南開大學
Nankai University

南　　开　　大　　学

人　工　智　能　学　院

《智能系统的设计与仿真》大作业报告

专　　业: 智能科学与技术
姓　　名: 石若川
学　　号: 2111381
指导教师: 霍卫光
课　　号: 0392
完成日期: 2024 年 6 月 7 日

目录

1 实验内容

设计一个水槽液位高度控制系统，如图1所示。被控系统为单容对象，被控量为液位高度 $h(t)$，控制量为进水阀门开度 $u(t)$。依据物料平衡原理，分析液位高度和进水阀门开度关系。采用 PID 控制器，实现液位高度的控制。

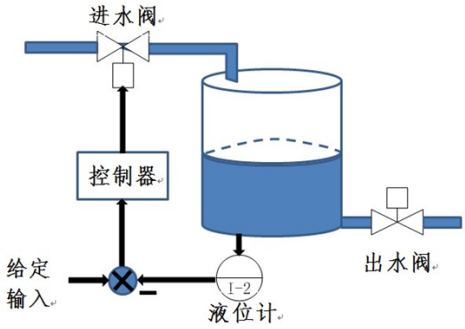

图 1: 水槽液位高度控制系统

对该水槽液位控制系统进行仿真。设计对话框能够对水槽模型参数、期望液位高度、仿真参数、PID 控制参数进行设定，如图2所示。在视图窗口对液位高度变化进行仿真，并将相关变量绘制在三幅图中，如图3所示:

- 进水流量 $Qi(t)$;
- 液位高度曲线 $h(t)$;
- 液位误差曲线 $e(t)$;

图 2: 参数设置对话框

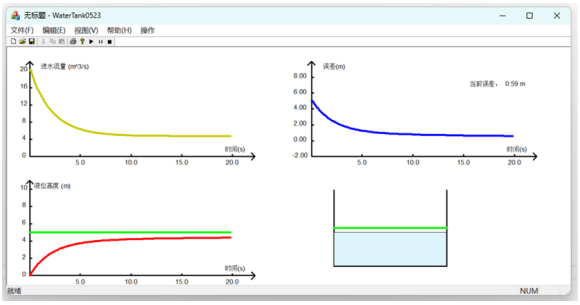

图 3: 视图窗口截图

工具条有“开始”,“暂停”和“停止”图标, 并设置相对应的快捷键“ALT+K”,“ALT+Z”,“ALT+T”, 实现曲线和水槽动画的绘制。能够保存对话框中的设置参数，并再次打开读取设置参数。

2 实验原理

根据物料平衡原理:

$$Q_i - Q_0 = A\frac{\mathrm{d}h}{\mathrm{d}t}$$

其中，Q_i 为进水流量（立方米/秒），Q_0 为出水流量，A 为截面积，$h(t)$ 为液位，并满足：

$$Q_i = k_u u, \quad Q_0 = A_0\sqrt{2gh}$$

其中，k_i 为进水阀门流量系数，A_0 为阀门系数。因此可以得到微分方程：

$$k_u u(t) - A_0\sqrt{2gh(t)} = A\frac{\mathrm{d}h(t)}{\mathrm{d}t}$$

定义期望给定信号为 $r(t)$，则误差信号为：

$$e(t) = r(t) - h(t)$$

对控制量 $u(t)$ 设计 PID 控制器：

$$u(t) = k_p e(t) + k_d\frac{de(t)}{dt} + k_i\int^t e(t)dt$$

其中，k_p, k_i, k_d 分别为比例、积分、微分增益。代入到微分方程中可得：

$$\frac{\mathrm{d}h(t)}{\mathrm{d}t} = \frac{k_u\left(k_p e(t) + k_d\frac{\mathrm{d}e(t)}{\mathrm{d}t} + k_i\int_0^t e(\tau)\mathrm{d}\tau\right) - A_0\sqrt{2gh(t)}}{A}$$

常微分方程的求解常用数值方法，实验中使用四阶龙格-库塔法进行求解。四阶龙格-库塔法的一般形式为：

$$\begin{aligned}
y_{i+1} &= y_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)\\
K_1 &= hf(x_i, y_i)\\
K_2 &= hf(x_i + \frac{h}{2}, y_i + \frac{1}{2}K_1)\\
K_3 &= hf(x_i + \frac{h}{2}, y_i + \frac{1}{2}K_2)\\
K_4 &= hf(x_i + h, y_i + K_3)
\end{aligned}$$

