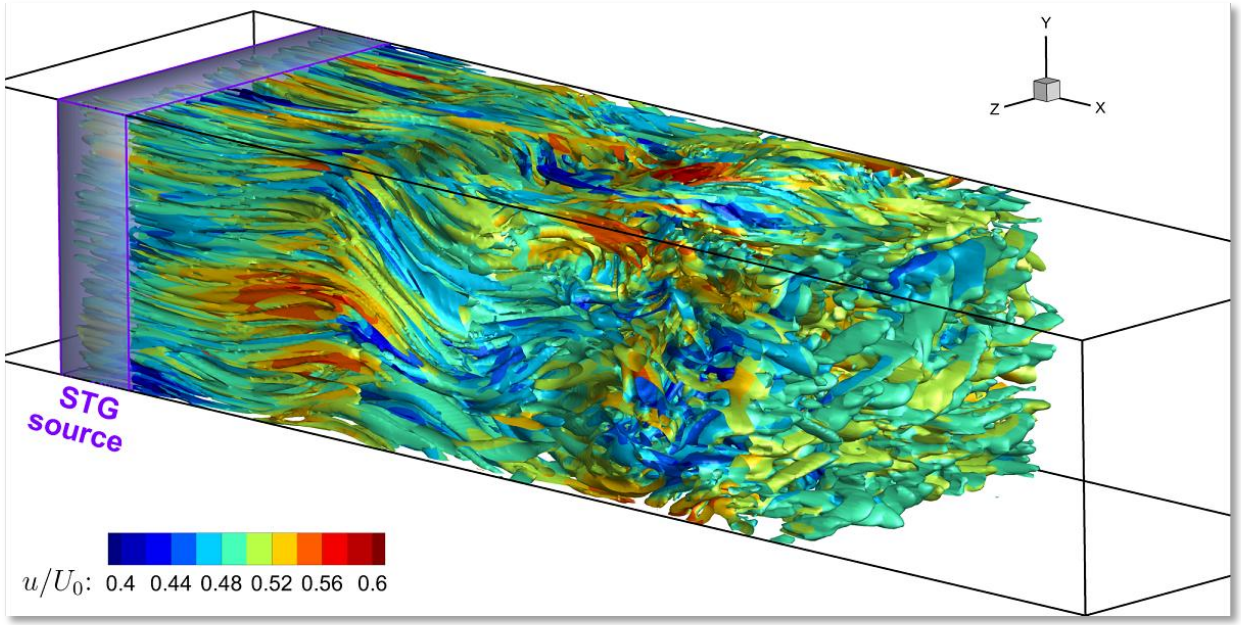
size matters



Технологии вычислительного эксперимента

- **Генерация синтетической турбулентности**
  моделирование фрагментов сложного объекта,
  RANS – LES интерфейс

- **Поглощающий слой**
  чтобы выйти обратно из леса

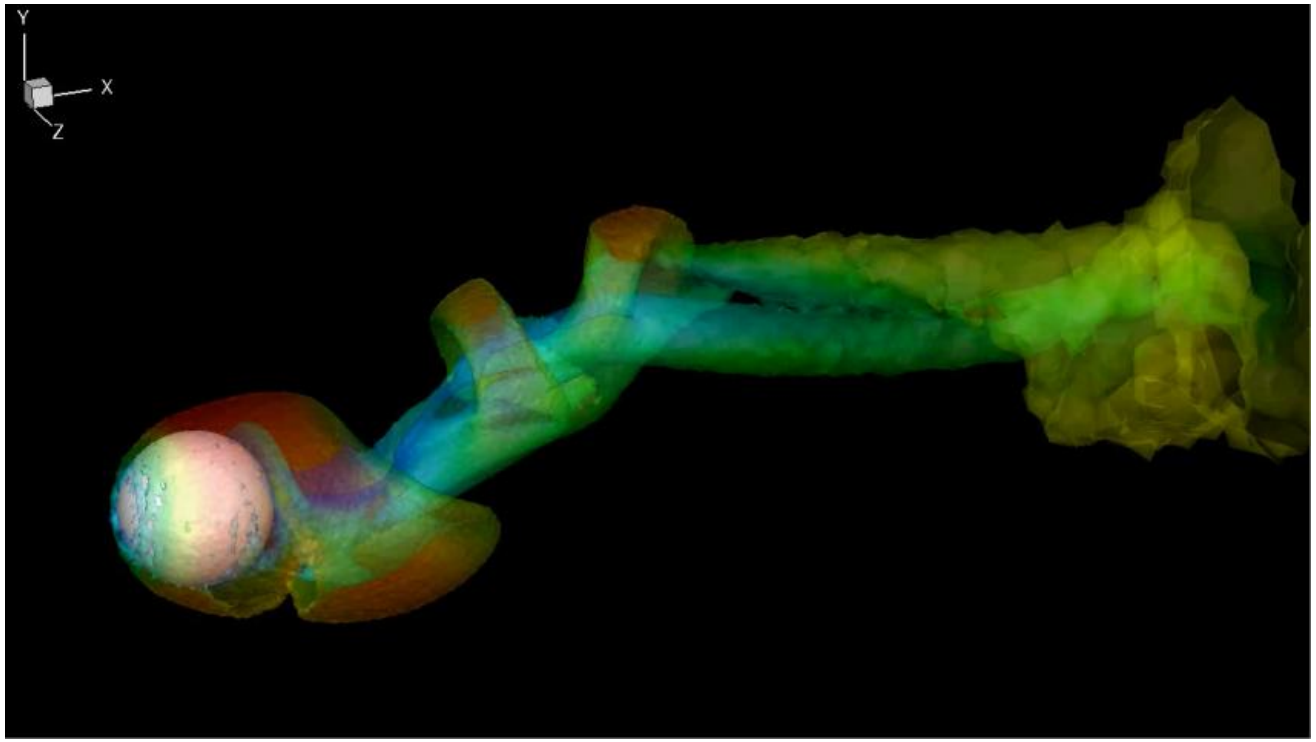


- **Погруженные граничные условия – IBC**

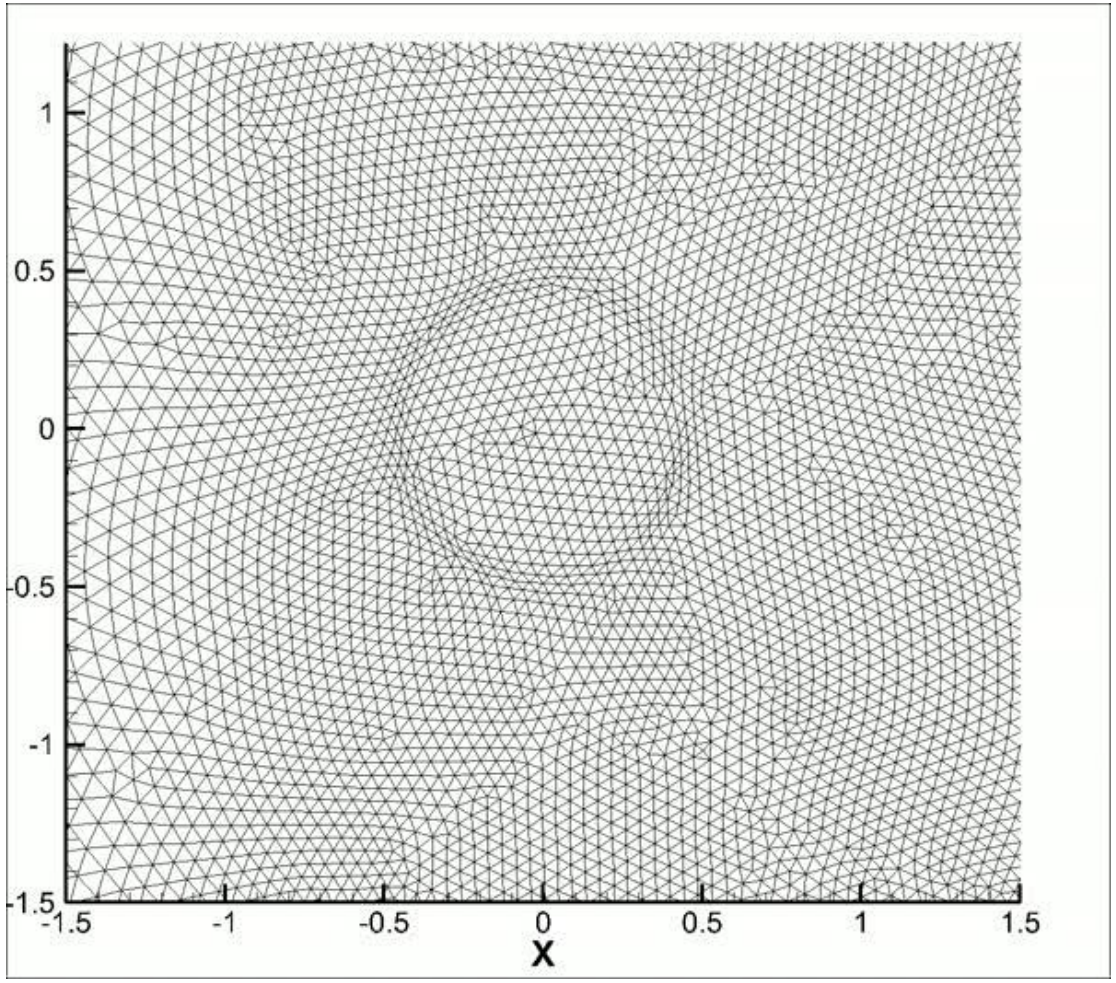
моделирование подвижных объектов

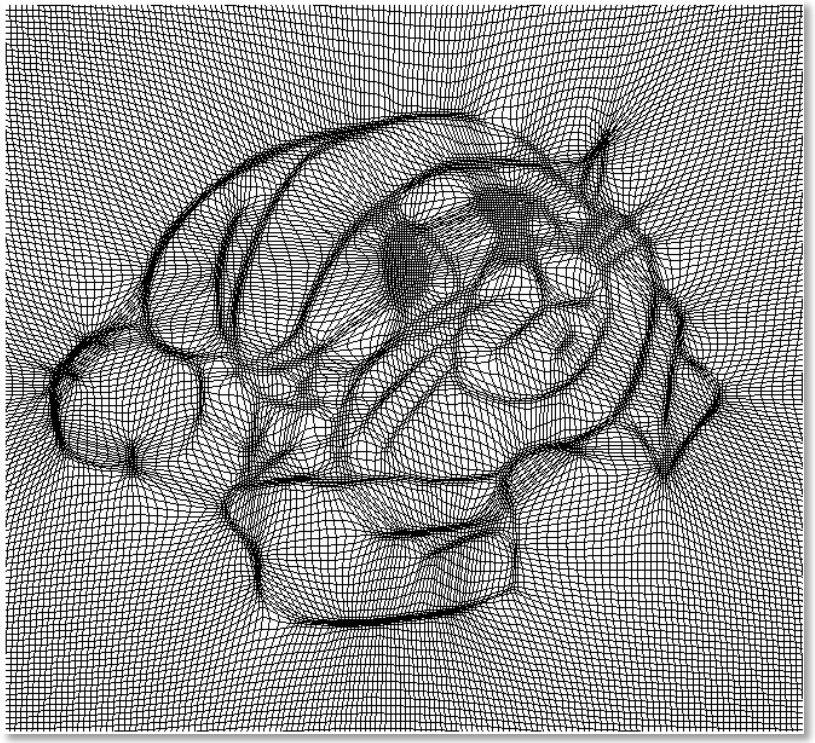
движение тел под действием сил, как внешних, так и индуцированных

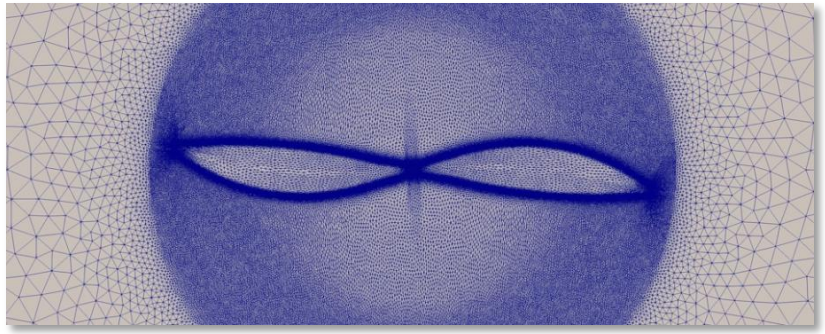
$$\begin{cases} \dfrac{\partial \mathbf{Q}_\eta}{\partial t} + \nabla \cdot \mathbf{F}\left(\mathbf{Q}_\eta\right) + \dfrac{1}{\eta}\chi \begin{pmatrix} 0 \\ \rho_\eta\left(\mathbf{u}_\eta - \mathbf{u}_{\Omega_o}\right) \\ E_\eta - E_{\Omega_o} \end{pmatrix} = \dfrac{1}{\mathrm{Re}} \nabla \cdot \mathbf{F}_v\left(\mathbf{Q}_\eta\right) \\ \mathbf{Q}_\eta\left(\mathbf{x}, t=0\right) = \mathbf{Q}_0\left(\mathbf{x}\right) \end{cases}$$



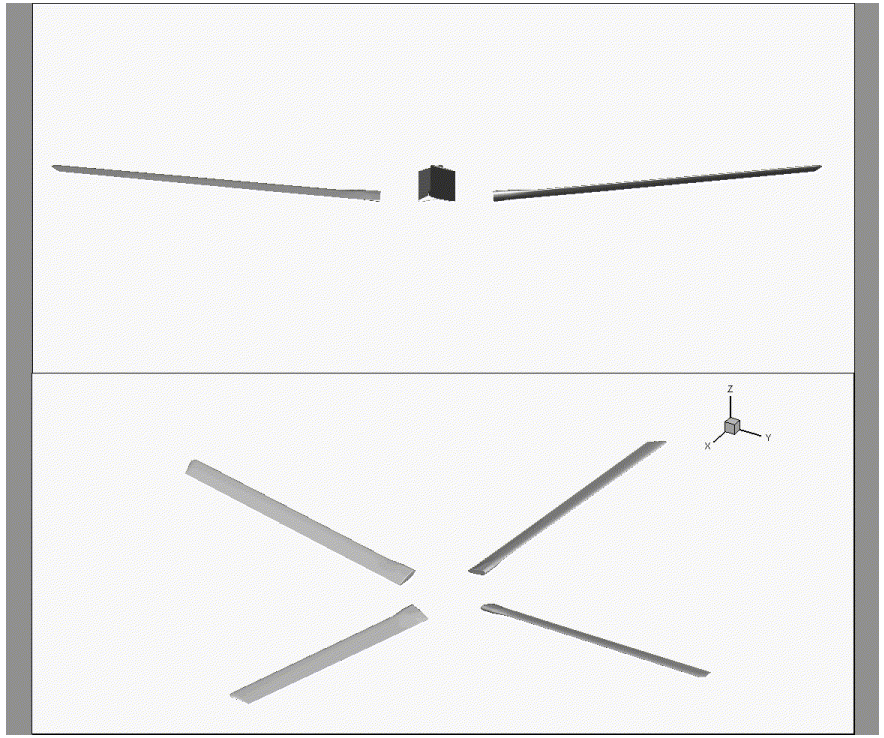
- **Динамическая адаптация сетки без изменения топологии**
  улучшение пространственного распределения узлов
  подвижные объекты IBC

