数学支援ソフト

有澤 健太郎　川口 泰良

１．　研究概要

「Visual C#.NET」を使った数学の予習復習を支援するソフトの作成した。

２．　研究の具体的内容

２．１Visual C#.NET について

Visual C#.NET とは 2000 年に Microsoft 社が発表したオブジェクト指向のプログラミング言語である。 C 言語や C++言語をベースに拡張しながら Java 風の性能や表記などが盛り込まれている。

２．２特徴

電卓のように２つの数のみを計算するのではなく，１つの式全体を入力し計算・処理することができる。それにより電卓ではなかなか処理し辛い複雑な計算もすることができる。(図１)

式全体を入力することで数値だけでなく x や sin などの文字を使った式も計算・処理が可能となった。

計算方法は式の前から順に計算するのではなく、逆ポーランド記法を用いて計算をしている。

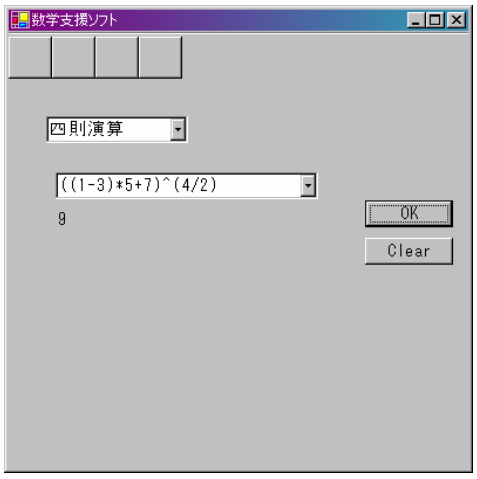


図１　四則演算

２．３逆ポーランド記法について

逆ポーランド記法とは，演算子をその計算対象の後に記す表記方法である。(図２)

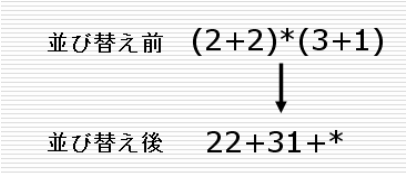


図２　逆ポーランド記法

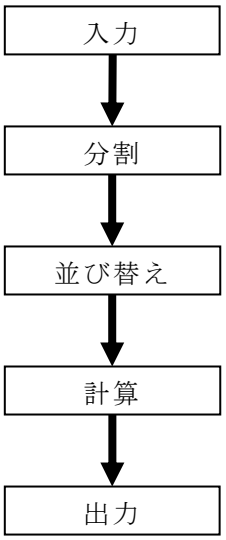
２．４機能

主な機能として,「四則演算」「グラフ化」「式の展開」があり，補助機能として「履歴」がある。

（１）四則演算

+，-，*，/，^，(，)を用いた計算が可能である。

計算には逆ポーランド記法を用いて計算をしている。(図１，フローチャート１)



フローチャート１

（２）グラフ化

四則演算のプログラムを基に x，sin，cos，tan，pi(π)の処理を追加したもので，x を数値として扱い，グラフを表示

する時に x をグラフの x 軸の値に置き換えて計算している。
グラフを表示時に拡大・縮小，極地，交点の表示が可能で，x 軸とグラフの交点から方程式を解くことも可能である。（図３，図４，フローチャート２）

