1 A1

1.1 Formulas

Taylor Series

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3$$
$$+ ... + \frac{f^n(a)}{n!}(x-a)^n$$

Maclaurin series

$$f(x) = f(0) + f'(0)(x-0) + \frac{f''(0)}{2!}(x-0)^2 + \frac{f'''(0)}{3!}(x-a)^3$$
$$+ ... + \frac{f^n(0)}{n!}(x-0)^n$$

$$sinx = \tfrac{x}{1!} - \tfrac{x^3}{3!} + \tfrac{x^5}{5!} - \tfrac{x^7}{7!} + ...$$

or

$$sinx = \textstyle\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}(x)^{2n+1}$$
$$cosx = 1 - \tfrac{x^2}{2!} + \tfrac{x^4}{4!} - \tfrac{x^6}{6!} + \tfrac{x^8}{8!} - ...$$

or

$$cosx = \textstyle\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}(x)^{2n}$$
$$e^x = 1 + \tfrac{x}{1!} + \tfrac{x^2}{2!} + \tfrac{x^3}{3!} + \tfrac{x^4}{4!} + ...$$

or

$$e^x = \textstyle\sum_{n=0}^{\infty} \frac{(x)^n}{(n)!}$$

2 A2

2.1 Formulas

Trapezoidal

$$\int_a^b f(x)dx \approx \frac{h}{2}\Big[f(a) + 2\sum_{j=1}^{n-1}[f(a+jh) + f(b)\Big]$$

error in trap

$$= -\frac{1}{12}h^3 n y''(\bar{x}) = -\frac{b-a}{12}h^2 y''(\bar{x})$$

Simpson

$$= \frac{h}{3}\Big[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\Big]$$

Where, $x_j = a + jh$ for $j = (0, 1, ..., n-1, n)$

$$error = -\frac{b-a}{180} h^4 y^{iv}(\bar{x})$$

simpson 3/8

$$\boxed{\int_{x_0}^{x_n} f(x)dx = \frac{3h}{8}\Big(y_0 + 3y_1 + 3y_2 + 2y3 + 3y_4 + 3y_5 + 2y_6 + ... + 2y_{n-3} + 3y_{n-2} + 3y_{n-1} + y_n)\Big]}$$

Gauss legendre

$$\boxed{\int_a^b f(x)dx = \int_{-1}^{1} f(\frac{b-a}{2}x + \frac{b+a}{2})\frac{b-a}{2}dx}$$

2.2 Inbuilt functions

np . polynomial . legendre . leggauss (n)
Takes the value of n for n point fomrula . return two array with weight and x values
scipy.integrate.quad(func, a, b)
takes func (the function to be integrated), a (lower limit) , b (upper limit), return the integration of function from a to b

3 A3

3.1 Formulas

1. **Forier series expansion:**

$$f(x) = a_0 + \Big[\sum_{n=1}^{\infty} a_n cos(\frac{n\pi x}{L}) + b_n sin(\frac{n\pi x}{L})\Big]$$

   Where,

$$\boxed{a_0 = \frac{1}{2L}\int_{-L}^{L} f(x)dx}$$

$$\boxed{a_n = \frac{1}{L}\int_{-L}^{L} f(x)cos(\frac{n\pi}{L}xdx}$$

$$\boxed{b_n = \frac{1}{L}\int_{-L}^{L} f(x)sin(\frac{n\pi}{L}xdx}$$

2. **Half range sine series**

$$\sum_{n=1}^{\infty} b_n sin(\frac{n\pi x}{L})$$

where,

$$\boxed{b_n = \frac{2}{L}\int_0^L f(x)sin(\frac{n\pi x}{L})dx}$$

3. **Half range cosine series**

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n cos(\frac{n\pi x}{L})$$

for $n = 1, 2, 3, \ldots$, where

$$a_0 = \frac{2}{L}\int_0^L f(x)dx$$

$$a_n = \frac{2}{L}\int_0^L f(x)cos(\frac{n\pi x}{L})dx$$

and

$$b_n = 0$$

3.1.1 Inbuilt Functions

kuch nahi hai

4 A4

4.1 Formulas

Algorithm 1 n-point Gauss Laguerre Quadrature rule

function MyLaguQuad(f, n)
  $[lagu\_zer, w] = l\_roots(n)$ ▷ Store the x values and weights in two lists
  lagu = 0 ▷ initialize the summation
  for i in range(1, n+1):
    $lagu+ = f(lagu\_zer[i-1]) * w[i-1]$ ▷ loop to sum all values for the integral
  return $lagu$ ▷ Returns the value of integral