3 代码分析

实验中的代码实现主要包括 CMyDlg 类、Doc 类和 View 类三部分。

- CMyDlg 类：控制对话框，用于获取对话框中的数据。
- Doc 类：控制文件的保存与加载，用于仿真效果的复现。
- View 类：控制视图窗口的绘制，用于仿真图像的绘制。

3.1 CMyDlg 类

在资源视图中，设计图4所示的对话框，对每个部件添加一个 ID 值，并添加一个新的类 CMyDlg。对填入参数的每个部件添加一个变量，并在该类的构造函数中添加参数的初始值。

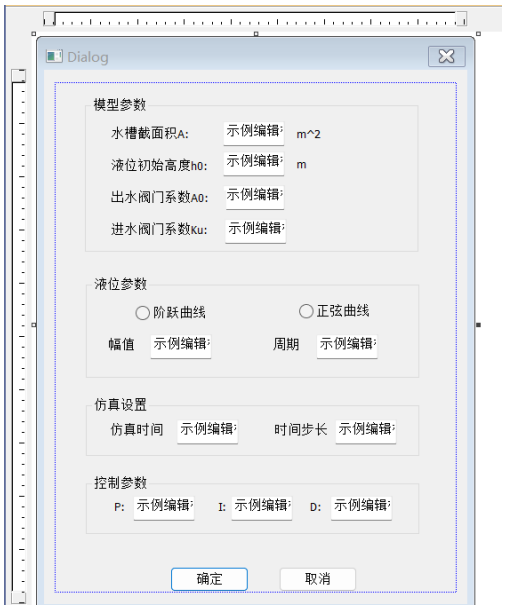

图 4: 对话框设计

为了能够修改对话框中显示的参数值，对构造函数进行重构，代码如下：

```
CMyDlg::CMyDlg(double A, double h0, double A0, double Ku, double Max, double T,
    double TimeLength, double TimeStep, double Kp, double Ki, double Kd, int
    InputType, CWnd* pParent /*=nullptr*/)
   : CDialogEx(IDD_DIALOG1, pParent)
   , m_A(A)
   , m_h0(h0)
   , m_A0(A0)
   , m_Ku(Ku)
   , m_Max(Max)
   , m_T(T)
   , m_TimeLength(TimeLength)
   , m_TimeStep(TimeStep)
   , m_Kp(Kp)
   , m_Ki(Ki)
   , m_Kd(Kd)
   , m_InputType(InputType)
{

}
```

由于阶跃函数和正弦函数的默认初始液位高度不一致，因此对点击对应控件添加响应程序，代码如下：

```
void CMyDlg::OnBnClickedJieyue()
{
```

```
    // TODO: 在此添加控件通知处理程序代码
    UpdateData();
    m_h0 = 0.0;
    UpdateData(false);
}

void CMyDlg::OnBnClickedSin()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData();
    m_h0 = 5.0;
    UpdateData(false);
}
```

3.2 Doc 类

在 Serialize 函数中添加保存和加载参数的代码，如下所示：

```
void CFinalHomeworkDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: 在此添加存储代码
        ar << m_A << m_h0 << m_A0 << m_Ku << m_Max << m_T << m_TimeLength <<
            m_TimeStep << m_Kp << m_Ki << m_Kd << m_InputType;
    }
    else
    {
        // TODO: 在此添加加载代码
        ar >> m_A >> m_h0 >> m_A0 >> m_Ku >> m_Max >> m_T >> m_TimeLength >>
            m_TimeStep >> m_Kp >> m_Ki >> m_Kd >> m_InputType;
    }
}
```