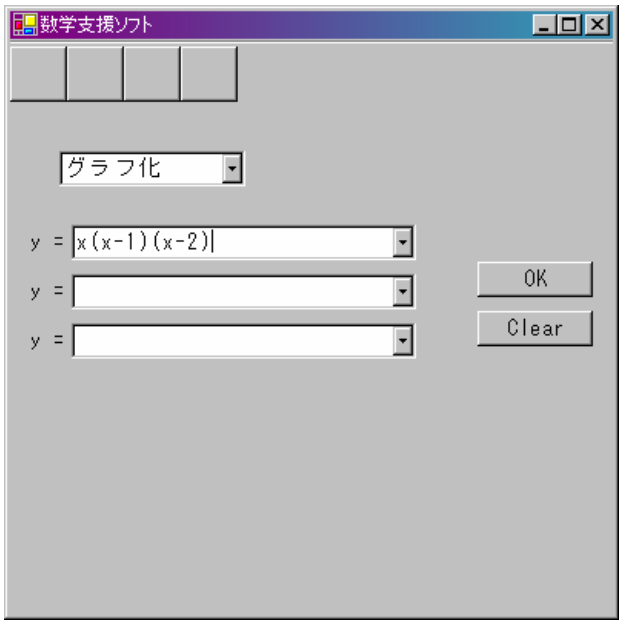


図３　グラフの式

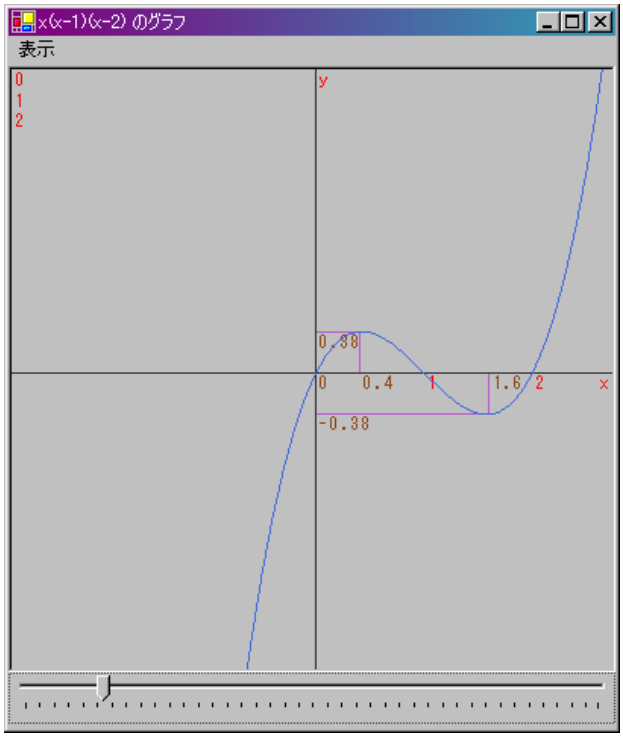
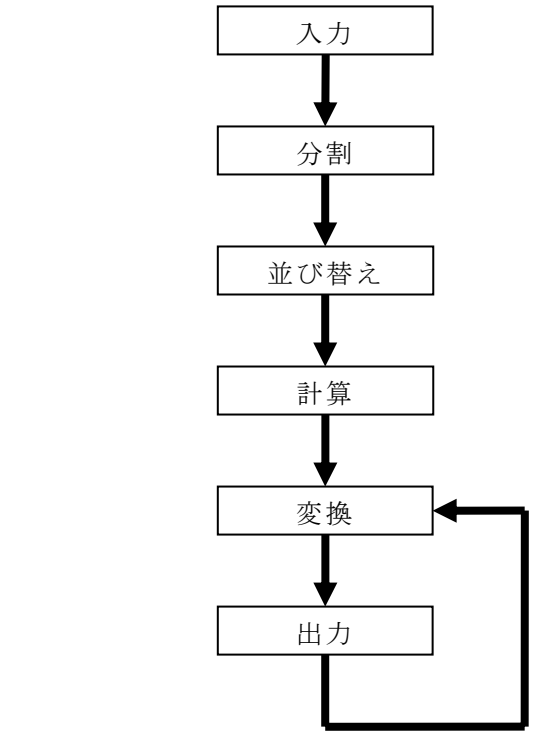


図４　グラフ



フローチャート２

（３）式の展開

このプログラムは逆ポーランド記法を一切使わず，文字の切り取り貼り付けのみで処理を行っている。このため数値の計算はできないが，文字列の処理が容易にできる。（図５，フローチャート３）

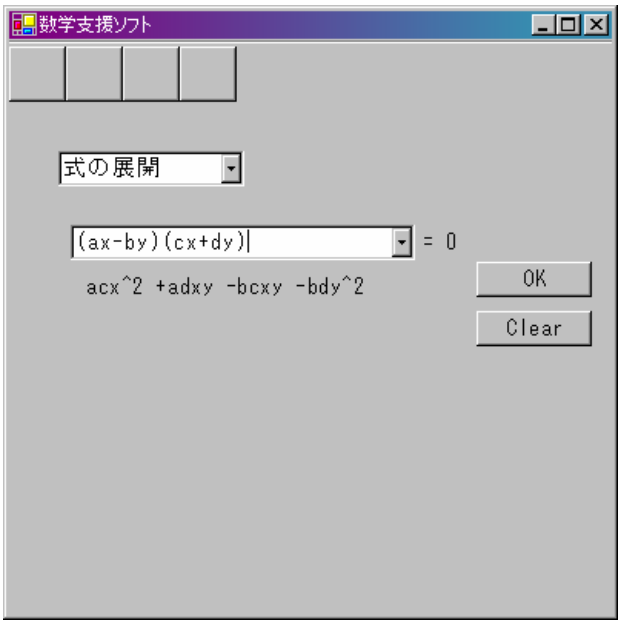
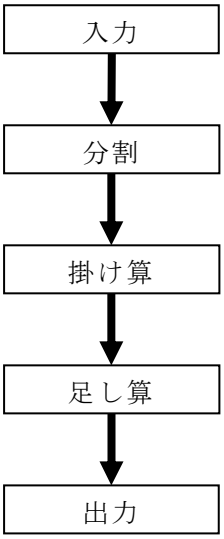


