First Name: **Signature:**

Last Name:

SCIPER ID: ____________________

EPFL

Practice exam

CS328 - Numerical Methods in Visual Computing

Procedures:

This is a timed examination. You will have a maximum of **2 hours** to complete this examination. Leave your **student ID card** on the table during the examination for verification. Write your **first and last name** as well as your **EPFL SCIPER number** on every sheet of paper. Use only black or blue ink pens and write legibly. Exam questions done in pencil or other colored pens will **NOT** be graded. Write your solutions on these sheets.

Multiple-choice items are each "worth" a fixed fraction of the question's score, while incorrect answers deduct a corresponding amount. The final score of multiple-choice question is clamped to a non-negative number.

Regulations:

With the exception of one (1) page of personal notes written on a A4 page, any use of textbooks or other books/printed materials, formula sheets, calculators, and other similar aids are **NOT** permitted during the examination. The use of electronic devices such as laptops, cell phones or other PDAs, MP3 players, headphones, etc. is strictly **PROHIBITED** during the examination.

The student is deemed to have failed the course if he or she is found cheating or violating any of the above regulations.

Course Name:	Numerical Methods in Visual Computing	**Date:**	21.12.2016
Course Number:	CS 328	**Time:**	14:15 - 16:00
Lecturer:	Wenzel Jakob		

1 The IEEE754 standard

1.1 Breaking things

Write three brief Python snippets that cause numerical overflow, underflow, and create NaNs, respectively. Anything is allowed (except simply returning existing constants from the standard library).

```
def overflow():
```

```
def underflow():
```

```
def nan():
```

1.2 Special values

In a few words, describe the purpose of the two special values `Inf` and `NaN` defined in the IEEE-754 standard.

1.3 Denormalized numbers

In a few words, describe the purpose of denormalized numbers defined in the IEEE-754 standard.

1.4 Cancellation

Suppose that a program subtracts two numbers `x` and `y` having the same sign and very similar magnitudes, which (by chance) yields an *exact* answer $\texttt{z} = \texttt{x} - \texttt{y}$ that is not affected by any rounding errors. Does `z` suffer from cancellation? Justify your answer.

1.5 Distributive law

Does floating point arithmetic satisfy the distributive law, i.e.

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

Justify your anwer.

2 Reverse engineering a nonlinear algorithm

You are reading some code from another programmer and you find numerical calculations without documentation and cryptic variable names. You eventually isolate the following iterative calculation that seems to be the most important part:

```
def compute_it(x):
    for i in range(10):
        x -= np.tan(x)
    return x
```

(i) What value does the iteration converge to? ? If there is more than one possible answer, any of them is fine

(ii) What algorithm is being used?

(iii) What equation is being solved? Explain the code in terms of the standard equation for the algorithm.

3 Linear systems

We are solving a square linear system $\mathbf{A}\mathbf{x} = \tilde{\mathbf{b}}$, where $\tilde{\mathbf{b}} = \mathbf{b} + \mathbf{n}$ is contaminated by noise $\mathbf{n}$ with $\|\mathbf{n}\|_2 < \varepsilon$. We shall assume that the only errors in the computation are due to $\mathbf{n}$, and that rounding errors etc. are negligible. You can assume the singular value decomposition of $\mathbf{A}$ is available, with left and right singular vectors named $\mathbf{u}_i$ and $\mathbf{v}_i$ and singular values named σ_i.

(i) What is the largest possible absolute error in $\mathbf{x}$? In other words, give a bound on the difference between the solution with the true value of $\mathbf{b}$ and the solution computed from the noisy value of $\tilde{\mathbf{b}}$.

(ii) What values of $\mathbf{b} \neq 0$ will lead to a large relative error in $\mathbf{x}$?

(iii) What values of $\mathbf{n}$ will lead to a large relative error in $\mathbf{x}$?