添加两个函数用于对类的数据成员进行设置和读取，代码如下所示：

```
void CFinalHomeworkDoc::Set(double A, double h0, double A0, double Ku, double Max,
    double T, double TimeLength, double TimeStep, double Kp, double Ki, double Kd,
    int InputType)
{
    m_A = A;
    m_h0 = h0;
    m_A0 = A0;
```

```
    m_Ku = Ku;
    m_Max = Max;
    m_T = T;
    m_TimeLength = TimeLength;
    m_TimeStep = TimeStep;
    m_Kp = Kp;
    m_Ki = Ki;
    m_Kd = Kd;
    m_InputType = InputType;
}

void CFinalHomeworkDoc::Get(double& A, double& h0, double& A0, double& Ku, double&
    Max, double& T, double& TimeLength, double& TimeStep, double& Kp, double& Ki,
    double& Kd, int& InputType)
{
    A = m_A;
    h0 = m_h0;
    A0 = m_A0;
    Ku = m_Ku;
    Max = m_Max;
    T = m_T;
    TimeLength = m_TimeLength;
    TimeStep = m_TimeStep;
    Kp = m_Kp;
    Ki = m_Ki;
    Kd = m_Kd;
    InputType = m_InputType;
}
```

3.3 View 类

3.3.1 四阶龙格-库塔法求解函数

根据四阶龙格-库塔法的计算公式，编写如下代码进行微分方程求解：

```
void CFinalHomeworkView::RungeKutta4()
{
    m_hValues.clear();
    m_uValues.clear();
    m_QValues.clear();
    m_errorValues.clear();

    double h = m_h0;
```

```
    double I = 0.0; // 积分项初值
    double previousError = 0.0;

    for (size_t i = 0; i < m_timePoints.size(); ++i) {
        double t = m_timePoints[i];
        double r = m_InputValue[i];
        double e = r - h;
        I += e * m_TimeStep;
        double D = (e - previousError) / m_TimeStep;
        double u = m_Kp * e + m_Ki * I + m_Kd * D;
        previousError = e;
        double q = m_Ku * u;

        double k1 = (m_Ku * u - m_A0 * sqrt(2 * 9.81 * h)) / m_A;
        double k2 = (m_Ku * u - m_A0 * sqrt(2 * 9.81 * (h + 0.5 * m_TimeStep * k1))) /
            m_A;
        double k3 = (m_Ku * u - m_A0 * sqrt(2 * 9.81 * (h + 0.5 * m_TimeStep * k2))) /
            m_A;
        double k4 = (m_Ku * u - m_A0 * sqrt(2 * 9.81 * (h + m_TimeStep * k3))) / m_A;

        h += (m_TimeStep / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4);

        m_hValues.push_back(h);
        m_uValues.push_back(u);
        m_QValues.push_back(q);
        m_errorValues.push_back(e);
    }
}
```

