Review of exam topics

1. Interpolation: Lagrange form of the interpolating polynomial and Newton's divided difference formula;
2. Matrices and systems of linear equations: Cholesky factorisation and iteration methods (Jacobi and Gauss-Seidel);
3. Numerical differentiation and ordinary differential equations/IVP's. Euler and modified Euler formulas, Runge-Kutta methods.
4. Numerical integration: Newton-Cotes quadrature formulas, Gaussian quadrature
5. Eigenvalues and eigenvectors: The power method, Gershgorin circle theorem and the inverse power method;
6. Approximation theory: Chebyshev polynomials, economization of power series.

Lagrange interpolating polynomials

Let $x_0, x_1, x_2, \dots x_n$ be $n+1$ points and $f_i = f(x_i)$ the function values at these points. An interpolating polynomial P_n is a polynomial of degree at most n such that $P_n(x_i) = f_i$.

The Lagrange Form of the Interpolating polynomial is

$$P_n(x) = \sum_{i=0}^{n} L_{n,i}(x) f_i \qquad \text{where} \qquad L_{n,i}(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k}$$

and the interpolation error is given by the formula

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0) \cdots (x - x_n)$$

Examples

Consider the function $f(x) = \sin(x)$.

1. Construct the Lagrange form of the interpolating polynomial for f passing through the points $(0, \sin(0))$, $(\pi/4, \sin(\pi/4))$ and $(\pi/2, \sin(\pi/2))$.
2. Use this polynomial to estimate $\sin(\pi/3)$. What is the error in this approximation?
3. Establish the theoretical error bound for using the polynomial found in part (a) to approximate $\sin(\pi/3)$. How does this error compare with that in part (b)?

Newton's divided difference formula

The Newton form of the interpolating polynomial is

$$P(x) = \sum_{k=0}^{n} f[x_0, x_1, \ldots x_k] \left(\prod_{i=0}^{k-1} (x - x_i) \right)$$
$$= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) +$$
$$+ \cdots f[x_0, x_1, \ldots x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

where

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}; \qquad f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \quad \text{etc.}$$

are the divided differences of $f(x)$ with respect to the interpolating points.

Exercises

1. Repeat example on previous slide for Newton interpolation polynomial
2. Starting with the interpolating polynomial in the form

$$P(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots$$

derive the coefficients a_0, a_1, etc. and show they are given by the divided differences formulas.

Matrices and systems of linear equations

- A matrix A is called **positive definite** if it is symmetric and $X^T AX > 0$ for any vector X.
- A matrix is called **strictly diagonally dominant** if, for each row, the absolute value of the diagonal element is strictly larger than the sum of absolute values of the other elements.
- A symmetric matrix A is positive definite if all its eigenvalues are positive.
- A symmetric matrix A is positive definite if it strictly diagonally dominant and all its diagonal elements are positive.

Cholesky factorisation

If A is symmetric positive definite then A can be factorised in the form $A = LL^T$, where L is a lower triangular matrix with nonzero diagonal elements.

Example: Show the following matrices are symmetric positive definite and calculate the Cholesky factorisation.

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 6 & -2 & 3 \\ -2 & 8 & 1 \\ 3 & 1 & 7 \end{pmatrix}$$

Example: Hence, solve the systems

$$AX = M; \qquad BX = M$$

where $M = (1, 2, -1)^T$.

Iterative methods: Jacobi and Gauss-Seidel

Example: Consider the system of equations

$$
\begin{aligned}
5x_1 + x_2 + 2x_3 &= 10 \\
-3x_1 + 9x_2 + 4x_3 &= -14 \\
x_1 + 2x_2 - 7x_3 &= -33
\end{aligned}
$$

The Jacobi algorithm is given by

$$
\begin{aligned}
x_1^{(k+1)} &= \frac{1}{5}\,[10 - x_2^{(k)} - 2x_3^{(k)}] \\
x_2^{(k+1)} &= \frac{1}{9}\,[-14 + 3x_1^{(k)} - 4x_3^{(k)}] \\
x_3^{(k+1)} &= -\frac{1}{7}\,[-33 - x_1^{(k)} - 2x_2^{(k)}]
\end{aligned}
$$

while the Gauss-Seidel algorithm is given by

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{5}\,[10 - x_2^{(k)} - 2x_3^{(k)}] \\ x_2^{(k+1)} &= \frac{1}{9}\,[-14 + 3x_1^{(k+1)} - 4x_3^{(k)}] \\ x_3^{(k+1)} &= -\frac{1}{7}\,[-33 - x_1^{(k+1)} - 2x_2^{(k+1)}] \end{aligned}$$

Note: If A is strictly diagonally dominant then both the Jacobi and Gauss-Seidel algorithms will converge for any choice of the initial vector approximation!

Ordinary differential equations

Euler's method

Consider the first order IVP

$$y'(t) = f(t, y(t)), \qquad a \le t \le b$$
$$y(a) = \alpha$$

Euler's method is used to determine an approximation w to the exact solution $y(t)$ of this problem. The algorithm is

$$w_0 = \alpha$$
$$w_{i+1} = w_i + hf(t_i, w_i)$$

where $t_i = a + ih$, $i = 0, 1, 2, \ldots N$, $h = (b-a)/N$ and $w_i \approx y(t_i)$.

Exercise 1: Derive Euler's algorithm from the Taylor series expansion of the solution $y(t)$.

Exercise 2: The IVP

$$x'(t) = \frac{t}{x}, \quad 0 \leq t \leq 5, \qquad x(0) = 1$$

has the exact solution $x(t) = \sqrt{t^2 + 1}$. Find an approximation to this solution using Euler's method. Use a step size $h = 0.5$ and calculate the absolute error at each step.

Exercise 3: Redo Exercise 2 with a step size $h = 0.25$. What do you notice about the errors?

The classical fourth-order Runge-Kutta method

The algorithm is given by

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$\begin{aligned}
k_1 &= hf(t_i, w_i), \\
k_2 &= hf(t_i + \frac{h}{2}, w_i + \frac{k_1}{2}), \\
k_3 &= hf(t_i + \frac{h}{2}, w_i + \frac{k_2}{2}), \\
k_4 &= hf(t_i + h, w_i + k_3),
\end{aligned}$$

where w_i is the approximate solution at time $t_i = t_0 + ih$.

Example

1. Using the Euler method and a step size of $h = 0.25$, find an approximate solution for the initial value problem

$$\frac{dx}{dt} = 3t - \frac{x}{t}, \qquad 1 \leq t \leq 3,$$
$$x(1) = 2.$$

   Given that the exact solution is $x(t) = t^2 + \frac{1}{t}$, calculate the approximation error at each step.

2. Approximate the solution of the problem in part (b) using the fourth order Runge-Kutta method with a step size of $h = 0.25$. How do the absolute errors in this case compare with those obtained from the Euler's method?

Numerical Integration

1. Derivation of open and closed Newton-Cotes formulas for $n = 1$, $n = 2$, $n = 3$ starting with the general Newton-Cotes quadrature formula and using Lagrange interpolating polynomials.
2. Composite Newton-Cotes formulas. Derivation of composite trapezoidal and Simpson's rule starting with the standard ones (with error terms).
3. Numerical verification of rates of convergence for composite rules.
4. Using the error term for a composite rule to calculate the number of subintervals required for a specifoed degree of accuracy.
5. Gaussian quadrature formulas (up to 3 points). Converting the interval of integration [-1,1] to a general interval $[a, b]$.