SVD decomposition using CUDA

In this example, we will create a random 4x5 matrix using uniform distribution and calculate its SVD decomposition matrix. The calculation will be achieved using the Nvidia GPU card and CUDA 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.

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

$$\text{afp\_backend}(\,) = \text{"cuda"}$$

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.

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

$$\text{afp\_get\_device}(\,) = 0$$

$$\text{afp\_set\_device}(0) = \text{true}$$

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

$$\text{afp\_device\_info}(\,) = \begin{bmatrix} \text{"NVIDIA\_GeForce\_940MX"} \\ \text{"CUDA"} \\ \text{"v11.2"} \\ \text{"5.0"} \end{bmatrix}$$

Finally, we have set OpenCL as a calculation backend and set the device with number 0 - Nvidia GeForce GPU with CUDA 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.

$$A = \begin{bmatrix} 0.225 & 0.902 & 0.396 & 0.281 & 0.093 \\ 0.552 & 0.745 & 0.140 & 0.626 & 0.28 \\ 0.082 & 0.37 & 0.633 & 0.418 & 0.098 \\ 0.613 & 0.163 & 0.430 & 0.71 & 0.956 \end{bmatrix}$$

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

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

$$B = \begin{bmatrix} \begin{bmatrix} -0.439 & 0.613 & -0.007 & -0.657 \\ -0.545 & 0.199 & 0.606 & 0.544 \\ -0.354 & 0.216 & -0.792 & 0.448 \\ -0.620 & -0.733 & -0.075 & -0.269 \end{bmatrix} & \begin{bmatrix} 2.014 \\ 0.905 \\ 0.523 \\ 0.208 \end{bmatrix} & \begin{bmatrix} -0.402 & -0.514 & -0.368 & -0.523 & -0.408 \\ -0.203 & 0.732 & 0.102 & -0.147 & -0.626 \\ 0.425 & 0.268 & -0.863 & -0.012 & 0.038 \\ 0.116 & -0.314 & -0.076 & 0.732 & -0.589 \\ -0.777 & 0.174 & -0.321 & 0.412 & 0.307 \end{bmatrix} \end{bmatrix}$$

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

$$\text{afp\_svd\_u}(A) = \begin{bmatrix} -0.439 & 0.613 & -0.007 & -0.657 \\ -0.545 & 0.199 & 0.606 & 0.544 \\ -0.354 & 0.216 & -0.792 & 0.448 \\ -0.620 & -0.733 & -0.075 & -0.269 \end{bmatrix}$$

Function afp_svd_v will calculate the unitary matrix V

$$\text{afp\_svd\_v}(A) = \begin{bmatrix} -0.402 & -0.514 & -0.368 & -0.523 & -0.408 \\ -0.203 & 0.732 & 0.102 & -0.147 & -0.626 \\ 0.425 & 0.268 & -0.863 & -0.012 & 0.038 \\ 0.116 & -0.314 & -0.076 & 0.732 & -0.589 \\ -0.777 & 0.174 & -0.321 & 0.412 & 0.307 \end{bmatrix}$$

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

$$\text{afp\_svd\_s}(A) = \begin{bmatrix} 2.014 \\ 0.905 \\ 0.523 \\ 0.208 \end{bmatrix}$$