SVD decomposition using OpenCL

In this example, we will create the random 4x5 matrix using uniform distribution and calculate its SVD decomposition matrix. The calculation will be achieved 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 the OpenCL as a calculation backend and set the device with number 1 - Intel CPU with OpenCL support 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 the variable A to check that the input matrix is generated.

$$A = \begin{bmatrix} 0.093 & 0.869 & 0.577 & 0.143 & 0.958 \\ 0.28 & 0.424 & 0.87 & 0.467 & 0.230 \\ 0.098 & 0.827 & 0.983 & 0.951 & 0.136 \\ 0.956 & 0.898 & 0.568 & 0.561 & 0.915 \end{bmatrix}$$

Now, we can do the SVD decomposition calculations on matrix A and place the resulting vector in variable B. Resulting vector contains unitary matrix U, non-zero diagonal elements as a sorted 1D vector S in descending order and unitary matrix V^T.

$$B := \text{afp\_svd}(A)$$

$$B = \begin{bmatrix} \begin{bmatrix} -0.462 & -0.414 & -0.784 & -0.027 \\ -0.382 & 0.326 & 0.083 & -0.861 \\ -0.521 & 0.696 & -0.078 & 0.487 \\ -0.607 & -0.487 & 0.611 & 0.144 \end{bmatrix} & \begin{bmatrix} 2.758 \\ 1.008 \\ 0.622 \\ 0.292 \end{bmatrix} & \begin{bmatrix} -0.283 & -0.558 & -0.528 & -0.392 & -0.42 \\ -0.342 & -0.083 & 0.448 & 0.478 & -0.668 \\ 0.846 & -0.260 & -0.177 & 0.313 & -0.295 \\ -0.199 & 0.494 & -0.696 & 0.473 & -0.089 \\ -0.217 & -0.608 & -0.067 & 0.545 & 0.531 \end{bmatrix} \end{bmatrix}$$

There are separate functions for every member of the resulting vector.
Function afp_svd_u will calculate the unitary matrix U

$$\text{afp\_svd\_u}(A) = \begin{bmatrix} -0.462 & -0.414 & -0.784 & -0.027 \\ -0.382 & 0.326 & 0.083 & -0.861 \\ -0.521 & 0.696 & -0.078 & 0.487 \\ -0.607 & -0.487 & 0.611 & 0.144 \end{bmatrix}$$

Function afp_svd_v will calculate the unitary matrix V

$$\text{afp\_svd\_v}(A) = \begin{bmatrix} -0.283 & -0.558 & -0.528 & -0.392 & -0.42 \\ -0.342 & -0.083 & 0.448 & 0.478 & -0.668 \\ 0.846 & -0.260 & -0.177 & 0.313 & -0.295 \\ -0.199 & 0.494 & -0.696 & 0.473 & -0.089 \\ -0.217 & -0.608 & -0.067 & 0.545 & 0.531 \end{bmatrix}$$

Finally, the function, afp_svd_s, will return non-zero diagonal elements as a sorted 1D vector S in descending order

$$\text{afp\_svd\_s}(A) = \begin{bmatrix} 2.758 \\ 1.008 \\ 0.622 \\ 0.292 \end{bmatrix}$$