Matrix-vector product: column-wise access

In this exercise, we will implement a function that performs a matrix-vector multiplication

$$y \leftarrow \alpha A x + \beta y$$

where A is an $m \times n$ matrix, x is a vector of length n, y is a vector of length m, and α and β are scalars.

Download the lab materials from Autolab using the “Download handout” link. This includes a template `my_dgemv_v2.c` for your implementation and a header file `my_dgemv_v2.h`, which defines the `array2d_t` and `array_t` data structures. You do not need to upload the header file when you submit your solution.

The function in the template has the following prototype:

```c
int my_dgemv_v2(
    double alpha,
    double beta,
    const array2d_t *A,
    const array_t *x,
    array_t *y
);
```

The function should return `1` in case of errors or invalid input, and other it should return `0`.

Use two nested loops: the *outer loop* should loop over j (corresponding to the sum over j) and the *inner loop* should loop over i (corresponding to the m elements of y). In other words, the elements of A should be visited column-by-column.

Submit your solution to Autolab using the “Submit file” link.

Solution

```c
#include "my_dgemv_v2.h"

int my_dgemv_v2(
    double alpha,
    double beta,
    const array2d_t *A,
    const array_t *x,
    array_t *y)
{
  if (!A || !x || !y)
    return 1;
  if (A->shape[1] != x->len || A->shape[0] != y->len)
    return 1;

  size_t m = A->shape[0];
  size_t n = A->shape[1];
  double *px = x->val;
  double *py = y->val;

  for (size_t i = 0; i < m; i++)
    py[i] *= beta;

  if (A->order == RowMajor)
  {
    for (size_t j = 0; j < n; j++)
    {
      double *pA = A->val + j;
      for (size_t i = 0; i < m; i++)
      {
        py[i] += alpha * pA[i * n] * px[j]; // stride n
      }
    }
  }
  else
  {
    for (size_t j = 0; j < n; j++)
    {
      double *pA = A->val + j * m;
      for (size_t i = 0; i < m; i++)
      {
        py[i] += alpha * pA[i] * px[j]; // stride 1
      }
    }
  }
  return 0;
}
```