Principali informazioni sull'insegnamento

Denominazione dell'insegnamento	**Calcolo Numerico**
Corso di studio	Informatica e Tecnologie per la Produzione del Software
Anno Accademico	2025/26
Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS)	6 CFU
Settore Scientifico Disciplinare	MAT/08
Lingua di erogazione	Italiano
Anno di corso	Secondo
Periodo di erogazione	2^semestre, le date esatte sono riportate nel manifesto/regolamento
Obbligo di frequenza	La frequenza è fortemente raccomandata
Sito web del corso di studio	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea

Docente/i

Nome e cognome	Roberto Garrappa
Indirizzo mail	roberto.garrappa@uniba.it
Telefono	080.5442685
Sede	Dipartimento di Matematica, Via Orabona 4, 70125, Bari. Stanza n.7, III piano.
Sede virtuale	Piattaforma e-learning UNIBA - https://elearning.uniba.it/
Sito web del docente	https://www.dm.uniba.it/it/members/garrappa
Ricevimento (giorni, orari e modalità, es. su appuntamento)	Giovedì dalle 15:00 alle 17:00 Si riceve anche in altri giorni previo appuntamento da richiedere via mail

Syllabus

Obiettivi formativi	*Il corso si propone di far acquisire allo studente una piena conoscenza delle modalità di trattamento di dati numerici attraverso il calcolatore. Inoltre, si punta a far*

	acquisire le principali tecniche per lo sviluppo di software per la risoluzione di problemi numerici ed a sviluppare le competenze necessarie per analizzare le proprietà dei diversi metodi (accuratezza, costo computazionale, ecc.). Lo studente svilupperà inoltre abilità nel testare i codici numerici e presentare in forma efficace il lavoro svolto.
Prerequisiti	Elementi di base dell'algebra lineare (vettori, matrici, determinante) e degli spazi vettoriali. Si tratta di argomenti generalmente trattati durante le scuole superiori e durante il corso di Matematica Discreta. L'insegnamento di Analisi Matematica è fortemente consigliato per poter seguire l'esame di Calcolo Numerico e sostenere con profitto il relativo esame. In particolare, lo studente dovrà conoscere gli elementi di base del calcolo differenziali (limiti, derivate e sviluppo in serie di Taylor) e del calcolo integrale per funzioni di una variabile. Lo studente deve inoltre possedere le basi di elementi di programmazione in modo da essere in grado di sviluppare codici numerici.
Contenuti di insegnamento (Programma)	1. Introduzione al Calcolo Numerico La risoluzione di problemi numerici al calcolatore, sorgenti di errore e misurazione degli errori, il processo di risoluzione numerica, efficienza, testing, errori computazionali, ambienti computazionali, linguaggi per il calcolo scientifico. – Ore 2.00 2. Il linguaggio di programmazione Python Introduzione al linguaggio, file di tipo script e function. Funzioni predefinite in NumPy. Introduzione alla grafica. – Ore 10.00 3. Aritmetica dei calcolatori Rappresentazione dei numeri. IEEE singola e doppia precisione. Troncamento e Arrotondamento. Precisione di macchina. Errore assoluto e relativo. Operazioni con i numeri di macchina. Cancellazione di cifre significative. Condizionamento di un problema. Stabilità degli algoritmi. Propagazione degli errori. Esperienze numeriche in Python sugli errori di arrotondamento e la loro propagazione. – Ore 10.00 3. Algebra lineare numerica e risoluzione di sistemi lineari Introduzione all'algebra lineare. Vettori, matrici ed operazioni su vettori e matrici. Norme di matrici e vettori. Matrici particolari (identità, nulla, matrici di permutazione, matrici triangolari, ecc.). Determinante di una matrice ed inversa di matrici. Sistemi di equazioni lineari. Studio del condizionamento di un sistema di equazioni lineari. Soluzione di sistemi di equazioni lineari con matrici triangolari inferiori e superiori: metodo di sostituzione in avanti ed indietro e costo di calcolo. Algoritmo di eliminazione di Gauss. Problematiche di stabilità e tecnica del pivoting. Fattorizzazione LU di una matrice. Esistenza della fattorizzazione LU con pivot. Matrici ortogonali. Cenni su decomposizione ai valori singolari e sue applicazioni (componenti principali, regressione lineare e multilineare, face recognition). Esercitazioni e sviluppo di codici Python, documentazione e testing, confronti con il software esistente. – Ore 18.00 4. Interpolazione Base delle potenze e metodo dei coefficienti indeterminati. Interpolazione di Lagrange e formule baricentriche. Interpolazione di Newton. Differenze divise. Interpolazione con nodi coincidenti. Errore nell'interpolazione polinomiale e studio del condizionamento. Cenni sulla scelta dei nodi di interpolazione ed utilizzo di nodi di Chebyshev. Interpolazione lineare a tratti. Approssimazione ai minimi quadrati nel discreto. Esercitazioni e sviluppo di codici Python, documentazione e testing, confronti con il software esistente. – Ore 10.00