図５　式の展開



フローチャート３

（４）履歴

入力し OK を押した時の式を履歴として保存する。

３．　研究のまとめ

この課題研究で使った Visual C#.NET は今まで実習等で使っていた C 言語と違いフォームを作るということが必要だった。

初めはフォームを作ったことが無く使い方がよく分からなかったが，使い方が分かっていくにつれて面白いと思うようになった。まだまだ分からない部分が多くすべての機能を使いこなすにはもっと時間が必要だと思った。

フォームの方はある程度理解出来たがプログラムはよく分からなかったのでプログラムの得意なやつがプログラムをして，僕がそれ以外のことをやるという風に作業の分担を経験することができた。

今回の課題研究で学んだことを今後の進学先で役立てていこうと思った。

有澤

この課題研究を通して Visual C#.NET がどういうものなのか，実習で使っている Linax の C とどう違うのか，Form の作り方などいろいろなことを学んだ。

３年になる前から課題研究でやることは決めていたので，２年の冬休みから Visual Stdio.NET の使い方など色々な下準備もしていました。Visual Basic .NET と基本的な使い方は一緒なので実習Ⅲの Viaual Basic を使った実習でも楽に終わらせることもできた。

３年になってからの２ヵ月ほぼ毎日家でこのプログラムを作っていた。しかし今より遥かにバグの多いプログラムだったので 700 行ほど書いていたが，今のようなプログラムに１ヵ月で作り変えた。その後も式によってはオーバーフローを起こすものもあったのでその処理にかなりの時間を費やしてしまった。その為当初予定していた微積分や式の変形をする時間がなくなってしまった。これでは「広く浅い」プログラムになってしまうと思い，微積分や式の変形をするプログラムをやめ「狭く深い」プログラムを目指すようした。そこでグラフをより正確に表示計算できるようにした。その結果 x 軸との交点を小数第７位まで正確に求めることができ，指数方程式も正確な値で解くことができるようになった。

今あるプログラムが完成品とは思ってはおらず，これからも時間があれば少しずつ進化させていきたいと思う。

川口

参考文献

緑のバイク★フリーソフト公開ホームページ
http://homepage3.nifty.com/midori_no_bike/index.html

ソースプログラム（一部抜粋）

（１）履歴

```
// 履歴
private void comboBox_KeyDown(object
sender,System.Windows.Forms.KeyEventArg
s e)
{
  // Enterが押された検出
  if(e.KeyCode==Keys.Enter&&((System.Wi
  ndows.Forms.ComboBox) sender).Text !=
  "")
  {
    // 押されたcomboBoxの名前の取り出し
    string name = ((System.Windows.
      Forms.ComboBox)sender).Name;
    // 関数comboBox2へ
    combobox2(ref name);
  }
}
// 関数comboBox2
void combobox2(ref string name)
{
  // comboBox2の中の履歴に今のテキスト
  の内容が含まれていない場合
  if(comboBox2.Items.Contains(comboBox2
  .Text)==false)
  comboBox2.Items.Add(""+comboBox2.Text
  );
}
```