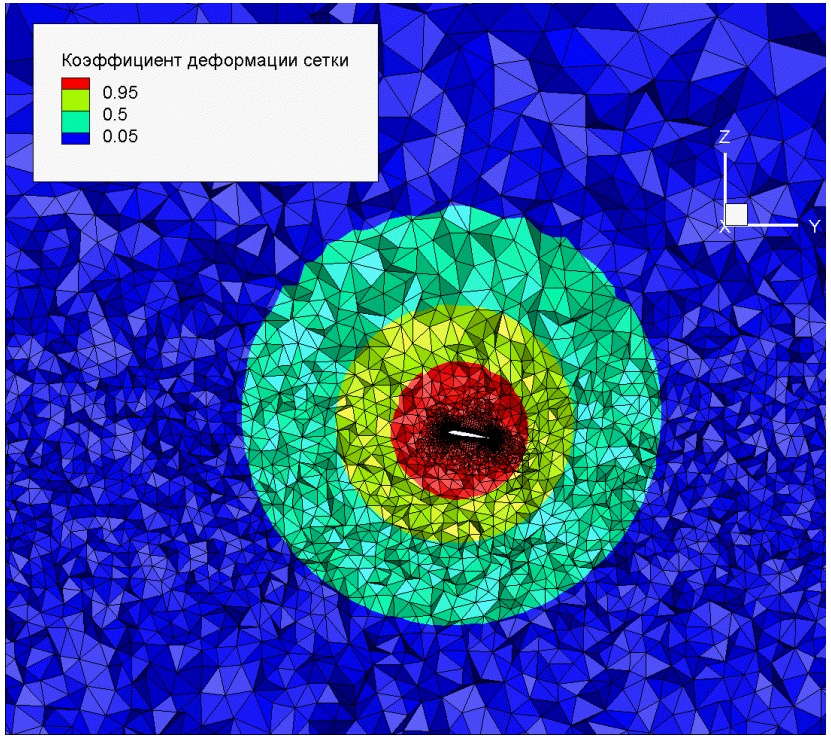


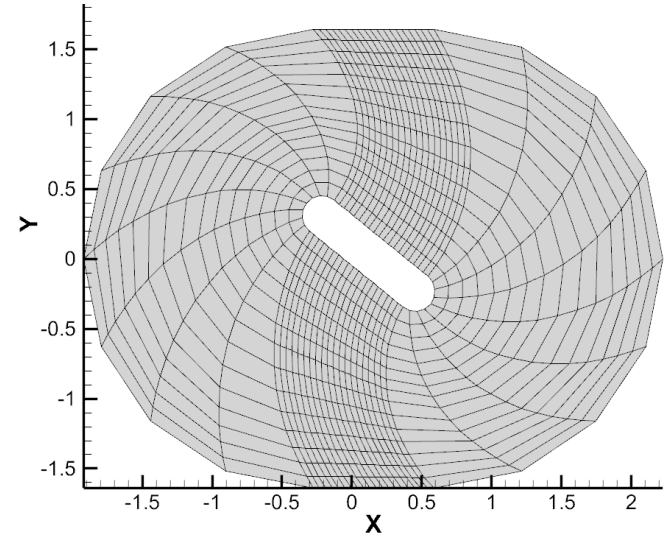


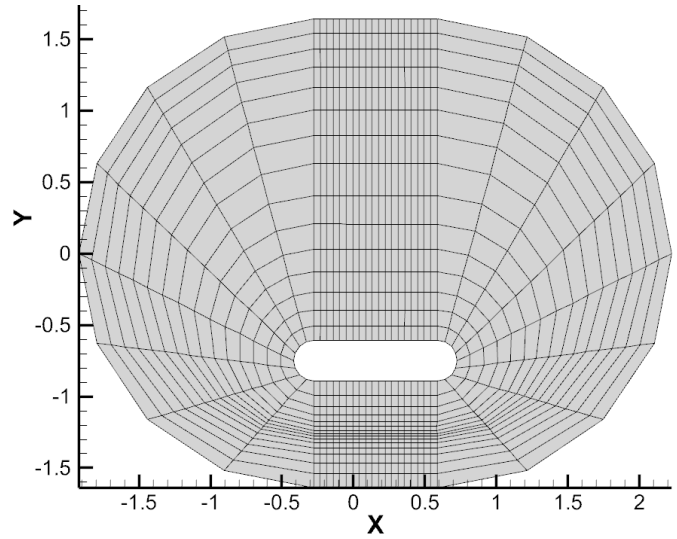


- **Деформируемые сетки**



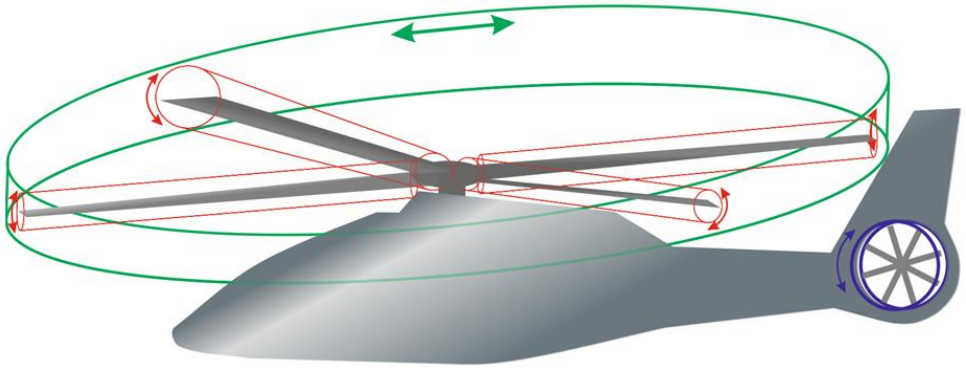


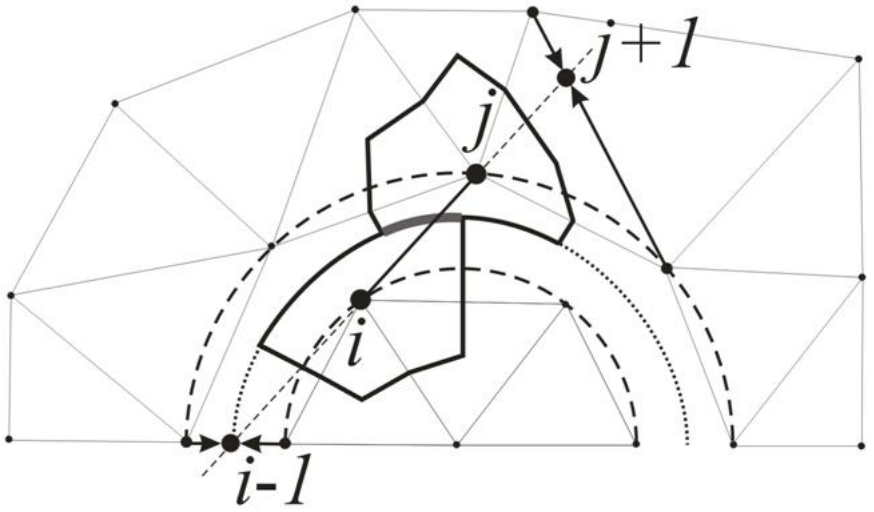


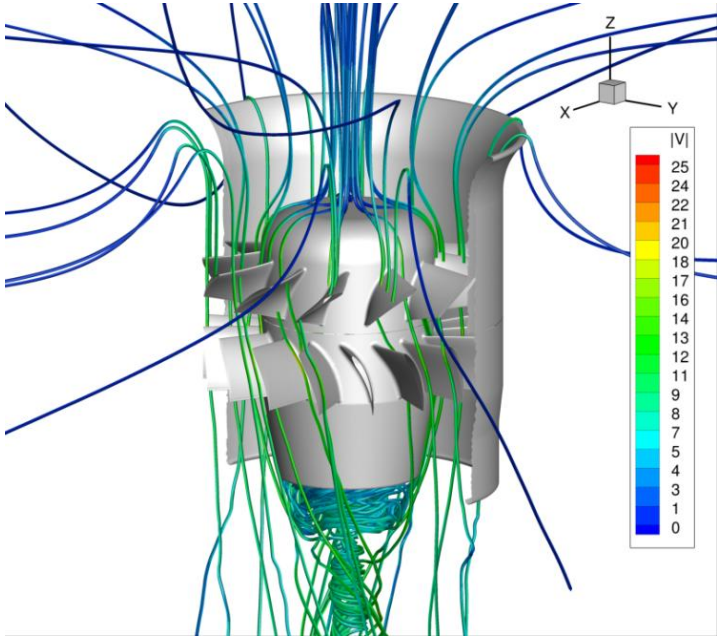


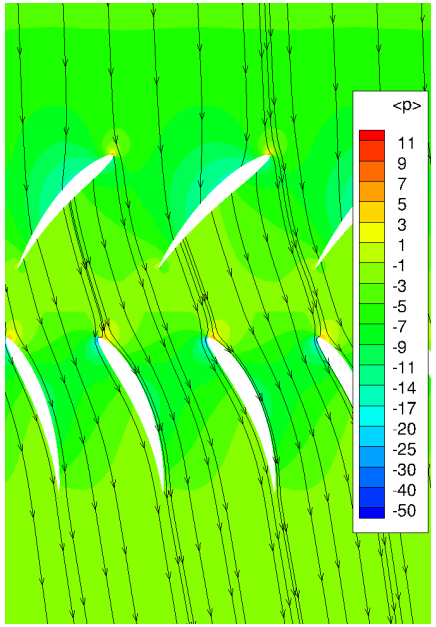
Технологии вычислительного эксперимента

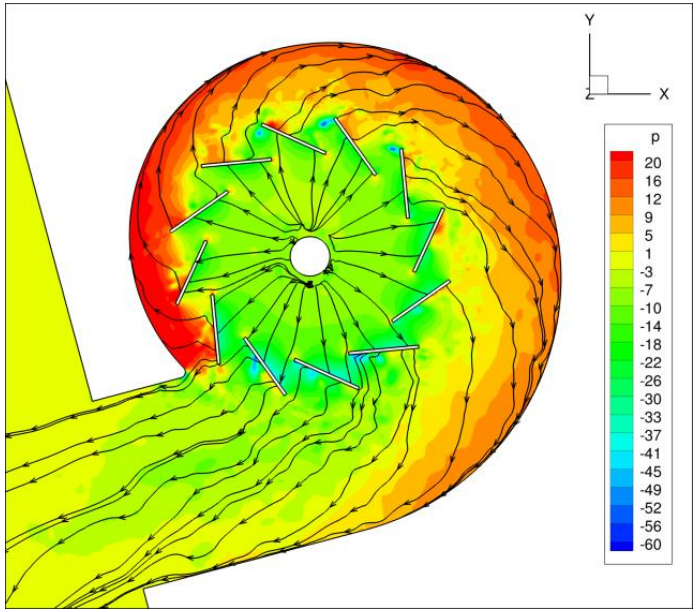
- **Скользящий интерфейс – вращающиеся сетки**
  ротор-статор взаимодействие
  cell-center, vertex-center, MPI+OpenMP распараллеливание



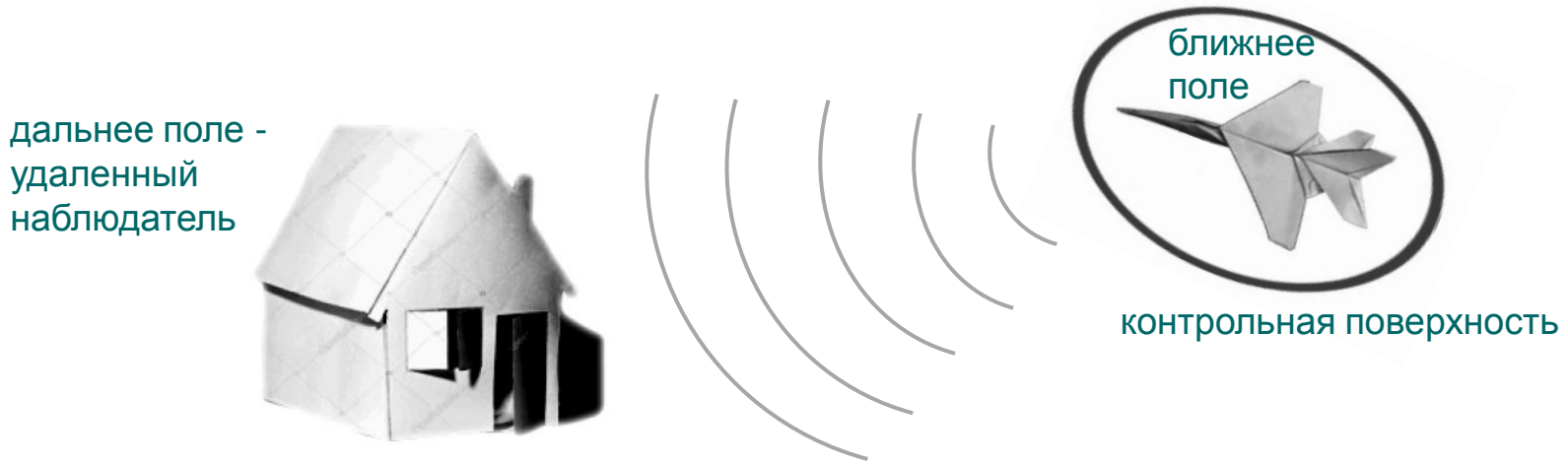


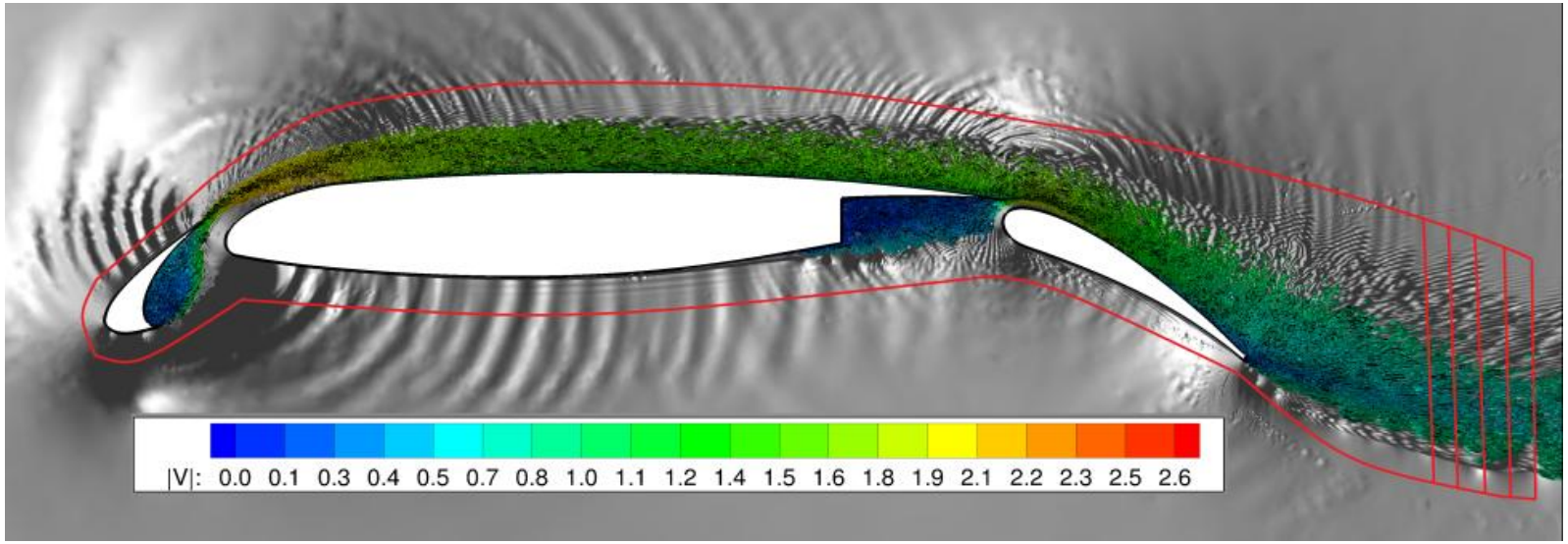




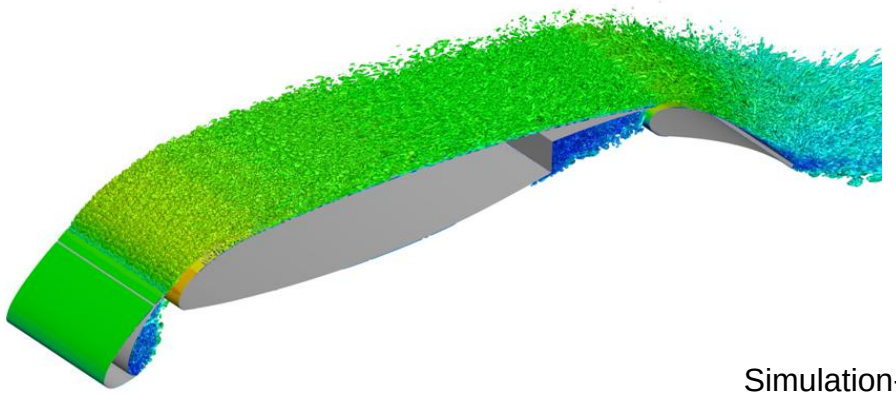


- **Акустика в дальнем поле – метод FW/H**
  параллельный постпроцессор для обработки акустических данных