3.3.2 对话框函数

首先，在资源视图的菜单中添加“操作-对话框”的选项，如图5所示，并添加对应的 ID 值。

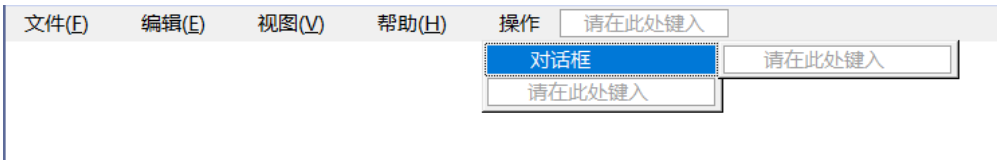

图 5: 菜单设置

添加对话框的响应函数对话框相应函数 OnOptDlg，代码如下所示。代码中主要实现了从对话框中进行参数读取、对于不同的输入曲线模式检测参数是否在范围内、利用四阶龙格-库塔法进行微分方程求解。

```
void CFinalHomeworkView::OnOptDlg()
{
   // TODO: 在此添加命令处理程序代码
   CMyDlg Dlg(m_A, m_h0, m_A0, m_Ku, m_Max, m_T, m_TimeLength, m_TimeStep, m_Kp,
       m_Ki, m_Kd, m_InputType, nullptr);

   if (Dlg.DoModal() == IDOK) {
      m_hValues.clear();
      m_uValues.clear();
      m_QValues.clear();
      m_errorValues.clear();
      m_timePoints.clear();
      m_InputValue.clear();

      m_A = Dlg.m_A;
      m_h0 = Dlg.m_h0;
      m_A0 = Dlg.m_A0;
      m_Ku = Dlg.m_Ku;
      m_Max = Dlg.m_Max;
      m_T = Dlg.m_T;
      m_TimeLength = Dlg.m_TimeLength;
      m_TimeStep = Dlg.m_TimeStep;
      m_Kp = Dlg.m_Kp;
      m_Ki = Dlg.m_Ki;
      m_Kd = Dlg.m_Kd;
      m_InputType = Dlg.m_InputType;

      if (m_InputType == 0) {
         if (m_Max < 0 || m_Max>10) {
            AfxMessageBox(_T("数值超出范围"), MB_ICONERROR | MB_OK);
                  return;
         }
         for (double t = 0.0; t < m_TimeLength - 1e-10; t += m_TimeStep)
         {
            m_timePoints.push_back(t);
            m_InputValue.push_back(m_Max);
         }
      }
      else {
         if (m_Max < 0 || m_Max>5 || m_T < 5 || m_T > 10) {
            AfxMessageBox(_T("数值超出范围"), MB_ICONERROR | MB_OK);
```

```
                return;
        }
        for (double t = 0.0; t < m_TimeLength - 1e-10; t += m_TimeStep)
        {
            double h = m_Max * sin(2 * 3.14159265 / m_TimeLength * t) + m_h0;
            m_timePoints.push_back(t);
            m_InputValue.push_back(h);
        }
    }
    RungeKutta4();
    m_Draw = true;
    Invalidate();
  }
}
```

3.3.3 OnDraw 函数

设置变量 m_draw，控制视图窗口是否进行绘图，默认为 false。当点击对话框中的“确定”后，m_draw 设置为 true，开始在视图窗口中绘图。

在 OnDraw 函数中绘制的是三个曲线图的坐标轴和水槽的边缘。绘制每个子图时，首先将设备坐标系转换为逻辑坐标系，并设置映射模式为 MM_LOMETRIC。根据绘图区域的大小控制绘图的比例，根据数据的最大值和最小值控制坐标轴纵轴的分度值。

```
void CFinalHomeworkView::OnDraw(CDC* pDC)
{
  CFinalHomeworkDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  if (!pDoc)
    return;

  if (!m_Draw)
    return;

  // TODO: 在此处为本机数据添加绘制代码
  // 获取数据大小
  int data_size = m_InputValue.size();
  // 获取流量的最大、最小值
  auto min_Q = std::min_element(m_QValues.begin(), m_QValues.end());
  auto max_Q = std::max_element(m_QValues.begin(), m_QValues.end());
  double min_QValue = *min_Q;
  double max_QValue = *max_Q;