（２）四則演算

```
// 関数 分割（四則・グラフ共通）
// 演算子の間に”“を入れる
Work = Work.Replace("("," ( ");
Work = Work.Replace(")"," ) ");
Work = Work.Replace("+"," + ");
Work = Work.Replace("-"," - ");
Work = Work.Replace("*"," * ");
Work = Work.Replace("/"," / ");
Work = Work.Replace("^"," ^ ");
// “ “をもとに配列に格納する
string[] For_ = Work.Split(new char[]{'
'});
int For_Num = Work.Split (new char[] {'
'}).Length;
// 空白を詰める
for(int i=0,j=0;i<For_Num;i++){
  if(For_[i]!=""){
    For[j,0] = For_[i];
    j++;
  }
}

// 関数 並び替え（四則・グラフ共通）
string[] For_Work = new string[100];
string[] Stk = new string[100];
int[] Sign = new int[2];
int[] cnt = new int[2];
// 移し変えと初期化
for(int i=0;i<100;i++)
{
  For_Work[i] = For[i,0];
  For[i,0] = "";
  Stk[i] = "";
}
// ループ
for(int i=0;;i++)
{
  // 式の終わり？
  if (For_Work[i] == "E")
  {
    // 次の段階への準備
    for(cnt[0]--;cnt[0]>=0;cnt[0]--)
    {
      For[cnt[1],0] = Stk[cnt[0]];
      cnt[1]++;
    }
    // ループ終了
```

```
    break;
  }
// "("？
else if (For_Work[i] == "(")
{
  Stk[cnt[0]] = For_Work[i];
  cnt[0]++;
}
")"？
else if (For_Work[i] == ")")
{
  // "("がでてくるまでループする
  for(;Stk[cnt[0]]!="(";)
  {
    For[cnt[1],0] = Stk[cnt[0]];
    Stk[cnt[0]] = "";
    cnt[1]++;
    cnt[0]--;
  }
  Stk[cnt[0]] = "";
}
// 演算子？
else if(For_Work[i] == "+" ||
For_Work[i] == "-" ||
For_Work[i] == "*" ||
For_Work[i] == "/" ||
For_Work[i] == "^" ||
For_Work[i] == "sin" ||
For_Work[i] == "cos" ||
For_Work[i] == "tan")
{
  // 無限ループ
  for(;;)
  {
    // スタックは空？
    if (cnt[0] == 0)
    {
      Stk[cnt[0]] = For_Work[i];
      cnt[0]++;
      break;
    }
    // 優先順位
    if (Stk[cnt[0]-1]=="") Sign[0]=0;
    if (Stk[cnt[0]-1]=="+") Sign[0]=1;
    if (Stk[cnt[0]-1]=="-") Sign[0]=1;
    if (Stk[cnt[0]-1]=="*") Sign[0]=2;
    if (Stk[cnt[0]-1]=="/") Sign[0]=2;
    if (Stk[cnt[0]-1]=="^") Sign[0]=3;
    if (Stk[cnt[0]-1]=="sin") Sign[0]=4;
    if (Stk[cnt[0]-1]=="cos") Sign[0]=4;
    if (Stk[cnt[0]-1]=="tan") Sign[0]=4;
    if (For_Work[i]=="+") Sign[1]=1;
    if (For_Work[i]=="-") Sign[1]=1;
    if (For_Work[i]=="*") Sign[1]=2;
    if (For_Work[i]=="/") Sign[1]=2;
    if (For_Work[i]=="^") Sign[1]=3;
    if (For_Work[i]=="sin") Sign[1]=4;
    if (For_Work[i]=="cos") Sign[1]=4;
    if (For_Work[i]=="tan") Sign[1]=4;
    // Stk＜For_Work
    if(Sign[0]<Sign[1]||Stk[cnt[0]-1] ==
    "(")
    {
      Stk[cnt[0]] = For_Work[i];
      cnt[0]++;
      break;
    }
    // Stk≧For_Work
    else
    {
      For[cnt[1],0] = Stk[cnt[0]-1];
      cnt[1]++;
      cnt[0]--;
      Stk[cnt[0]] = For_Work[i];
    }
  }
}
```

```
// 数値？
else
{
  For[cnt[1],0] = For_Work[i];
  cnt[1]++;
}
// 空白になっている箱の除去
for(int i=0;i<100;i++){
  if(For[i,0].Length==0){
    for(int j=i+1;j<100;j++)
      For[j-1,0] = For[j,0];
  }
}
// 関数 計算
double[] Reg = new double[100];
int Counter_Reg = 0;
for(int i=0;i<100;i++)
  Reg[i] = 0;
for(int i=0;For[i,0]!="";i++){
  // 演算子？
  if(For[i,0] == "+" ||
  For[i,0] == "-" ||
  For[i,0] == "*" ||
  For[i,0] == "/" ||
  For[i,0] == "^" ){
    if (For[i,0]=="+")
      Reg[Counter_Reg-2]+=
      Reg[Counter_Reg-1];
    if (For[i,0]=="-")
      Reg[Counter_Reg-2]-=
      Reg[Counter_Reg-1];
    if (For[i,0]=="*")
      Reg[Counter_Reg-2]*=
      Reg[Counter_Reg-1];
    if (For[i,0]=="/")
      Reg[Counter_Reg-2]/=
      Reg[Counter_Reg-1];
    if (For[i,0]=="^")
Reg[Counter_Reg-2]=
      Math.Pow(Reg[Counter_Reg-2],
      Reg[Counter_Reg-1]);
    Counter_Reg--;
  }
  // 数値
  else{
    Reg[Counter_Reg] =
    double.Parse(For[i,0]);
    Counter_Reg++;
  }
}
// 結果
For[0,0] = ""+Reg[0];
```

