

ภาคผนวก ก.

รายละเอียดขั้นตอนวิธีการคำนวณค่าต่าง ๆ ที่เกิดขึ้นในกระบวนการผลิตแบบ Duncan Process สามารถแสดงให้เห็นได้ดัง Flow Chart ที่แสดงถึงความสัมพันธ์กันของคำสั่งต่าง ๆ ที่มีอยู่ทั้งหมดของโปรแกรมภาษา C # โดยใช้เครื่องมือช่วยเขียนจาก Microsoft Visual Basic Studio 2008 ซึ่งรายละเอียดของตัวโปรแกรมสามารถพิจารณาได้จาก Coding

Coding

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.Reflection;
using System.IO;
namespace Duncan
{
    public partial class Form1 : Form
    {
        #region declare attribute.
        private double lamda;
        private double g;
        private double D;
        private double V;
        private double b;
        private double c;
        private double W;
        private double C1;
        private double C2;
        private double n0;
        private double n1;
        private double p0;
        private double delta = 3;
        public Form1()
        {
            InitializeComponent();
        }
        #endregion
        #region utilities function
        private double Factorial(double number)
        {
            if (number <= 1.00)
                return 1.00;
            else
                return number * Factorial(number - 1.00);
        }
        #endregion
        private void RetrieveInformation()
        {
            string msg = "";
```

```

if (this.LBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า lamda\n";
if (this.GBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า g\n";
if (this.DBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า d\n";
if (this.VBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า V\n";
if (this.BBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า b\n";
if (this.CBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า c\n";
if (this.WBOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า W\n";
if (this.C1BOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า C1\n";
if (this.C2BOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า C2\n";
if (this.N0BOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า n0\n";
if (this.P0BOX.Text.ToString().Trim().Length < 1)
msg += "ยังไม่ระบุค่า p0\n";
if (msg.Length > 1)
throw new Exception(msg);
try
{
this.lamda = Convert.ToDouble(LBOX.Text.ToString().Trim());
this.g = Convert.ToDouble(GBOX.Text.ToString().Trim());
this.D = Convert.ToDouble(DBOX.Text.ToString().Trim());
this.V = Convert.ToDouble(VBOX.Text.ToString().Trim());
this.b = Convert.ToDouble(BBOX.Text.ToString().Trim());
this.c = Convert.ToDouble(CBOX.Text.ToString().Trim());
this.W = Convert.ToDouble(WBOX.Text.ToString().Trim());
this.C1 = Convert.ToDouble(C1BOX.Text.ToString().Trim());
this.C2 = Convert.ToDouble(C2BOX.Text.ToString().Trim());
this.n0 = Convert.ToDouble(N0BOX.Text.ToString().Trim());
this.p0 = Convert.ToDouble(P0BOX.Text.ToString().Trim());
if (N1BOX.Text.ToString().Trim().Length > 0)
this.n1 = Convert.ToDouble(N1BOX.Text.ToString().Trim());
else
this.n1 = -1;
}
catch (Exception) { throw new Exception("ข้อมูลไม่ถูกต้อง กรุณาตรวจสอบอีกครั้ง!\n");
}
}
private void ComputeNow_Click(object sender, EventArgs e)
{
double a = 0.00;
double beta = 0.00;
double p = 0.00;
double h = 0.00;
double EL = 0.00;
double aOld = 0.00;
double betaOld = 0.00;

```

```

double pOld          = 0.00;
double hOld          = 0.00;
double ELOld         = 0.00;
resultsBindingSource.Clear();
//step0 get variable value from form. and validate it.
this.RetrieveInformation();
//เป็นค่าที่ต้องคำนวณเพื่อใช้ในสมการที่ 4
double p1 = p0 + (delta * Math.Sqrt(p0 * (1.00 - p0)));
//loop for increase n number.
if (n1 > n0) //คำนวณและแสดงค่าแบบช่วง
{
    for (double n = n0; n <= n1; n++)
    {
        for (double d = 0.00; d <= n; d++)
        {
            if (d == n)
            {
                resultsBindingSource.Add(new Results(n, hOld, d, ELOld, aOld, betaOld));
                continue;
            }
            a          = GetAlpha(n, d);
            beta       = GetBeta(n, d);
            p          = GetPN(n, d, p1);
            h          = GetHN(n, p, a);
            EL         = GetEL(n, h, d, a, p);
            if (a > 0 && beta > 0 && p > 0 && h > 0 && EL > 0)
            {
                resultsBindingSource.Add(new Results(n, h, d, EL, a, beta));
                aOld      = a;
                betaOld   = beta;
                pOld      = p;
                hOld      = h;
                ELOld     = EL;
            }
            else
            {
                resultsBindingSource.Add (new Results(n, hOld, d, ELOld, aOld, betaOld));
            }
        }
    }
}
else // คำนวณและแสดงค่าเฉพาะที่ค่า n0 เท่านั้น
{
    for (double d = 0.00; d <= n0; d++)
    {
        if (d == n0)
        {
            resultsBindingSource.Add(new Results(n0, hOld, d, ELOld, aOld, betaOld));
            continue;
        }
        a          = GetAlpha(n0, d);
        beta       = GetBeta(n0, d);
        p          = GetPN(n0, d, p1);
        h          = GetHN(n0, p, a);
        EL         = GetEL(n0, h, d, a, p);
        if (a > 0 && beta > 0 && p > 0 && h > 0 && EL > 0)
        {
            resultsBindingSource.Add(new Results(n0, h, d, EL, a, beta));
            aOld      = a;
            betaOld   = beta;
            pOld      = p;
            hOld      = h;
        }
    }
}

```