```

```
20  CRect rect; GetClientRect(&rect);
21
22  // 流量（左上角）
23  pDC->SetMapMode(MM_LOMETRIC);        // 设置映射模式
24  CPoint ptLeft = CPoint(rect.left, rect.top);
25  CPoint ptRight = CPoint(rect.right / 2, rect.bottom / 2);
26  pDC->DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
27  pDC->DPtoLP(&ptRight);
28
29  // x和y方向数据和绘图位置的比例关系，留出两侧空白
30  double dXInter = (ptRight.x - ptLeft.x - m_interval * 2) / data_size;
31  double dYInter = (ptLeft.y - ptRight.y - m_interval * 2) / (max_QValue -
        min_QValue);
32
33  pDC->SetWindowOrg(-m_interval - ptLeft.x,
34     (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
35
36  CPen NewPen;
37  CPen* pOldPen;
38  NewPen.CreatePen(PS_SOLID, 4, RGB(0, 0, 0));
39  pOldPen = pDC->SelectObject(&NewPen);
40
41  // 绘制横纵轴
42  pDC->MoveTo(0, 0);
43  pDC->LineTo(dXInter*data_size + 30, 0);
44  pDC->MoveTo(0, 0);
45  pDC->LineTo(0, dYInter * (max_QValue - min_QValue));
46
47  //设置文本颜色和背景色
48  pDC->SetTextColor(RGB(0, 0, 0));
49  pDC->SetTextAlign(TA_CENTER | TA_TOP);
50
51  // 绘制x坐标刻度
52  for (int i = data_size / 5; i < data_size + data_size / 5; i = i + data_size/5) {
53     if (i >= data_size) {// x轴末端绘制箭头
54        pDC->MoveTo(data_size * dXInter, 10);
55        CString str1;
56        str1.Format(_T("%.1f"), (double(i) / double(data_size) * m_TimeLength));
57        pDC->TextOut(data_size * dXInter, -20, str1);
58        pDC->LineTo(data_size * dXInter, 0);
59
60        pDC->MoveTo(data_size * dXInter + 50, 0);
```

```
61          pDC->LineTo(data_size * dXInter + 40, 10);
62          pDC->MoveTo(data_size * dXInter + 50, 0);
63          pDC->LineTo(data_size * dXInter + 40, -10);
64          CString str;
65          str.Format(_T("时间 (s) "));
66          pDC->TextOut(data_size * dXInter + 80, -40, str);
67       }
68       else {
69          pDC->MoveTo(i * dXInter, 10);
70          CString str1;
71          str1.Format(_T("%.1f"), ceil(double(i) / double(data_size) * m_TimeLength));
72          pDC->TextOut(i * dXInter, -20, str1);
73          pDC->LineTo(i * dXInter, 0);
74
75       }
76    }
77
78    // 绘制y坐标刻度
79    for (int i = floor(min_QValue); i <= max_QValue + (max_QValue - min_QValue) / 5;
          i = i + (max_QValue-min_QValue)/5) {
80       if (i >= int(max_QValue)) {// y轴末端绘制箭头
81          pDC->MoveTo(0, dYInter * (max_QValue - min_QValue));
82          pDC->LineTo(0, (max_QValue - floor(min_QValue)) * dYInter);
83
84          pDC->MoveTo(0, (max_QValue - floor(min_QValue)) * dYInter );
85          pDC->LineTo(-10, (max_QValue - floor(min_QValue)) * dYInter - 10);
86          pDC->MoveTo(0, (max_QValue - floor(min_QValue)) * dYInter);
87          pDC->LineTo(10, (max_QValue - floor(min_QValue)) * dYInter - 10);
88          CString str;
89          str.Format(_T("进水流量 (m^3/s) "));
90          pDC->TextOut(60, (max_QValue - floor(min_QValue)) * dYInter + 50, str);
91       }
92       else {
93          pDC->MoveTo(0, (i- floor(min_QValue)) * dYInter);
94          pDC->LineTo(10, (i - floor(min_QValue)) * dYInter);
95          CString str1;
96          str1.Format(_T("%d"), (i));
97          pDC->TextOut(-20, (i - floor(min_QValue)) * dYInter, str1);
98       }
99    }
100
101   // 恢复坐标系
```

```
102   pDC->SetMapMode(MM_TEXT);
103   pDC->SetWindowOrg(0, 0);
104
105
106   // 误差（右上角）
107   pDC->SetMapMode(MM_LOMETRIC);
108   ptLeft = CPoint(rect.right / 2, rect.top);
109   ptRight = CPoint(rect.right, rect.bottom / 2);
110   pDC->DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
111   pDC->DPtoLP(&ptRight);
112   pDC->SetWindowOrg(-m_interval - ptLeft.x,
113     (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