（３）グラフ化

```
// 関数 グラフ
// 定義
Graphics g=Graphics.FromImage (p.Image);
// 文字列を描画する
g.DrawString("0",font,C_R,200,200);
g.DrawString("y",font,C_R,200,0);
g.DrawString("x",font,C_R,400-15,200);
// 拡大・縮小
Graf_A[(int)j+mem].X=Graf_B[(int)j+mem]
.X*100/trackBar1.Value+pictureBox1.Widt
h/2;
Graf_A[(int)j+mem].Y=Graf_B[(int)j+mem]
.Y*10/trackBar1.Value+pictureBox1.Heigh
t/2;
// 軸の表示
g.DrawLine(pen_1,0,pictureBox1.Height/2
,pictureBox1.Width,pictureBox1.Height/2
);
g.DrawLine(pen_1,pictureBox1.Width/2,0,
pictureBox1.Width/2,pictureBox1.Height)
;
// グラフの描画
g.DrawLines(pen_2,Graf_A);
```

（４）式の展開

```
// 分割
// ( で分割する
string[] For_Work = comboBox2.Text.Split
(new char[] {'('});
// ( の数
int For_Num = comboBox2.Text.Split (new
char[] {'('}).Length;
// 最初の ( より前の空白詰
if(For_Work[0].Length==0)
{
  for(int i=1;i<For_Num;i++)
    For_Work[i-1] = For_Work[i];
  For_Num --;
}

// ( の除去
for(int i=0;i<For_Num;i++)
{
  For_Work[i] =
  For_Work[i].Replace(")","");
}
// + , - の前で分割
for(int i=0;i<For_Num;i++)
{
  int Len = (For_Work[i]).Length;
  int cnt = 0;
  // + , - で配列の位置をひとつずらす
  for(int j=0;j<Len;j++)
  {
    if(For_Work[i].Substring(j, 1)
    == "+"
    || For_Work[i].Substring(j, 1)
    == "-")
      cnt++;
    For[i,cnt] +=
    For_Work[i].Substring(j, 1);
  }
}
```

```
// + の除去
for(int i=0;i<For_Num;i++)
  for(int j=0;For[i,j]!="";j++)
    For[i,j] = For[i,j].Replace("+","");
// 関数　分割
// 終了？
if(For[1,0]=="")        break;
for(int i=0;i<500;i++)
  Work[i] = "";
  for(int i=0;For[0,i]!="";i++)
  {
    for(int j=0;For[1,j]!="";j++)
    {
      // a～z までの個数をカウント
      for(int k=0;k<26;k++)
      {
        // a～z
        char a_z = (char)(0x61+k);
        A_AA[0,k] = For[0,i].Split
        (new char[] {a_z}).Length-1;
        A_AA[1,k] = For[1,j].Split
        (new char[] {a_z}).Length-1;
      }
      // 係数の取り出し
      Num[0] = Regex.Replace
      (For[0,i],"[a-z]","");
      Num[1] = Regex.Replace
      (For[1,j],"[a-z]","");

      int che = 0;
      for(int k=0;k<26;k++)
        if(A_AA[0,k]!=A_AA[1,k])
          che++;
        if(che==0)
        {
          // 計算
          stringmult=(double.Parse(Num[
          0])+double.Parse(Num[1]));
          if(mult=="0")
```

```
          {
            Work[i] = "";
            Work[j] = "";
            for(int k=i+1;k<500;k++)
              Work[k-1] = Work[k];
          }
          else
          {
            Work[i] = mult;
            // 文字の追加
            for(int k=0;k<26;k++)
            {
            char a_z = (char)(0x61+k);
              Work[i] += new String
              (a_z,A_AA[0,k]);
            }
            Work[j] = "";
          }
        }
        for(int l=k+1;l<500;l++)
        Work[l-1] = Work[l];
    }
  }

// 関数　出力準備
for(int i=0;For[0,i]!="";i++)
{
  for(int j=0;j<26;j++)
  {
    char a_z = (char)(0x61+j);
    int A_Z     = For[0,i].Split(new
char[] {a_z}).Length-1;
    string Rep;
    if(A_Z!=0)
    {
      if(A_Z==1)
        Rep = ""+a_z;
      else
        Rep = a_z+"^"+A_Z;
      string REP = new String(a_z,A_Z);
      For[0,i] = For[0,i].Replace
      (REP,Rep);
    }
  }
}
for(int i=1;For[0,i]!="";i++)
{
  int M = For[0,i].
  Split(new char[]{'-'})  .Length-1;
  if(M==0)
    For[0,i] = "+"+For[0,i];
}
```