ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computational Engineering

Spring semester 2014

Prof. P. Koumoutsakos, G. Tauriello
ETH Zentrum, CLT F 12, C 11
CH-8092 Zürich

Mockup Exam

Issued: May 26, 2014

Exam directives. During the exam you will have to meet the following requirements:

- Clear your desk (no cell phones, cameras, etc.): on your desk you should have your Legi, your pen and your notes. We provide you with the necessary paper and the exam sheets.
- Read carefully the first two pages of the exam. Write your name, Legi-ID and the computer IP (or the hostname) where requested. Before handing in the exam, **PUT YOUR SIGNATURE ON PAGE 2**.
- The personal summary consists of no more than 4 pages (2 sheets). The personal summary can be handwritten or machine-typed. In case it is machine-typed, the text has to be single-spaced and the font size has to be at least 8 pts. You are not allowed to bring a copy of somebody else's summary.
- The teaching assistants will give you the necessary paper sheets. You are not allowed to use any other paper sheets.
- You can answer in English or in German; the answers should be handwritten and clearly readable, written in blue or black. Only one answer per question is accepted. Invalid answers should be clearly crossed out. Whenever you write a C++ code, include also the associated comments.
- To answer new questions (e.g. Question 1, not sub-questions!), always use a new page. On the top-right corner of every page write your complete name and Legi-ID. Unless otherwise noted in the question, you should hand-in your answers on paper! The only exception is the code for the programming questions to be solved using the computer.
- If something is disturbing you during the exam, or it is preventing you from peacefully solving the exam, please report it immediately to an assistant. Later notifications will not be accepted.
- You must hand in: the exam cover, the sheets with the exam questions and your solutions. The exam cannot be accepted if the cover sheet or the question sheets are not handed back.
- Although we suggest to solve simple questions without the use of the computer, you are allowed to use it to help you answer any question. The "exam mode" of the computers is that they are offline (no internet). You can use any text editor, GNU make and gcc.

Exam details. **This mockup exam was written for the lecture in 2013**. It is meant to give you an idea of the **style of questions that you may expect in the exam**. Last year's exam was divided in four different types of exercises and we will try to keep a similar structure this year:

1. *Numerical Problems*. This part is pen and paper only and covers the mathematical and numerical concepts described during the course. **This part corresponds to approximately 1/3 of the total points available for the exam**.

2. *Compiling and Running C++ programs*. In this part, you are asked to behave as a human compiler. Snippets of C++ code are proposed to you, and you have to determine if the code compiles (and if not why) or to understand what the code does and what will be its output. **This part corresponds to approximately 1/6 of the total points available for the exam**.

3. *Developing C++ programs*, In this part, you are asked to answer to theoretical questions about the different Design Patterns and Algorithms encountered during the course. Some programming related to such concepts may be required as well. **This part corresponds to approximately 1/6 of the total points available for the exam**.

4. *C++ in action*. This part is primarily concerned with programming and is the only part which you have to solve using the computer. The questions closely follow the style of the exercises proposed during the course. The problems may involve code to be written from scratch as well as code reading where gaps need to be filled. In the latter case, the codes will be smaller and simpler than the codes you have seen in the exercises. This part is **not** covered in this mock up exam and we refer to the exercises to get practice on this matter. **This part corresponds to approximately 1/3 of the total points available for the exam**.

Numerical Problems

Question 1: Theory

Please answer the following questions with one to two sentences.

a) You learned three methods to numerically estimate derivatives in the computer. Name two of them.
interpolation, taylor expansion, integration/quadrature

b) Name two criteria to select a numerical integration scheme for a physical system.
consistency, accuracy, stability, efficiency

c) Apart from accuracy, name one other feature that we look for in interpolation schemes for mesh-to-particle and particle-to-mesh.
Conservation of moments or efficiency in computation (compact support etc)

d) What physical quantity does the Hamiltonian $H(\mathbf{p}, \mathbf{q})$ represent in a system of ODEs?
The total energy, an invariant (conserved quantity).

Question 2: Accuracy of a numerical scheme

Consider the following ODE:

$$\frac{dx}{dt} = f(x(t)),\ x(0) = 0, \tag{1}$$

and the discretization $x^n \approx x(t^n) = x(\Delta t \cdot n)$.

