



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО

ЦЕНТР КОМПЕТЕНЦИЙ ONEAPI В ННГУ

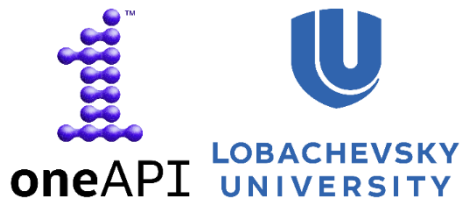
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО
ЦЕНТР КОМПЕТЕНЦИЙ oneAPI в ННГУ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ

***ВВЕДЕНИЕ В АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ
И ОПТИМИЗАЦИЮ ПРОГРАММ***

**Оптимизация структур данных
при работе с разреженными матрицами**



Мееров И.Б. при участии
Лебедева С.А., Пировой А.Ю.

Содержание

- ❑ Понятие разреженной матрицы
- ❑ Задачи алгебры разреженных матриц
- ❑ Форматы хранения разреженных матриц
- ❑ Заключение

1. ПОНЯТИЕ РАЗРЕЖЕННОЙ МАТРИЦЫ

1. Понятие разреженной матрицы

1.1 Алгебра разреженных матриц

- ❑ **Алгебра разреженных матриц (Sparse algebra)** – важный раздел вычислительной математики, имеющий очевидное практическое применение.
- ❑ Разреженные матрицы возникают естественным образом при постановке и решении задач из разных научных и инженерных областей:
 - При постановке и решении оптимизационных задач.
 - При численном решении дифференциальных уравнений в частных производных.
 - В теории графов.
 - При моделировании сетей.

1. Понятие разреженной матрицы

1.2. Варианты определения

- ❑ Встречаются разные формулировки:
 - Разреженной называют матрицу, имеющую малый процент ненулевых элементов.
 - Матрица размера $N \times N$ называется разреженной, если количество ее ненулевых элементов есть $O(N)$.
 - Известны и другие определения.
- ❑ Приведенные варианты являются не вполне точными в математическом смысле.
- ❑ **Вопрос:** какое определение вам кажется наиболее логичным?

1. Понятие разреженной матрицы

1.3. Определение

- ❑ На практике классификация матрицы зависит не только от количества ненулевых элементов, но и от:
 - размера матрицы,
 - распределения ненулевых элементов,
 - архитектуры конкретной вычислительной системы и используемых алгоритмов

- ❑ Матрицу имеет смысл представлять в специальном *разреженном формате* в том случае, если это позволяет экономить память/ускорять обработку на конкретной вычислительной системе при решении определенного класса задач

1. Понятие разреженной матрицы

1.4. Проблема выбора

- ❑ В ряде случаев матрица имеет регулярную структуру (например, ленточная матрица), что позволяет разработать специализированную структуру хранения.
- ❑ Иногда размерность матрицы такова, что ее плотное представление попросту не укладывается в память.
- ❑ Если и разреженное, и плотное представление допустимы, а количество ненулевых элементов невелико, необходимо проанализировать, какая структура хранения будет более эффективной в рамках используемых алгоритмов и конкретной архитектуры.
- ❑ В зависимости от результатов этого анализа можно рассматривать матрицу либо как разреженную, либо как плотную.

1. Понятие разреженной матрицы

1.5. Примеры разреженных матриц

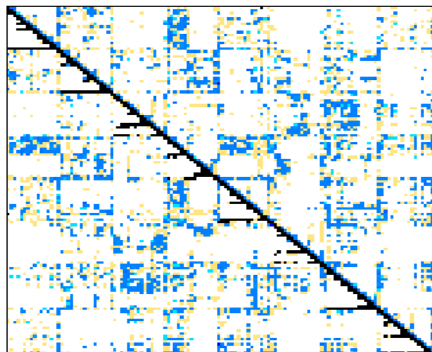
- ❑ Матрицы из коллекции университета Флориды (в настоящее время – коллекция SuiteSparse)



