Лекция 8

Прикладные математические библиотеки:
cuBLAS (***B****asic* ***L****inear* ***A****lgebra* ***S****ubroutines*),
cuFFT (***F****ast* ***F****ourier* ***T****ransformation*), ...

Особенности реализации и использования cuBLAS

- Хранение по столбцам (*column-major storage*), совместимость с Фортраном, для копирования и инициализации матриц следует использовать специальный API;
- линейная индексация массивов;

Имена функций образуются по схеме: *cublas<T><function>*.

Например: cublasSgemv, тип данных float, generic matrix-vector умножение.

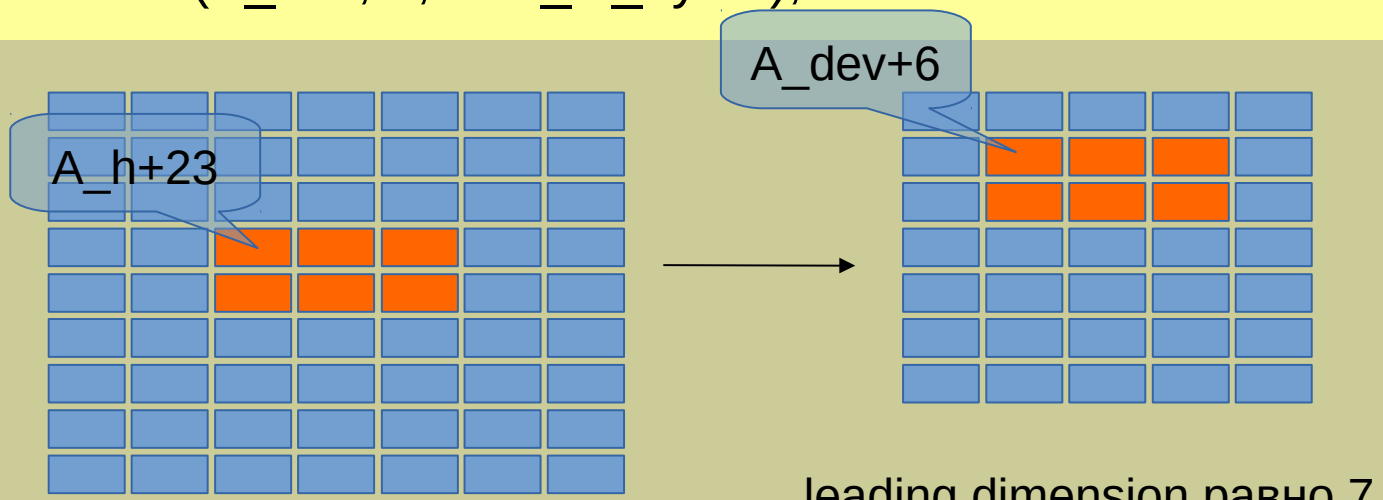
```
#include <stdio.h>
#include <stdlib.h>
#include <cublas_v2.h>

__host__ void print_array(float * data1,
                          float * data2,
                          int num_elem,
                          const char * prefix){
  printf("\n%s", prefix);
  for (int i=0; i<num_elem; i++){
    printf("\n%2d: %2.4f %2.4f ", i+1, data1[i], data2[i]);
  }
}
```