Given x^n, we find x^{n+1} by computing:

$$x^* = x^n + \frac{\Delta t}{2} \cdot f(x^n), \tag{2}$$

$$x^{n+1} = x^n + \Delta t \cdot f(x^*). \tag{3}$$

This is called the Midpoint method.

a) Derive the order of accuracy of a single time-step of the Midpoint method (local error).
The answer is $O(\Delta t^3)$.
Starting from Eq. 1 and the fact that

$$x^* = x^n + \frac{\Delta t}{2} f(x^n), \tag{4}$$

we apply the Taylor expansion as follows:

$$f(x^*) = \frac{dx\left(n\Delta t + \frac{\Delta t}{2}\right)}{dt}, \tag{5}$$

$$= \frac{dx(n\Delta t)}{dt} + \frac{\Delta t}{2}\frac{d^2x(n\Delta t)}{dt^2} + O(\Delta t^2), \tag{6}$$

$$= f(x^n) + \frac{\Delta t}{2}\frac{df}{dt}(x^n) + O(\Delta t^2). \tag{7}$$

Alternatively, we could also write:

$$f(x^*) = f(x^n) + \frac{\Delta t}{2} f(x^n) f'(x^n) + O(\Delta t^2),$$
$$= f(x^n) + \frac{\Delta t}{2} \frac{df}{dt}(x^n) + O(\Delta t^2),$$

where we used $\frac{df}{dt} = f'(x(t))\frac{dx}{dt} = f'(x(t))f(x(t))$. Either way, we see that

$$\frac{df}{dt} = \frac{2(f(x^*) - f(x^n))}{\Delta t} + O(\Delta t). \tag{8}$$

Now, we write the Taylor series expansion of x^{n+1} around x^n, which is

$$x^{n+1} = x^n + \Delta t f(x^n) + \frac{\Delta t^2}{2} \frac{df(x^n)}{dt} + O(\Delta t^3). \tag{9}$$

Substituting Eq. 8 in above equation we get

$$x^{n+1} = x^n + \Delta t f(x^n) + \Delta t(f(x*) - f(x^n)) + O(\Delta t^3), \tag{10}$$
$$= x^n + \Delta t f(x^*) + O(\Delta t^3). \tag{11}$$

b) What is the order of accuracy of the scheme to compute the solution $x(t)$ at a fixed final time $t = t_{end}$ (global error)? Explain.
The answer is $O(\Delta t^2)$.
Given that the local order of accuracy is $O(\Delta t^3)$, we compute the global order of accuracy as $O(\Delta t^3) \cdot N$, where N is the total number of time-steps. Since $N = O(1/\Delta t)$, the global error is $O(\Delta t^2)$.

Question 3: Conservation of energy and moments

A particle with one degree of freedom evolves according to:

$$\frac{d^2x}{dt^2} = f(x(t)). \tag{12}$$

The kinetic energy T and the potential energy U are given by:

$$T(x(t)) = \frac{1}{2}\dot{x}^2, \tag{13}$$
$$U(x(t)) = -\int_{x_0}^{x} f(\zeta) d\zeta. \tag{14}$$

The total energy is given by $E = T + U$ and the force is given by $f(x(t)) = -\frac{dU}{dx}$.

a) Show that the energy for this particle is conserved: $\frac{dE}{dt} = 0$.

$$\frac{dE}{dt} = \frac{dT}{dt} + \frac{dU}{dt} \tag{15}$$
$$= \frac{dT}{dt} + \frac{dU}{dx}\frac{dx}{dt} \tag{16}$$
$$= \dot{x}\ddot{x} - f(x)\dot{x} \tag{17}$$
$$= \dot{x}\ddot{x} - \ddot{x}\dot{x} \tag{18}$$
$$= 0 \tag{19}$$

b) Consider a particle at position x_p, carrying a charge q_p. We would like to map the charge to the two closest grid points at positions x_i and x_{i+1} which will then have charges q_i and q_{i+1}. The mapping should conserve the zeroth and first moments. Derive formulas to compute the charges q_i and q_{i+1}.

We need to solve the following system of equations to conserve both moments:

$$q_i + q_{i+1} = q_p, \tag{20}$$
$$x_i q_i + x_{i+1} q_{i+1} = x_p q_p. \tag{21}$$

We multiply the first equation by x_i and subtract it from the second one:

$$(x_{i+1} - x_i) q_{i+1} = (x_p - x_i) q_p. \tag{22}$$

We finally obtain:

$$q_{i+1} = \frac{x_p - x_i}{x_{i+1} - x_i} q_p. \tag{23}$$

We insert this into Eq. 21 to obtain:

$$q_i = \left(1 - \frac{x_p - x_i}{x_{i+1} - x_i}\right) q_p = \frac{x_{i+1} - x_p}{x_{i+1} - x_i} q_p. \tag{24}$$