114
115   // 误差的最值
116   auto min_error = std::min_element(m_errorValues.begin(), m_errorValues.end());
117   auto max_error = std::max_element(m_errorValues.begin(), m_errorValues.end());
118   double min_errorValue = *min_error;
119   double max_errorValue = *max_error;
120
121   // x和y方向数据和绘图位置的比例关系，留出两侧空白
122   dXInter = (ptRight.x - ptLeft.x - m_interval * 2) / data_size;
123   dYInter = (ptLeft.y - ptRight.y - m_interval * 2) / (max_errorValue -
        min_errorValue);
124
125   // 绘制横纵轴
126   pDC->MoveTo(0, 0);
127   pDC->LineTo(dXInter * data_size + 30, 0);
128   pDC->MoveTo(0, 0);
129   pDC->LineTo(0, dYInter * (max_errorValue - min_errorValue));
130
131   //设置文本颜色和背景色
132   pDC->SetTextColor(RGB(0, 0, 0));
133   pDC->SetTextAlign(TA_CENTER | TA_TOP);
134
135   // 绘制x坐标刻度
136   for (int i = data_size / 5; i < data_size + data_size / 5; i = i + data_size /
        5) {
137     if (i >= data_size) {// x轴末端绘制箭头
138       pDC->MoveTo(data_size * dXInter, 10);
139       CString str1;
140       str1.Format(_T("%.1f"), (double(i) / double(data_size) * m_TimeLength));
141       pDC->TextOut(data_size * dXInter, -20, str1);
```

```
142         pDC->LineTo(data_size * dXInter, 0);
143
144         pDC->MoveTo(data_size * dXInter + 50, 0);
145         pDC->LineTo(data_size * dXInter + 40, 10);
146         pDC->MoveTo(data_size * dXInter + 50, 0);
147         pDC->LineTo(data_size * dXInter + 40, -10);
148         CString str;
149         str.Format(_T("时间 (s) "));
150         pDC->TextOut(data_size * dXInter + 80, -40, str);
151       }
152       else {
153         pDC->MoveTo(i * dXInter, 10);
154         CString str1;
155         str1.Format(_T("%.1f"), ceil(double(i) / double(data_size) * m_TimeLength));
156         pDC->TextOut(i * dXInter, -20, str1);
157         pDC->LineTo(i * dXInter, 0);
158
159       }
160     }
161
162     // 绘制y坐标刻度
163     for (double i = min_errorValue; i <= max_errorValue + (max_errorValue -
          min_errorValue) / 5; i = i + ceil((max_errorValue - min_errorValue) / 5)) {
164       if (i >= int(max_errorValue)) {// y轴末端绘制箭头
165         pDC->MoveTo(0, dYInter* (max_errorValue - min_errorValue));
166         pDC->LineTo(0, (floor(max_errorValue) - floor(min_errorValue))* dYInter);
167
168         pDC->MoveTo(0, (floor(max_errorValue) - floor(min_errorValue)) * dYInter);
169         pDC->LineTo(-10, (floor(max_errorValue) - floor(min_errorValue))* dYInter -
              10);
170         pDC->MoveTo(0, (floor(max_errorValue) - floor(min_errorValue))* dYInter );
171         pDC->LineTo(10, (floor(max_errorValue) - floor(min_errorValue))* dYInter -
              10);
172         CString str;
173         str.Format(_T("误差 (m) "));
174         pDC->TextOut(-70, (floor(max_errorValue) - floor(min_errorValue)) *
              dYInter, str);
175       }
176       else {
177         pDC->MoveTo(0, (floor(i) - floor(min_errorValue)) * dYInter);
178         pDC->LineTo(10, (floor(i) - floor(min_errorValue)) * dYInter);
179         CString str1;
```

```
180         int a = floor(i);
181         int b = floor(i) - floor(min_errorValue);
182         str1.Format(_T("%.2f"), (floor(i)));
183         pDC->TextOut(-40, (floor(i) - floor(min_errorValue)) * dYInter, str1);
184       }
185     }
186
187     // 恢复坐标系
188     pDC->SetMapMode(MM_TEXT);
189     pDC->SetWindowOrg(0, 0);
190
191     // 液位（左下角）
192     pDC->SetMapMode(MM_LOMETRIC);
193     ptLeft = CPoint(rect.left, rect.bottom / 2);
194     ptRight = CPoint(rect.right / 2, rect.bottom);
195     pDC->DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
