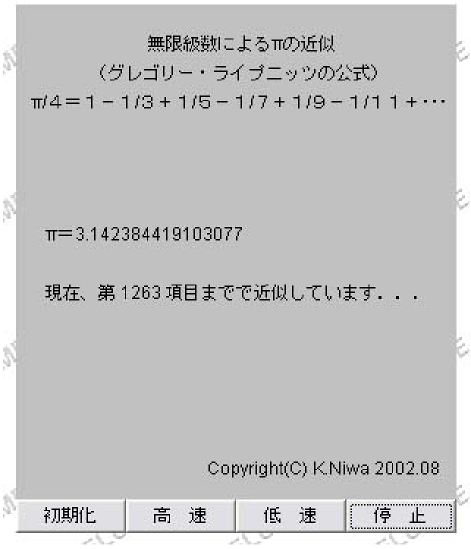
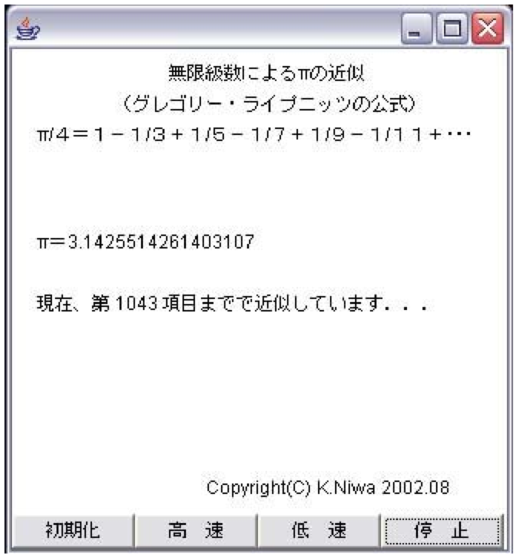
【グレゴリー・ライプニッツの公式】