Особенности реализации и использования cuBLAS

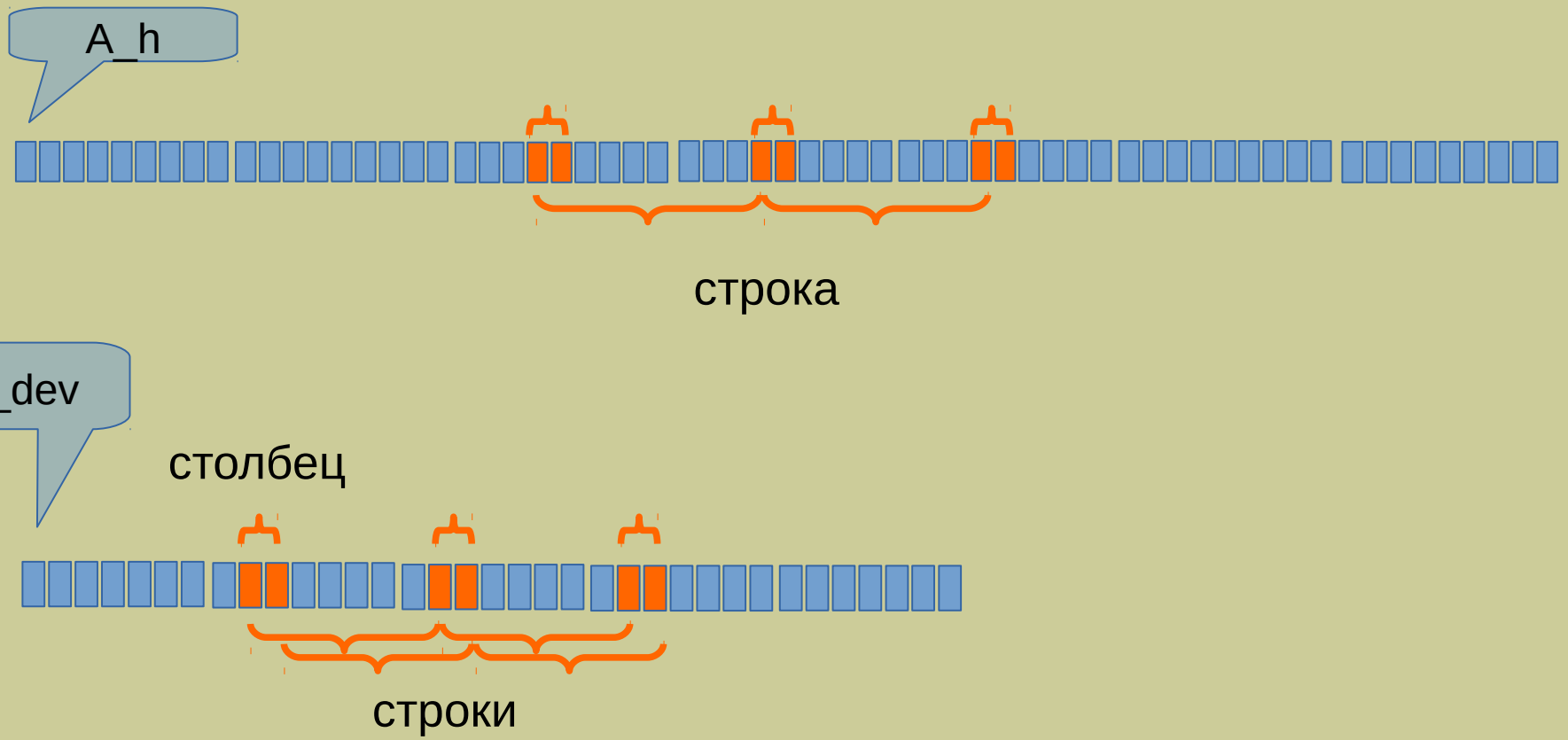
```
int main(){
  const int num_elem = 8;
  const size_t size_in_bytes = (num_elem * sizeof(float));

  float * A_dev;
  cudaMalloc( (void **) &A_dev, size_in_bytes );
  float * B_dev;
  cudaMalloc( (void **) &B_dev, size_in_bytes );

  float * A_h;
  cudaMallocHost( (void **) &A_h, size_in_bytes );
  float * B_h;
  cudaMallocHost( (void **) &B_h, size_in_bytes );

  memset(A_h, 0, size_in_bytes);
  memset(B_h, 0, size_in_bytes);

  // Инициализация библиотеки  CUBLAS
  cublasHandle_t cublas_handle;
  cublasCreate(&cublas_handle);
```

```
for (int i=0; i < num_elem; i++){
  A_h[i] = (float) i;
}
print_array(A_h, B_h, num_elem, "Before Set");

const int num_rows = num_elem;
const int num_cols = 1;
const size_t elem_size = sizeof(float);

//Копирование матрицы с числом строк num_elem и одним столбцом с хоста на устройство
cublasSetMatrix(num_rows, num_cols, elem_size, A_h,
                num_rows, A_dev, num_rows);

//Очищаем массив на устройстве
cudaMemset(B_dev, 0, size_in_bytes);
```