<https://sparse.tamu.edu/>

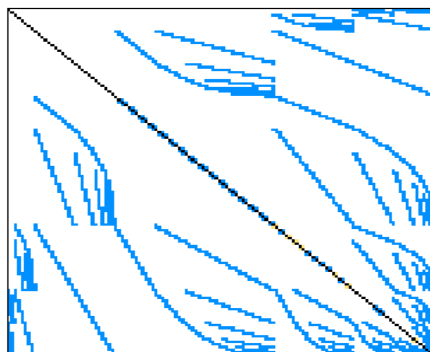
ASIC_680ks

Моделирование схем



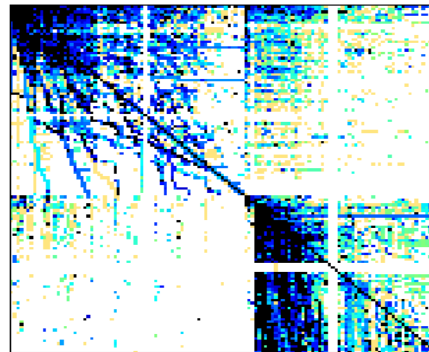
parabolic_fem

Метод конечных элементов



Webbase_1M

Матрица web-связей



2. АЛГОРИТМЫ И ПРОГРАММНЫЕ СРЕДСТВА

2. Алгоритмы и программные средства

2.1. Основные алгоритмы

- ❑ Базовые операции:
 - Транспонирование матрицы
 - Умножение матрицы на плотный вектор
 - Умножение двух разреженных матриц
- ❑ Решение систем линейных уравнений $Ax = b$:
 - Прямые методы (LL^T , LU , LDL^T , QR разложения),
 - Итерационные методы (CG, GMRES, Чебышева и др.)
- ❑ Решение задачи на собственные значения, задачи наименьших квадратов
 - Итерационные методы

2. Алгоритмы и программные средства

2.2. Библиотеки

- ❑ Коммерческий решатель Intel MKL для общей памяти, распределенной памяти: Sparse BLAS, MKL PARDISO, FEAST
- ❑ Академические прямые решатели:
 - MUMPS, SuiteSparse, SuperLU, SPRAL, PaStiX...
- ❑ Академические итерационные решатели:
 - PETSc, SOL, PARALUTION, Lis, HYPRE...
- ❑ Академические решатели задачи на собственные значения:
 - SLEPc, BLOPEX, FEAST, PRIMME...
- ❑ Актуальный обзор ПО: <http://www.netlib.org/utk/people/JackDongarra/la-sw.html>

3. ФОРМАТЫ ХРАНЕНИЯ

3. Форматы хранения

3.1. Сложность задачи

- ❑ Эффективные методы хранения и обработки разреженных матриц на протяжении последних десятилетий вызывают интерес у широкого круга исследователей.
- ❑ Для учета разреженной структуры приходится существенно усложнять как методы хранения, так и алгоритмы обработки.
- ❑ Многие тривиальные с точки зрения программирования алгоритмы в разреженном случае становятся весьма сложными.
- ❑ Для оптимизации производительности приходится разрабатывать нетривиальные алгоритмы и использовать возможности современной аппаратуры.

3. Форматы хранения

3.2. Координатный формат (COO)

- ❑ Элементы матрицы и ее структура хранятся в трех массивах, содержащих ненулевые значения **Values**, их номера строк **Row** и столбцов **Col**.
- ❑ Объем необходимой памяти (M):
 - Плотное представление: $M = 8 N^2$ байт.
 - В координатном формате: $M = 8 NZ + 4 NZ + 4 NZ = 16 NZ$ байт (предполагается, что $N < 2^{32}$; в противном случае необходимо использовать 64-битный вариант беззнакового целого для хранения индексов).

3. Форматы хранения

3.2. Координатный формат (COO)

Пример:

A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Value

0	0	1	1	3	3	5	5	5
---	---	---	---	---	---	---	---	---

Row

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

Col

3. Форматы хранения

3.2. Координатный формат (COO)

- ❑ Координатный формат можно использовать
 - как в упорядоченном виде, когда в массиве **Value** хранятся элементы матрицы построчно (например, слева направо, сверху вниз),
 - так и в неупорядоченном.
- ❑ Неупорядоченное представление существенно упрощает операции вставки/удаления новых элементов, но приводит к переборным поискам.
- ❑ Упорядоченный вариант позволяет быстрее находить все элементы нужной строки (столбца), но приводит к перепаковкам при вставках/удалениях элементов.