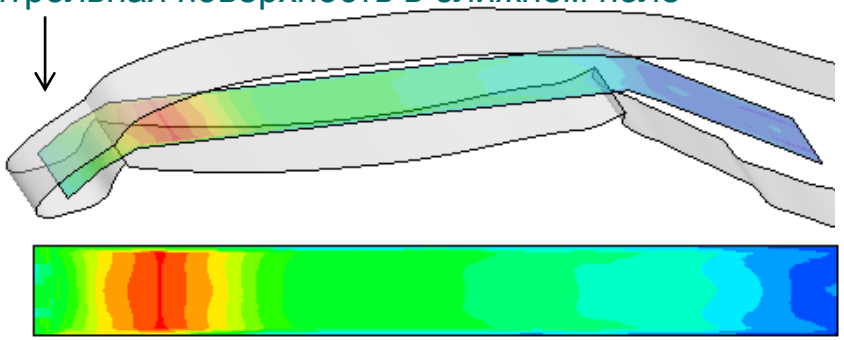




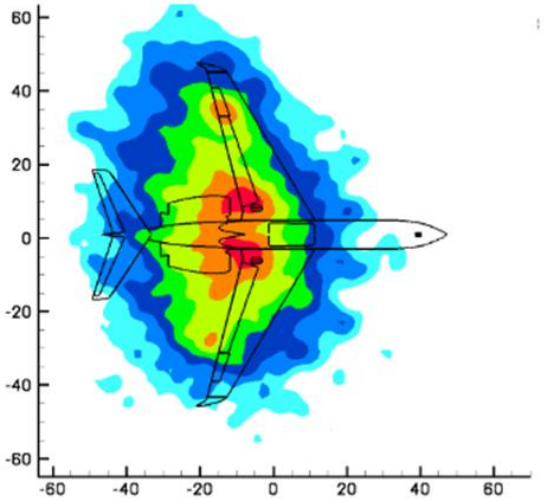
- **Численный beamforming**
  восстановление источника волнового уравнения во внутренней области

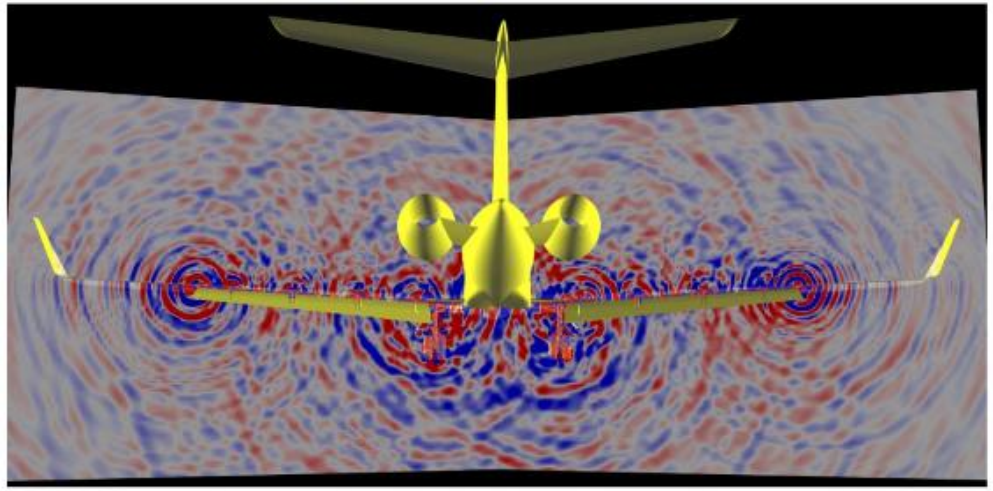


Контрольная поверхность в ближнем поле



Simulation-Based Airframe Noise Prediction of a Full-Scale, Full Aircraft
Mehdi R. Khorrami (NASA), Ehab Fares (Exa GmbH)





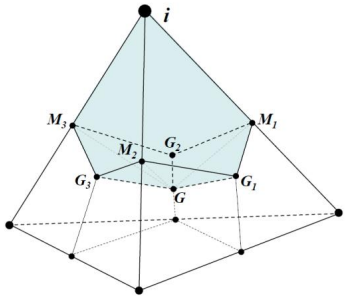
https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160009089.pdf

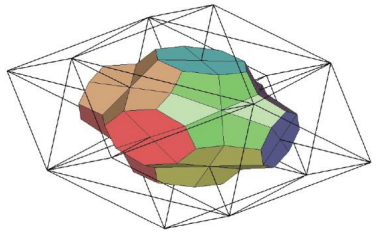
Прогресс 2020

- **Стало не на чем считать**
  всё кругом позанято, а что не позанято – то в дауне на профилактике
- **А еще надо столько всего посчитать**
  чтобы хоть немного оправдать свое существование
- **Хотим переползти на графон**
  чтобы хоть как-то это все успеть посчитать
- **А на GPU все сложно, туда просто так на хромой козе не подъедешь**
  код изначально должен быть хотя бы надежный и простой
- **А у нас в коде проблемки**
  в нем развелся бардак, что чёрт ногу сломит и без всяких ускорителей
- **А еще надо впиливать в код столько всякой ерунды**
  чтобы было о чем наспамить столько ненужных статей

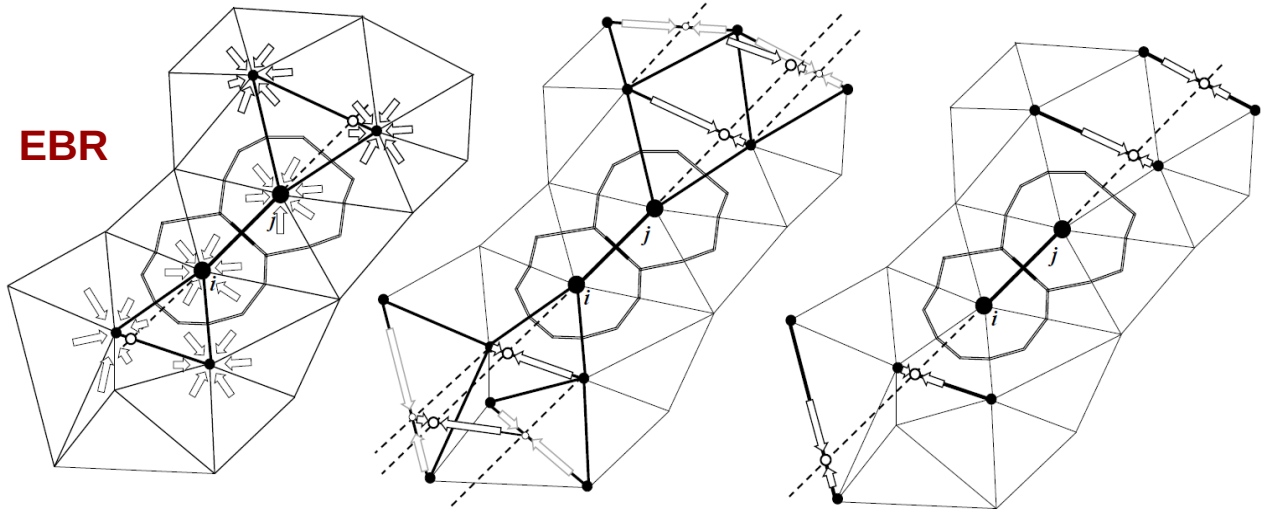
И как с этим жить!?

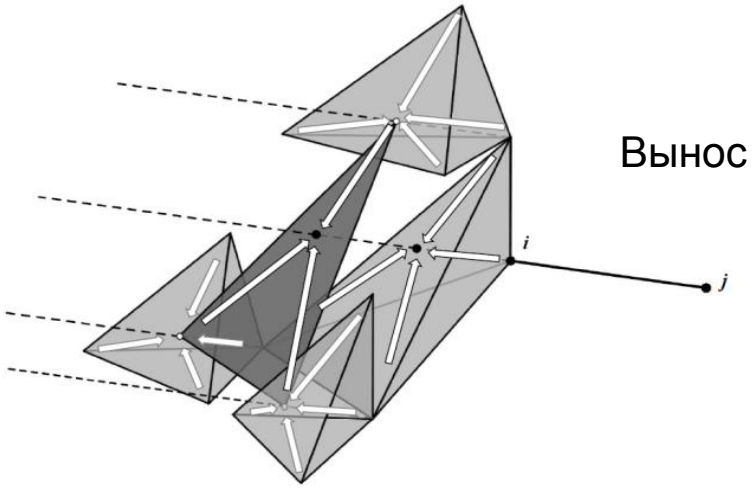
Песочница для разработки новых схем – хаос и бардак



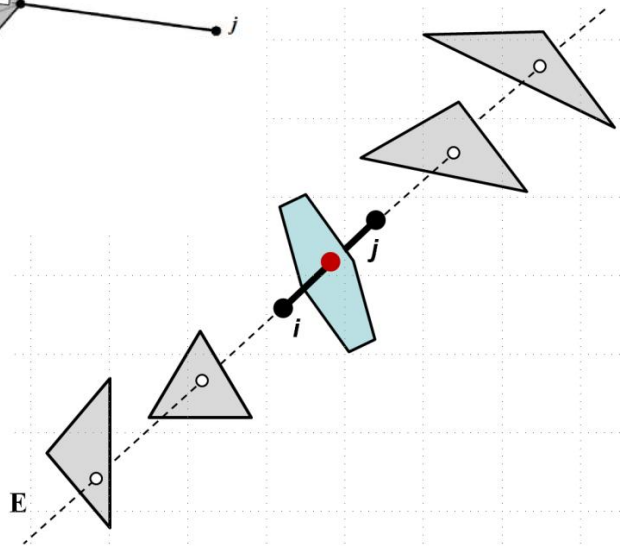


EBR

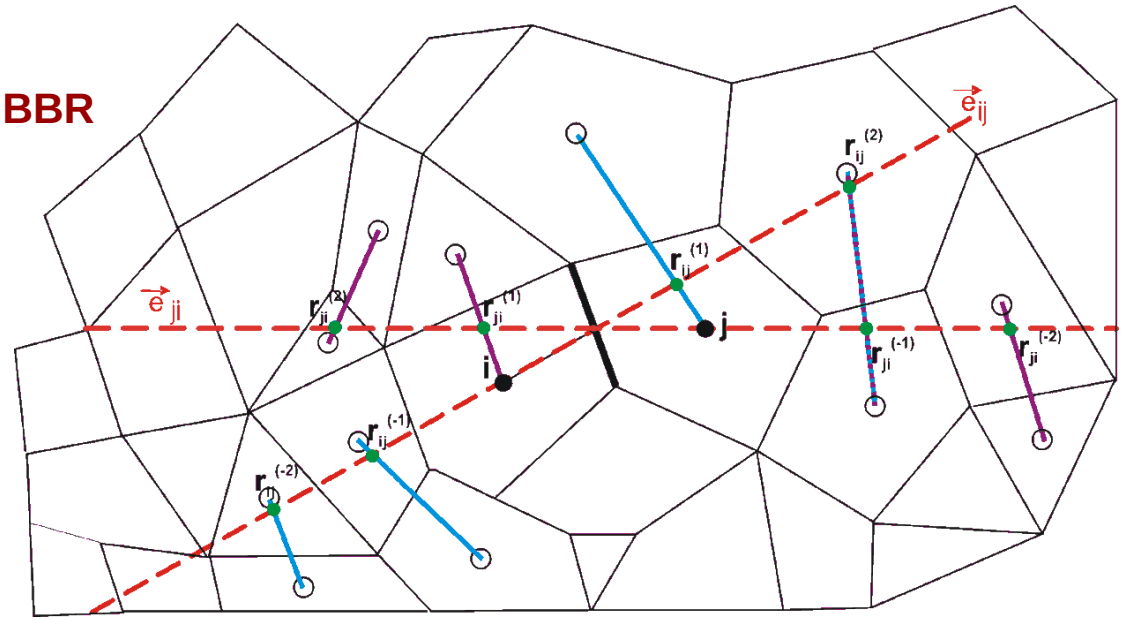




Вынос мозга коллекцией интерполяционных конструкций



BBR



Конфигурация SAFE MODE

Программные уровни с точки зрения производительности

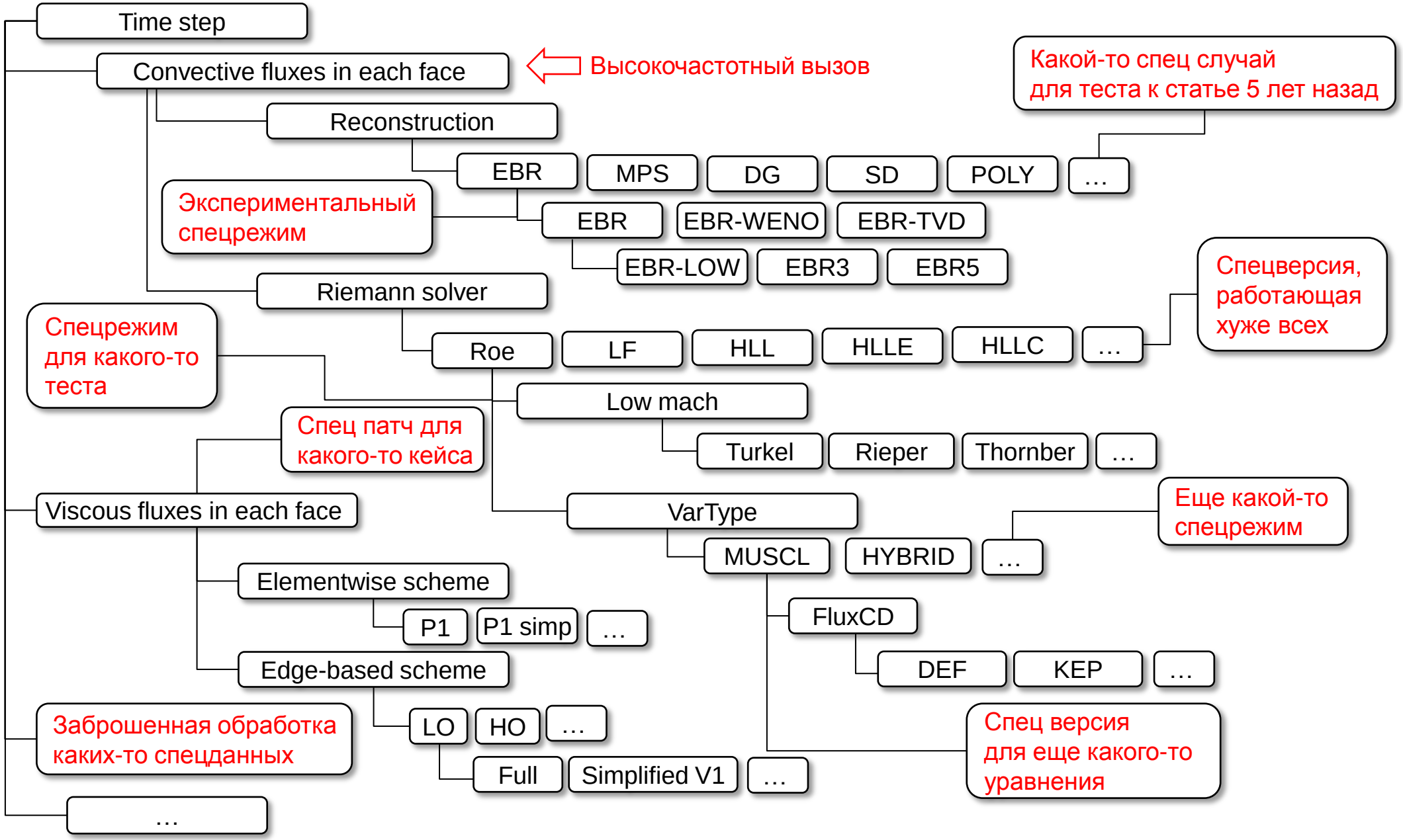
LL – Low Level – высокочастотные функции
performance-critical, обработка одного сеточного объекта
ифы, свичи, ассёрты, вызовы функций – создают оверхэд