4.2 Inbuilt Functions

- from scipy.special.orthogonal import l_roots
  Usage
  l_roots(n), where n is the number of points.
  Output
  List of zeros and weights respectively.
- from scipy.integrate import quad
  Usage
  scipy.integrate.quad(func, a, b, args=(), full_output=0, epsabs=1.49e-08, epsrel=1.49e-08, limit=50, points=None, weight=None, wvar=None, wopts=None, maxp1=50, limlst=50), where func is the function to be integrated, a and b are the lower and upper limit respectively.
  Output
  The integral of func from a to b.

5 A5

Algorithm 2 n-point Gauss Hermite Quadrature rule

function MyHermiteQuad(f, n)
  $[herm\_zer, w] = h\_roots(n)$ ▷ Store the x values and weights in two lists
  herm = 0 ▷ initialize the summation

  for i in range(1, n+1):
    $herm+ = f(herm\_zer[i-1]) * w[i-1]$ ▷ loop to sum all values for the integral
  return *herm* ▷ Returns the value of integral

5.1 Inbuilt Functions

- from scipy.special.orthogonal import h_roots
  Usage
  h_roots(n), where n is the number of points.
  Output
  List of zeros and weights respectively.

- from scipy.integrate import quad
  Usage
  scipy.integrate.quad(func, a, b, args=(), full_output=0, epsabs=1.49e-08, epsrel=1.49e-08, limit=50, points=None, weight=None, wvar=None, wopts=None, maxp1=50, limlst=50), where func is the function to be integrated, a and b are the lower and upper limit respectively.
  Output
  The integral of func from a to b.

6 A6

6.1 Formulas

Weighted Least Square Fitting

$$\boxed{m = \frac{\Sigma w_i \Sigma w_i x_i y_i - \Sigma w_i y_i \Sigma x_i}{\Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2}}$$

$$\boxed{c = \frac{\Sigma w_i x_i^2 \Sigma w_i y_i - \Sigma w_i x_i \Sigma w_i x_i y_i}{\Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2}}$$

$$\boxed{\Delta = \Sigma w_i \Sigma w_i x_i^2 - (\Sigma w_i x_i)^2, S_{xy} = \sum w_i x_i y_i}$$

$$\boxed{\sigma_m = \sqrt{\frac{\Sigma w_i}{\Delta}}}$$

$$\boxed{\sigma_c = \sqrt{\frac{\Sigma w_i x_i^2}{\Delta}}}$$

$$\boxed{r = \frac{\Sigma w_i (x_i - \overline{X})(y_i - \overline{Y})}{\sqrt{\Sigma w_i (x_i - \overline{X})^2 \Sigma w_i (y_i - \overline{Y})^2}}}$$

Least Square Fitting

$$\boxed{m = \frac{\Sigma x_i \Sigma y_i - N * \Sigma x_i y_i}{\Sigma w_i \Sigma x_i^2 - N * (\Sigma w_i x_i)^2}}$$

$$\boxed{c = \frac{\Sigma x_i^2 \Sigma y_i - \Sigma x_i \Sigma x_i y_i}{N * \Sigma x_i^2 - (\Sigma x_i)^2}}$$

$$\boxed{y\_calc_i = x_i * m + c}$$

$$\boxed{\sigma_m = \sqrt{\frac{N * \Sigma (y_i - y\_calc_i)^2}{\Sigma xisq - ((\Sigma x_i) * *2) * (N - 2))}}}$$

$$\boxed{\sigma_c = \sqrt{\frac{(\sigma m)^2 * N}{\Sigma xisq}}}$$

$$\boxed{r = \sqrt{\frac{(\Sigma x_i y_i)^2}{\Sigma x_i^2 \Sigma y_i^2}}}$$

6.2 Inbuilt Functions

scipy.stats.linregress(x, y)
Parameters x, y :array_like
Returns (slope,intercept,rvalue,pvalue,slope_stderr,intercept_stderr)

7 A7

7.1 Formulas

DIRAC DELTA FUNCTION

- Pulse function

$$\delta_\epsilon(x-a) = \begin{cases} \frac{1}{2\epsilon} & \text{if } -\epsilon < x-a < \epsilon \\ 0 & \text{otherwise.} \end{cases}$$