3. Форматы хранения

3.3. Работа с координатным форматом

□ Пример: $y = Ax$.

```
// порядок элементов не важен
for (i = 0; i < NZ; i++)
{
    y[Col[i]] += Value[i] * x[Col[i]];
}
```

3. Форматы хранения

3.3. Работа с координатным форматом

□ Пример: $y = Ax$.

```
// элементы упорядочены по строкам
j = 0;
for (i = 0; i < N; i++)
{
    sum = 0;
    while ((j < NZ) && (Row[j] == i))
    {
        sum += Value[j] * x[Col[j]]; j++;
    }
    y[i] = sum;
}
```

3. Форматы хранения

3.3. Формат CRS (CSR)

- ❑ Формат хранения **CSR** (Compressed Sparse Rows) или **CRS** (Compressed Row Storage), призван устранить некоторые недоработки координатного представления.
- ❑ Для хранения ненулевых элементов используются три массива:
 - Массив **Values** хранит значения ненулевых элементов построчно (строки рассматриваются по порядку сверху вниз),
 - массив **Col** хранит номера столбцов для каждого элемента,
 - массив **RowIndex** хранит индексы начала элементов каждой строки в массиве **Col**.

3. Форматы хранения

3.3. Формат CRS (CSR)

Пример:

A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

1	2	3	4	8	5	7	1	6
---	---	---	---	---	---	---	---	---

Value

0	4	2	3	3	5	1	2	5
---	---	---	---	---	---	---	---	---

Col

0	2	4	4	6	6	9
---	---	---	---	---	---	---

RowIndex

3. Форматы хранения

3.3. Формат CRS (CSR)

- ❑ Количество элементов массива **RowIndex** равно $N + 1$.
- ❑ i -ый элемент массива **RowIndex** указывает на начало i -ой строки.
- ❑ Элементы строки i в массиве **Value** находятся по индексам от **RowIndex[i]** до **RowIndex[i + 1] – 1** включительно.
- ❑ Т.о. обрабатывается случай пустых строк, а также добавляется «лишний» элемент в массив **RowIndex** – устраняется особенность при доступе к элементам последней строки. **RowIndex[N] = NZ**.

3. Форматы хранения

3.3. Формат CRS (CSR)

- ❑ Объем необходимой памяти.
 - Плотное представление: $M = 8 N^2$ байт.
 - В координатном формате: $M = 16 NZ$ байт.
 - В формате CRS: $M = 8 NZ + 4 NZ + 4 (N + 1) = 12 NZ + 4 N + 4$.
- ❑ В часто встречающемся случае, когда $N + 1 < NZ$, данный формат является более эффективным, чем координатный, с точки зрения объема используемой памяти.

3. Форматы хранения

3.3. Формат CRS (CSR)

□ Пример: $y = Ax$.

```
// Цикл по строкам матрицы A
for (i = 0; i < N; i++)
{
    // Вычисляем i-ю компоненту вектора y
    sum = 0;
    j1 = RowIndex[i];
    j2 = RowIndex[i + 1];
    for (j = j1; j < j2; j++)
        sum = sum + Value[j] * x[Col[j]];
    y[i] = sum;
}
```


3. Форматы хранения

3.4. Формат CRS (CSR) – модификации

- ❑ Возможна индексация как с нуля, так и с единицы (C/Fortran).
- ❑ В базовом варианте CRS строки рассматриваются по порядку, но элементы внутри каждой строки могут быть как упорядочены по номеру столбца, так и не упорядочены.
- ❑ Модификация CRS с четырьмя массивами. Четвертый массив хранит индексы элементов массива **Col**, соответствующих последнему элементу строки. Строки могут не быть упорядочены – быстрая перестановка.
- ❑ Блочный CRS (BCRS) рассматривает матрицу как совокупность плотных блоков (обычно малого размера)

3. Форматы хранения

3.4. Формат CRS (CSR) – модификации

- ❑ В некоторых источниках рассмотренный формат упоминается как Yale format, в некоторых – как AIJ.
- ❑ Иногда применяется модификация, состоящая в том, что массивы хранятся в памяти в другом порядке.
- ❑ Все, что было описано выше, может быть с равным успехом применено не к строкам, а к столбцам. В итоге получается столбцовый формат CSC (CCS) вместо строчного CSR (CRS).
- ❑ Для ряда алгоритмов более удобен строчный формат, для ряда – столбцовый.