196     pDC->DPtoLP(&ptRight);
197     pDC->SetWindowOrg(-m_interval - ptLeft.x,
198       (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
199
200     // 误差的最值
201     auto min_h = std::min_element(m_hValues.begin(), m_hValues.end());
202     auto max_h = std::max_element(m_hValues.begin(), m_hValues.end());
203     double min_hValue = *min_h;
204     double max_hValue = *max_h;
205
206     auto min_i = std::min_element(m_InputValue.begin(), m_InputValue.end());
207     auto max_i = std::max_element(m_InputValue.begin(), m_InputValue.end());
208     double min_iValue = *min_i;
209     double max_iValue = *max_i;
210
211     if (max_iValue > max_hValue)
212       max_hValue = max_iValue;
213
214     // x和y方向数据和绘图位置的比例关系，留出两侧空白
215     dXInter = (ptRight.x - ptLeft.x - m_interval * 2) / data_size;
216     dYInter = (ptLeft.y - ptRight.y - m_interval * 2) / (max_hValue - 0);
217
218     // 绘制横纵轴
219     pDC->MoveTo(0, 0);
220     pDC->LineTo(dXInter* data_size + 30, 0);
221     pDC->MoveTo(0, 0);
```

```
222   pDC->LineTo(0, dYInter* (max_hValue - 0));
223
224   //设置文本颜色和背景色
225   pDC->SetTextColor(RGB(0, 0, 0));
226   pDC->SetTextAlign(TA_CENTER | TA_TOP);
227
228   // 绘制x坐标刻度
229   for (int i = data_size / 5; i < data_size + data_size / 5; i = i + data_size /
          5) {
230     if (i >= data_size) {// x轴末端绘制箭头
231       pDC->MoveTo(data_size * dXInter, 10);
232       CString str1;
233       str1.Format(_T("%.1f"), (double(i) / double(data_size) * m_TimeLength));
234       pDC->TextOut(data_size * dXInter, -20, str1);
235       pDC->LineTo(data_size * dXInter, 0);
236
237       pDC->MoveTo(data_size * dXInter + 50, 0);
238       pDC->LineTo(data_size * dXInter + 40, 10);
239       pDC->MoveTo(data_size * dXInter + 50, 0);
240       pDC->LineTo(data_size * dXInter + 40, -10);
241       CString str;
242       str.Format(_T("时间 (s) "));
243       pDC->TextOut(data_size * dXInter + 80, -40, str);
244     }
245     else {
246       pDC->MoveTo(i * dXInter, 10);
247       CString str1;
248       str1.Format(_T("%.1f"), ceil(double(i) / double(data_size) * m_TimeLength));
249       pDC->TextOut(i * dXInter, -20, str1);
250       pDC->LineTo(i * dXInter, 0);
251
252     }
253   }
254
255   // 绘制y坐标刻度
256   for (int i = 0; i <= max_hValue; i = i + max_hValue / 4) {
257     if (i == int(max_hValue)) {// y轴末端绘制箭头
258       pDC->MoveTo(0, i * dYInter);
259       pDC->LineTo(-10, i * dYInter - 10);
260       pDC->MoveTo(0, i * dYInter);
261       pDC->LineTo(10, i * dYInter - 10);
262       CString str;
```

```
263         str.Format(_T("液位高度 (m) "));
264         pDC->TextOut(60, i * dYInter + 60, str);
265       }
266       else {
267         pDC->MoveTo(0, i * dYInter);
268         pDC->LineTo(10, i * dYInter);
269         CString str1;
270         str1.Format(_T("%d"), (int(i)));
271         pDC->TextOut(-20, i * dYInter, str1);
272       }
273     }
274
275     // 恢复坐标系
276     pDC->SetMapMode(MM_TEXT);
277     pDC->SetWindowOrg(0, 0);
278
279     // 水槽 (右下角)
280     pDC->SetMapMode(MM_LOMETRIC);
281     ptLeft = CPoint(rect.right / 2, rect.bottom / 2);
282     ptRight = CPoint(rect.right, rect.bottom);
283     pDC->DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
284     pDC->DPtoLP(&ptRight);
285     pDC->SetWindowOrg(-m_interval - ptLeft.x,
286       (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
287
288     // 绘制水桶
289     pDC->MoveTo(0, 0);
290     pDC->LineTo(600, 0);
291     pDC->MoveTo(0, 0);
292     pDC->LineTo(0, dYInter * (max_hValue - 0) + 50);
293     pDC->MoveTo(600, 0);
294     pDC->LineTo(600, dYInter * (max_hValue - 0) + 50);
295 }
```

