Principali informazioni sull'insegnamento		
Denominazione dell'insegnamento	***Calcolo Numerico***	
Corso di studio	*Informatica e Tecnologie per la Produzione del Software*	
Anno di corso	*2023/2024*	
Crediti formativi universitari (CFU) / European Credit Transfer and Accumulation System (ECTS):		: 6
SSD	*MAT/09*	
Lingua di erogazione	*Italiano*	
Periodo di erogazione	*Secondo semestre 1/3/2023 – 7/6/2023*	
Obbligo di frequenza	*No, ma fortemente consigliata*	

Docente	
Nome e cognome	Francesca Mazzia; Antonella Falini
Indirizzo mail	francesca.mazzia@uniba.it; antonella.falini@uniba.it;
Telefono	0805443284
Sede	*Dipartimento di Informatica, V° piano*
Sede virtuale	*Team di Microsoft Teams per ricevimento: h0hat71*
Ricevimento (giorni, orari e modalità)	Venerdì 10--12 Si riceve anche in altri giorni previo appuntamento da richiedere via mail

Syllabus	
Obiettivi formativi	Il corso si propone come raccordo costruttivo fra la matematica e l'informatica, fornendo allo studente gli strumenti specifici di base per risolvere i problemi matematici usando il calcolatore mettendo in evidenza i rischi che si corrono con un uso ingenuo delle risorse di calcolo. Il corso presenta i metodi fondamentali per risolvere numericamente problemi matematici, mettendo in evidenza gli aspetti computazionali. Il programma comprende i metodi iterativi per equazioni non lineari, i metodi diretti per sistemi lineari, interpolazione, approssimazione, integrazione e calcolo di autovalori.
Prerequisiti	*Elementi di base di algebra lineare; calcolo differenziale e integrale per funzioni di una variabile; elementi di programmazione. L'insegnamento di Analisi Matematica è propedeutico all'insegnamento di Calcolo numerico*
Contenuti di insegnamento (Programma)	1. Introduzione al Calcolo Scientifico Modelli matematici e metodi numerici, sorgenti di errori, il processo di risoluzione numerica, efficienza, testing, errori computazionali, ambienti computazionali, linguaggi per il calcolo scientifico, problem solving enviroments: MATLAB, PYTHON. 2. Analisi dell'errore. 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. Stabilita' degli algoritmi. Propagazione degli errori. Introduzione a Python, il linguaggio, file di tipo script e function. Funzioni predefinite in NumPy. Introduzione alla grafica. Esempi Python sugli errori di arrotondamento.

	3. Elementi di algebra lineare Matrici e vettori. Operazioni con matrici. Inversa di una matrice. Sistemi lineari. Prodotto esterno fra vettori. Dipendenza lineare di vettori. Autovalori e autovettori. Norme di vettori e matrici. Spazi vettoriali lineari. Base di uno spazio vettoriale. Trasformazioni lineari e matrici. Sottospazi vettoriali. Spazio vettoriale generato da un insieme di vettori. Nucleo e immagine di una trasformazione lineare. Sottospazio ortogonale.
	4. Algoritmi per la soluzione di sistemi lineari Il Python per l'algebra lineare, Memorizzazioni di vettori e matrici, operazioni sulle matrici. Sistemi triangolari inferiori e superiori. Matrici di permutazione e proprietà. Algoritmo di eliminazione di Gauss. Problematiche di stabilità. Teorema di esistenza della fattorizzazione LU con pivot. Studio del condizionamento di un sistema lineare. Studio del residuo. Metodo dei minimi quadrati. Pseudoinversa. Esempi di codici Python, documentazione e testing, confronti con il software esistente. 5. Interpolazione e approssimazione Base delle potenze. Interpolazione di Lagrange. Interpolazione di Newton. Differenze divise. Interpolazione con nodi coincidenti. Interpolazione di Hermite. Errore nell'interpolazione polinomiale. Scelta dei nodi per l'interpolazione. Studio del condizionamento. Interpolazione lineare a tratti. Spline lineare. Approssimazione ai minimi quadrati nel discreto. Fitting di dati: polinomio di migliore approssimazione nel senso dei minimi quadrati. Retta di regressione lineare. Esempi di codici Python, documentazione e testing, confronti con il software esistente. 6. Calcolo degli zeri di funzione Metodo delle bisezioni. Convergenza. Criteri di arresto e stime dell'errore. Ordine di convergenza. Iterazione funzionale. Teorema di contrazione. Ordine di convergenza. Il metodo di Newton. Criteri di stop. Metodi quasi newtoniani. Metodo della direzione costante. Metodo della falsa posizione. Il metodo delle secanti. Metodo di Brent. Errore accuratezza e numero di condizione. Esempi di codici Python, documentazione e testing, confronti con il software esistente. 7. Calcolo degli autovalori. Metodo delle potenze. Applicazione: google page rank. Esempi di codici Python. 8. Calcolo degli integrali. Metodo dei trapezi, di Simpson, dei trapezi composto, di Simpson composto per il calcolo di integrali definiti. Stime dell'errore e metodo dei trapezi adattativo. Esempi di codici Python.
Testi di riferimento	*Uri M. Ascher, Chen Greif, A First Course in Numerical Methods, SIAM, 2011 (testo di riferimento)*
Note ai testi di riferimento	I libri di testo sono integrati con le slide, le dispense e gli appunti (elettronici) del docente distribuiti sulla piattaforma ADA e sul canale Teams della classe