3.1 Using factorizations

Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ along with one of the factorizations shown below, explain how you would compute a solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^n$. List all required steps and specify the associated big-O runtime complexity for each one (constant factors don't matter). Use the following format in your answer:

1. Invert $\mathbf{A}$ (cost: $O(...)$)
2. Multiply $\mathbf{A}^{-1}$ by $\mathbf{b}$ to obtain $\mathbf{x}$ (cost: $O(...)$)

You should assume that all involved steps are implemented efficiently, and that matrices with special structure (e.g. diagonal matrices) use an efficient encoding.

(i) $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ (Cholesky decomposition)

(ii) $\mathbf{P}\mathbf{A} = \mathbf{L}\mathbf{U}$ (LU decomposition with partial pivoting)

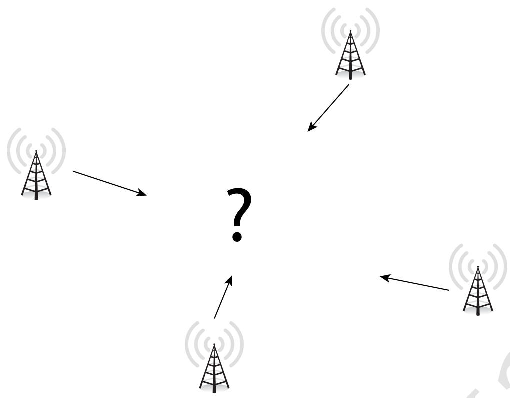
(iii) $\mathbf{A} = \mathbf{Q}\mathbf{R}$ (QR decomposition)

(iv) $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$ (Singular value decomposition)

Are all approaches in the same complexity class, or is one of them asymptotically faster?

4 Radar localization

A signal source is briefly observed by n radar dishes with *distinct* positions (x_i, y_i) $(i = 1, \ldots, n)$ scattered on a 2D plane. Following the observation, the operator of each dish reports back a unit vector (u_i, v_i) representing the most likely direction towards the signal.



We'll now focus on the problem of computing the source position (s, t) given these observations.

(i) Suppose that $n = 3$. For what dish arrangement can we expect the associated numerical problem to be ill-conditioned? Does the problem still admit solutions in this case? Justify your answers.

(ii) Let's assume from now on that the radar dish locations were intelligently chosen to avoid the problems discussed in the previous question.

For what values of n is the problem always underdetermined or overdetermined regardless of the source position? Why?

(iii) Let's assume that there are only two dishes. Construct a square linear system that can be used to solve for (s, t).

(iv) Now suppose that n is very large. List three algorithms that can be used to solve this problem and discuss performance and accuracy tradeoffs that would need to be considered to choose a particular one.

5 Numerical and analytic integration

5.1 Multiple-Choice

The following multiple-choice questions cover characteristic properties of analytic and numerical techniques covered in CS-328. Circle all items that are true.
You can assume that the functions in question are expressed using standard mathematical notation and only consist of elementary operations (addition, subtraction, square root, etc.)

A. The integral of a Riemann-integrable function can always be found using analytic techniques (i.e. calculus).

B. The derivative of a differentiable function can always be found using analytic techniques (i.e. calculus).

C. Integrating a quadratic polynomial using Simpson's rule will always yield an exact result.

D. A higher-order quadrature rule such as Simpson's rule is always more accurate than a lower-order quadrature rule such as the Trapezoid rule.

E. Monte Carlo integration is an effective technique to compute integrals of high-dimensional functions.

F. Monte Carlo integration requires that the function to be integrated is evaluated on a regular n-dimensional grid.

5.2 Orthogonal functions

Suppose that the following inner product is defined on the space of polynomials

$$\langle f, g \rangle := \int_0^1 f(x)g(x)\,\mathrm{d}x.$$

In this abstract space, the two functions

$$f_1(x) := x \qquad f_2(x) := x + 1$$

span a two-dimensional "plane" (i.e. a subspace containing all linear combinations of f_1 and f_2).

(i) Apply the Gram-Schmidt algorithm using pencil & paper to compute an orthogonal basis $(f_1, \tilde{f}_2)$ of this plane.

(ii) Describe why orthonormal sets of functions are sometimes preferred to non-orthonormal ones (e.g. in the context of regression).