HL – High Level – низкочастотные функции
обработка наборов сеточных объектов
ифы, свичи, ассёрты, вызовы функций – пренебрежимы

- **Low-level проверки включены в конфиге SAFE MODE, выключены в релизе**
- **QA процедура – в сейфмоде**
- **Проверка доступа к контейнерам – массивам V[i], блочным массивам V[i][j] по обоим индексам**
- **Сейфмод медленнее релиза раза в полтора из-за множества высокочастотных проверок**

A. Gorobets, P. Bakhvalov. Improving Reliability of Supercomputer CFD Codes on Unstructured Meshes. Supercomputing Frontiers and Innovations. 2019. Vol. 6, No. 4, pp. 44-56. http://dx.doi.org/10.14529/jsfi190403

Темная сторона исследовательского кода: запутанный клубок из беспорядка



Проблема не из программной, а из социальной сферы

Может и не надо бы держать в коде устаревший неиспользуемый функционал.

Но это же исследовательской код.

Нельзя просто взять и выбросить чужой ненужный хлам.

Ученые-разрабы воспринимают это как личное оскорбление.

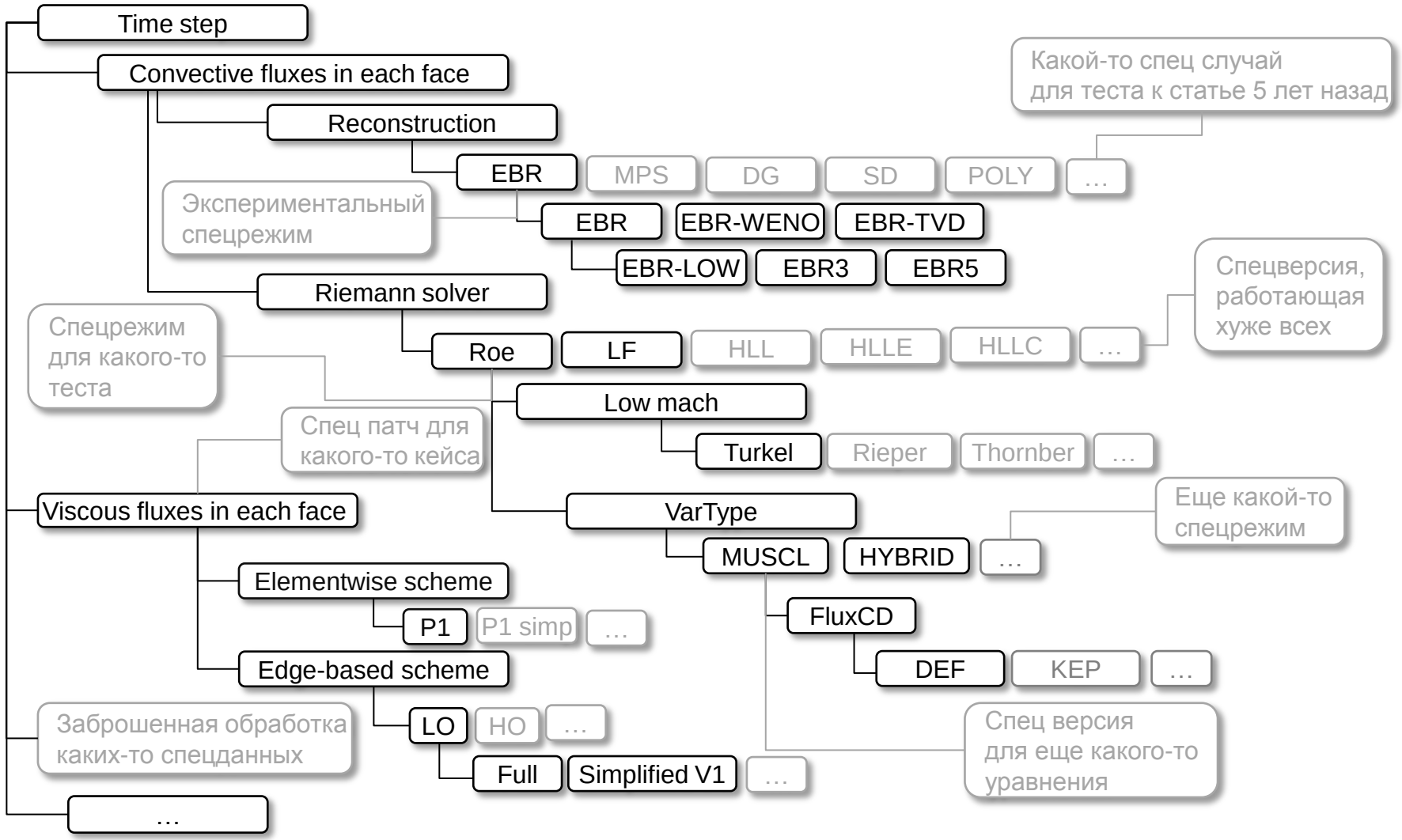
Кто посмел убрать из кода мою любимую игрушку??
Ну и что, что она торчит по всему коду ветвлениями в 20 местах?
Почему это она не используется!? А 5 лет назад для какого-то теста для какой-то статьи!?
Что значит она больше не нужна!? А вдруг через 10 лет еще понадобится какой-то тест?
…

Но у социальной проблемы есть программное решение: #ifdef PLAYGROUND

Берем весь мусор на нижнем уровне и заметаем под ковер!

Конфигурация PLAYGROUND

```
#ifdef PLAYGROUND_MODE
   SomeExperimentalUnusedAbandonedThing();
#endif
```



Заброшенный хлам и редко используемые режимы больше не снижают перф

Конфигурация смешанной точности – экономим память

- **FP32 – линал в неявке и тяжелые коэффициенты для вязкости**
- **FP64 – все остальное**

```
#define LINAL_FLOAT // FP32 in solver

#ifdef LINAL_FLOAT
typedef double linal_real; // Solver data
typedef double dif_real;   // Visc coefficients
#else
typedef float linal_real;
typedef float dif_real;
#endif
```

~2X по памяти, ~1.7X по скорости

и никакого влияния на точность результата

A. Gorobets, P. Bakhvalov, A. Duben, P. Rodionov. Acceleration of NOISEtte Code for Scale-resolving Supercomputer Simulations of Turbulent Flows. Lobachevskii Journal of Mathematics. Vol 41, No 8, pp. 1463–1474, 2020. https://doi.org/10.1134/S1995080220080077

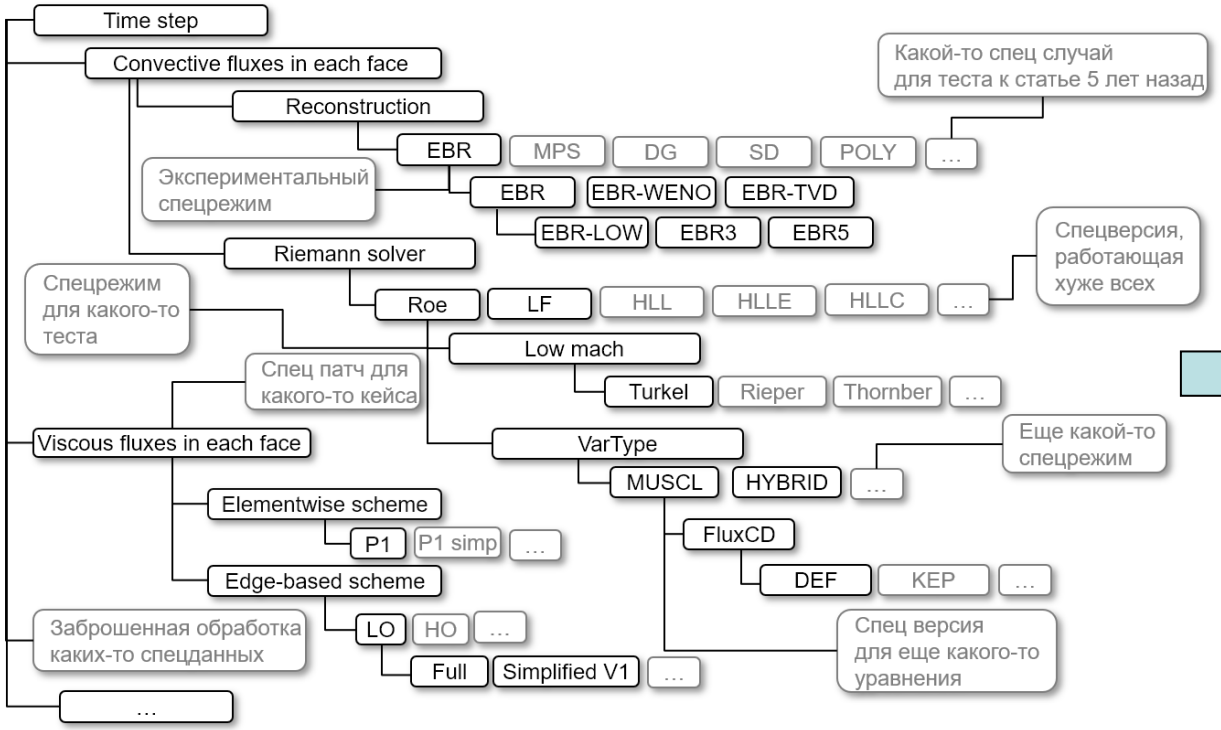
Конфигурация БИЗНЕСЛАНЧ

Множественные функции слиты в одну общую функцию с фиксированным набором методов

- **Намного выше производительность**

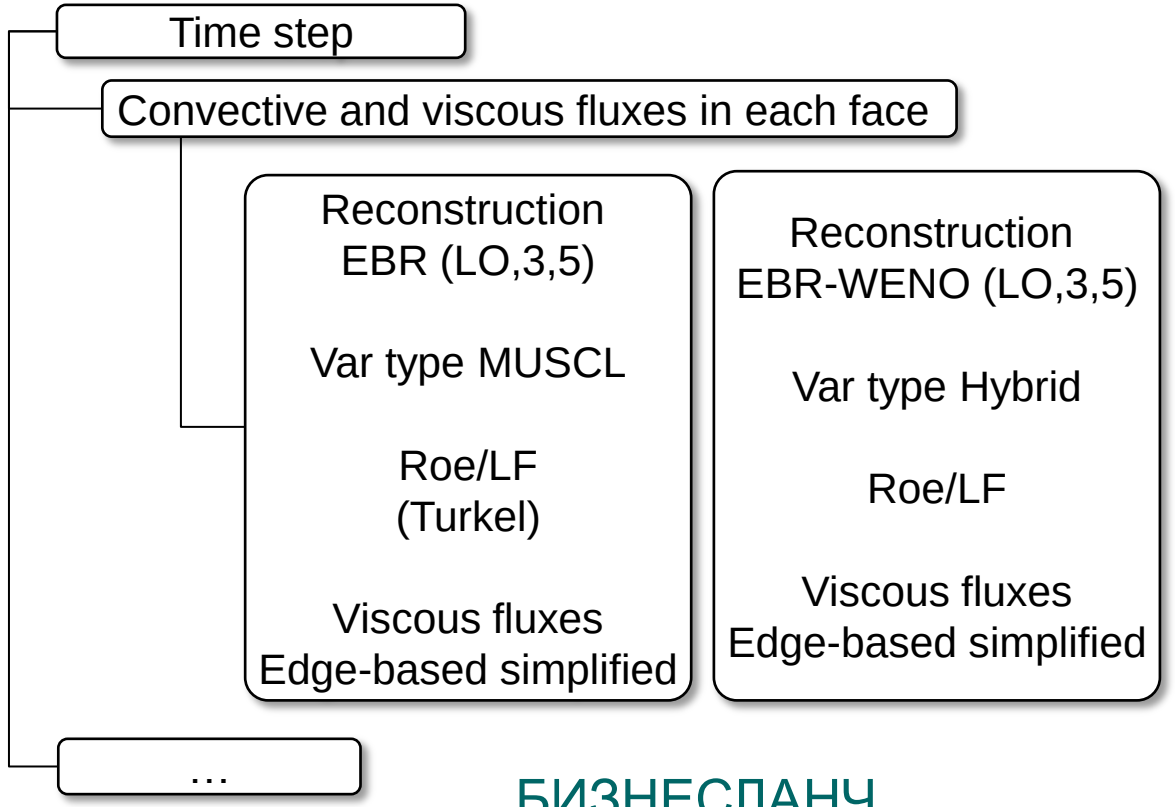
  Нет оверхэда на вызовы функций, намного меньше ветвлений, свичей;

  нет повторного чтения-записи потоков и особенно якобиана, самой большой штуковины



