|  |  |
| --- | --- |
| public class RK4Order1 {  static double myFunction(double x, double y) {  *//y = y - 2x;* return (y - 2 \* x \* y);  }   static void rungeKutta(double start\_x, double start\_y, double step, double end\_x){   double x = start\_x;  double y = start\_y;   int n = (int)((end\_x - start\_x) / step);  System.*out*.println(n);  System.*out*.println("X\t\t\t\tY");  System.*out*.printf("x = %.3f and y = %.6f\n", x, y);   for(int i = 0; i < n; i++){   double k1 = step \* *myFunction*(x, y);  double k2 = step \* *myFunction*(x+(step/2), y+(k1/2));  double k3 = step \* *myFunction*(x+(step/2), y+(k2/2));  double k4 = step \* *myFunction*(x+step, y+k3);  double k = (k1 + (2\*k2) + (2\*k3) + k4) / 6;   y += k;  x += step;  System.*out*.printf("x = %.3f and y = %.6f\n", x, y);  }  }    public static void main(String[] args){   double init\_x = 0.0;  double init\_y = 1.0;  double step = 0.2;  double final\_x = 2.0;   *rungeKutta*(init\_x, init\_y, step, final\_x);   }  *//int n = (int)((xn - x0) / h);* } | public class NumericalIntegration {  static double myFunction(double input) {  *//System.out.println("log(" + input + "): " +Math.log(input));* return Math.*log*(input);  }  static double trapezoidal(double l, double u, int inters) {  double result = 0;  double h = (u - l) / inters;  result += 0.5 \* (*myFunction*(u) + *myFunction*(l));  for (int i = 1; i < inters; i++) {  result += *myFunction*(l + i \* h);  }  return (h \* result);  }  static double simpsonOneEight(double l, double u, int inters) {  double result = 0;  double h = (u - l) / inters;  result += *myFunction*(u) + *myFunction*(l);  for (int i = 1; i < inters; i++) {  if (i % 2 == 0) {  result += 2 \* *myFunction*(l + i \* h);  } else {  result += 4 \* *myFunction*(l + i \* h);  }  }  return ((h \* result) / 3);  }  static double simpsonThreeEight(double l, double u, int inters) {  double result = 0;  double h = (u - l) / inters;  result += *myFunction*(u) + *myFunction*(l);  for (int i = 1; i < inters; i++) {  if (i % 3 == 0) {  result += 2 \* *myFunction*(l + i \* h);  } else {  result += 3 \* *myFunction*(l + i \* h);  }  }  return (3 \* ((h \* result) / 8));  }   public static void main(String[] args) {  *//Scanner myScanner = new Scanner(System.in);* double lower = 4;  double upper = 5.2;  int intervals = 6; *//n* double straight = *myFunction*(upper) - *myFunction*(lower);  System.*out*.println("Straight Answer: " + straight);  double trap = *trapezoidal*(lower, upper, intervals);  System.*out*.println("Trapezoidal: \t" + trap);  double simpOE = *simpsonOneEight*(lower, upper, intervals);  System.*out*.println("Simpson One Eighth: " + simpOE);  double simpTE = *simpsonThreeEight*(lower, upper, intervals);  System.*out*.println("Simpson Three Eighth: " + simpTE);  } } |
| import java.util.Arrays; import java.util.Scanner;  import static java.lang.System.*exit*;  public class gauss1 {   static void printM(double[][] matrix) {  for (double[] row : matrix) {  System.*out*.println(Arrays.*toString*(row));  }  }   public static void main(String[] args) {   int n\_variables;  Scanner myScanner = new Scanner(System.*in*);  System.*out*.println("Enter the number of variables:");  n\_variables = myScanner.nextInt();  System.*out*.println("Size of augmented matrix is :" + n\_variables + " rows and " + (n\_variables + 1) + " columns");   double[][] aug\_mat = new double[n\_variables][n\_variables + 1];  System.