Question 4: Fast Fourier transforms

The discrete Fourier transform for a 2π periodic function defined at N uniformly-placed, discrete locations, $x_j = 2\pi j/N$, is given by

$$f_j = \sum_{k=0}^{N-1} \hat{f}_k \mathrm{e}^{\mathrm{i}kx_j}. \tag{25}$$

Because the transformation is linear, it can be represented as a matrix–vector multiplication of the form

$$\mathbf{f} = \mathbf{A}_N \hat{\mathbf{f}}, \tag{26}$$

where $\mathbf{f}$ and $\hat{\mathbf{f}}$ are vectors of the discrete values in physical and Fourier space, respectively.

a) For $N = 2$, determine $\mathbf{A}_2$ (please expand all complex numbers, i.e. $\mathrm{i}^2 = -1, \mathrm{e}^{\mathrm{i}\pi/2} = \mathrm{i}, \ldots$).

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

b) For $N = 4$, determine $\mathbf{A}_4$.

$$\mathbf{A}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \mathrm{i} & -1 & -\mathrm{i} \\ 1 & -1 & 1 & -1 \\ 1 & -\mathrm{i} & -1 & \mathrm{i} \end{bmatrix}.$$

c) Count the number of additions in the matrix-vector multiplication $\mathbf{f} = \mathbf{A}_4\hat{\mathbf{f}}$ with $\mathbf{A}_4$ as given in 4b. We note that we ignore the multiplications by 1 and i in this simple case as they may be thought as "shuffling" data.

12 additions.

d) One algorithm for the fast Fourier transform for a $N = 2M$ length vector is based on recursive calls of the following algorithm:

Step 1. Split $\hat{\mathbf{f}}$ into

$$\begin{aligned} \hat{\mathbf{f}}' &= [\hat{f}_0, \hat{f}_2, ..., \hat{f}_{N-2}]^T \\ \hat{\mathbf{f}}'' &= [\hat{f}_1, \hat{f}_3, ..., \hat{f}_{N-1}]^T \end{aligned} \tag{27}$$

Step 2. Compute the two half-size transforms on the split data

$$\begin{aligned} \mathbf{f}' &= \mathbf{A}_M\hat{\mathbf{f}}' \\ \mathbf{f}'' &= \mathbf{A}_M\hat{\mathbf{f}}'' \end{aligned} \tag{28}$$

Step 3. Combine the intermediate transformed data

$$\begin{aligned} f_j &= f'_j + \omega_N^j f''_j \\ f_{j+M} &= f'_j - \omega_N^j f''_j \end{aligned} \tag{29}$$

for $j = 0, 1, 2, ...M-1$ where $\omega_N = \exp(2\pi \mathrm{i}/N)$.

These three steps can be written as $\mathbf{f} = \mathbf{A}_4^c\mathbf{A}_4^t\mathbf{A}_4^s\hat{\mathbf{f}}$. Determine $\mathbf{A}_4^c$, $\mathbf{A}_4^t$, and $\mathbf{A}_4^s$.

$$\mathbf{A}_4^c = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & \mathrm{i} \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\mathrm{i} \end{bmatrix},$$

$$\mathbf{A}_4^t = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

$$\mathbf{A}_4^s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

e) Count the number of additions in $\mathbf{f} = \mathbf{A}_4^c\mathbf{A}_4^t\mathbf{A}_4^s\hat{\mathbf{f}}$ using the result in 4d and comment on the algorithm.

8 additions, which is 4 less than if we had used $\mathbf{A}_4$ directly.

Compiling and Running C++ programs

Question 5: Compiling C++ programs

a) If we would leave the class definition below as it is and continue coding on the following lines, the compiler would give an error. Where is the error and how can it be fixed?

```
1  class A
2  {
3  private:
4      int m;
5  }
```

Every class definition has to end with a final ; after the closing curly brace.

b) Does this code generate compilation errors? If yes, please indicate the line numbers and how to fix the problems.

```
1  class Simulation
2  {
3      static const int NPARTICLES = 20;
4      double particles[NPARTICLES][4];
5
6  public:
7
8      // save the simulation status to an array of floats
9      void to_chars(const double * ptr) const;
10 };
11
12 void Simulation::to_chars(const double * ptr)
13 {
14     int c = 0;
15
16     for(int i=0; i<NPARTICLES*4; i++, c++)
17         ptr[c] = particles[i/4][i%4];
18 }
```