Organizzazione della didattica			
Ore			
Totali	Didattica frontale	Pratica (laboratorio, campo, esercitazione, altro)	Studio individuale
150	*32*	*30*	*88*
CFU/ETCS			
6	*4*	*2*	

Metodi didattici	
	Lezioni frontali ed esercitazioni/laboratorio in aula

Risultati di apprendimento previsti	
Conoscenza e capacità di comprensione	○ Conoscere le tecniche e i metodi per la programmazione numerica finalizzati alla risoluzione di problemi nell'ambito delle discipline matematiche ed affini, con particolare enfasi ai problemi fondamentali nell'ambito dell'algebra lineare. ○ Comprendere e saper illustrare le problematiche relative dell'uso del calcolatore per la risoluzione di problemi matematici
Conoscenza e capacità di comprensione applicate	○ Capacità di risolvere problemi matematici mediante algoritmi ottimizzati dal punto di vista del costo computazionale e della stabilità. ○ Sviluppo delle capacità di programmare, documentare e testare algoritmi numerici, interpretandone correttamente i risultati. ○ Sviluppo delle capacità di risolvere problemi matematici usando problem solving enviroments.
Competenze trasversali	○ *Autonomia di giudizio:* Saper individuare il metodo numerico più idoneo per risolvere numericamente un problema matematico tra quelli trattati nel corso. ○ *Abilità comunicative:* Saper definire in modo rigoroso i problemi matematici trattati nel corso e saper esporre i relativi metodi numerici, delineandone le proprietà fondamentali. ○ *Capacità di apprendere in modo autonomo:* Capacità di studiare e risolvere problemi numerici simili ma non necessariamente uguali a quelli affrontati durante le lezioni.

Valutazione	
Modalità di verifica dell'apprendimento	L'esame consiste in una prova orale che verterà su tutti gli argomenti svolti a lezione, inclusi le parti teoriche (definizioni, teoremi e dimostrazioni) e gli esercizi ad esse relative. L'esame prevede anche la discussione relativa alla soluzione di alcuni problemi matematici usando gli algoritmi trattati a lezione ed implementati in ambiente Python.
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*

	scientifico 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	Lo studente deve comprendere e saper illustrare le problematiche relative dell'uso del calcolatore per la risoluzione dei problemi matematici analizzati durante il corso. Saper individuare il metodo numerico più idoneo per risolvere numericamente un problema matematico tra quelli trattati nel corso, conoscere le tecniche e i metodi per la programmazione numerica finalizzati alla sua risoluzione. Saper definire in modo rigoroso i problemi matematici trattati nel corso e saper esporre i relativi metodi numerici, delineandone le proprietà fondamentali.
Altro	
	Durante le lezioni verranno discussi, in modo partecipato, diversi quesiti ed esercizi simili per tipologia a quelli comunemente somministrati durante gli esami e gli esoneri. La finalità è duplice: monitorare in tempo reale lo stato di preparazione degli studenti frequentanti, perfezionandone la preparazione in vista dell'esame o degli esoneri; agevolare lo studio in itinere degli aspetti pratici della disciplina, motivando concretamente i corsisti a sostenere l'esame in tempi brevi, sfruttando possibilmente la modalità degli esoneri. Sono previsti due esoneri: il primo durante l'interruzione delle lezioni a metà corso, il secondo a fine corso. Entrambe le date sono concordate, nei limiti consentiti, con gli studenti frequentanti.

General information		
Academic subject	***Numerical computing***	
Degree course	*Informatica e Tecnologie per la Produzione del Software*	
Academic Year	*2023/2024*	
European Credit Transfer and Accumulation System (ECTS)		6
Language	*MAT/09*	
Academic calendar (starting and ending date)	*Second semester 1/3/2023 – 7/6/2023*	
Attendance	*Not mandatory, but strongly suggested*	

Professor/ Lecturer	
Name and Surname	Francesca Mazzia; Antonella Falini
E-mail	francesca.mazzia@uniba.it; antonella.falini@uniba.it;
Telephone	0805443284
Department and address	*Computer Science department, V° floor*
Virtual headquarters	*Microsoft Teams code for tutoring: h0hat71*
Tutoring (time and day)	Friday 10--12 (by appointment on other days)

