Student Name: Rohan Tiwari

PS2 Questions

=============

Add your answers to this file in plain text after each question. Leave a blank line between the text of the question and the text of your answer and at least two lines between your answer and the next question.

.. role:: cpp(code)

   :language: c++

Inner Product

----

1. How would you use :cpp:`dot` to implement :cpp:`two_norm`?

To implement two_norm using dot, we take the dot product of the vector with itself. After this, we take the square root of the result. The two_norm_d function in amath583.cpp implements this logic.

Student Name: Rohan Tiwari

Tensors

----

2. What is the formula that we would use for indexing if we wanted "slab-major" ordering? Hint: What is the size of a slab? It might be helpful to draw a picture of the data in 3D and see what the size is of a slab.

We want to do slab-major ordering. We assume the dimensions of Tensor are `[L,M,N]`. `L=num_slabs_, M=num_rows_, N=num_cols_.`

For a 2d row-major matrix with dimensions `[num_rows_, num_cols_]`, we know element at `[i,j] = j + i*num_cols_.` j is used to index within a row and i is used to jump rows. We can see that index j changes the fastest. We can build on this to find the location of element [i,j,k] in the Tensor in slab-major order.

k will be used to index within a slab, now we think of each slab as a matrix. In this case, index k will change the fastest and index i will change the slowest.

`j*num_cols_` will be used to jump rows and `i*num_rows*num_cols_` will be used to jump the matrix within each slab.

So indexing strategy is **`k + j*num_cols_ + i*num_rows_*num_cols_`**

This is how we will implement the indexing methods in Tensor class:

```
double& operator()(size_t i, size_t j, size_t k) {

    return storage_[k + num_cols_*j + i*num_rows_* num_cols_ ];

  }

const double& operator()(size_t i, size_t j, size_t k) const {

    return storage_[ k + num_cols_*j + i*num_rows_* num_cols_];

  }
```

3. (Extra credit.) I claim that there are six distinct indexing formulas. Show that this claim is correct or that it is incorrect.

Tensor has 3 dimensions, and each element is uniquely identified by specifying a value for each of the 3 dimensions. The order of specifying these could be anything thereby leading us to 3! combinations, which is 6 in total.

The claim is correct.