LU decomposition using GPU and OpenCL

In this example, we will create a random 4x5 matrix using uniform distribution and calculate its LU decomposition matrix. The calculation will be achieved by using the GPU card and OpenCL with a group of MatDeck functions that incorporate ArrayFire functionalities.

First, we will set the environment to use the GPU for calculations. Using the function, afp_supported_backends, a list of all supported backends that can be used for calculations will be produced. In our case, calculations can be made on the CPU, using OpenCL or CUDA framework.

$$\text{afp\_supported\_backends}() = \begin{bmatrix} \text{"cpu"} \\ \text{"opencl"} \\ \text{"cuda"} \end{bmatrix}$$

Default environment for calculations is the CPU. We can change the current environment with the function, afp_set_backend, and check which environment is currently in use with the afp_backend function.

$$\text{afp\_set\_backend}(\text{"opencl"}) = \text{true}$$

$$\text{afp\_backend}() = \text{"opencl"}$$

In each environment, there can be several devices which support calculations within it. To check the number of devices which support calculations in the current environment, use the function, afp_get_device_count, and the functions afp_get_device and afp_set_device to check/change current device.

$$\text{afp\_get\_device\_count}() = 3$$

$$\text{afp\_get\_device}() = 1$$

$$\text{afp\_set\_device}(1) = \text{true}$$

To display information about currently selected devices, use the function afp_device_info

$$\text{afp\_device\_info}() = \begin{bmatrix} \text{"Intel(R)\_HD\_Graphics\_620"} \\ \text{"OpenCL"} \\ \text{"Intel(R) OpenCL"} \\ \text{"2.1"} \end{bmatrix}$$

Finally, we have set OpenCL as a calculation backend and set the device with number 1 - integrated Intel graphic card as a device on which we will do all calculations.

Let's create a uniformly random 4x5 matrix with real values.

$$A := \text{afp\_randu}(4\ ,\ 5\ ,\ \text{"real"})$$

We can print variable A to check that the input matrix is generated.

$$A = \begin{bmatrix} 0.614 & 0.663 & 0.903 & 0.943 & 0.454 \\ 0.736 & 0.231 & 0.370 & 0.785 & 0.842 \\ 0.839 & 0.586 & 0.709 & 0.987 & 0.722 \\ 0.252 & 0.623 & 0.49 & 0.113 & 0.328 \end{bmatrix}$$

Now we can do LU decomposition calculations on matrix A and place the resulting vector in variable B. Resulting vector contains lower triangle matrix L, upper triangle matrix U and pivot vector.

$$B := \mathrm{afp\_lu}(A)$$

$$B = \left[\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.300 & 1 & 0 & 0 \\ 0.732 & 0.523 & 1 & 0 \\ 0.877 & -0.633 & -0.319 & 1 \end{bmatrix} \begin{bmatrix} 0.839 & 0.586 & 0.709 & 0.987 & 0.722 \\ 0 & 0.447 & 0.277 & -0.183 & 0.111 \\ 0 & 0 & 0.239 & 0.316 & -0.133 \\ 0 & 0 & 0 & -0.096 & 0.236 \end{bmatrix} \cdots \right]$$

There are separate functions for every member of the resulting vector.
Function afp_lu_low will calculate the lower triangle matrix L

$$\mathrm{afp\_lu\_low}(A) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.300 & 1 & 0 & 0 \\ 0.732 & 0.523 & 1 & 0 \\ 0.877 & -0.633 & -0.319 & 1 \end{bmatrix}$$

Function afp_lu_upp will calculate the upper triangular matrix U

$$\mathrm{afp\_lu\_upp}(A) = \begin{bmatrix} 0.839 & 0.586 & 0.709 & 0.987 & 0.722 \\ 0 & 0.447 & 0.277 & -0.183 & 0.111 \\ 0 & 0 & 0.239 & 0.316 & -0.133 \\ 0 & 0 & 0 & -0.096 & 0.236 \end{bmatrix}$$

And finally, function afp_lu_piv will return the pivot vector of the LU decomposition

$$\mathrm{afp\_lu\_piv}(A) = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$