3. Форматы хранения

3.5. Формат CCS

□ Пример:

A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

1	7	3	1	4	8	2	5	6
---	---	---	---	---	---	---	---	---

Value

0	5	1	5	1	3	0	3	5
---	---	---	---	---	---	---	---	---

Row

0	1	2	4	6	7	9
---	---	---	---	---	---	---

ColIndex

3. Форматы хранения

3.6. Работа с форматом CCS

□ Пример: $y = Ax$.

```
for (i = 0; i < N; i++)
    y[i] = 0.0;

// Цикл по столбцам матрицы A
for (i = 0; i < N; i++)
{
    j1 = ColIndex[i];
    j2 = ColIndex[i + 1];
    for (j = j1; j < j2; j++)
        y[Row[j]] += Value[j] * x[i];
}
```

3. Форматы хранения

3.7. Проблемы стандартных форматов

- ❑ Низкая арифметическая интенсивность
 - ❑ Нерегулярный доступ к памяти в большинстве базовых алгоритмов
 - ❑ Проблемы с векторизацией расчетов (разная длина строк...)
 - ❑ Трудности распараллеливания алгоритмов (балансировка нагрузки)
-
- ❑ **Пример:** $y = Ax$, A в формате CRS
 - ❑ **Вопрос:** найдите проблемы, вычислите арифметическую интенсивность

3. Форматы хранения

3.7. Проблемы стандартных форматов

- ❑ Низкая арифметическая интенсивность
- ❑ Нерегулярный доступ к памяти в большинстве базовых алгоритмов
- ❑ Проблемы с векторизацией расчетов (разная длина строк...)
- ❑ Трудности распараллеливания алгоритмов (балансировка нагрузки)

- ❑ **Итог:** низкая эффективность использования современных процессоров с большим числом ядер и широкими векторными регистрами
- ❑ **Направление работ:** разработка новых форматов, лучше приспособленных для современных и будущих архитектур

3. Форматы хранения

3.8. Формат Ellpack

- ❑ Формат ELLPACK предложен для эффективного представления разреженных матриц на графических ускорителях
- ❑ В сравнении с CRS:
 - улучшена локальность данных,
 - данные выровнены в памяти

3. Форматы хранения

3.8. Формат Ellpack

- ❑ Для каждой строки матрицы хранится одинаковое число ненулевых элементов
- ❑ Для хранения ненулевых элементов используются два массива:
 - Массив **Values** хранит значения ненулевых элементов построчно (строки рассматриваются по порядку сверху вниз),
 - массив **Col** хранит номера столбцов для каждого элемента,
 - если количество элементов в строке меньше максимального, массивы Values и Col дополняются нулями

3. Форматы хранения

3.8. Формат Ellpack

□ Пример:

A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

1	2	0
3	4	0
0	0	0
8	5	0
0	0	0
7	1	6

Value

0	4	0
2	3	0
0	0	0
3	5	0
0	0	0
1	2	5

Column

3. Форматы хранения

3.8. Формат Ellpack

□ Объем необходимой памяти.

– Плотное представление: $M = 8 N^2$ байт.

– В координатном формате: $M = 16 NZ$ байт.

– В формате CRS: $M = 8 NZ + 4 NZ + 4 (N + 1) = 12 NZ + 4 N + 4$.