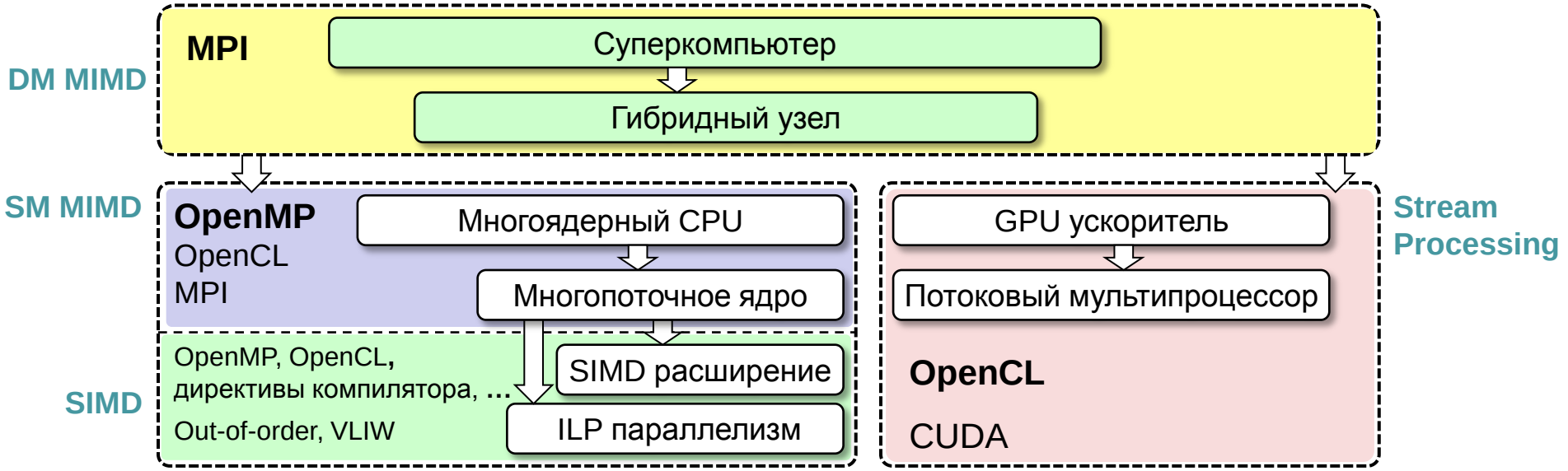
САРАЙ

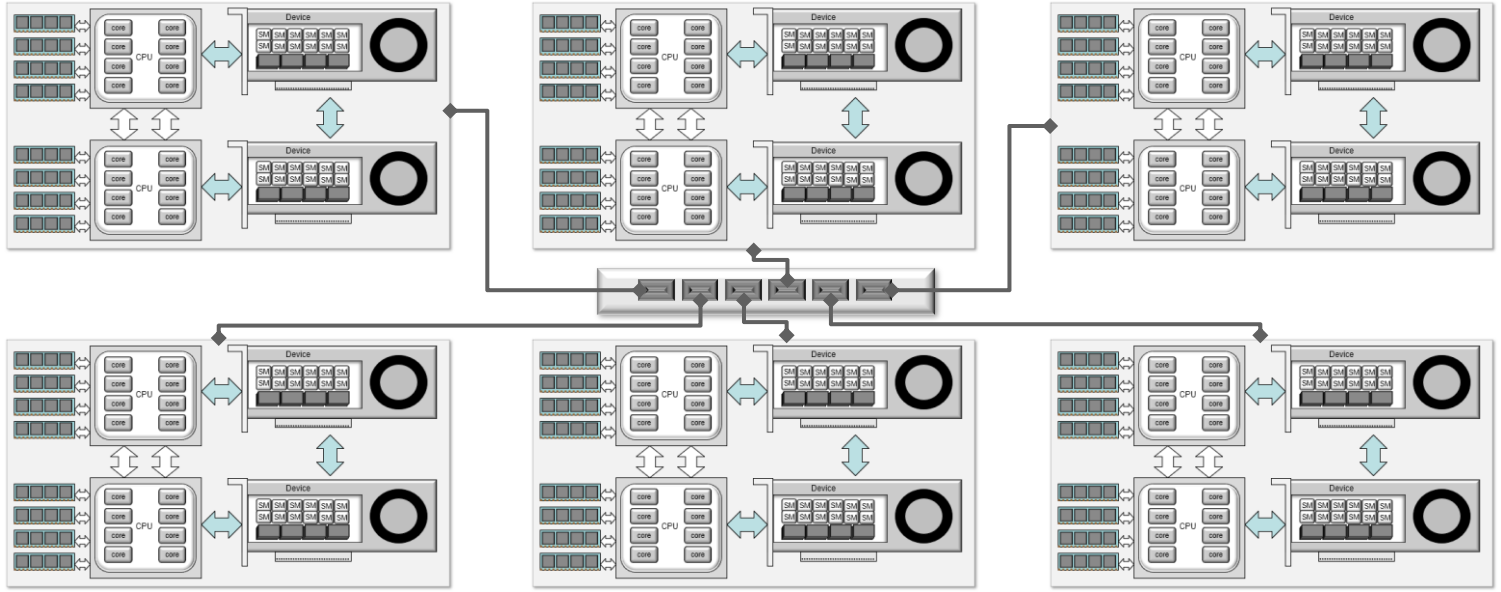
2 варианта комплексного обеда: дозвук и сверхзвук



БИЗНЕСЛАНЧ

Уровни параллелизма и средства разработки

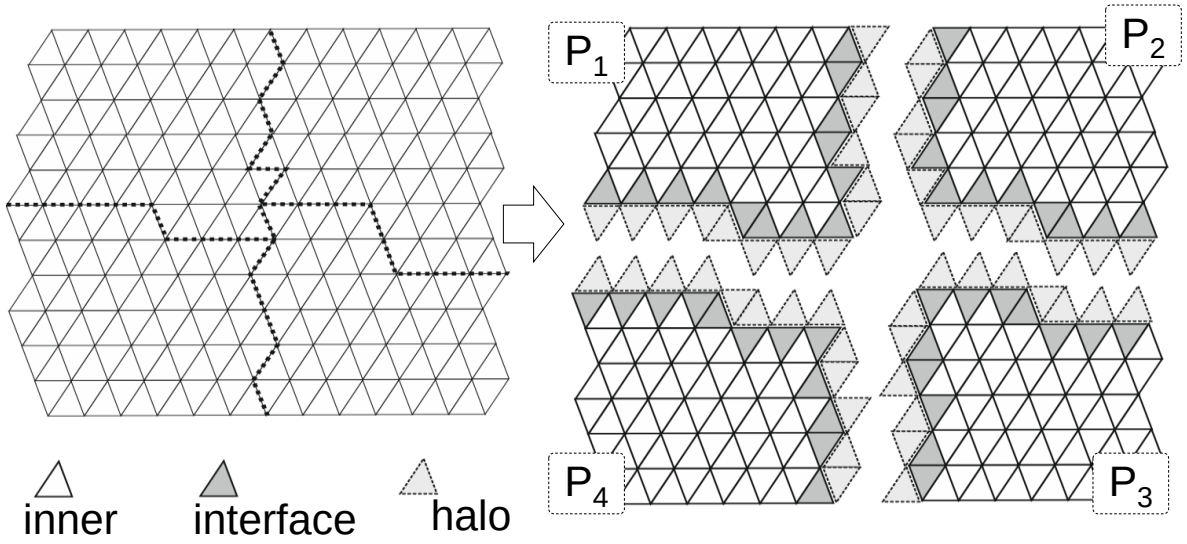


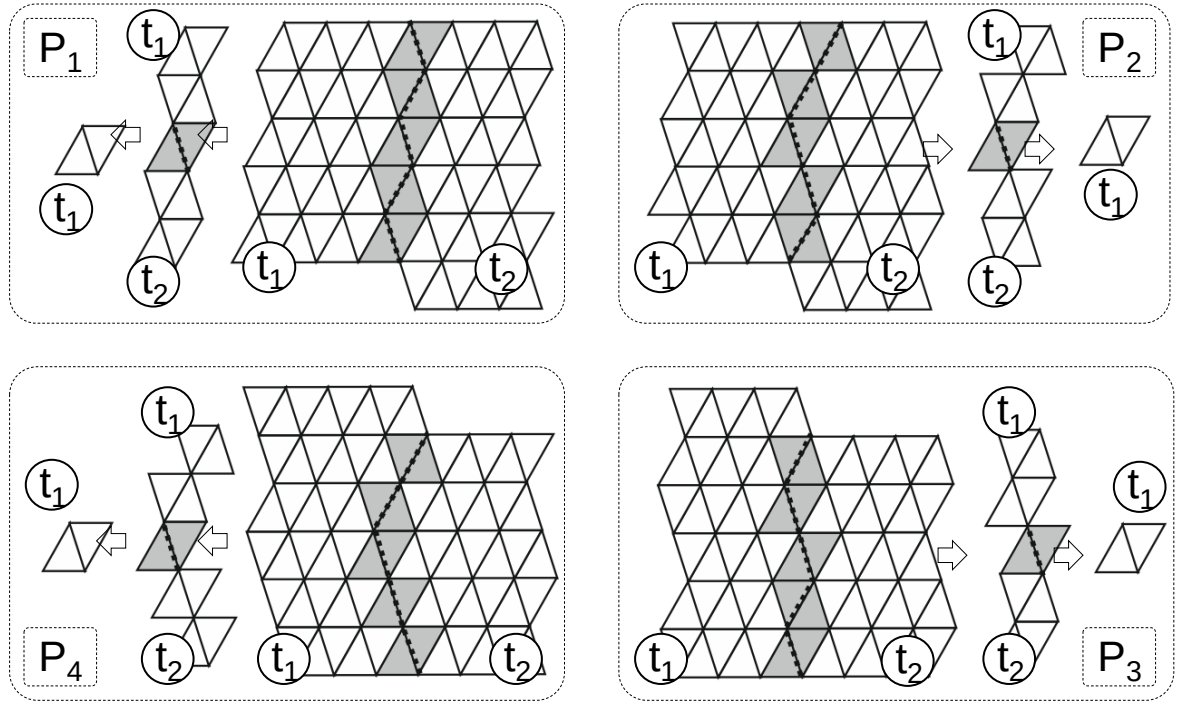


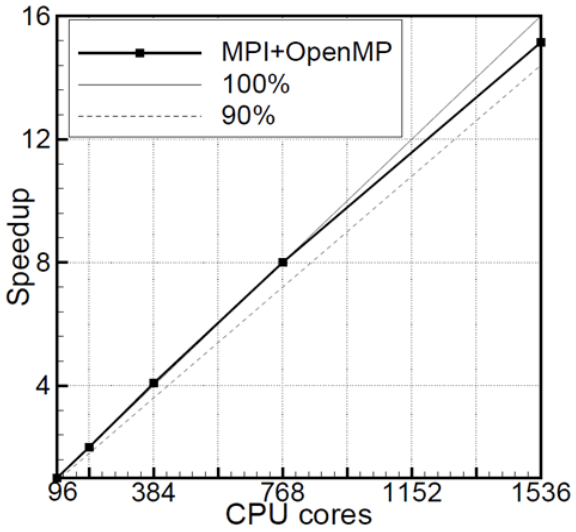
Многоуровневое распараллеливание

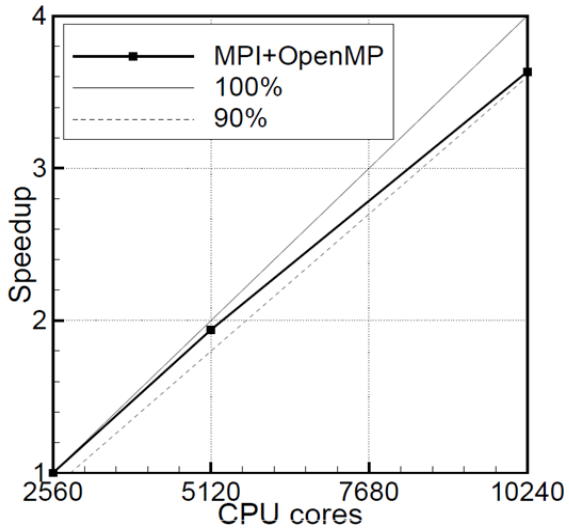
DM-MIMD: MPI – распараллеливание с распределенной памятью

SM-MIMD: OpenMP – распараллеливание с общей памятью







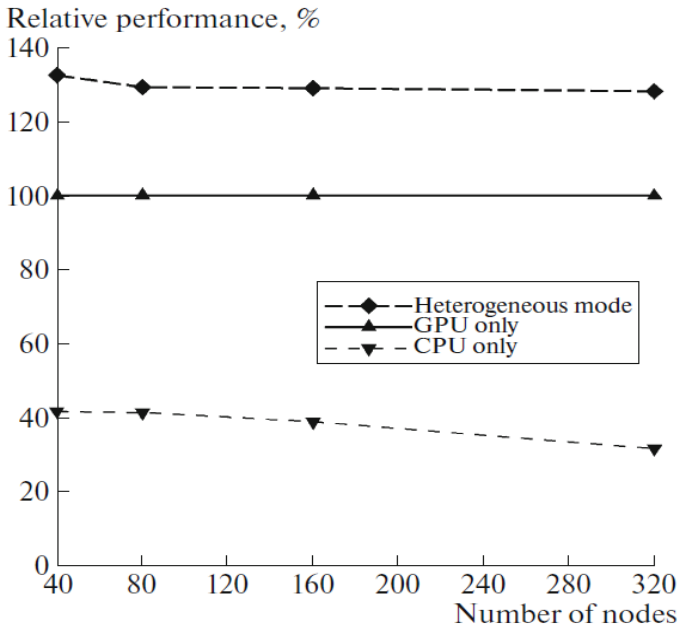


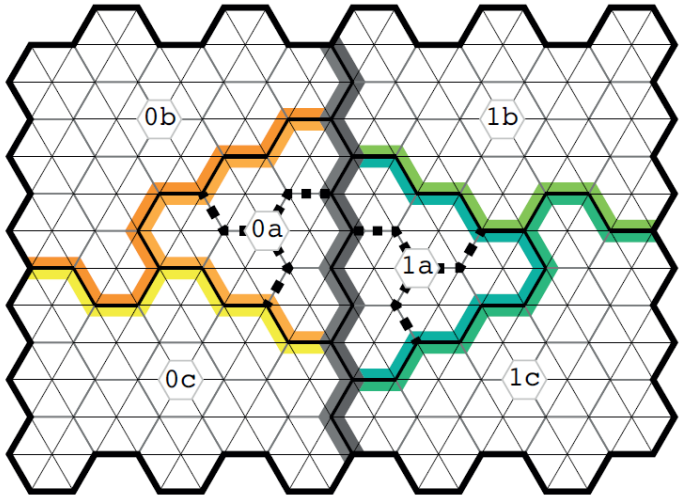
A. Gorobets. Parallel Algorithm of the NOISEtte Code for CFD and CAA Simulations. Lobachevskii J Math (2018) 39: 524. https://doi.org/10.1134/S1995080218040078

Гетерогенные вычисления: задел

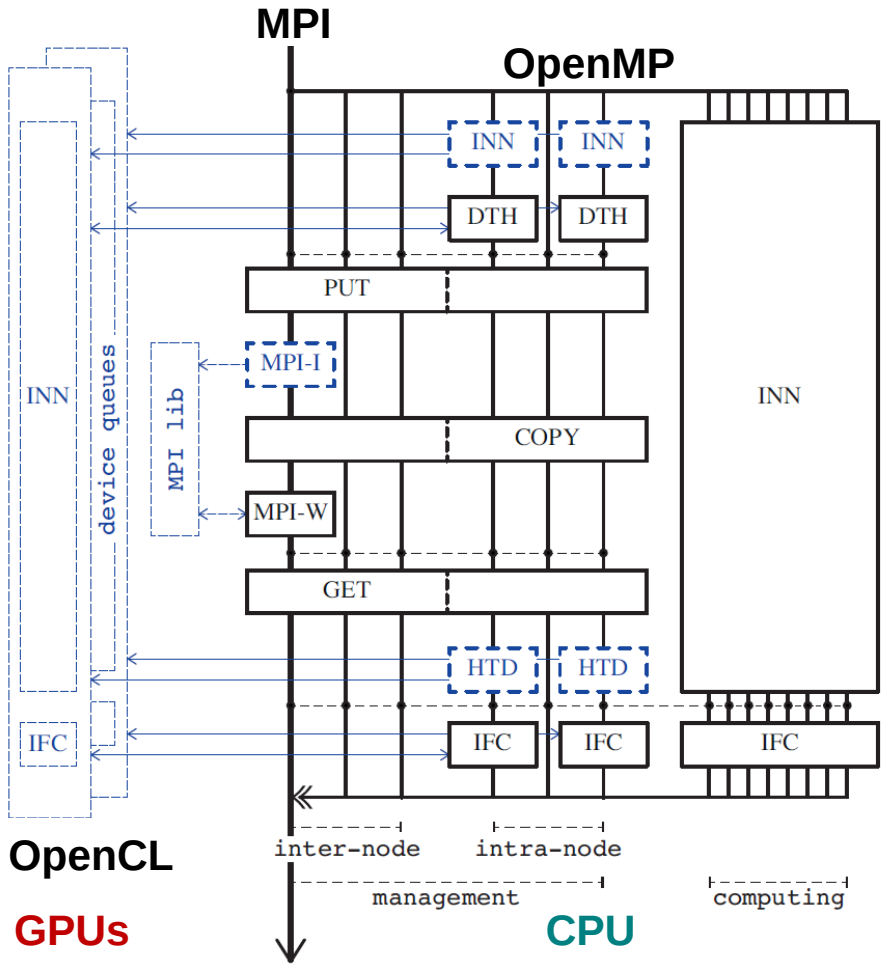
Коды HPC^2 и Тапир: 1 MPI процесс – 1 гибридный узел