［Ｊａｖａアプレット］



［Ｊａｖａアプリケーション］

１．はじめに

次のグレゴリー・ライプニッツの公式を用いてπの近似値を求めてみましょう。

［グレゴリー・ライプニッツの公式］

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots$$

シミュレーションソフト「グレゴリー・ライプニッツの公式によるπの近似」を使って、πの近似値が求まる様子を観察してみてください。

２．Ｊａｖａアプレット

（１）Ｊａｖａプログラムリスト

```
/////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                  //
//            「グレゴリー・ライプニッツの公式によるπの近似」            //
//                 Copyright（C）K.Niwa  2002.08.05                 //
//                       （Ｊａｖａアプレット）                       //
//                                                                  //
/////////////////////////////////////////////////////////////////////////////////////////////////////

// ライブラリーからのクラスの読み込み
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

/****** public class Gregory extends Applet implements Runnable ************************************/
public class Gregory extends Applet implements Runnable{        //スレッドを使えるようにする

//変数とオブジェクトの型宣言
        Thread myTh;                    //スレッド型で宣言する
        Button[] myBtn;                 //ボタン型配列で宣言する
```

```
        Panel myPanel;                          //パネル型で宣言する
        int flag=0;
        int Speed=100;                          //速度
        int ct=0;                               //実験回数
        int count;                              //ループカウンター
        double pai;                             //πの近似値
        double s;                               //π／４の近似値を求める過程での無限級数

/****** public void init()ﾒｿｯﾄﾞ ************************************************************/
        public void init(){
                setBackground(Color.lightGray);                 //背景色をグレーにする
                myTh=null;                                      //スレッドの初期化

                myBtn=new Button[4];                            //ボタンの実体化
                myBtn[0]=new Button("初期化");
                myBtn[1]=new Button("高　速");
                myBtn[2]=new Button("低　速");
                myBtn[3]=new Button("停　止");

                myPanel=new Panel();                            //パネルの実体化
                myPanel.setLayout(new GridLayout(1,4));         //ﾊﾟﾈﾙをｸﾞﾘｯﾄﾞﾚｲｱｳﾄにする
                for (count=0;count<=3;count++){
                        myPanel.add(myBtn[count]);              //パネルにボタンを貼り付ける
                }
                setLayout(new BorderLayout());                  //全体をボーダーレイアウトにする
                add("South",myPanel);                           //パネルを南に貼り付ける

                myBtn[0].addActionListener(new ActionListener(){        //初期化ボタンの定義
                        public void actionPerformed(ActionEvent e){
                                flag=0;                                 //ボタン識別子
                                repaint();
                        }
                });

                myBtn[1].addActionListener(new ActionListener(){        //高速ボタンの定義
                        public void actionPerformed(ActionEvent e){
                                flag=1;                                 //ボタン識別子
                                Speed=20;                               //速度
                                repaint();
                        }
                });

                myBtn[2].addActionListener(new ActionListener(){        //低速ボタンの定義
                        public void actionPerformed(ActionEvent e){
                                flag=2;                                 //ボタン識別子
                                Speed=200;                              //速度
                                repaint();
                        }
                });

                myBtn[3].addActionListener(new ActionListener(){        //停止ボタンの定義
                        public void actionPerformed(ActionEvent e){
                                flag=3;                                 //ボタン識別子
                                repaint();
                        }
                });

        }//public void init()
```

```
/******  public  void  start()ﾒｿｯﾄﾞ  ************************************************************/