– В формате ELLPACK:

$M = 4 K N + 8 K N = 12 K N$, K – максимальное число элементов в строке

3. Форматы хранения

3.8. Формат Ellpack

□ Пример: $y = Ax$.

```
// Цикл по строкам матрицы A
for (i = 0; i < N; i++)
{
    // Вычисляем i-ю компоненту вектора y
    sum = 0;
    for (j = 0; j < K; j++)
        sum += Value[i * K + j] * x[Col[i * K + j]];
    y[i] = sum;
}
```

3. Форматы хранения

3.9. Формат Ellpack – модификации

- ❑ ELLPACK эффективен для матриц, содержащих близкое количество ненулевых элементов во всех строках.
- ❑ Модификации ELLPACK ориентированы на сокращение числа дополненных элементов:
 - Sliced ELLPACK (SELL-C) разделяет матрицу на группы строк, каждая группа хранится в формате ELLPACK
 - Sliced ELLPACK с сортировкой (SELL-C- σ) дополнительно вводит области сортировки таким образом, что строки с близким числом ненулевых элементов будут собраны в одном блоке
 - Внутри блока элементы хранятся **по столбцам**

3. Форматы хранения

3.10. Формат Sliced ELLPACK (SELL-C)

□ Пример: $C = 2$

A

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

1	2	
3	4	
0	0	
8	5	
0	0	0
7	1	6

Value

0	4	
2	3	
0	0	
3	5	
0	0	0
1	2	5

Column

SlicePtr

0	4	8	14
---	---	---	----

Value

1	3	2	4	0	8	0	5	0	7	0	1	0	6
---	---	---	---	---	---	---	---	---	---	---	---	---	---

Col

0	2	4	3	0	3	0	5	0	1	0	2	0	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---

3. Форматы хранения

3.10. Формат Sliced ELLPACK (SELL-C)

□ Пример: $y = Ax$.

```
// Цикл по блокам строк матрицы A
for (i = 0; i < N / C; i++)
{
    nb = (SlicePtr[i + 1] - SlicePtr[i]) / C;
    for (k = 0; k < C; k++)
    { // Вычисляем (i*C+k)-ю компоненту вектора y
        sum = 0;
        start = SlicePtr[i] + k;
        for (j = 0; j < nb; j++)
            sum += Value[start + j * nb] *
                    x[Col[start + j * nb]];
        y[i * C + k] = sum;
    }
}
```

3. Форматы хранения

3.10. Формат Sliced ELLPACK (SELL-C)

□ Пример:

```
// Цикл по строкам матрицы A
for (i = 0; i < N; i += C)
{
    for (k = i; k < i + C; k++)
    { // Вычисляем k-ю компоненту вектора y
        start = SlicePtr[i / C] + k - C;
        end = SlicePtr[i / C + 1];
        sum = 0;
        for (j = start; j < end; j += C)
            sum += Value[j] * x[Col[j]];
        y[k] = sum;
    }
}
```

