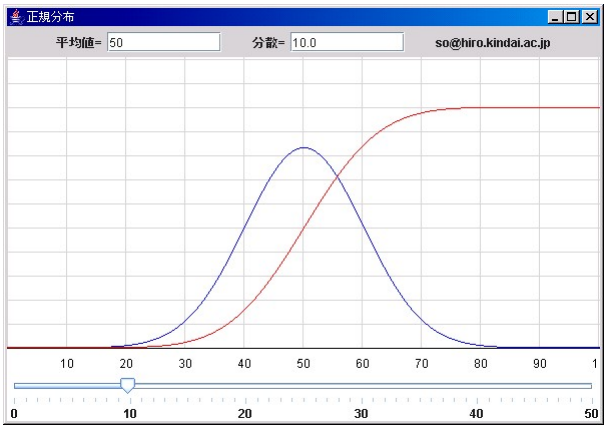
正規分布




```
/*
 * 作成日: 2006/08/07  @author 徐　丙鉄
 * 正規分布の確率密度関数/分布関数の表示
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class NormalDistGraph extends JApplet implements
ActionListener,ChangeListener{

        double meanValue=50.0;  //平均値の初期値
        double sigma=10.0;      //分散の初期値

        JTextField meanValueJTxtF;      // 平均値の入力欄
        JTextField sigmaJTxtF ;         // 分散の値の入力欄
        JSlider sigmaJSlider;           // 分散の値のスライダー
        GraphJPanel graphJPanel;        //グラフ描画領域

    //--- applet の初期化 init()  その他に必要であれば  start(), stop(), destroy();
        public void init(){
                Container c = this.getContentPane();  //アプレット内の表示領域の取得
        //--- パネル：ラベル, ボタン, テキストフィールド
                JPanel control_panel = new JPanel();
                        control_panel.setBackground(Color.lightGray);
                JLabel meanJLabel = new JLabel("平均値=");
                meanValueJTxtF = new JTextField("50",10);
                        meanValueJTxtF.addActionListener(this);
                JLabel sigmaJLabel = new JLabel("    分散=");
                sigmaJTxtF = new JTextField("10",10);
                        sigmaJTxtF.addActionListener(this);
                JLabel myJLabel = new JLabel("    so@hiro.kindai.ac.jp");
                control_panel.add(meanJLabel);   control_panel.add(meanValueJTxtF);
                control_panel.add(sigmaJLabel);  control_panel.add(sigmaJTxtF);
                control_panel.add(myJLabel);
        //--- グラフ描画パネル
                graphJPanel = new GraphJPanel(); //パネル：グラフ描画エリア
                        graphJPanel.setBackground(Color.lightGray);
        //--- 分散スライダー
                sigmaJSlider = new JSlider(JSlider.HORIZONTAL, 0, 50, 10); //分散スライダー
                        sigmaJSlider.setPaintLabels(true);        sigmaJSlider.setMajorTickSpacing(10);
                        sigmaJSlider.setMinorTickSpacing(1);      sigmaJSlider.setPaintTicks(true);
                        sigmaJSlider.addChangeListener(this);
          //画面レイアウト：北（ラベル, テキストフィールド, ボタン), 中央（グラフ）,南（分散：スライダー）
                        c.add(control_panel,BorderLayout.NORTH);
                                c.add(graphJPanel,BorderLayout.CENTER);
                                c.add(sigmaJSlider,BorderLayout.SOUTH);
                        this.setSize(600,400);
                }

        public static void main( String[] args ) {
                NormalDistGraph nd = new NormalDistGraph();
                        nd.init(); nd.start();
                JFrame windowJFrame = new JFrame("正規分布");
                        windowJFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                        windowJFrame.getContentPane().add(nd);
                        windowJFrame.setBounds(200,100,600,420); // ウインドウの表示位置とサイズ
                        windowJFrame.setVisible(true);
        }
    //---  平均と分散の値が変更された場合：java.awt.event.ActionListener ---
        public void actionPerformed(ActionEvent e){
                meanValue = Double.parseDouble( meanValueJTxtF.getText() );
                sigma = Double.parseDouble( sigmaJTxtF.getText() );
```

```
                graphJPanel.setParameters( meanValue, sigma);
                sigmaJSlider.setValue( (int)sigma );
        }

    //---          スライダーのアクションリスナー ---
        public void stateChanged(ChangeEvent e) {
                sigma = (double)( sigmaJSlider.getValue() );
                if( sigma > 0){ graphJPanel.setParameters( meanValue, sigma);
                                sigmaJTxtF.setText(""+sigma); }
        }
}
```

```
/*
 * 作成日: 2006/08/05  @author 徐　丙鉄
 * グラフ表示用 JPanel
 */

import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JPanel;

public class GraphJPanel extends JPanel{

