Programming (ERIM)

5. Exercise

Deadline for submission: 2014-12-07 23:59 CET

Instructions

Testing your code in an automated manner is very important for developing working programs. Most programs are not written for one-time use, but they develop iteratively, and new features are added with time. However, adding new features can break existing functionality. *Unit tests* can help to give you confidence in your program code by testing functionality at the granularity of a single method call. For example, if you have a function that computes the input + 2:

```
function a = addTwo(b)
        a = b+2;
end
```

you can make a unit test for testing the function with some input values:

```
function testAddTwo()
        assert(addTwo(2) == 4);
        assert(addTwo(-2) == 0);
        "addTwo OK"
        end
end
```

Now although you cannot be sure about whether the addTwo works properly with all possible inputs, the unit test gives you some assurance that the addTwo-method works.

Exercise

Implement a function that does the "schoolbook" multiplication of matrices, i.e. $C = AB$ in a way that you would do it on paper. Implement the function in a test-first manner, that is, *first* write tests for the function and only afterwards the actual implementation. Document and check pre-conditions for the implemented function.

Make a script file for assessing how many times faster is the built-in matrix multiplication to your own "schoolbook" multiplication implementation. This script should perform computational tests with matrices of sizes $n \times n$, where $n \in \{2, 3, \ldots, 100\}$. In each iteration of the computational test, construct the matrices (A and B) to be multiplied: they should random numbers from the interval [1, 10]. Make a plot with two y-axes (Matlab: `plotyy`, R: `twoord.plot` from the `plotrix` library) so, that on the first axis you have the running times of the two methods, and on the second one the factor with which the built-in multiplication method is faster. Remember to include axis titles and a legend in the graph.