```

    EL0ld      = EL;
}
else
{
    resultsBindingSource.Add(new Results(n0, h0ld, d, EL0ld, a0ld, beta0ld));
}
}
}
}
private double FactorialMultiply(double n, double x, double multiply)
{
    if (n < 2.00)
        return multiply;
    if (n == x)
        throw new Exception("Can not compute value at n==d");
    double fac1      = multiply;
    double start     = x + 1.00;
    double stopdevice = n - x;
    for (double i = start; i <= n; i++)
        fac1 = fac1 * i;
    for (double j = 1.00; j <= stopdevice; j++)
        fac1 = fac1/j;
    return fac1;
}
private double Power(double numBase, double power)
{
    double r = 1.00;
    power = power+1;
    for (double i = 1.00; i < power; i++)
        r = r * numBase;
    return r;
}
private double GetAlpha(double n, double d)
{
    double al = 0.00;
    for (double x = 0.00; x <= d; x++)
        al += (FactorialMultiply(n, x, (Power(p0, x) * Power(1.00 - p0, n - x))));
    return 1.00 - al;
}
private double GetBeta(double n, double d)
{
    double beta = 0.00;
    for (double x = 0.00; x <= d; x++)
        beta += (FactorialMultiply(n, x, (Power(1.00-p0, x) * Power(p0, n - x))));
    return beta;
}
private double GetPN(double n, double d, double p1)
{
    double p = 0.00;
    for (double x = 0.00; x <= d; x++)
        p += (FactorialMultiply(n, x, (Power(p1, x) * Power(1.00 - p1, n - x))));
    return 1.00 - p;
}
private double GetHN(double n, double p, double a)
{
    return Math.Sqrt((a * (C1 + C2) + (b + (c * n))) / (lamda * V * ((1.00 / p) - (0.5))));
}
private double GetEL(double n, double h, double d, double a, double p)
{
    double duncan      = GetTDuncan(h, p, n);
    double part1       = ((b + (c * n)) * (duncan / h));
    double part2       = ((C1 * p) + (C2 * duncan * a) + w);
}

```

```

double part3          = (V * (h * ((1.00 / p) - (0.5) + (lamda * h / 12.00))
+ (g * n) + D));
return ((part1 + part2 + part3 ) / duncan);
}
private double GetTDuncan(double h, double p, double n)
{
return ((1.00 / lamda) + h * ((1.00 / p) - 0.5 + (lamda * h / 12.00)) + (g *
n) + D);
}
#region ExporterDataGridVersExcel
private void ToExcel_Click(object sender, EventArgs e)
{
using (SaveFileDialog saveFileDialog = GetExcelSaveFileDialog())
{
if (saveFileDialog.ShowDialog(this) == DialogResult.OK)
{
string fileName = saveFileDialog.FileName;
ExporterDataGridVersExcel(this.dataGridView1, fileName);
Process.Start(fileName);
}
}
}
private SaveFileDialog GetExcelSaveFileDialog()
{
SaveFileDialog saveFileDialog = new SaveFileDialog();
saveFileDialog.CheckPathExists = true;
saveFileDialog.AddExtension = true;
saveFileDialog.ValidateNames = true;
saveFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
saveFileDialog.DefaultExt = ".xls";
saveFileDialog.Filter = "Microsoft Excel Workbook (*.csv)|*.csv";
return saveFileDialog;
}
public void ExporterDataGridVersExcel(DataGridView dgView, String fileName)
{
try
{
//Create the CSV file to which grid data will be exported.
StreamWriter sw = new StreamWriter(fileName, false)
//First we will write the headers.
int iColCount = dgView.ColumnCount;
int i = 0;
foreach (DataGridViewColumn ch in dgView.Columns)
{
sw.Write(ch.Name.Replace("DataGridViewTextBoxColumn", "").Trim());
if (i < iColCount - 1)
{
sw.Write(",");
}
i++;
}
sw.Write(sw.NewLine);
//Now write all the rows.
foreach (DataGridViewRow uneLigne in dgView.Rows)
{
i = 0;
foreach (DataGridViewColumn uneColonne in dgView.Columns)
{
sw.Write("'" + uneLigne.Cells[uneColonne.Name].Value.ToString().Trim());
if (i < iColCount - 1)
{
sw.Write(",");
}
}
}
}
}

```

```
i++;  
}  
sw.Write(sw.NewLine);  
}  
sw.Close();  
}  
catch (Exception ex)  
{  
    MessageBox.Show(ex.ToString());  
}  
}  
#endregion  
}  
}
```

Flow Chart