        public void start() {
                if (myTh==null) {
                        myTh=new Thread(this);                  //スレッドの実体化
                        myTh.start();                           //スレッドの開始
                }
        }

/******  public  void  run()ﾒｿｯﾄﾞ  ********  イベントがなくても動作する  ************************/
        public void run() {
                while (true) {
                        try{
                                myTh.sleep(Speed);
                        }
                        catch (InterruptedException e)  {}
                        if (flag==1 || flag==2) {
                                repaint();
                        }
                }
        }

/******  public  void  paint(Graphics  g)ﾒｿｯﾄﾞ  ******************************************************/
        public void paint(Graphics g) {

        //初期状態または初期化ボタンを押したときのイベント処理
                if(flag==0) {
                        //g.clearRect(0,0,300,360);             //全体のクリア
                        ct=0;                                   //第何項目までの和であるかの初期化
                        s=0;                    //π／４の近似値を求める過程での無限級数の初期化
                        pai=0;                                  //πの近似値の初期化

                        g.drawString("無限級数によるπの近似",100-10,20+10);
                        g.drawString("（グレゴリー・ライプニッツの公式）",70-20,40+10);
                        g.drawString("π/４＝１－１/３＋１/５－１/７＋１/９－１/１１＋･･･"
                                                                        ,20-10,60+10);
                        g.drawString("π＝",30-10,160);
                //第何項目までの和であるかを表示
                        g.drawString("現在、第 "+"  "+" 項目までで近似しています...  "
                                                                      ,30-10,300-100);
                        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);       //作者表示
                }//if(flag==0)

        //高速ボタンまたは低速ボタンを押したときのイベント処理
                else if (flag==1 || flag==2) {
                        if (ct<2147483647) {
                                ct=ct+1;
                        }
                        else{
                                flag=3;
                        }

                        if (ct%2==1) {
                                s=s+(double)1/(2*ct-1);
                        }
                        else if (ct%2==0) {
                                s=s-(double)1/(2*ct-1);
                        }
                        pai=(double)4*s;
                        g.drawString("無限級数によるπの近似",100-10,20+10);
                        g.drawString("（グレゴリー・ライプニッツの公式）",70-20,40+10);
                        g.drawString("π/４＝１－１/３＋１/５－１/７＋１/９－１/１１＋･･･"
                                                                        ,20-10,60+10);
                        g.drawString("π＝"+pai,30-10,160);
```

```
                //第何項目までの和であるかを表示
                        g.drawString（"現在、第 "+ct+" 項目までで近似しています．．． "
                                                                                ,30-10,300-100）;
                        g.drawString（"Copyright（C） K.Niwa 2002.08",130,330-10）;        //作者表示
                ｝//else if （flag==1 || flag==2）

        //停止ボタンを押したときのイベント処理
                if（flag==3）｛
                        g.drawString（"無限級数によるπの近似",100-10,20+10）;
                        g.drawString（"（グレゴリー・ライプニッツの公式）",70-20,40+10）;
                        g.drawString（"π/４＝１－１/３＋１/５－１/７＋１/９－１/１１＋･･･"
                                                                                ,20-10,60+10）;
                        g.drawString（"π＝"+pai,30-10,160）;
                //第何項目までの和であるかを表示
                        g.drawString（"現在、第 "+ct+" 項目までで近似しています．．． "
                                                                                ,30-10,300-100）;
                        g.drawString（"Copyright（C） K.Niwa 2002.08",130,330-10）;        //作者表示
                ｝//if（flag==3）

        ｝//public void paint（Graphics g）

｝//public class Gregory extends Applet implements Runnable
```