несжимайка и сжимайка, неструктурированные сетки





Многоуровневая декомпозиция



Overlap – сокрытие обменов за вычислениями

NUMA-aware OpenMP распараллеливание

MPI + OpenMP + (CUDA и OpenCL)

A.Gorobets, S.Soukov, P.Bogdanov. Multilevel parallelization for simulating turbulent flows on most kinds of hybrid supercomputers. Computers and Fluids. Volume 173, Pages 171-177. 2018. https://doi.org/10.1016/j.compfluid.2018.03.011

S. A. Soukov, A. V. Gorobets. Heterogeneous Computing in Resource-Intensive CFD Simulations. https://doi.org/10.1134/S1064562418060194

X.Álvarez-Farré, A. Gorobets F. X. Trias. A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers. Computers and Fluids. Vol 214, 2021, 104768 https://doi.org/10.1016/j.compfluid.2020.104768

X.Álvarez, A.Gorobets, F.X.Trias, R.Borrell, and G.Oyarzun. HPC^2 - a fully portable algebra-dominant framework for heterogeneous computing. Application to CFD. Computers and Fluids. Volume 173. Pages 285-292. 2018. https://doi.org/10.1016/j.compfluid.2018.01.034

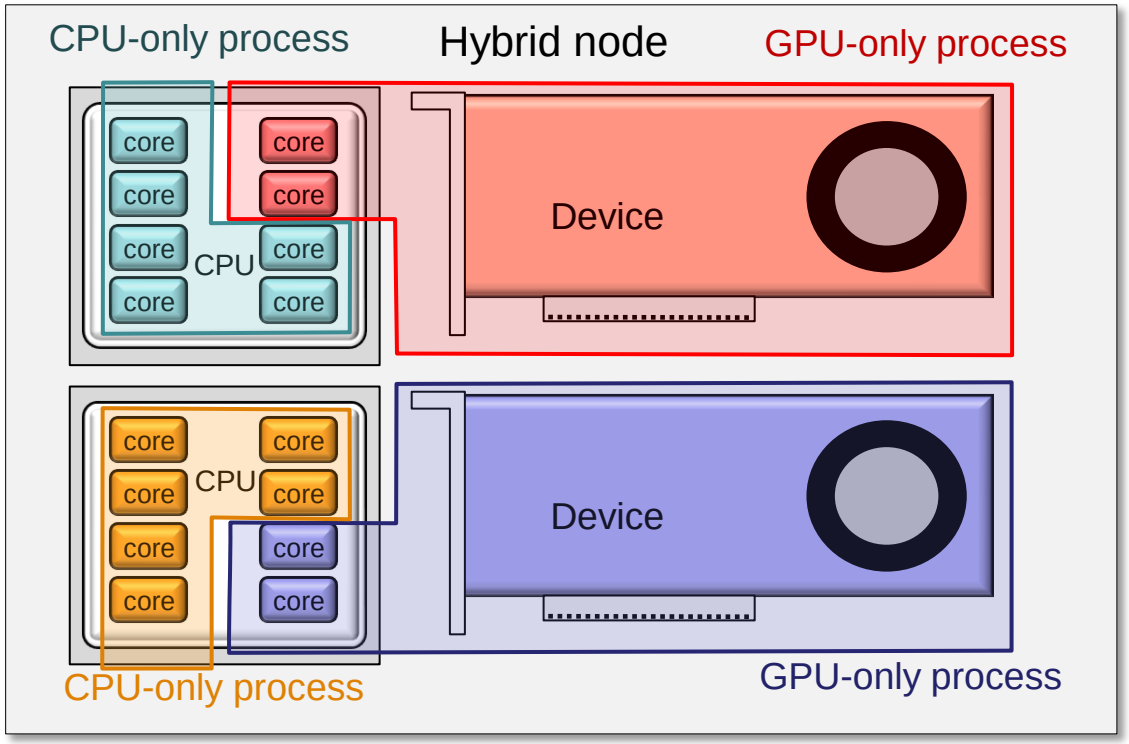
NOISETTE GPU

Минимизация сложности и страданий

- **Гетерогенщина CPU + GPU: 1 процесс – 1 девайс**
  процессы делятся по ролям CPU-only и GPU-only
  никакого усложнения внутренней программной логики
  небольшой апгрейд декомпозиции по балансировке

- **Большинство кернелов – наивный порт**
  высокоинтеллектуальным методом copy-paste
  только несколько кернелов специально адаптированы
  (градиенты, SpMV, редукция, …)

- **Кернел-код конфигурируется в рантайме**
  чем устраняется большинство ветвлений
  все, что можно, выносится на препроцессор и в константы

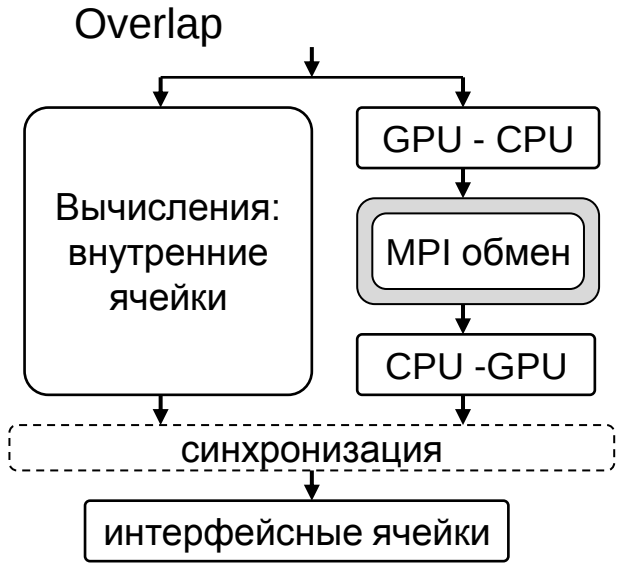


NOISETTE GPU

Бизнесланч (CPU) ⟺ **OpenCL версия (GPU или CPU)**

- **EBR-LO, EBR-3, EBR-5, EBR-WENO-LO, EBR-WENO-3, EBR-WENO-5**
- **Распадник: Roe, Roe+Turkel**
- **Упрощенная вязкость* – сильно быстрее и ест меньше памяти**
  при этом практике вообще не влияет на результат
- **Явный Р – К, неявный Ньютон + PBiCGSTAB солвер**
- **Турбуль: RANS, LES, (I)(D)DES**

- **Multi-GPU, multi-node – все как положено**
- **Overlap – одновременное выполнение вычислений и обменов**
  переупорядочивание ячеек: внутренние, интерфейс, гало



*A. Gorobets, P. Bakhvalov, A. Duben, P. Rodionov. Acceleration of NOISEtte Code for Scale-resolving Supercomputer Simulations of Turbulent Flows. LJM. Vol 41, No 8, pp. 1463–1474, 2020. https://doi.org/10.1134/S1995080220080077

Показательный пример

```
double *sum = 0.0;
for(int i=0; i<N; ++i) sum += X[i];
```

OpenMP

```
double sum = 0.0;
#pragma omp parallel for reduction(+:sum)
for(int i=0; i<N; ++i) sum += X[i];
```

MPI+OpenMP

```
double lsum = 0.0, gsum;
#pragma omp parallel for reduction(+:lsum)
for(int i=0; i<N; ++i) lsum += X[i];
MPI_Allreduce(&lsum, &gsum, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
```

А теперь то же самое на OpenCL

```
cl_kernel knlSUM;
knlSUM   = clCreateKernel(clProgram, "knlSUM", &clErr); // создаем кернел
if(clErr != CL_SUCCESS) crash("clCreateKernel knlSUM error: %d\n",clErr);
#define REDUCTION_LWS 256 // размер рабочей группы для суммы
#define REDUCTION_ITEM 8  // сколько элементов будет суммировать один work-item
int REDUCTION_BUFSIZE = ((N/REDUCTION_ITEM)/REDUCTION_LWS) +
                        ((N/REDUCTION_ITEM)%REDUCTION_LWS>0);
// буфер под частичные суммы рабочих групп
cl_mem clSum = clCreateBuffer(clContext, CL_MEM_READ_WRITE,
                              REDUCTION_BUFSIZE*sizeof(double), NULL, &clErr);
if(clErr != CL_SUCCESS) crash("clCreateBuffer clSum error %d\n",clErr);
// выставляем параметры запуска
size_t lws = REDUCTION_LWS; // размер рабочей группы
size_t gws = (N/REDUCTION_ITEM); // общее число заданий
if(gws%lws>0) gws += lws-gws%lws; // делаем кратное lws
// выставляем аргументы кернелу
clSetKernelArg(knlSUM, 0, sizeof(int), &N);
clSetKernelArg(knlSUM, 1, sizeof(cl_mem), &clX);
clSetKernelArg(knlSUM, 2, sizeof(cl_mem), &clSum);
// отправляем на исполнение
clErr= clEnqueueNDRangeKernel(clQueue, knlSUM, 1, NULL, &gws, &lws, 0, NULL, NULL);
if(clErr != CL_SUCCESS) crash("clEnqueueNDRangeKernel error %d\n",clErr);
clFinish(clQueue); // ждем завершения

// забираем результат с девайса
double *Sum = new double[REDUCTION_BUFSIZE];
clErr = clEnqueueReadBuffer(clQueue, clSum, CL_TRUE, 0,
                            REDUCTION_BUFSIZE*sizeof(double), Sum, 0, NULL, NULL);
if(clErr != CL_SUCCESS) crash("clEnqueueReadBuffer clSum error %d\n", clErr);
clFinish(clQueue);
double lsum = 0.0; // досуммируем результат на хосте
#pragma omp parallel for reduction(+:lsum)
for(int i=0; i<REDUCTION_BUFSIZE; ++i) lsum += Sum[i];
double gsum; // досуммируем по всей MPI группе, если у нас имеется MPI распараллеливание
MPI_Allreduce(&lsum, &gsum, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
```

```
double *sum = 0.0;
for(int i=0; i<N; ++i) sum += X[i];
```

```
#define REDUCTION_LWS 256 // пусть размер рабочей группы - фиксирован
__kernel void knlSUM(int n, __global double* x, __global double *res){
    const int gid = get_global_id(0); // глобальный номер work-item
    const int lid = get_local_id(0); // номер внутри рабочей группы
    __local double sdata[REDUCTION_LWS]; // сюда будем собирать сумму группы.
    sdata[lid] = gid < N ? x[gid] : 0.0;
    barrier(CLK_LOCAL_MEM_FENCE); // барьер, чтобы sdata весь был заполнен

    // делаем сборку сдваиванием
    if(lid <128) sdata[lid] += sdata[lid +128]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid < 64) sdata[lid] += sdata[lid + 64]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid < 32) sdata[lid] += sdata[lid + 32]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid < 16) sdata[lid] += sdata[lid + 16]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid <  8) sdata[lid] += sdata[lid +  8]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid <  4) sdata[lid] += sdata[lid +  4]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid <  2) sdata[lid] += sdata[lid +  2]; barrier(CLK_LOCAL_MEM_FENCE);
    if(lid <  1) sdata[lid] += sdata[lid +  1]; barrier(CLK_LOCAL_MEM_FENCE);

    // записываем результат группы
    if(lid == 0) res[get_group_id(0)] = sdata[0];
}
```

Версии кода

NOISETTE – PLAYGROUND: сначала новые вещи реализуются тут

NOISETTE – САРАЙ: потом полезные вещи внедряются сюда

это одна версия, разные конфиги

NOISETTE – БИЗНЕСЛАНЧ: потом отлаженные и нужные на практике кочуют сюда

NOISETTE – OpenCL: потом реально нужные в ресурсоемких задачах портируются на графон

- **QA процедура контролирует согласованность версий**
- **работать с тремя версиями – это плохо**
- **Бизнесланч и OpenCL версии очень маленькие – это хорошо**
  PLAYGROUND ~**200K** строк, kernel-код ~**4K** строк (пока)

Большой простор для ошибок и расхождений. Поэтому – тотальный контроль и диагностика.
Но код не запустится на графоне, пока не выполнит полный самотест
CPU vs GPU, kernel-to-kernel, сравнение полных шагов по времени, …
для данной конкретной задачи

Использование OpenCL в продакшене

GPU vs CPU – сколько ядер можно снять с графона
“все включено”: EBR5, неявка, турбуль, mixed accuracy

Соотношение по GB/s CPU vs GPU: **1:7.5** (очень примерно)

- **Обтекание какой-то сферы, сетка 1M ячеек**
  1 GPU = 119 CPU cores, 1:7.4
  Узел 4 GPU = 374 CPU cores

- **Обтекание какой-то лопасти, сетка 2M ячеек**
  1 GPU = 140 CPU cores, 1:8.7
  Node 4 GPU = 491 CPU cores

- **Обтекание модельной решетки турбины T106C, 80M ячеек**
  3 узла, 12 GPU = ~1500 CPU ядер

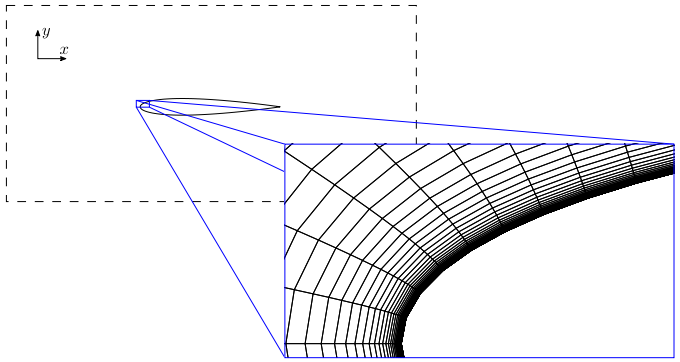
- **С одной Вольты снимаем эквивалент 100 – 150 ядер, что согласуется с соотношением бэндвиса**

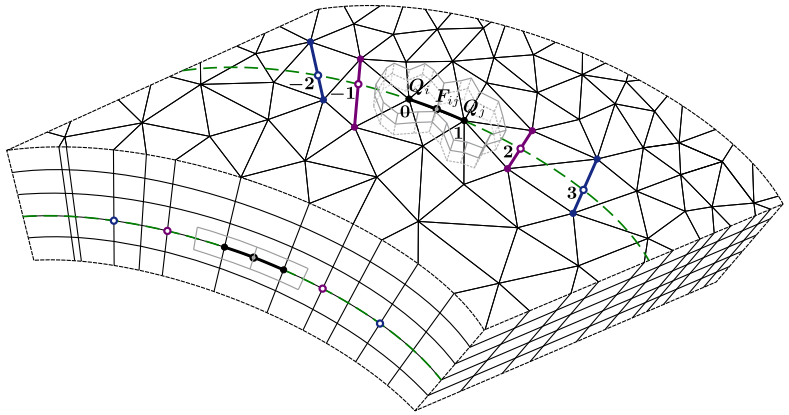
- **Чем больше сетка, тем больше ускорение**