leading dimension



Размещение матрицы в памяти



Выполнение *SAXPY* средствами *cuBLAS*

```
// выполнение SingleAlphaXPlusY
  const int stride=1;
  float alpha = 2.0F;
  cublasSaxpy(cublas_handle, num_elem, &alpha, A_dev,
                    stride, B_dev, stride);

 //Копирование матриц с числом строк num_elem и одним столбцом с устройства на хост
  cublasGetMatrix(num_rows, num_cols, elem_size, A_dev,
                     num_rows, A_h, num_rows);
  cublasGetMatrix(num_rows, num_cols, elem_size, B_dev,
                          num_rows, B_h, num_rows);

  print_array(A_h, B_h, num_elem, "Intermediate Set");
```

```
  alpha = 3.0F;
// повторное выполнение SingleAlphaXPlusY
  cublasSaxpy(cublas_handle, num_elem, &alpha, A_dev,
                    stride, B_dev, stride);

  //Копирование матриц с числом строк num_elem и одним столбцом с
устройства на хост
  cublasGetMatrix(num_rows, num_cols, elem_size, A_dev,
                        num_rows, A_h, num_rows);
  cublasGetMatrix(num_rows, num_cols, elem_size, B_dev,
                        num_rows, B_h, num_rows);

  // Удостоверяемся, что все асинхронные вызовы выполнены
  const int default_stream = 0;
  cudaStreamSynchronize(default_stream);

  // Print out the arrays
  print_array(A_h, B_h, num_elem, "After Set");
  printf("\n");
```

```
  // Освобождаем ресурсы на устройстве
  cublasDestroy(cublas_handle);
  cudaFree(A_dev);
  cudaFree(B_dev);

  // Освобождаем ресурсы на хосте
  cudaFreeHost(A_h);
  cudaFreeHost(A_h);
  cudaFreeHost(B_h);

  //сброс устройства, подготовка для выполнения новых программ
  cudaDeviceReset();

  return 0;
}
```

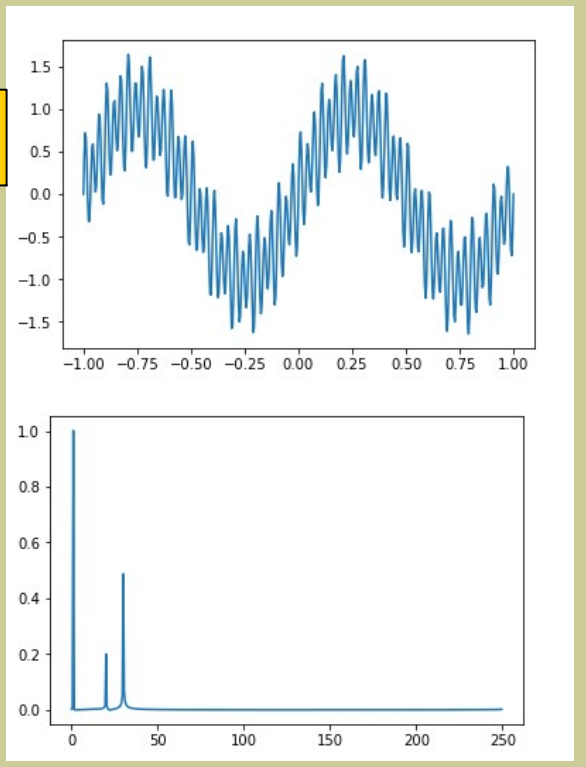
Дискретное преобразование Фурье

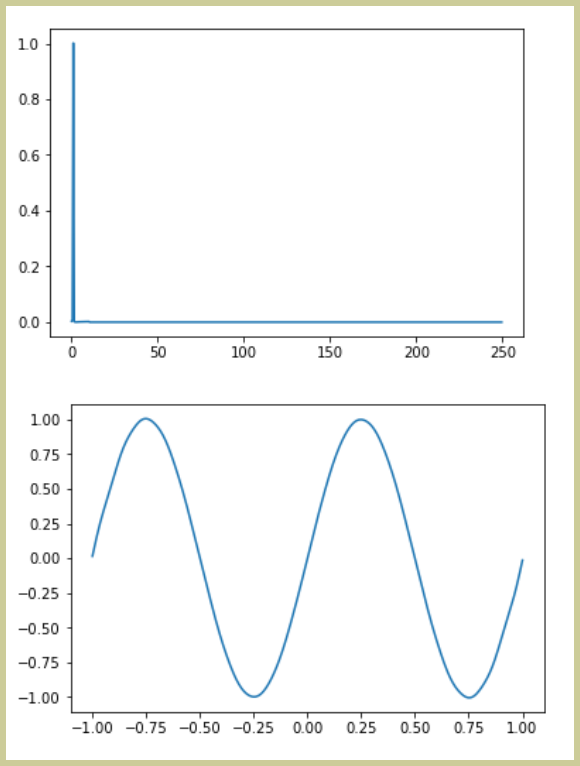
$$A_k = \sum_{n}^{N-1} a_n \exp(-2\pi i \frac{nk}{N})$$