- Gaussian Function

$$\delta_\epsilon(x-a) = \frac{1}{\sqrt{2\pi\epsilon}} e^{-(x-a)^2/(2\epsilon)}$$

- Lorentz form

$$\delta_\epsilon(x) = \frac{\epsilon}{\pi(x^2+\epsilon^2)}$$

- Exponential form

$$\delta_\epsilon(x) = \frac{e^{-|x|/\epsilon}}{2\epsilon}$$

- Sine form

$$\delta_\epsilon(x) = \frac{sin(x/\epsilon)}{\pi x}$$

- secant hyperbola form

$$\delta_\epsilon(x) = \frac{sech^2(x/\epsilon)}{2\epsilon}$$

7.2 Inbuilt Functions

8 A8

FORMULAS:

1) EULER METHOD:

$$Y_n = Y_{n-1} + hF(X_{n-1}, Y_{n-1})$$

2) RK-4 METHOD:

$$K_1 = hf(x_n, y_n)$$
$$K_2 = hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2})$$
$$K_3 = hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2})$$
$$K_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + k_1/6 + k_2/3 + k_3/3 + k_4/6$$

3) RK-2 METHOD:

$$K_1 = h\ f(t, x)$$
$$K_2 = h\ f(t+h, x+K_1)$$
$$x(t+h) = x(t) + \frac{1}{2}(K_1 + K_2)$$

ALGORITHMS

1) *EULER:*

1. define f(x,y)
2. input x_0, y_0
3. input h, n
4. for j from 0 to (n-1) do
   - $y_{j+1} = y_j + hf(x_j, y_j)$
   - $x_{j+1} = x_j + h$
   - Print x_{j+1} and y_{j+1}
5. end

2) *RK-4 METHOD:*

i) Define f(t,y)

ii) For t= $t_0, t_1, t_2, \ldots\ldots\ldots, t_f$:

$$t_{n+1} = t_n + h$$

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f(t_n + h, y_n + hk_3).$$

$$y_{n+1} = y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

3) *RK-2 METHOD:*

i) Define f(t,x)

ii) For t= $t_0, t_1, t_2, \ldots\ldots\ldots, t_f$:

$$t_{n+1} = t_n + h$$

$$K_1 = h\ f(t, x)$$
$$K_2 = h\ f(t+h, x+K_1)$$
$$x(t+h) = x(t) + \frac{1}{2}(K_1 + K_2)$$

9 A9

9.1 formulas

1. **Dircihlet condition**

$$y(x) = y_1(x) + cy_2(x)$$
$$y_1 \implies y(a) = \alpha, y'(a) = 0; y_2 \implies r(x) = 0, y(a) = 0, y'(a) = 1$$
$$c = \frac{\beta - y_1(b)}{y_2(b)}$$

2. **Neumann condition**

$$y(x) = y_1(x) + cy_2(x)$$
$$y_1 \implies y(a) = 0, y'(a) = \alpha; y_2 \implies r(x) = 0, y(a) = 1, y'(a) = 0$$
$$c = \frac{\beta - y_1'(b)}{y_2'(b)}$$

3. **Robin Condition**

$$y(x) = y_1(x) + c_1y_2(x) + c_2y_3(x)$$
$$y_1 \implies y(a) = 0, y'(a) = 0; y_2 \implies r(x) = 0, y(a) = 1, y'(a) = 0; y_3 \implies r(x) = 0, y(a) = 0, y'(a) = 1$$
$$\alpha_1c_1 + \alpha_2c_2 = \alpha_3$$
$$[\beta_1y_2(b) + \beta_2y_2(b)]c_1 + [\beta_1y_3(b) + \beta_2y_3(b)]c_2 = \beta_3 - \beta_1y_1(b) - \beta_2y_1(b)$$

9.2 Inbuilt functions

1. **from scipy import stats**

stats.linregress(x,y = None)
Calculate a linear least-squares regression for two sets of measurements.
Parameters: x,y : Array like
Returns: slope,intercept,rvalue,pvalue,stderr,intercept_stderr:float
rvalue: Pearson correlation coefficient
stderr: standard error of estimated slope

10 A10

10.1 Formulas

1. **Dirichlet condition**
solve for

$$y(a) = \alpha, y(b) = \beta \qquad y'(a) = s$$
$$\phi(s) = \beta - y(b, s) = 0$$