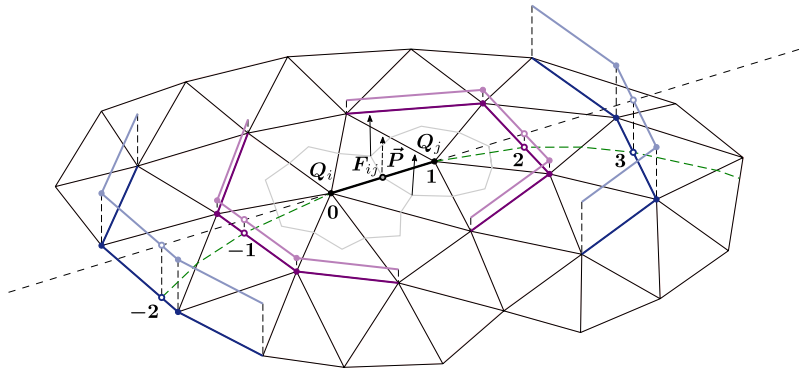
K-60 hybrid cluster
Узлы:
2 16C Intel Xeon **Gold 6142** (120 GB/s)
4 GPU NVIDIA **V100** (32 GB, 900GB/s)
Сеть: InfiniBand FDR

Схема EBR “полуструкт”

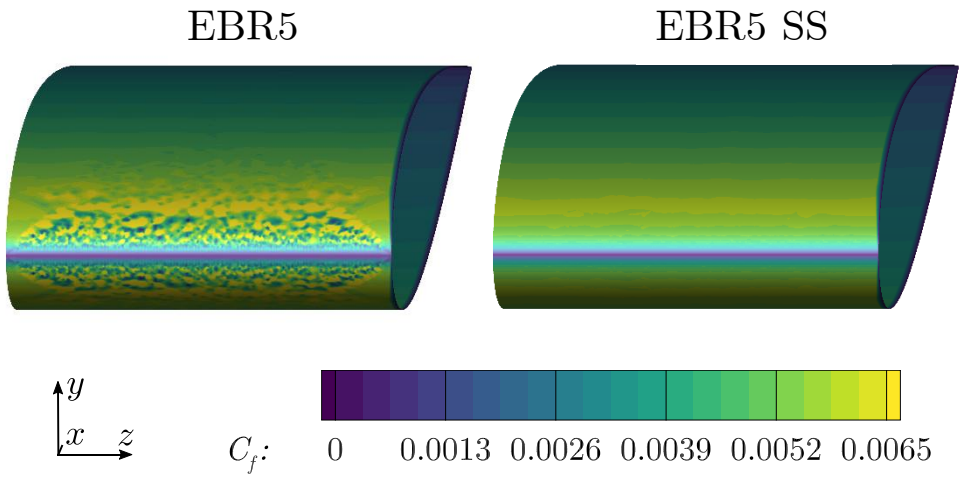
- **EBR Semi-Structured – модификация EBR для полуструктурированных (призматических) сеток, использующая в области погранслоя криволинейные реконструкции**

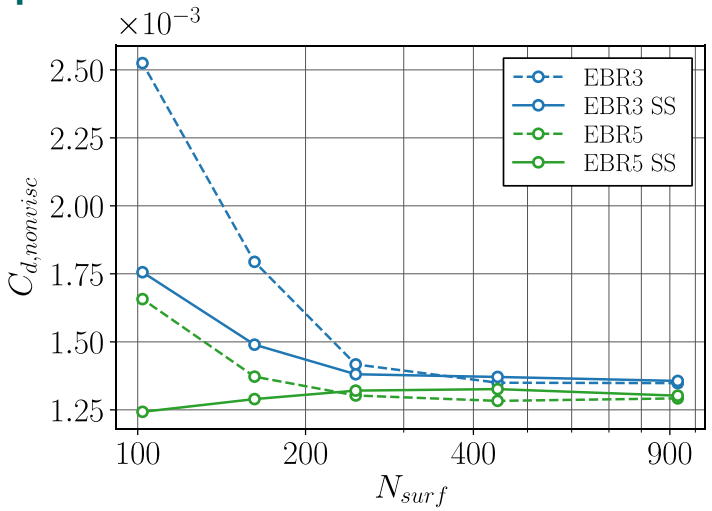


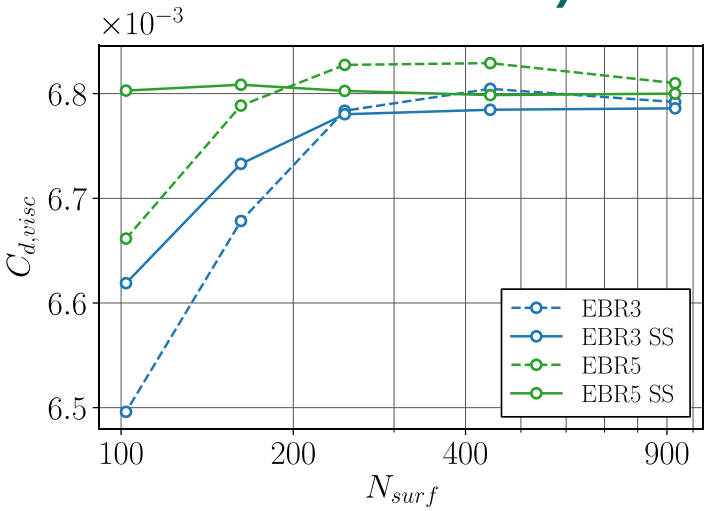




- **Результаты тестирования EBR SS на задаче об обтекании NACA0012 (AOA=0° Re=6·10⁶ M=0.15):**

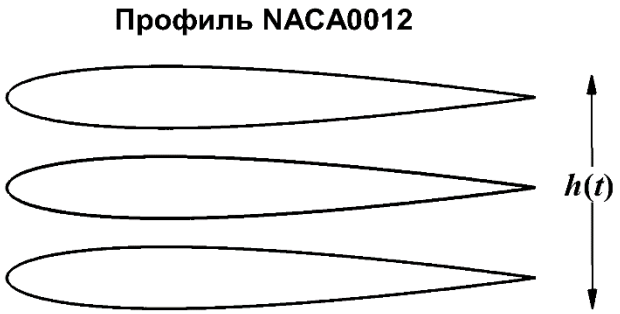






Грант РФФИ "Рёберно-ориентированные схемы в областях пограничных слоёв и скользящих интерфейсов неструктурированных сеток", №18-01-00445

Метод погруженных границ & адаптация сетки



Постановка задачи:

- вертикальное движение профиля по закону $h(t) = -h_0 \cos(k\,t)$

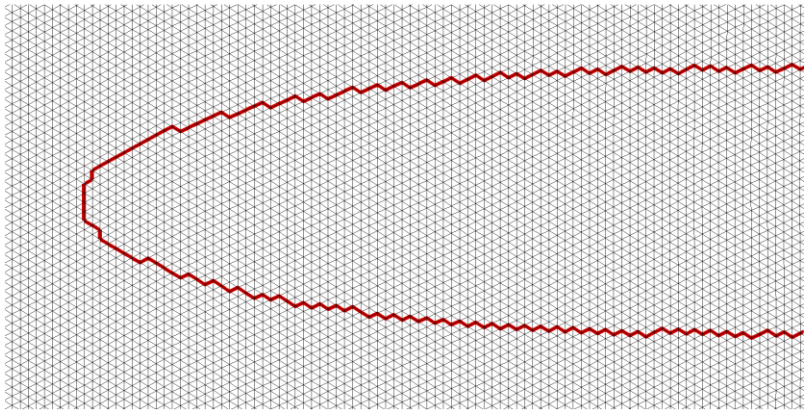
$h_0 = 0.175, \quad kh_0 = 0.3 \ldots 2.0$

- внешний поток M = 0.1, Re = 20 000

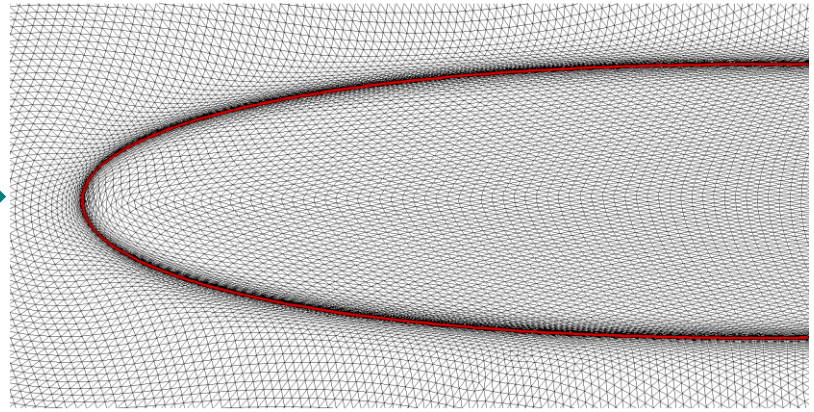
Адаптация при движении профиля

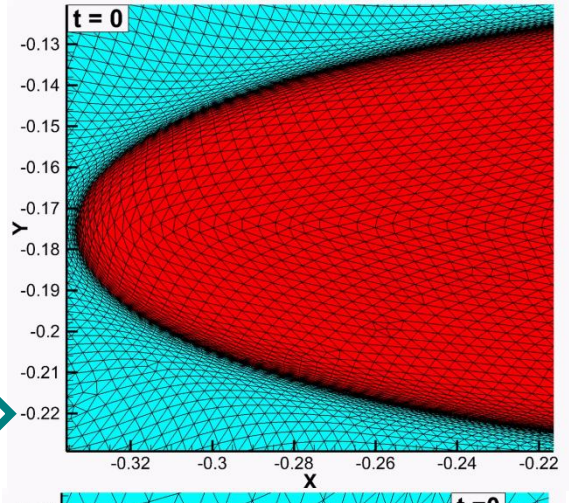
Этапы адаптации сетки с сохранением начальной топологии*):

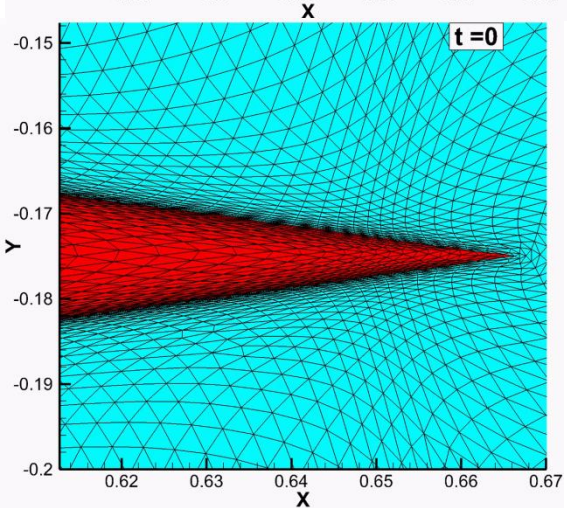
Начальная сетка



Адаптация перед первым шагом по времени t = 0







Отслеживание границы профиля: интерполяционный алгоритм Сукова С.А. с использованием ортогональных адаптивных решёток**)

*) Garanzha V., Kudryavtseva L. (2018) Hypoelastic Stabilization of Variational Algorithm for Construction of Moving Deforming Meshes // Communications in Computer and Information Science 974: 497-511.

**) Суков С.А. (2020) Комбинированный алгоритм вычисления расстояния со знаком для задач численного моделирования физических процессов и визуализации движения твердых тел // Научная визуализация

Метод погруженных границ & адаптация сетки

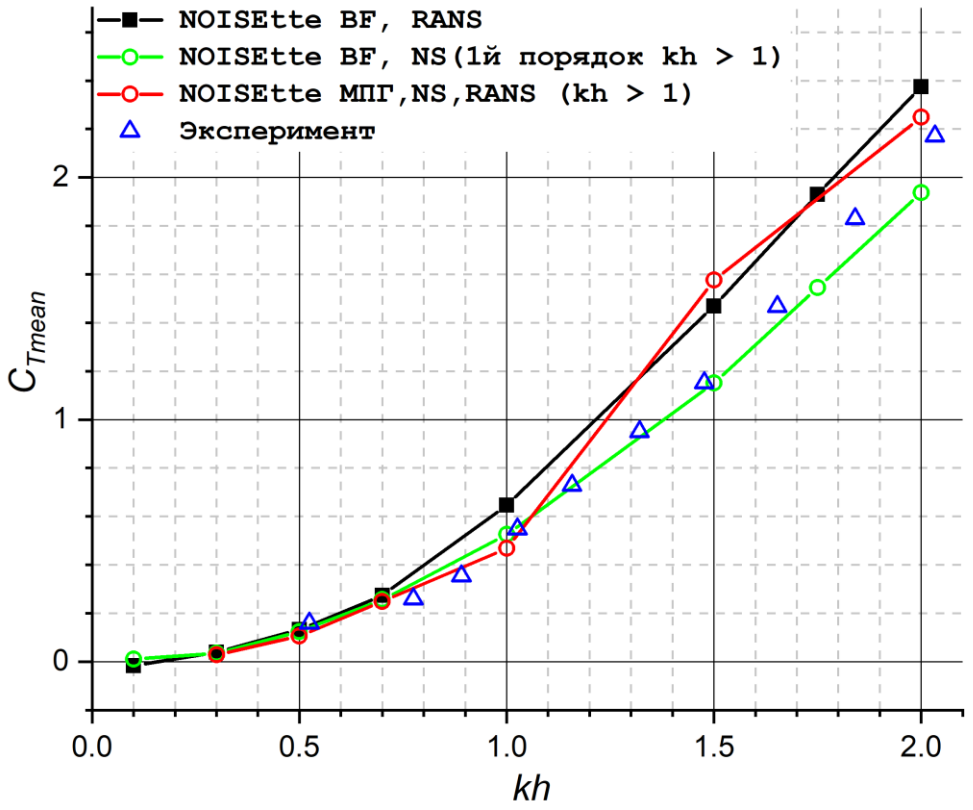
Расчётные модели: уравнения NS и уравнения RANS (SA)

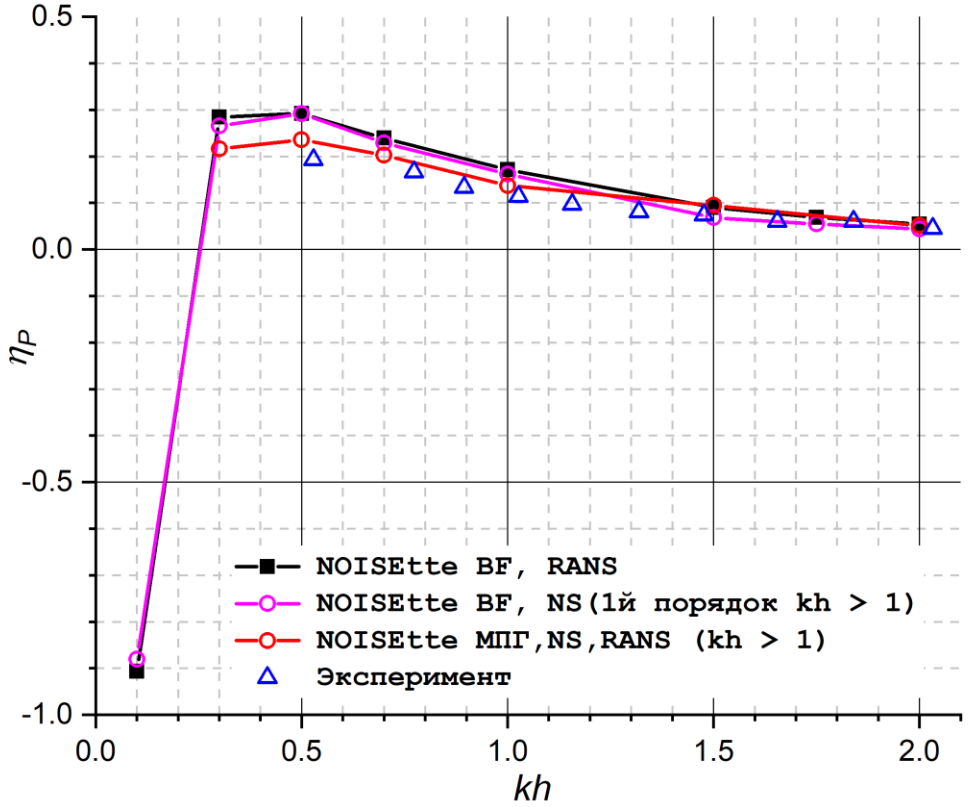
Интегральные характеристики течения:

коэффициент силы тяги — $C_T = -C_D$, тяговый КПД — $\eta_P = -\langle C_L(t)\, dh(t)/dt \rangle / C_T$

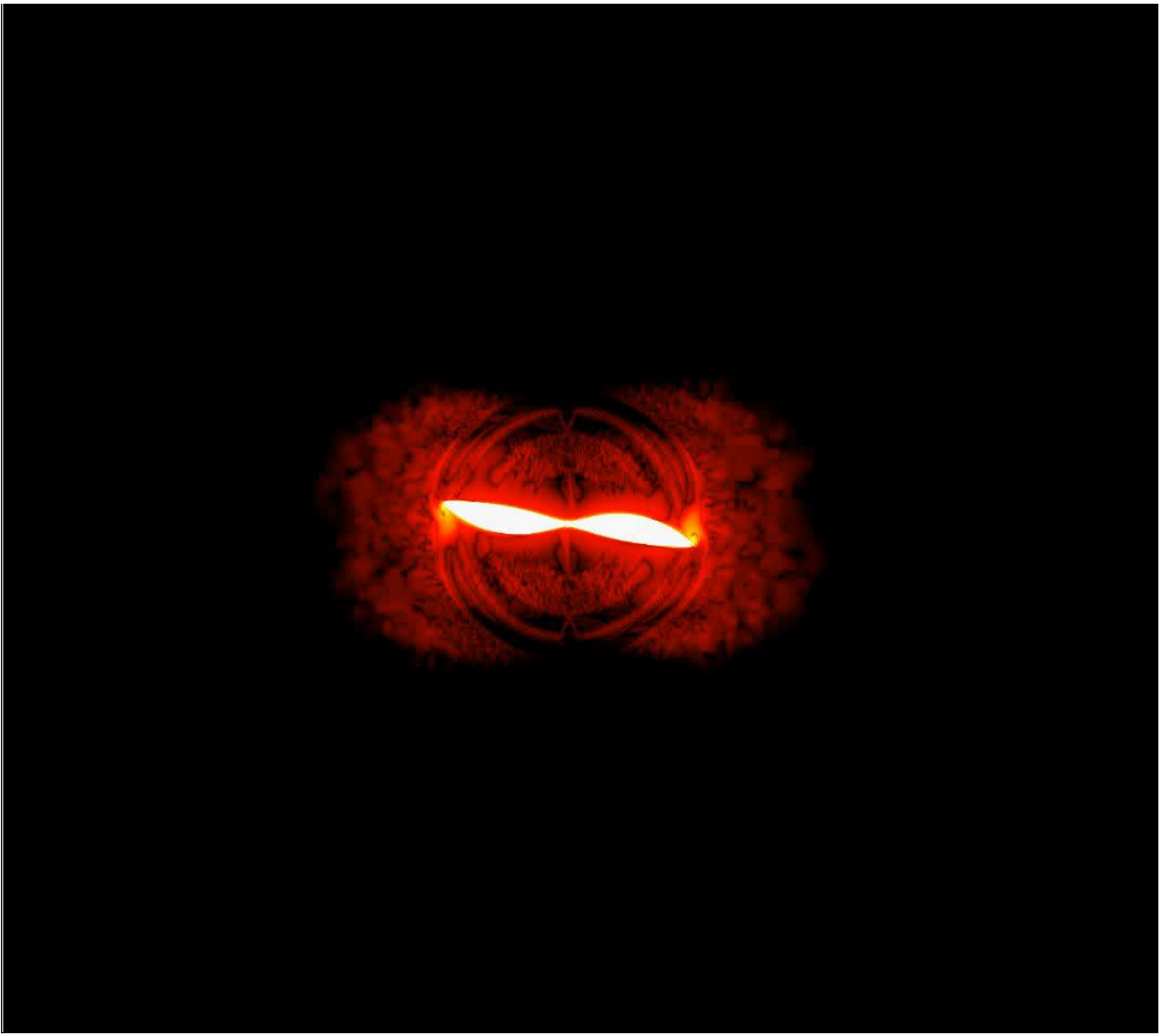
C_D — коэффициент сопротивления, C_L — коэффициент подъёмной силы, < > — осреднение по времени

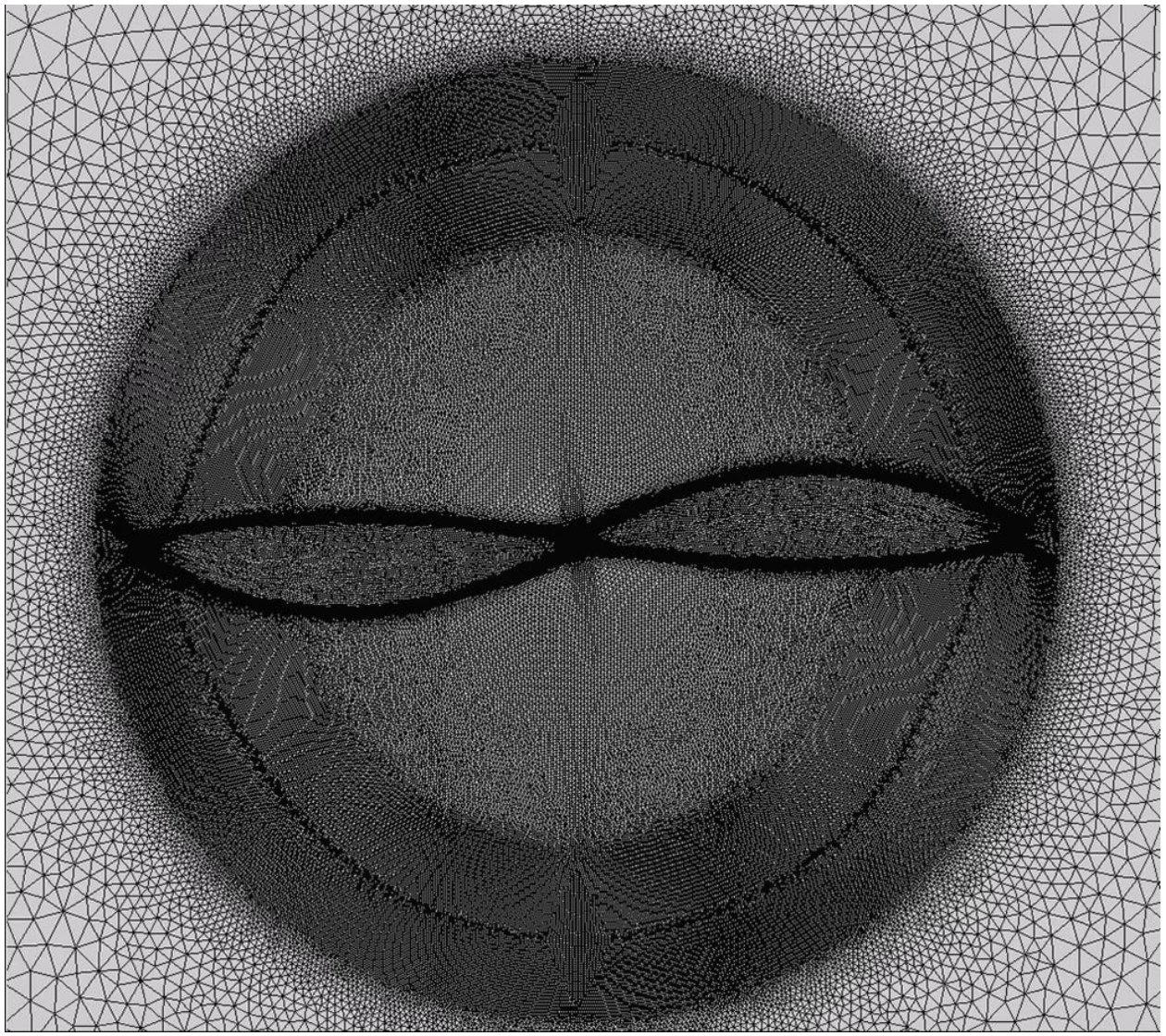
Сравнение: с расчётами уравнений NS в неинерциальной системе координат на сетке согласованной с границей (BF) и экспериментальными данными





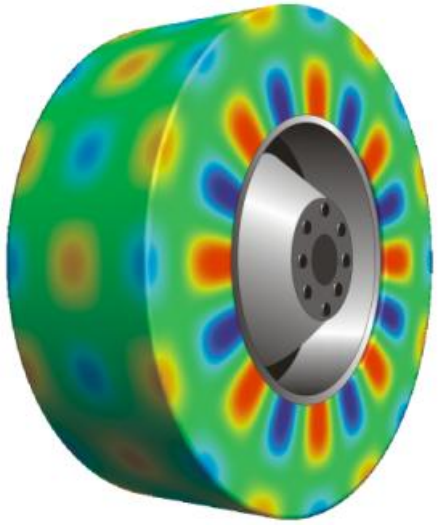
Метод погруженных границ & адаптация сетки





РНФ 16-11-10350

ColESo – библиотека точных решений



Collection of Exact Solutions
П. А. Бахвалов

ES->PointValue(Time, Coor, Vars);
время, координаты, блок переменных

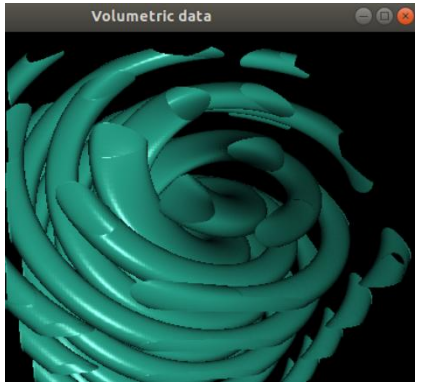
O(30) точных решений

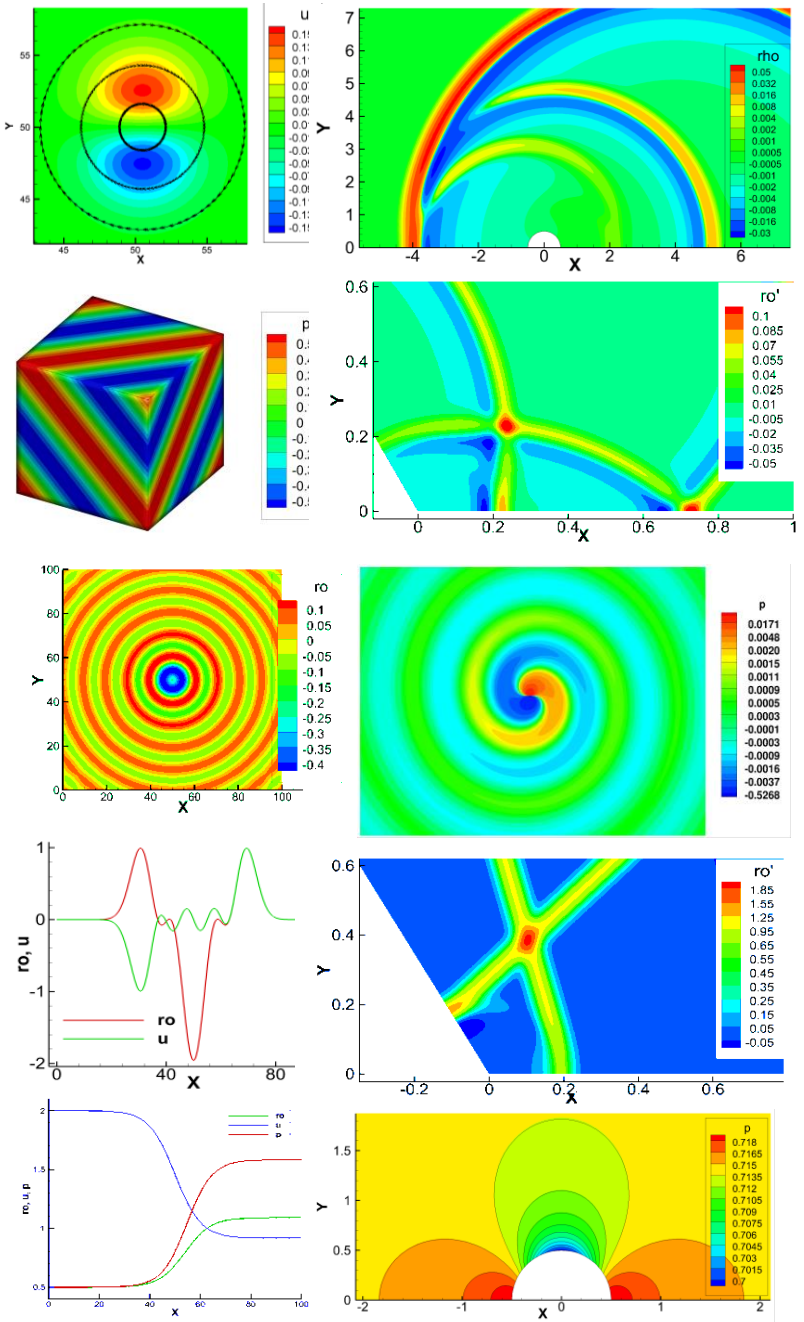
https://github.com/bahvalo/ColESo

- **Акустика в свободном пространстве**
- **Акустика в канале при наличии вязкости**
- **Задачи дифракции**
- **Низкорейнольдсовые задачи**
  (структура УВ, течение Куэтта, …)
- **Плоские вихри**
- **Одномерные ударно-волновые задачи**

…

первый пользователь ☺
визуализаторы из СПбПУ
система **S3VS**





Заключение

- **Бизнесланч и прочистка кода ~1.5X ускорение**
  за счет слияния конвекции и вязкости
  за счет устранения ветвлений и вызовов функций на нижнем уровне

- **Смешанная точности ~1.7X ускорение**

- **OpenCL версия – удои более 100 ядер с девайса**
  сложнее поддерживать код и вносить изменения
  но кернел-код (пока) маленький ~4K строк
  много мучений, тягот, лишений и возни, но хоть заметен результат

- **Гетерогенщина CPU+GPU очень легко при подходе 1 процесс – 1 девайс**
  нет сложной программной логики, никаких вложенных OpenMP-шечек
  несколько срок кода для раздачи весов на декомпозиции

- **Остается потенциал для дальнейшего ускорения 20 – 30%**
  не на все пока хватило сил и времени

Работы по развитию кода для использования современных гибридных суперкомпьютеров поддержаны **РНФ, проект 19-11-00299** “Сверхмасштабируемые параллельные алгоритмы и гетерогенные вычисления для вихреразрешающего моделирования задач гидродинамики, аэродинамики и аэроакустики”