	5. Calcolo degli zeri di funzione Metodo delle bisezioni successive: derivazione, convergenza, criteri di arresto e stime dell'errore. Iterazione funzionale e ordine di convergenza. Il metodo di Newton: derivazione geometrica ed analitica, criteri di arresto e studio della convergenza. Metodi quasi newtoniani: metodo della direzione costante, metodo della falsa posizione, metodo delle secanti. Esercitazioni e sviluppo di codici Python, documentazione e testing, confronti con il software esistente. – Ore 6.00 6. Formule di quadratura per il calcolo degli integrali. Metodo dei trapezi, di Simpson, dei trapezi composto, di Simpson composto per il calcolo di integrali definiti. Stime dell'errore. Esercitazioni e sviluppo di codici Python, documentazione e testing, confronti con il software esistente. – Ore 6.00
Testi di riferimento	Uri M. Ascher, Chen Greif, A First Course in Numerical Methods, SIAM, 2011 (testo di riferimento) Michael R. Overton, Numerical Computing with IEEE Floating Point Arithmetic, SIAM 2001 (lettura consigliata) James F. Epperson, Introduzione all'analisi numerica, teoria, metodi, algoritmi. McGraw-Hill, Milano, 2003 (lettura consigliata)
Note ai testi di riferimento	Il docente integrerà i testi di riferimento con propri appunti che saranno distribuiti per mezzo della piattaforma di e-learning.

Organizzazione della didattica

Ore

Totali	Didattica frontale	Esercitazione	Studio individuale
150 ore	32 ore	30 ore	88 ore

CFU/ETCS

6 CFU	4 CFU	2 CFU	

Metodi didattici

	Il corso viene erogato mediante lezioni frontali integrate da esercitazioni teoriche ed esercitazioni pratiche al calcolatore svolte dal docente e che gli studenti possono replicare sui propri calcolatori. Tutto il materiale prodotto in aula, unitamente a dispense, viene distribuito agli studenti attraverso la piattaforma didattica di e-learning del Corso di Laurea.

Risultati di apprendimento previsti	
Conoscenza e capacità di comprensione	Conoscere le tecniche per lo sviluppo di software pe la risoluzione di problemi matematici Conoscere le tecniche per analizzare le proprietà dei metodi numerici
Conoscenza e capacità di comprensione applicate	Capacità di ottimizzare gli algoritmi in base alle caratteristiche del problema ed alle risorse di calcolo a disposizione. Capacità di programmare, documentare e testare gli algoritmi numerici e di interpretare i risultati.
Competenze trasversali	**Autonomia di giudizio** acquisire la capacità di individuare i metodi più adatti in base al problema ed alle risorse di calcolo a disposizione. **Abilità comunicative** saper descrivere con adeguato linguaggio il problema da affrontare, le tecniche di risoluzione e le principali proprietà di ciascuna di esse; saper inoltre presentare i risultati del proprio lavoro di programmazione. **Capacità di apprendere in modo autonomo** saper derivare metodi per problemi non trattati partendo dai metodi studiati a lezione.

Valutazione	
Modalità di verifica dell'apprendimento	L'esame consiste in una prova orale, nel quale vengono poste allo studente domande sugli argomenti del corso: sviluppo degli algoritmi, proprietà teoriche, aspetti di implementazione, ecc. Inoltre per superare la prova di esame lo studente dovrà produrre i programmi (consigliato il linguaggio Python) relativi a tutti gli argomenti ed algoritmi studiati, accludendo una presentazione dei risultati ottenuti. Lo studente dovrà implementare gli algoritmi studiati, in particolare quelli non già implementati in classe, ed effettuare dei test con dati significativi. La presentazione dei risultati delle esperienze numeriche, mirante a mettere in evidenza le proprietà salienti di ciascun algoritmo, dovrà essere corredata da commenti ai risultati stessi; durante l'esame si discuteranno e commenteranno anche i programmi Python ed i risultati delle esperienze numeriche. La presentazione dei codici, con risultati e commenti, dovrà essere riportati in un file pdf o power-point da inviare via mail al docente qualche giorno prima dell'esame. Il voto è espresso in trentesimi.