*out*.println("Enter the values of augmented matrix:");  for (int i = 0; i < n\_variables; i++) {  for (int j = 0; j < (n\_variables + 1); j++) {  System.*out*.print("Enter vale for aug mat: " + (i + 1) + " and " + (j + 1) + ": ");  aug\_mat[i][j] = myScanner.nextDouble();  }  }   System.*out*.println("Augmented matrix: ");  *printM*(aug\_mat);   *//elimination process* for (int i = 0; i < n\_variables; i++) {   if (aug\_mat[i][i] == 0) {  System.*out*.println("Mathematical Error");  *exit*(0);  }  for (int j = i + 1; j < n\_variables; j++) {  double x = aug\_mat[j][i];  for (int k = 0; k < (n\_variables + 1); k++) {  */\*  cross multiply with first co-efficients ->  r[i-1]\*r[1][0] - r\*r[0][0]  \*/  //System.out.println(aug\_mat[j][k] + " \* " + aug\_mat[i][i] + " - " + aug\_mat[i][k] + " \* " + aug\_mat[j][i]);* aug\_mat[j][k] = aug\_mat[j][k] \* aug\_mat[i][i] - aug\_mat[i][k] \* x;  }  }  }  System.*out*.println("Updated Augmented matrix: ");  *printM*(aug\_mat);  *//solution by back substitution* double[] soln = new double[n\_variables];  *//get z* soln[n\_variables - 1] = aug\_mat[n\_variables - 1][n\_variables] / aug\_mat[n\_variables - 1][n\_variables - 1];   *//get the remaining by back substitution iteratively* for (int i = (n\_variables - 2); i >= 0; i--) {  soln[i] = aug\_mat[i][n\_variables];  for (int j = i + 1; j < n\_variables; j++) {  soln[i] -= soln[j] \* aug\_mat[i][j];  }  soln[i] /= aug\_mat[i][i];  }  System.*out*.println("Solution: ");  System.*out*.println(Arrays.*toString*(soln));  } }  //Chi Square Test  #include <stdio.h>  #include <math.h>  int main() {  // Given data  int accidents[6] = {14, 18, 12, 11, 15, 14};  int days = 6;  double chi\_square = 0.0;  double expected\_frequency;  double table\_value = 11.07; // Given table value for 5% significance level, 5 d.f.    // Calculate total number of accidents  int total\_accidents = 0;  for(int i = 0; i < days; i++) {  total\_accidents += accidents[i];  }  // Calculate expected frequency for uniform distribution  expected\_frequency = (double)total\_accidents / days;  // Calculate chi-square statistic  for(int i = 0; i < days; i++) {  chi\_square += pow(accidents[i] - expected\_frequency, 2) / expected\_frequency;  }  // Output the result  printf("Calculated Chi-Square value: %.2f\n", chi\_square);  printf("Table value for Chi-Square at 5%% level with 5 d.f.: %.2f\n", table\_value);  // Check if the calculated value is less than the table value  if(chi\_square < table\_value) {  printf("The accidents are uniformly distributed over the week.\n");  } else {  printf("The accidents are not uniformly distributed over the week.\n");  }  return 0;  } | //t-test  #include <stdio.h>  #include <math.h>  int main()  {  int i, n;  float x[20], sum = 0, sum1=0, xbar, ttab, tcal, mu,se;    printf("Enter the population mean: ");  scanf("%f", &mu);  printf("Enter the sample size : ");  scanf("%d",&n);  printf("Enter the sample values: ");  for(i=0; i < n; i++){  scanf("%f",&x[i]);  sum=sum+x[i];  }    printf("%f", sum);    xbar=sum/n;  printf("Mean of Sample: \n%f", xbar);    //calculate standard deviation  for(i=0; i < n; i++){  sum1=sum1+(x[i]-xbar)\* (x[i]-xbar);  }  se=sqrt(sum1/(n-1));  //calculate t-value  tcal =fabs((xbar-mu)/(se/sqrt(n-1)));  printf("\nEnter the t-table value: ");  scanf("%f", &ttab);  printf("Results: \n");  printf("tcal=%f \n tab=%f \n df=%d \n los = 0.05\n", tcal,ttab,n-1);  if (tcal < ttab){  