（２）ＨＴＭＬリスト

```
<HTML>
        <HEAD>
< !---------------------------------------------------------------------------------------------------------------------
                        「グレゴリー・ライプニッツの公式によるπの近似」
                                Copyright（C）K.Niwa 2002.08.05
----------------------------------------------------------------------------------------------------------------------->
        </HEAD>
        <BODY>
                <CENTER>
                        <B>「グレゴリー・ライプニッツの公式によるπの近似」</B>
                        <BR><BR>
                        <APPLET  CODE="Gregory.class"  WIDTH="300"  HEIGHT="360"></APPLET>
                </CENTER>
        </BODY>
</HTML>
```

３．Ｊａｖａアプリケーション・プログラムリスト

```
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                         //
//                      「Ｆグレゴリー・ライプニッツの公式によるπの近似」                                   //
//                              Copyright（C）K.Niwa  2002.08.16                                           //
//                                  （Ｊａｖａアプリケーション）                                           //
//                                                                                                         //
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//ライブラリーからのクラスの読み込み
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

/****** public class FGregory extends Frame implements Runnable   *********************************/
public class FGregory extends Frame implements Runnable｛          //スレッドを使えるようにする

//変数とオブジェクトの型宣言
        Thread myTh;                                    //スレッド型で宣言する
        Button［］ myBtn;                                //ボタン型配列で宣言する
```

```
    Panel myPanel;                          //パネル型で宣言する
    int flag=0;
    int Speed=100;                          //速度
    int ct=0;                               //実験回数
    int count;                              //ループカウンター
    double pai;                             //πの近似値
    double s;                               //π／４の近似値を求める過程での無限級数

/******  フレームとイベント処理の定義 ***************************************************/
    public FGregory(){

        setSize(300+30,360);                        //フレームの大きさ

        addWindowListener(new WindowAdapter(){    //閉じるボタンのイベント処理
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });

        myTh=null;                                  //スレッドの初期化
        if (myTh==null){
            myTh=new Thread(this);                  //スレッドの実体化
            myTh.start();                           //スレッドの開始
        }

        myBtn=new Button[4];                        //ボタンの実体化
        myBtn[0]=new Button("初期化");
        myBtn[1]=new Button("高　速");
        myBtn[2]=new Button("低　速");
        myBtn[3]=new Button("停　止");

        myPanel=new Panel();                        //パネルの実体化
        myPanel.setLayout(new GridLayout(1,4));     //ﾊﾟﾈﾙをｸﾞﾘｯﾄﾞﾚｲｱｳﾄにする
        for (count=0;count<=3;count++){
            myPanel.add(myBtn[count]);              //パネルにボタンを貼り付ける
        }
        setLayout(new BorderLayout());              //全体をﾎﾞｰﾀﾞｰﾚｲｱｳﾄにする
        add("South",myPanel);                       //パネルを南に貼り付ける

        myBtn[0].addActionListener(new ActionListener(){      //初期化ボタンの定義
            public void actionPerformed(ActionEvent e){
                flag=0;                                       //ボタン識別子
                repaint();
            }
        });

        myBtn[1].addActionListener(new ActionListener(){      //高速ボタンの定義
            public void actionPerformed(ActionEvent e){
                flag=1;                                       //ボタン識別子
                Speed=20;                                     //速度
                repaint();
            }
        });

        myBtn[2].addActionListener(new ActionListener(){      //低速ボタンの定義
            public void actionPerformed(ActionEvent e){
                flag=2;                                       //ボタン識別子
                Speed=200;                                    //速度
                repaint();
            }
        });
```

```
		myBtn[3].addActionListener(new ActionListener(){	//停止ボタンの定義
			public void actionPerformed(ActionEvent e){
				flag=3;					//ボタン識別子
				repaint();
			}
		});

	}//public FGregory()

/******  public void run()ﾒｿｯﾄﾞ  ****************************************************************/
	public void run(){
		while (true){
			try{
				myTh.sleep(Speed);
			}
			catch (InterruptedException e) {}
			if (flag==1 || flag==2){
				repaint();
			}
		}
	}//public void run()

/******  public void paint(Graphics g)ﾒｿｯﾄﾞ  ********************************************************/
	public void paint(Graphics g){

	//初期状態または初期化ボタンを押したときのイベント処理
		if(flag==0){

			//g.clearRect(0,0,300+30,360);		//全体のクリア
			ct=0;					//第何項目までの和であるかの初期化
			s=0;			//π／４の近似値を求める過程での無限級数の初期化
			pai=0;					//πの近似値の初期化

			g.drawString("　　無限級数によるπの近似",70,20+10+20);
			g.drawString("（グレゴリー・ライプニッツの公式）",70,40+10+20);
			g.drawString("π/４＝１－１/３＋１/５－１/７＋１/９－１/１１＋･･･"
								,20,60+10+20);
			g.drawString("π＝",30-10,160);
		//第何項目までの和であるかを表示
			g.drawString("現在、第 "+" "+" 項目までで近似しています... "
								,30-10,300-100);
			g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);	//作者表示
		}//if(flag==0)

	//高速ボタンまたは低速ボタンを押したときのイベント処理
		else if (flag==1 || flag==2){
			if (ct<2147483647){
				ct=ct+1;
			}
			else{
				flag=3;
			}

			if (ct%2==1){
				s=s+(double)1/(2*ct-1);
			}
			else if (ct%2==0){
				s=s-(double)1/(2*ct-1);
			}
			pai=(double)4*s;
			g.drawString("　　無限級数によるπの近似",70,20+10+20);
			g.drawString("（グレゴリー・ライプニッツの公式）",70,40+10+20);
			g.drawString("π/４＝１－１/３＋１/５－１/７＋１/９－１/１１＋･･･"
								,20,60+10+20);
```

```
                    g.drawString("π ＝"+pai,30-10,160);
                //第何項目までの和であるかを表示
                    g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                                                                    ,30-10,300-100);
                    g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);      //作者表示
                }//else if (flag==1 || flag==2)

        //停止ボタンを押したときのイベント処理
                if(flag==3){
                    g.drawString("        無限級数によるπの近似",70,20+10+20);
                    g.drawString("（グレゴリー・ライプニッツの公式）",70,40+10+20);
                    g.drawString("π/４＝１－１/３＋１/５－１/７＋１/９－１/１１＋･･･"
                                                                    ,20,60+10+20);
                    g.drawString("π ＝"+pai,30-10,160);
                //第何項目までの和であるかを表示
                    g.drawString("現在、第 "+ct+" 項目までで近似しています... "
                                                                    ,30-10,300-100);
                    g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);      //作 者 表 示
                }//if(flag==3)

        }//public void paint(Graphics g)

/****** public static void main ﾒｿｯﾄﾞ ***********************************************************/
        public static void main(String[] args){
                Frame w=new FGregory();
                w.show();
        }//public static void main(String[] args)

}//public class FGregory extends Frame implements Runnable
```