Criteri di valutazione	• *Conoscenza e capacità di comprensione: Lo studente dovrà mostrare di aver compreso le tecniche di base per lo sviluppo di software numerico del calcolo numerico e di saper illustrare i principali metodi oggetto del corso di studio.* • *Autonomia di giudizio*: lo studente dovrà mostrare di essere in grado di valutare le principali caratteristiche di ciascun metodo e di saper effettuare confronti tra i diversi metodi. • Abilità comunicative: lo studente dovrà mostrare di essere in grado di presentare in maniera efficace i risultati del proprio lavoro. Capacità di apprendere: lo studente dovrà mostrare di essere in grado di applicare le tecniche studiate anche in contesti leggermente differenti da quelli illustrati durante il corso.
Criteri di misurazione dell'apprendimento e di attribuzione del voto finale	Gli studenti devono mostrare di essere in grado di comprendere le principali problematiche relative alla risoluzione di problemi numerici, allo sviluppo di metodi ed allo studio delle loro proprietà. Gli studenti dovranno essere quindi in grado di implementare i metodi studiati, di testarli e di presentare in maniera efficace i risultati ottenuti. L'esame mira anche a valutare la capacità dello studente nell'illustrare in maniera incisiva il lavoro di programmazione svolto.
Altro	Si suggerisce agli studenti di affidarsi esclusivamente alle informazioni/comunicazioni fornite sui siti ufficiali del Dipartimento di Informatica, ovvero sui gruppi social solo se costituiti e amministrati esclusivamente dai docenti dei relativi insegnamenti: • https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea • https://www.uniba.it/it/ricerca/dipartimenti/informatica • https://elearning.uniba.it/ I programmi degli insegnamenti sono disponibili qui: • https://elearning.uniba.it/ Le informazioni che tutti gli studenti dovrebbero conoscere sono scritte nei Regolamenti didattici e manifesti degli studi disponibili nel sito: • https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea Si suggerisce agli studenti di diffidare delle informazioni e dei materiali circolanti su siti o gruppi social non ufficiali, poiché spesso sono risultati non affidabili, non corretti o incompleti. Per ogni dubbio, chiedere un incontro al docente secondo le modalità previste per il ricevimento.

Main information on the course

Course name	Calcolo Numerico	
Degree	Informatica e Tecnologie per la Produzione del Software	
Academic year	2025/26	
European Credit Transfer and Accumulation System (ECTS), in Italian Crediti Formativi Universitari (CFU)		6 CFU (each CFU corresponds to 25 hours (h) of student's time); CFU are of type T1, T2 or T3 T1 = 8 h lecture + 17 h individual study T2 = 15 h practice + 10 h individual study T3 = 25 h individual study
Settore Scientifico Disciplinare	MAT/08	
Course language	Italian	
Anno di corso	Second	
Course period	Second semester	
Course attendance requirement	It is highly recommended to attend classes	
Website of the Degree	https://www.uniba.it/it/ricerca/dipartimenti/informatica/didattica/corsi-di-laurea/corsi-di-laurea	

Teacher(s)

Name and Surname	Roberto Garrappa
email	roberto.garrappa@uniba.it
phone	080.5442685
office	Dipartimento di Matematica, Via Orabona 4, 70125, Bari. Stanza n.7, III piano.
e-learning platform	Piattaforma e-learning UNIBA - https://elearning.uniba.it/
Teacher's homepage	https://www.dm.uniba.it/it/members/garrappa
Office hours	Thursday 15:00 - 17:00 (by appointment on other days)

Syllabus

Course goals	*The course aims to give to students a full knowledge of methods of processing numerical data by means of computers. Moreover, students will learn the main techniques for the development of numerical software and for analyzing their main features (accuracy, computational cost, etc.). Students will also develop skills in testing numerical codes and presenting their work in an effective way.*
Prerequisites/requirements	Linear Algebra (vectors, matrices and determinant) and vector spaces. Elements of calculus (limits, derivatives, integrals and Taylor's series). Programming skills.
Course program	1. Introduction to the course – Hours 2.00 2. Introduction to Python – Hours 10.00 3. Computer arithmetic. Floating-point representation. IEEE single and double precision. Chopping and rounding. Machine precision. Absolute and relative errors. Floating-point arithmetic. Analysis of conditioning of some problems. Stability of algorithms.