Syllabus	
Learning Objectives	*The course aims to bridge the gap between mathematics and computer science by providing students with the fundamental tools to numerically solve mathematical problems by using computers.*
Course prerequisites	*Linear algebra; differential and integral calculus; programming skills. The exam of Analisi Matematica is mandatory to take this exam.*
Contents	1. Introduction to Scientific Calculus Mathematical models and numerical methods; errors; numerical process; efficiency; testing; computational errors; computational environments; problem solving environments; MATLAB; PYTHON. 2. Error Analysis. Machine representation of numebrs. IEEE single and double precision. Truncation and Rounding techniques. Machine precision. Absolute and relative error. Operations with floating numbers. Cancellation phenomenon. Conditioning. Algorithms stability. Error propagation analysis. Introduction to Python language, script and functions files. Primitive functions from NumPy and SciPy. Introduction to use the graphics tools. 3. Fundamentals of linear algebra Matrices and vectors. Matrix operations. Inverse matrix calculation. Linear systems. Inner and outer product. Linear dipendency. Eigenvalues and eigenvectos. Linear vector spaces. Basis for a vector space. Linear transformations. Subspaces. Kernel and Range spaces for a linear transformation. Orthogonal subspaces. 4. Linear systems algorithms Python for linear algebra. Storing of arrays and vectors, matrix operations. Triangular linear systems. Permutation matrices and properties. Gauss elimination algorithm. Stability issues. Theorem for the LU factorization existence. Conditioning of a linear system. Residual study. Least squares method. Pseudoinverse. Codes examples in Python, documentations and testing, comparisons with existing software. 5. Interpolation and approximantion Power basis. Lagrange interpolation. Newton interpolation. Divided differences. Hermite interpolation. Error for polynomial interpolation. Conditioning. Linear piecewise interpolation. Linear splines. Discrete least

	squares approximation. Data fitting: best polynomial approximation. Linear regression. Python codes examples, documentation and testing, comparisons with existing software. 6. Root finding algorithms Bisection method. Convergence analysis. Stopping criterion and error estimate. Order of convergence. Functional iteration methods. Order of convergence analysis. Local convergence theorem. Newton method. Stopping criteria. Quasi-newton methods. Hybrid methods: Brent method. Error analysis, accuracy evaluation and conditioning. Python codes examples, documentation and testing, comparisons with existing software. 7. Eigenvalues computation Power method. Application: Google Page Ranking. Python codes examples, documentation and testing, comparisons with existing software. 8. Numerical evaluation of integrals Trapezoidal rule. Simpson rule. Composite trapezoidal rule. Composite Simpson rule. Error estimates and adaptive trapezoidal rule. Python codes examples, documentation and testing, comparisons with existing software.

Books and bibliography	*Uri M. Ascher, Chen Greif, A First Course in Numerical Methods, SIAM, 2011 (testo di riferimento)* *L. Brugnano, C. Magherini, A. Sestini, Calcolo numerico, seconda edizione, Master, Università & Professioni, Firenze 2010.* *James F. Epperson, Introduzione all'analisi numerica, teoria, metodi, algoritmi. McGraw-Hill, Milano, 2003*
Additional materials	Books are integrated with slides and (electronic) lecture notes, shared on the E-learning ADA platform and on the MS Teams class channel.

Work schedule			
Total	Lectures	Hands on (Laboratory, working groups, seminars, field trips)	Out-of-class study hours/ Self-study hours
Hours			
150	*32*	*30*	*88*
ECTS			
6	*4*	*2*	
Teaching strategy	*Standard lectures and lab sessions in class*		
Expected learning outcomes			
Knowledge and understanding on:	○ Learn techniques and methodologies for numerical programming, especially in the context of numerical linear algebra. ○ Understunding and illustrate issues related to the use of computers to solve numerical problems.		
Applying knowledge and understanding on:	○ 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. ○ Probelm soving skills development.		
Soft 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 and feedback	
Methods of assessment	Oral exam on all the explained topics, including both theoretical parts (definitions, theorems and proffs) and practical exercises. Some mathematical problems will be required to be numerically solved by using the presented and implemented numerical algorithms.
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 mean features of each method and to be able of compare 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.
Criteria for assessment and attribution of the final mark	Students should understand and be able to explain the issues related to the use of computers in numerically solve the proposed mathematical problems. They also should be able to identify the most appropriate methodology among the presented ones and should know the algorithmic techniques used to solve the assayed problems. Students are also required to define rigorously the mathematical framework for every problem treated during the lectures and they also should illustrate the used numerical methods by explaining their fundamental properties.
Additional information	
	During every lecture, the teacher will encourage discussions on questions and typical exercises that will be assigned to students during the exams. There is a twofold benefit: the preparation of the students gets monitored and perfectioned; the learning process is levelled. The students will also be encouraged to do the exams at the end of the course by taking advantage of the mid-term tests. In particular, there are two midterms: the first one takes place during the first lecture break, the second one takes place at the end of the course. Both the dates are agreed beforehand with the attending students, whenever possible.