printf("\nAccept Null hypothesis H0, The given samples drawn from the assumed population of mean %f", mu);  }else{  printf("\nReject the null hypotheses H0 , the given samples are not drawn from the assumed population of mean value %f", mu);  }    return 0;  }  //f-test  #include <stdio.h>  #include <math.h>  // Function to calculate mean  double calculate\_mean(double data[], int n) {  double sum = 0.0;  for (int i = 0; i < n; i++) {  sum += data[i];  }  return sum / n;  }  // Function to calculate standard deviation  double calculate\_standard\_deviation(double data[], int n, double mean) {  double sum = 0.0;  for (int i = 0; i < n; i++) {  sum += pow(data[i] - mean, 2);  }  return sqrt(sum / (n - 1));  }  int main() {  int n1, n2;  double ftable;  // Read the number of items in the sample1  printf("Enter the number of items in the sample 1: ");  scanf("%d", &n1);    // Read the number of items in the sample2  printf("Enter the number of items in the sample 2: ");  scanf("%d", &n2);  // Read the items in the sample 1  double data1[n1];  printf("Enter the items in the sample 1:\n");  for (int i = 0; i < n1; i++) {  scanf("%lf", &data1[i]);  }    // Read the items in the sample 1  double data2[n2];  printf("Enter the items in the sample 2:\n");  for (int i = 0; i < n2; i++) {  scanf("%lf", &data2[i]);  }    // Calculate the sample means  double mean1 = calculate\_mean(data1, n1);  double mean2 = calculate\_mean(data2, n2);  // Calculate the sample standard deviations  double sd1 = calculate\_standard\_deviation(data1, n1, mean1);  double sd2 = calculate\_standard\_deviation(data2, n2, mean2);    // Calculate the f-value  double f\_value = sd1 > sd2 ? (sd1/sd2) : (sd2/sd1);  // Display the results  printf("Calculated f-value: %.2f\n", f\_value);  printf("Enter the table value of f: ");  scanf("%lf", &ftable);    printf("\nDegrees of freedom: %d, %d\n", (n1-1), (n2-1));  if(f\_value > ftable){  printf("\nNull hypothesis is rejected");  }else{  printf("\nNull hypothesis is accepted");  }  return 0;  } |
| //regression equations  import java.util.Scanner;  public class RegressionEq {   *//read x, y, x values, y values  //calculate sumx, sumy, sumx\*sumy, sum(x square), (sumx square), sum(x\*y)  //a = ([n \* (sum(x\*y))] - [sumx \* sumy]) / ([n \* sum(x square)] - (sumx square)  //b = (sumy - a\*sumx) / n;* public static void main(String[] args) {  int n;  Scanner myScanner = new Scanner(System.*in*);  System.*out*.println("Enter number of x values: ");  n = myScanner.nextInt();   double[] x\_values = new double[n];  double[] y\_values = new double[n];   System.*out*.println("Enter value for x: ");  for(int i = 0; i < n; i++){  System.*out*.printf("x[%d]: ", i);  x\_values[i] = myScanner.nextDouble();  }   System.*out*.println("Enter value for y: ");  for(int i = 0; i < n; i++){  System.*out*.printf("y[%d]: ", i);  y\_values[i] = myScanner.nextDouble();  }   double sumx2 = 0, sumx = 0, sumy = 0, sumxy = 0;  for(int i = 0; i < n; i++){  sumx += x\_values[i];  sumx2 += x\_values[i] \* x\_values[i];   sumy += y\_values[i];  sumxy += x\_values[i] \* y\_values[i];  }   double sq\_sumx = sumx \* sumx;  double a = ((n \* sumxy) - (sumx\*sumy)) / ((n \* sumx2) - sq\_sumx);  double b = (sumy - a\*sumx)/n;  double c = 1 / a;  double d = -b / a;   System.*out*.printf("Equation 1: y = %8.5fx + %8.5f%n", a, b);  System.*out*.printf("Equation 2: x = %8.5fy + %8.5f", c, d);   } } |  |
|  |  |
|  |  |