	Errors propagation. Python experiments on errors and their propagation. – Ore 10.00 3. Linear algebra and solution of linear systems Introduction to linear algebra. Vectors, matrices and particular matrices (identity permutation matrices, triangular matrices). Vector and matrix norms. Systems of linear equations. Conditioning number of a matrix and conditioning of linear systems. Solution of linear systems with triangular matrices: backward and forward substitution algorithms Gauss elimination algorithm. Stability and Pivoting. LU factorization. Overview of singular value decomposition and its applications (principal component analysis, linear and multilinear regression, face recognition). Python coding and experiments. – Hours 18.00 4. Interpolation Power basis and Vandermonde matrix. Lagrange interpolation formula and barycentric formulas. Newton formula. Remainder and errors in polynomial interpolation. Chebyshev nodes. Piecewise linear interpolant and linear splines. Discrete least squares approximation. – Hours 10.00 5. Roots computation Successive bisection method. Function iterations, convergence and stopping criteria. Newton method: geometric and analytical derivation, convergence and stopping criteria. Quasi-Newton methods: false position and secant methods. – Hours 6.00 6. Quadrature rules. Trapezoidal rule, Simpson rules and compound formulas. – Hours 6.00
Books of reference	Uri M. Ascher, Chen Greif, A First Course in Numerical Methods, SIAM, 2011 Michael R. Overton, Numerical Computing with IEEE Floating Point Arithmetic, SIAM 2001 (suggested reading) James F. Epperson, Introduzione all'analisi numerica, teoria, metodi, algoritmi. McGraw-Hill, Milano, 2003 (suggested reading)
Notes to the books	Books will be integrated by lecture notes provided by the teacher.

Organization of the didactic activities

Hours			
Total	Lectures	Practice sessions	Individual study
150	32	30	88
CFU/ETCS			
6	4	2	

Teaching methods

Standard lectures

Expected learning outcomes

Knowledge and understanding	○ Techniques for developing software for solving mathematical problems Techniques for analyzing performances of numerical software
Applying knowledge and understanding	○ Optimization of algorithms with respect to features of the problem and computing resources availability. Development, documentation and testing of software and capability of interpretation of results.
Other skills	• *Making informed judgments and choices:* identify suitable methods for each specific problem. • *Communicating knowledge and understanding:* describing in a rigorous way and with proper language the problem, the method used to solve it and is main features. *Capacities to continue learning:* applying techniques and methods to slightly different problems.

Assessment	
Assesment methods	The examination consists of an oral exam, during which the student is asked questions on the course topics, including algorithm development, theoretical properties, implementation aspects, and related issues. In addition, in order to pass the exam, the student is required to produce programs (Python is recommended) covering all the topics and algorithms studied, together with a presentation of the results obtained. The student must implement studied algorithms, in particular those not already implemented during the lectures, and perform tests using meaningful data. The presentation of the numerical results, aimed at highlighting the key features of each algorithm, should be accompanied by appropriate comments. During the exam, the Python programs and the results of the numerical experiments will also be discussed. The codes, together with results and comments, must be included in a PDF or PowerPoint file and sent to the teacher by email a few days before the exam. Grades are given on a scale of 30.
Evaluation criteria	• *Knowledge and understanding*: students must show the understanding of main techniques for developing numerical software and they must be able to describe the main methods illustrated during the course. • *Autonomy of judgment*: students must show to be able to evaluate main features of each method and comparing performances of different methods • *Communication skills*: students must be able to present in an effective way the outcomes of their work on programming and testing numerical methods. • *Capacities to continue learning*: students must show to be able to apply main numerical technique to slightly different problems with respect to those illustrated during the course.

Measurements and final grade	Students must show to understand the main issues related to solving numerical problems, to develop methods and study their property. Students must be able to implement methods, test them and present, in an effective way, results from execution and testing. For the final mark will be also considered the ability to present in a correct and effective way methods and outcomes from programming tasks.
Further information	