We need to change the const keywords for the to_chars method:

```
1  class Simulation
2  {
3      static const int NPARTICLES = 20;
4      double particles[NPARTICLES][4];
5
6  public:
7
8      // save the simulation status to an array of floats
9      void to_chars(double * const ptr) const;
10 };
11
12 void Simulation::to_chars(double * const ptr) const
13 {
14     int c = 0;
15
16     for(int i=0; i<NPARTICLES*4; i++, c++)
17         ptr[c] = particles[i/4][i%4];
18 }
```

c) Consider the following code:

```
#include <iostream>

template <int T>
struct X
{
    static const int val = T * X<T-1>::val;
};

template <>
struct X<0>
{
    static const int val = 1;
};

int main()
{
    std::cout << X<5>::val << std::endl;

    return 0;
}
```

Is there any compilation or runtime error? If yes, indicate the error, otherwise write down the program output.

The code does not contain any error. The program output is 120.

Question 6: Memory allocation and Copy Constructor

Let us consider the following class, which does compile:

```
class FloatVector {
    int numelems;
    float * elems;
public:
    void clear() {
        for(int i=0; i<numelems; i++) elems[i] = 0;
    }

    FloatVector(const int n) : elems(new float[n]), numelems(n) { }
    ~FloatVector() { delete [] elems; }

    FloatVector(const FloatVector& src):
        elems(src.elems), numelems(src.numelems) { }
};
```

a) The following code leads to two potential execution errors.

```
int main() {
    FloatVector * a = new FloatVector(5e3);
    FloatVector b = *a;
    delete a;
    b.clear();
    return 0;
}
```

Please indicate line number and reason for the execution errors.

- line 5: the copy constructor initializes the variable `elems` with the same memory address as `src.elems`. When `src.elems` is deallocated, the memory pointed by `elems` becomes invalid. The function `clear()` attempts to access this invalid memory and will cause an error.
- line 7: `b` is destroyed, deleting the memory that was already deallocated at line 4.

b) Concisely, change the implementation of the copy constructor of `FloatVector`, so that the execution errors disappear.

We change the copy constructor into:

```
1 FloatVector(const FloatVector& src): numelems(src.numelems)
2 {
3     elems = new float[src.numelems];
4     for(int i = 0;i<numelems;i++) elems[i] = src.elems[i];
5 }
```

Question 7: Polymorphism

Consider the following code:

```
1  #include <iostream>
2  class ODEsolver {
3  protected:
4      void _rhs() {
5          std::cout << "Euler rhs" << std::endl;
6      }
7
8      void _update() {
9          std::cout << "Euler update" << std::endl;
10     }
11
12 public:
13     void step() {
14         _rhs();
15         _update();
16     }
17 };
18
19 class ODEsolver_Leapfrog : public ODEsolver {
20     void _update() {
21         std::cout << "Leapfrog update" << std::endl;
22     }
23 };
```

a) What would be the output of the following code?

```
1 ODEsolver * solver = new ODEsolver_Leapfrog;
2 solver->step();
```

The output is:

```
Euler rhs
Euler update
```

b) What would be the output of the following code?

```
ODEsolver_Leapfrog solver;
solver.step();
```

The output is:
Euler rhs
Euler update

Question 8: STL vectors

Let us consider the following program:

```
#include <iostream>
#include <vector>

using namespace std;

int main (int argc, char * const argv[]) {
    vector<int> v;

    for (int i=0; i<10; ++i) v.push_back(i);

    vector<int>::reverse_iterator it;
    for (it = v.rbegin(); it != v.rend(); it++)
        cout << *it << " ";

    cout << endl;

    return 0;
}
```

What is the output of the program?

The output is: 9 8 7 6 5 4 3 2 1 0.

Developing C++ programs

Question 9: Design Pattern

In order to generate geometrical objects of different shapes, you decide to implement the abstract factory pattern. Below, you find an unfinished code containing the implementation of the subclasses of `Shape`:

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Triangle: public Shape
6  {
7  public:
8      void imA() { cout << "I am a Triangle" << endl; }
9  };
10
11 class Rectangle: public Shape
12 {
13 public:
14     void imA() { cout << "I am a Rectangle" << endl; }
15 };
```

a) The implementation of the base class `Shape` is still missing. You are given the following two possible implementations:

```
1  // version A
2  class Shape
3  {
4  public:
5      virtual void imA() { cout << "I am a Shape" << endl; }
6  };
7  // version B
8  class Shape
9  {
10 public:
11     virtual void imA() = 0;
12 };
```

Will they both work? What is the difference?

Both versions work. Version B implements an abstract class and this enforces the explicit implementation of the virtual function in all derived classes.

b) Given a working implementation for the `Shape` class, we want to define an abstract factory `ShapeFactory` with subclasses `TriangleFactory` and `RectangleFactory` such that the `main` method below works and produces the following output:

```
I am a Triangle
I am a Rectangle
```

```
1  int main()
2  {
3      ShapeFactory *TF = new TriangleFactory;
```

```
4       ShapeFactory *RF = new RectangleFactory;
5       Shape *s1 = TF->produce();
6       Shape *s2 = RF->produce();
7       s1->imA();
8       s2->imA();
9
10      return 0;
11  }
```

Implement the classes `ShapeFactory`, `TriangleFactory` and `RectangleFactory` and state which version of the `Shape` class you are using.