$$a_n = \frac{1}{N} \sum_{n}^{N-1} A_k \exp(2\pi i \frac{nk}{N})$$

Фильтрация сигнала

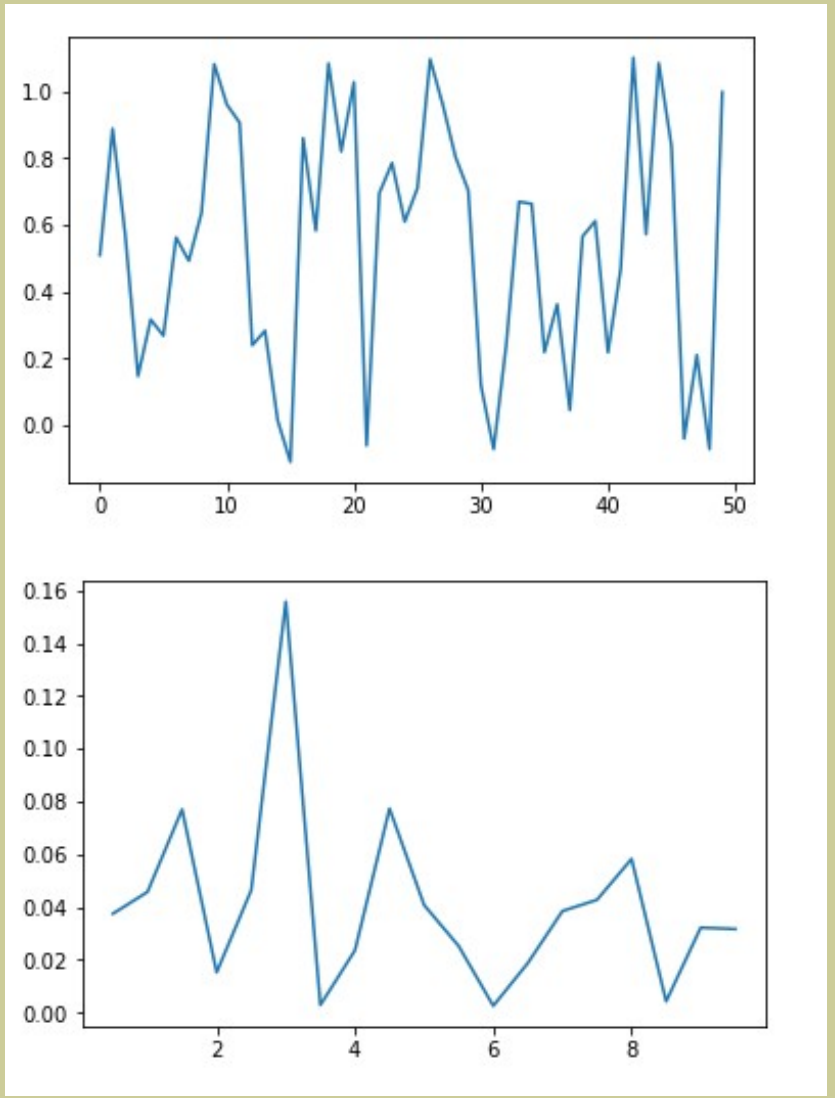
```
import numpy as np
import matplotlib.pyplot as plt

N = 500
x= np.linspace(-1.0,1.0,N)
k=np.arange(N)

y=np.exp(2*np.pi*1j*x)+0.2*np.exp(2*np.pi*1j*x*20)+\
                  0.5*np.exp(2*np.pi*1j*x*30)
yt=np.fft.fft(y)

plt.plot(x,y.imag)
plt.show()

plt.plot(k/2,np.sqrt(yt.real**2+yt.imag**2)/N)
plt.show()

for fr in range(N):
    if fr>20:
        yt[fr]=0+0j

plt.plot(k/2,np.sqrt(yt.real**2+yt.imag**2)/N)
plt.show()

zt=np.fft.ifft(yt)

plt.plot(x,zt.imag)
plt.show()
```





