Sample viva questions and topics

Lecturer: Colm Connaughton

List of examinable topics covered in 2021

- Floating point arithmetic, truncation error and loss of significance.
- Amplification of truncation error by instability
- Iteration, recursion and memoization, divide and conquer paradigm.
- Computational complexity of recursive algorithms
- Solving linear recursion relations
- Sorting: insertion sort, Shellsort and mergesort
- Structural recursion: linked lists and binary trees. Basic idea of how to represent recursive data structures in Julia.
- Data structures and the operations they are designed for: arrays, linked lists, stacks, queues, hash tables, binary trees.
- Binary tree search
- Interval membership and Fenwick trees - connection to Gillespie algorithm for stochastic simulation
- Bracket-and-bisect method for root finding.
- Derivation and properties of Newton-Raphson method for root finding, convergence properties.
- Newton-Raphson method in $\mathbb{R}^n$.
- Convex functions, convex sets and convex optimisation
- Linear programming: infeasible, unbounded and solvable problems, fundamental theorem of linear programming
- Writing a linear programme in standard form, slack variables, basic feasible vectors.
- Idea of Dantzig Simplex Algorithm (but **not** detailed calculations).
- Ideas of golden section search for one-dimensional minimisation (but **not** detailed derivation.)
- Method of steepest descent for unconstrained optimisation in $\mathbb{R}^n$.
- Stochastic gradient descent.
- Dual numbers and automatic differentiation.
- Taylor's theorem and derivation of finite difference formulae for numerical approximation of derivatives.
- Error analysis of timestepping algorithms
- Adaptive timestepping, stiffness and numerical solution of ordinary differential equations

Sample viva questions

1. Explain the concept of machine precision and how loss of significance occurs in floating point arithmetic.
2. Solve recurrence relations like the following:
$$a_n = a_{n-1} + a_{n-2}$$
with $a_0 = 1$ and $a_1 = 1$.
3. Write down an example of a recursive function (other than the factorial function!) and explain how it works. Explain the idea of a "divide-and-conquer" algorithm and give an example.
4. Explain how the Shellsort algorithm works and why it is faster than insertion sort. Given an array of 8 numbers in a random order, write down the intermediate partial sorts obtained for Shellsort with strides $\{4, 2, 1\}$.
5. Explain how the mergesort algorithm works and write down an equation for its computational complexity
6. Consider a recurrence relation like the following for the computational complexity of a hypothetical divide-and-conquer algorithm:
$$F(n) = 2F\left(\frac{n}{2}\right) + n$$
with $F(1) = 1$. Explain how to solve this recursion.
7. What is meant by structural recursion and give some examples.
8. Explain what is a linked list and describe how it compares to a linear array for storing a sequence of objects. Describe how to create a linked list in Julia.
9. What are stacks and queues?
10. Explain what is a hash table and how it works.
11. Explain what is a binary tree and outline how it can be used to perform search on a set of key-value pairs in $O(\log n)$ time where n is the number of elements in the set to be searched.
12. What is a Fenwick tree and how does it differ from a binary search tree? Explain how a Fenwick tree can be used to solve the interval membership problem efficiently.
13. Explain what it means for an interval (a, b) to bracket a root of a function $f(x)$ of a single variable and explain how the bracket-and-bisect algorithm works.
14. Derive the Newton Raphson method for finding roots of a function of a single variable and explain its advantages and disadvantages.
15. Consider a set of n nonlinear equations in $\mathbb{R}^n$:
$$\mathbf{F}(\mathbf{x}) = 0.$$
Derive the Newton Raphson method for multi-dimensional root finding.
16. Explain the concepts of convex sets, convex functions and convex optimisation problems. Why is convexity so important in the theory of optimisation?

17. What is a linear programme? Explain why the feasible set is a polygon in $\mathbb{R}^n$ and why the solution (if it exists) must be at a vertex of the feasible set.
18. Explain how to put a general linear programme in standard form.
19. Explain how Dantzig's simplex algorithm works.
20. Explain how to find a starting vertex for the simplex algorithm.
21. Explain how to formulate the quantile regression problem as a linear programme.
22. What does it mean for a triple (a, b, c) to bracket a minimum of a function $f(x)$ of a single variable? Explain the golden section search algorithm to find a local minimum of a function of a single variable starting from a bracketing triple.
23. Given a function, $f(\mathbf{x})$ of n variables, what is the gradient of f? Given a point $\mathbf{x} \in \mathbb{R}^n$ and a direction $\mathbf{d} \in \mathbb{R}^n$, what is the line minimiser of $f(\mathbf{x})$ from $\mathbf{x}$ in the direction $\mathbf{d}$? Explain how the Method of Steepest Descent works.
24. Describe the stochastic gradient descent algorithm and explain what types of optimisation problems it is intended to solve.
25. Write down the addition, multiplication and conjugation rules for dual numbers. Explain how dual arithmetic provides automatic differentiation of functions of a single variable.
26. Derive the forward and backward Euler algorithms for solving the system of ordinary differential equations

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}\left(\mathbf{u}\right) \qquad \text{with } \mathbf{u}(0) = \mathbf{U},$$

    and explain the difference between global and stepwise error.
27. Explain the difference between explicit and implicit time-stepping algorithms. Derive the implicit trapezoidal method:

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \frac{h}{2}(\mathbf{F}_i + \mathbf{F}_{i+1}).$$

28. Derive the improved Euler method

$$\begin{aligned} \mathbf{u}^*_{i+1} &= \mathbf{u}_i + h\mathbf{F}_i \\ \mathbf{F}^*_{i+1} &= \mathbf{F}(\mathbf{u}^*_{i+1}) \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \frac{h}{2}\left[\mathbf{F}_i + \mathbf{F}^*_{i+1}\right]. \end{aligned}$$

29. Explain how adaptive timestepping works and show that for a method with stepwise error of $O(h^n)$ that

$$h_{\mathbf{new}} = \left(\frac{\varepsilon}{\Delta}\right)^{\frac{1}{n}} h_{\mathbf{old}}$$

    where ε is the absolute error tolerance and Δ is the current estimated stepwise error.
30. What is a stiff problem and why are they difficult to solve?