```
1  class ShapeFactory
2  {
3  public:
4      virtual Shape* produce() = 0;
5  };
6
7  class TriangleFactory : public ShapeFactory
8  {
9  public:
10     Shape* produce() { return new Triangle(); }
11 };
12
13 class RectangleFactory : public ShapeFactory
14 {
15 public:
16     Shape* produce() { return new Rectangle(); }
17 };
```

These classes will work with both versions of `Shape`. Note that a variant without a pure virtual `ShapeFactory::produce` returning a `Shape` instance would only work in combination with version A of `Shape`.

Question 10: Performance of cell lists

In the lecture and exercises you have seen that cell lists are an efficient way to compute short range interactions. Yet, it does not come for free. You require to have a certain number of particles for it to be worth the effort. In this exercise you will derive when it is better not (!) to use a cell list.

a) Assume you are working in a 2D computational domain with $(x, y) \in [0, 1]^2$ and your pairwise interactions have a cutoff of 0.08. What is the maximum number M of cells that you could use?

The cell side's length has to be big enough to fit the cutoff radius: $1.0/0.08 = 12.5$. In this case, 12 boxes fit in one direction. Thus, the answer is $M = 12 \times 12 = 144$ cells.

b) Assume you have N particles and that the cost (or time) to compute a single pairwise interaction between them is c_{pw}. This means that the time required to compute x pairwise interactions is xc_{pw}. What is the cost t_{naive} to compute pairwise interactions between all particles?

Each of the N particles interacts with $N - 1$ other particles. So we have $N(N - 1)$ interactions. The total cost is $t_{naive} = c_{pw}N(N - 1)$.

As a sidenote: If interactions in this system are symmetric (or antisymmetric) we have to do $N(N-1)/2$ interactions. The cost would be $t_{naive} = c_{pw}N(N-1)/2$.

c) If we were to use a cell list, we would first need to go through all particles to create the cell list and then we would only compute interactions of a particle with particles in its own cell and neighboring ones. Assume that the N particles are nicely distributed over all cells so that each cell contains $N_c = N/M$ particles. Furthermore assume that the cost to put a single particle in a cell when creating the cell list is c_{cl} and the cost to compute a single pairwise interaction between 2 particles is again c_{pw}. What is the cost t_{cl} of creating the cell list and of computing pairwise interactions between all particles using a cell list?

We have to spend N times c_{cl} to setup the cell lists and then we go through N particles and for each we interact with all particles in the same cell plus the eight neighboring cells. The total cost is $t_{cl} = c_{cl}N + c_{pw}N(9N/M - 1)$.

d) Derive for what number of particles N it is better not (!) to use a cell list using your results from questions 10b and 10c. You can assume $M > 9$ and that we create the cell list at each time-step.

We have to solve for N in the inequality

$$c_{pw}N(N-1) < c_{cl}N + c_{pw}N(9N/M - 1)\,. \tag{30}$$

The results would tell us that is better not to use a cell list when dealing with $N < c_{cl}M/(c_{pw}(M-9))$ particles.

e) For large values of N, you would use cell lists. Consider the speed up $A = t_{naive}/t_{cl}$ where t_{naive} is the cost (or time to solution) for the naive algorithm as computed in question 10b while t_{cl} is the cost using cell lists as computed in question 10c. What is the expected speed up A in the limit of $N \to \infty$?

$$\begin{aligned}\lim_{N->\infty} A &= \lim_{N->\infty} \frac{c_{pw}N(N-1)}{c_{cl}N + c_{pw}N(9N/M-1)} \\ &= \lim_{N->\infty} \frac{c_{pw}(1-1/N)}{c_{cl}/N + c_{pw}(9/M - 1/N)} = \frac{c_{pw}}{c_{pw}9/M} = \frac{M}{9}\end{aligned} \tag{31}$$

So for very large values of N we expect a speed up of $A = M/9$.