Math 4997-3 Quiz 6: Due by 2021/10/07

Exercises

1. Programming on paper (2 credits):
   Write a program that squares all elements $(a_i \cdot a_i)$ in a `std::vector<double> a` and compute the sum of all elements using `std::for_each` and `std::execution::par`.

2. Understanding code (2 credits):
   What does this program do?

```
#include <iostream>
#include <vector>
#include <numeric>
#include <future>

using namespace std;

int func1(vector<int> values){


    return accumulate(values.begin(),values.end(),0);


}

int main()
{
    std::vector<int> values = {1,2,3,4,5,6,7,8,9,10};

    auto f1 = std::async(func1,values);

    auto f2 = std::async([](const vector<int> values )
          {return std::inner_product(values.begin(), values.end(), values.begin(), 0);}
          ,values);


    cout << f1.get() + f2.get() << std::endl;

    return 0;
}
```

Programming exercise

1. Communication matrix: (2 credits)
   Use the following matrix as the network of people

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

   and compute $\mathbf{M}^4$, where $\mathbf{M}^4 = \mathbf{M} * \mathbf{M} * \mathbf{M} * \mathbf{M}$ and print the resulting matrix to the terminal or in the Jyputer notebook.

2. Conjugate gradient method (4 credits)
   To solve a equation system $\mathbf{Ax} = \mathbf{b}$, we can use the conjugate gradient methods (CG) by using following algorithm

   (a) $\mathbf{r_0} = \mathbf{b} - \mathbf{Ax}_0$
   (b) If $|\mathbf{r}_0| < \epsilon$ return $\mathbf{x}_0$
   (c) $\mathbf{p}_0 = \mathbf{r}_0$
   (d) $k = 0$
   (e) $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$
   (f) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
   (g) $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$
   (h) If $|\mathbf{r}_{k+1}| < \epsilon$ exit loop
   (i) $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
   (j) $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
   (k) $k = k + 1$
   (l) go to (e) return $\mathbf{x}_{k+1}$

   Implement the conjugate gradient algorithm using the Blaze library[1]. Note that the Blaze library is installed on the course's server `#include <blaze/Math.h>` and I recommend to use the server or Jupyter notebooks for this exercise.

   Hints:

   - Note that $\mathbf{x}_0$ can be chosen, if we know some assumption of the solution or set to zero.
   - The symbol $\mathbf{v}^T$ denotes the transpose[2] of the vector $\mathbf{v}$.
   - The symbol $|\mathbf{v}|$ denotes the norm[3] of a vector $\mathbf{v}$.
   - Before you start coding, please read Blaze's documentation[4] first. You will find plenty of functions there, *e.g.* printing a matrix to the terminal, and will have less work to implement.

[1] https://bitbucket.org/blaze-lib/blaze/wiki/Home
[2] https://bitbucket.org/blaze-lib/blaze/wiki/Vector%20Operations#!trans
[3] https://bitbucket.org/blaze-lib/blaze/wiki/Vector%20Operations#!norms
[4] https://bitbucket.org/blaze-lib/blaze/wiki/Home

Validation This code produces a matrix **A** and a vector **b**, such that the vector **x** is the solution for $\mathbf{Ax} = \mathbf{b}$

```
for(int i=0; i<N; ++i) {
        A(i,i) = 2.0;
        b[i] = 1.0*(1+i);
        x[i] = 0.5*(i+1);
}
```

You can use the matrix **A** and the vector **b** as the input of your CG implementation and compare your solution with the vector **x** to validate your code. You should not use this vector as the input of the CG algorithm, since your code might stop at step (2) already.

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.