        double mean ; double sigma ;
        double x_min = 0.0; double x_max = 100.0; //変域
        int H = 320; int W=600;  //描画領域の縦・横の初期値
        int H0=(H/12)*11; int H10=(H/12)*10; //H0：グラフの x 軸の画面座標, H10：グラフの高さ

    //--- コンストラクタ ---
        public GraphJPanel(){ setSize(H,W); setParameters( 50.0, 10.0 ); }

    //--- ユーザが入力したパラメータをセット ---
        public void setParameters( double m , double s ){
                this.mean = m ; this.sigma = s ; this.repaint();
        }

    //--- グラフ描画 ---
        public void paintComponent(Graphics g){
                W = getWidth(); H = getHeight();
                H0=(H/12)*11; H10=(H/12)*10;
                g.clearRect( 0, 0, W, H );

                // グリッド目盛
                g.setColor(Color.lightGray);
                // 縦線
                for( int i=0; i < W ; i += W/10 ){ g.drawLine( i, 0, i, H0 ); }
                // 横線
                for( int j = H0; j >= 0 ; j -= H10/10 ){ g.drawLine( 0, j, W, j );}
                g.setColor(Color.black);int n=0;
                for( int i=0; i < W ; i += W/10 ){ g.drawString( ""+n , i-6, H-10 ); n += 10; }
                // x 軸
                g.setColor(Color.black);        g.drawLine(0,H0,W,H0);

                //--- 確率密度関数のグラフ ---
                g.setColor(Color.blue);
                double x0 = 0;        double y0 = NormalDistribution.ndf( x0, mean, sigma );
                double dx = (x_max - x_min)/W ;
                for(int i=1; i <= W ; i++){
          double x1 = 1.0*i*dx; double y1 = NormalDistribution.ndf( x1, mean, sigma );
                        g.drawLine( (i-1), (int)(H0-5000*y0), i, (int)(H0-5000*y1) );  x0=x1; y0=y1;
                }

    //--- 分布関数のグラフ ---
                g.setColor(Color.red);
```

```
                    x0 = 0;    y0 = NormalDistribution.df( x0, mean, sigma );
```

```
                    for(int i=1; i <= W ; i++){
            double x1 = 1.0*i*dx; double y1 = NormalDistribution.df( x1, mean, sigma );
                            g.drawLine( (i-1), (int)(H0-H10*y0), i, (int)(H0-H10*y1) );    x0 = x1;  y0 = y1;
                    }
          }
}
```

```
/*
 * 作成日: 2006/08/05 @author 徐　丙鉄
 * 正規分布関数
 *  g(x) ：規格化正規分布密度関数　平均値：0　分散：1
 *  ndf(x,m,s) ：正規分布密度関数　平均値：m　分散：s
 *  df(x,m,s)  ：正規分布　　　　平均値：m　分散：s
 */

public class NormalDistribution {

          static int N=100;
          static double x_min=-10.0; //分布関数を求めるときの積分の開始点

          //--- 規格正規分布 ---
          static double g( double x){
                     double y = Math.exp(-x*x*0.5)/Math.sqrt(2.0*Math.PI);
                     return y;
          }

          //--- 確率分布 ---
          static double ndf(double x, double m, double s){
                              double z = (x-m)/s;
                              double y = NormalDistribution.g(z)/s;
                              return y;
          }

          //--- 分布関数 ---
          static double df(double x, double m, double s){
                    double z = (x-m)/s;
                    double y = NormalDistribution.nd(z);
                    return y;
          }

          //--- 正規化分布関数
          public static double nd( double z ){

                    double sum=0.0;  double dx = (z - x_min)/2/N;
                    //シンプソン法
                    for( int n = 0; n < 2*N ; n += 2){
                              double x = x_min + dx*n;
                              double y0 = NormalDistribution.g(x) ;
                              double y1 = NormalDistribution.g(x+dx) ;
                              double y2 = NormalDistribution.g(x+2*dx) ;
                              double s = y0+4.0*y1+y2;
                              sum += s;
                    }
                    sum = sum*dx/3.0;
                    return sum;
   }

}
```