3. Форматы хранения

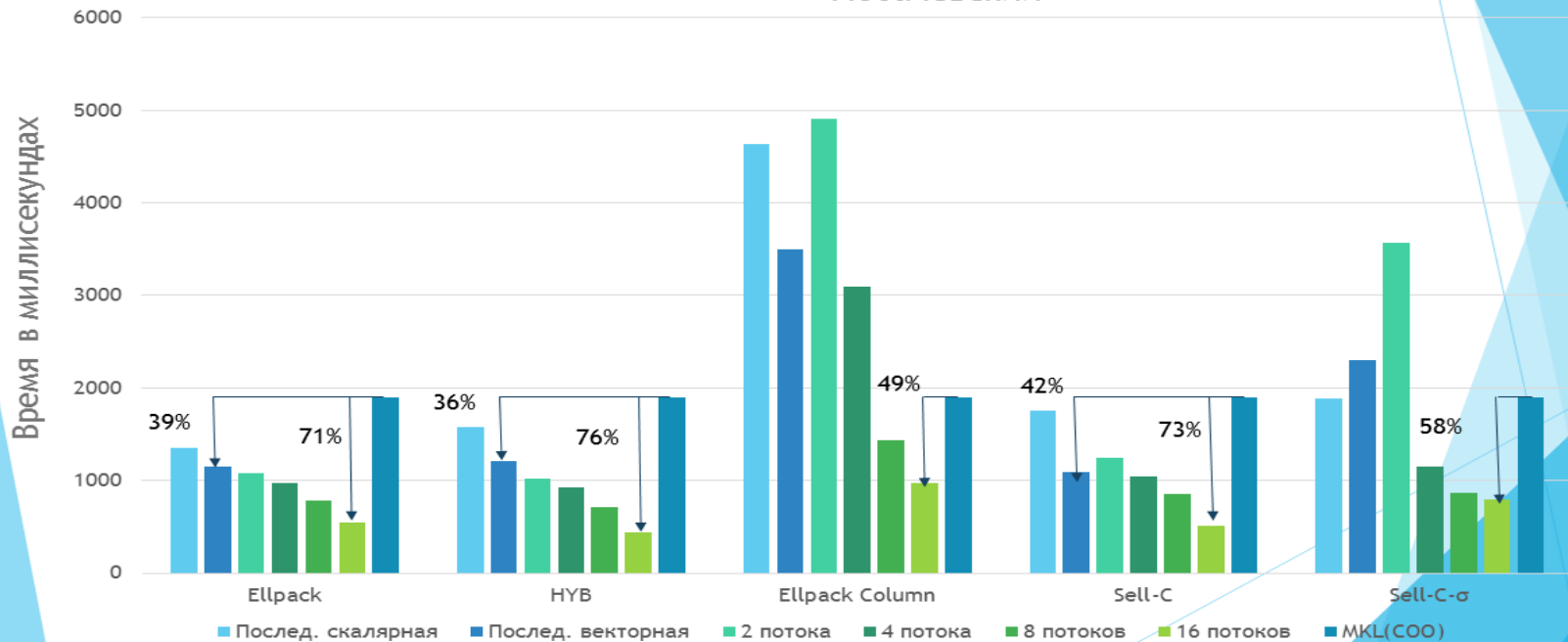
3.11. Гибридные форматы

- Гибридные форматы: часть элементов хранится в формате ELLPACK, остальные – в другом формате. Примеры:
 - ELLPACK + координатный формат (HYB)
 - ELLPACK с выделением маленьких плотных блоков 1×2 или 1×4
 - ELLPACK с выделением диагоналей

3. Форматы хранения

3.12. Эксперимент (А. Миронова, диплом бак.)

Матрица $\text{nlpkkt } 240$ ($n = 27\,993\,600$, $\text{nnz} = 760\,648\,352$); суперкомпьютер «Лобачевский»



4. ЗАКЛЮЧЕНИЕ

4. Заключение

- ❑ Алгебра разреженных матриц – важный прикладной раздел вычислительной математики: многие алгоритмы являются наиболее вычислительно-трудоемкими составляющими расчетных приложений
- ❑ Вопрос выбора подходящих структур данных крайне важен для практического использования
- ❑ Разработано много форматов хранения разреженных матриц
- ❑ При создании новых форматов идет борьба за преодоление основных проблем: нерегулярный доступ к памяти, малая эффективность векторизации и распараллеливания

Литература

1. Писсанецки С. Технология разреженных матриц. – М.:Мир, 1988.
2. Саад Ю. Итерационные методы для разреженных линейных систем: Учеб. пособие. – В 2-х томах. – М.: Изд-во МГУ, 2013.
3. Davis T. A. Direct methods for sparse linear systems. – Siam, 2006. – Т. 2.
4. Saad Y. Iterative methods for sparse linear systems. – Siam, 2003.
5. Saad Y. Numerical Methods for Large Eigenvalue Problems: Revised Edition. – Siam, 2011. – Т. 66.

Авторский коллектив

- ❑ Мееров Иосиф Борисович, к.т.н., доцент, зам. зав. каф. МОСТ
- ❑ Сысоев Александр Владимирович, к.т.н., доцент каф. МОСТ
- ❑ Линев Алексей Владимирович, зав. лаб. интернета вещей, каф. ПРИН
- ❑ Волокитин Валентин Дмитриевич, программист лаборатории СТиВВ, каф. МОСТ
- ❑ Козинов Евгений Александрович, к.т.н., преподаватель каф. МОСТ
- ❑ Панова Елена Анатольевна, инженер лаборатории СТиВВ, каф. МОСТ

Контакты

Нижегородский государственный университет

<http://www.unn.ru>

Центр компетенций oneAPI в ННГУ

<http://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу>

Институт информационных технологий, математики и механики

<http://www.itmm.unn.ru>