2. **Neumann condition**
solve for

$$y'(a) = \alpha, y'(b) = \beta \qquad y(a) = s$$
$$\phi(s) = \beta - y'(b, s) = 0$$

3. **Robin condition**
   solve for

$$\alpha_1 y(a) + \alpha_2 y'(a) = \alpha_3, \beta_1 y(b) + \beta_2 y'(b) = \beta_3 \qquad y(a) = s \quad or \quad y'(a) = s$$
$$\phi(s) = \beta_3 - \beta_1 y(b,s) - \beta_2 y'(b,s) = 0$$

$$s_k = s_{k-1} - \phi(s_{k-1}) \left[\frac{s_{k-1} - s_{k-2}}{\phi(s_{k-1}) - \phi(s_{k-2})} \right]$$
$$Stop, |\phi(s)| < tol$$

10.2 Inbuilt Functions

1. **from scipy.optimize import fsolve**

fsolve(func,x_0)
Find the roots of a function.
Parameters: func,x_0
func: Callable(x,*args) - function takes one arguemnt(maybe a vector if there are two or more than two variables)
x_0:ndarray - Initial guess for the roots
Return:x - ndarray : The solution

11 A11

11.1 Formulas

General Matrix Formulation of linear BVP with linear BC

$$y''(x) = p(x)y(x) + q(x)y'(x) + r(x)$$
$$x\epsilon[a,b]$$

$A\omega = B$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & & & & \\ l_1 & d_1 & u_1 \dots & & & & \\ & l_2 & d_2 & u_2 \dots \vdots & \ddots & & \\ & & & & l_{n-1} & d_{n-1} & u_{n-1} \\ & & & & & a_{n+1,n} & a_{n+1,n+1} \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ -h^2 r_1 \\ -h^2 r_2 \\ \vdots \\ -h^2 r_{n-1} \\ b_{n+1} \end{bmatrix} \qquad \omega = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \\ \omega_{n+1} \end{bmatrix}$$

$d_i = 2 + h^2 p_i$, $u_i = -1 + \frac{h}{2} q_i$, $l_i = -1 - \frac{h}{2} q_i$

$$a_{11} = \begin{cases} 1, & \text{Dirichlet BC at x=a} \\ \mathrm{d}_0, & \text{Neumann BC at } x = a \\ \mathrm{d}_0 + 2hl_0 \dfrac{\alpha_1}{\alpha_2}, & \text{Robin BC at } x = a \end{cases}$$

$$a_{12} = \begin{cases} 0, & \text{Dirichlet BC at x=a} \\ -2, & \text{Otherwise} \end{cases}$$

$$a_{N+1,N+1} = \begin{cases} 1, & \text{Dirichlet BC at x=b} \\ \mathrm{d}_N, & \text{Neumann BC at x=b} \\ \mathrm{d}_N + 2hu_N\dfrac{\beta_1}{\beta_2}, & \text{Robin BC at x=b} \end{cases}$$

$$a_{N+1,N} = \begin{cases} 0, & \text{Dirichlet BC at x=b} \\ -2, & \text{Otherwise} \end{cases}$$

$$b_1 = \begin{cases} \alpha, & \text{Dirichlet BC at x=a} \\ \text{-h}^2 r_0 + 2hl_0\alpha, & \text{Neumann BC} \\ \text{-h}^2 r_0 + 2hl_0\dfrac{\alpha_3}{\alpha_2}, & \text{Robin BC} \end{cases}$$

$$b_{N+1} = \begin{cases} \beta, & \text{Dirichlet BC at x=b} \\ \text{-h}^2 r_N + 2hu_N\beta, & \text{Neumann BC} \\ \text{-h}^2 r_N + 2hu_N\dfrac{\beta_3}{\beta_2}, & \text{Robin BC} \end{cases}$$

- DBC : $y(a) = \alpha, y(b) = \beta$
- NBC : $y'(a) = \alpha, y('b) = \beta$
- RBC : $\alpha_1 y(a) + \alpha_2 y'(a) = \alpha_3, \beta_1 y(b) + \beta_2 y'(b) = \beta_3$

11.2 Inbuilt Functions

1. scipy.sparse.diags : Construct a sparse matrix from diagonals.
parameters : diagonals, offsets=0, shape=None, format=None, dtype=None

2. numpy.linalg.solve(a, b)
Solve a linear matrix equation, or system of linear scalar equations.