Поиск периодичностей

```
import numpy as np
import matplotlib.pyplot as plt

N = 50
x= np.linspace(-6.28,6.28,N)
x=0.2*np.sin(3*x)
x1=np.random.rand(N)
X=x+x1

k=np.arange(N)
plt.plot(k,X)
plt.show()

xt=np.fft.fft(X)

S=np.sqrt(xt.real**2+xt.imag**2)/N
plt.plot(k[1:20]/2,S[1:20])
plt.show()
```



cuFFT, cuFFTW

```
#include <cufft.h>
#include <stdio.h>
#include <malloc.h>
#define NX 64
#define BATCH 1
#define pi 3.141592

__global__ void gInitData(cufftComplex *data){
  int i=threadIdx.x+blockDim.x*blockIdx.x;

  float x=i*2.0f*pi/(NX);
  data[i].x=cosf(x)-3.0f*sinf(x);
  data[i].y=0.0f;
}
```

Инициализация [- эмуляция получения экспериментальных] данных

```
int main(){
  cufftHandle plan;
  cufftComplex *data;
  cufftComplex *data_h=(cufftComplex*)calloc(NX,sizeof(cufftComplex));;

   cudaMalloc((void**)&data, sizeof(cufftComplex)*NX*BATCH);
   if (cudaGetLastError() != cudaSuccess){
     fprintf(stderr, "Cuda error: Failed to allocate\n");
     return -1;
  }

  gInitData<<<1, NX>>>(data);
  cudaDeviceSynchronize();
```

Конфигурация и выполнение *cuFFT*

```
if (cufftPlan1d(&plan, NX, CUFFT_C2C, BATCH) != CUFFT_SUCCESS){
  fprintf(stderr, "CUFFT error: Plan creation failed");
  return -1;
}

if (cufftExecC2C(plan, data, data, CUFFT_FORWARD) != CUFFT_SUCCESS){
   fprintf(stderr, "CUFFT error: ExecC2C Forward failed");
   return -1;
}
if (cudaDeviceSynchronize() != cudaSuccess){
  fprintf(stderr, "Cuda error: Failed to synchronize\n");
  return -1;
}
```

```
  cudaMemcpy(data_h, data, NX*sizeof(cufftComplex),
                                                    cudaMemcpyDeviceToHost);

  for(int i=0;i<NX;i++)
    printf("%g\t%g\n", data_h[i].x, data_h[i].y);

  cufftDestroy(plan);
  cudaFree(data);
  free(data_h);

  return 0;
}
```

3.1665e-07	0
0.999998	-3
2.28871e-07	-8.2206e-07
1.63913e-07	-4.56348e-07
9.80322e-08	-3.23816e-07
7.98545e-08	-2.5686e-07
7.73086e-08	-2.28101e-07
9.77997e-08	-1.39602e-07
1.18862e-07	-1.54969e-07
9.28675e-08	-1.11384e-07
6.40492e-08	-8.61264e-08
-4.85517e-09	-5.77259e-08
6.43692e-08	-5.47365e-08
1.63796e-07	-9.20975e-08
7.76398e-08	9.34922e-09
8.9407e-08	-2.98023e-08
-2.04891e-08	1.11759e-08
7.45058e-08	-4.47035e-08
1.74435e-08	-7.70034e-08
8.75443e-08	-8.56817e-08
6.6242e-08	-2.56215e-08
9.06268e-08	-6.91981e-08
1.40801e-07	-5.71873e-08
9.21028e-08	2.889e-08
7.67154e-08	4.43338e-08
4.37996e-08	5.50851e-08
-4.38268e-09	4.02646e-08
-6.66299e-09	5.26171e-09
2.46764e-08	-7.85383e-09
4.83042e-08	2.96609e-08
5.39201e-08	1.64677e-08
-2.98023e-08	0