3.3.4 动画绘制函数

在资源视图中“Toolbar”添加“开始”“暂停”和“停止”的图标，并添加对应的 ID 值，如图6所示。

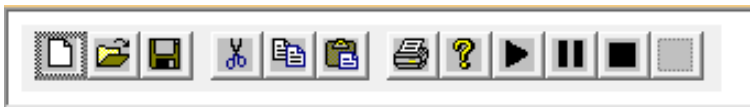

图 6: 工具条资源

在资源视图“Accelerator”中对“开始”“暂停”和“停止”的 ID 值添加快捷键，如图7所示。

ID	修饰符	键	类型
ID_EDIT_COPY	Ctrl	C	VIRTKEY
ID_EDIT_COPY	Ctrl	VK_INSERT	VIRTKEY
ID_EDIT_CUT	Shift	VK_DELETE	VIRTKEY
ID_EDIT_CUT	Ctrl	X	VIRTKEY
ID_EDIT_PASTE	Ctrl	V	VIRTKEY
ID_EDIT_PASTE	Shift	VK_INSERT	VIRTKEY
ID_EDIT_UNDO	Alt	VK_BACK	VIRTKEY
ID_EDIT_UNDO	Ctrl	Z	VIRTKEY
ID_FILE_NEW	Ctrl	N	VIRTKEY
ID_FILE_OPEN	Ctrl	O	VIRTKEY
ID_FILE_PRINT	Ctrl	P	VIRTKEY
ID_FILE_SAVE	Ctrl	S	VIRTKEY
ID_NEXT_PANE	无	VK_F6	VIRTKEY
ID_Pause	Alt	Z	VIRTKEY
ID_PREV_PANE	Shift	VK_F6	VIRTKEY
ID_Start	Alt	K	VIRTKEY
ID_Stop	Alt	T	VIRTKEY

图 7: 快捷键设置

动画效果利用 WM_TIMER 消息实现。添加 WM_TIMER 消息，在对应的函数 OnTimer 中通过变量 m_num 获得当前绘制的数据，实现曲线和液位的绘制，代码如下：

```
void CFinalHomeworkView::OnTimer(UINT_PTR nIDEvent)
{
  // TODO: 在此添加消息处理程序代码和/或调用默认值
  if (nIDEvent == 1) {
    // 数据量
    int data_size = m_InputValue.size();
    // 获取流量的最大、最小值
    auto min_Q = std::min_element(m_QValues.begin(), m_QValues.end());
    auto max_Q = std::max_element(m_QValues.begin(), m_QValues.end());
    double min_QValue = *min_Q;
    double max_QValue = *max_Q;

    CRect rect; GetClientRect(&rect);
    CClientDC dc(this);

    // 流量（左上角）
    dc.SetMapMode(MM_LOMETRIC);          // 设置映射模式
    CPoint ptLeft = CPoint(rect.left, rect.top);
    CPoint ptRight = CPoint(rect.right / 2, rect.bottom / 2);
    dc.DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
    dc.DPtoLP(&ptRight);

```

```
23      // x和y方向数据和绘图位置的比例关系，留出两侧空白
24      double dXInter = (ptRight.x - ptLeft.x - m_interval * 2) / data_size;
25      double dYInter = (ptLeft.y - ptRight.y - m_interval * 2) / (max_QValue -
            min_QValue);
26
27      dc.SetWindowOrg(-m_interval - ptLeft.x,
28         (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
29
30      CPen m_brownPen;
31      m_brownPen.CreatePen(PS_SOLID, 3, RGB(210, 160, 90));
32      dc.SelectObject(m_brownPen);
33
34      int x, y;
35      x = (int)(m_num * dXInter);
36      y = (int)((m_QValues[m_num] - floor(min_QValue)) * dYInter);
37      dc.MoveTo(x, y);
38      x = (int)((m_num + 1) * dXInter);
39      y = (int)((m_QValues[m_num + 1] - floor(min_QValue)) * dYInter);
40      dc.LineTo(x, y);
41
42      // 恢复坐标系
43      dc.SetMapMode(MM_TEXT);
44      dc.SetWindowOrg(0, 0);
45
46
47      // 误差（右上角）
48      dc.SetMapMode(MM_LOMETRIC);
49      ptLeft = CPoint(rect.right / 2, rect.top);
50      ptRight = CPoint(rect.right, rect.bottom / 2);
51      dc.DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
52      dc.DPtoLP(&ptRight);
53      dc.SetWindowOrg(-m_interval - ptLeft.x,
54         (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
55
56      // 误差的最值
57      auto min_error = std::min_element(m_errorValues.begin(), m_errorValues.end());
58      auto max_error = std::max_element(m_errorValues.begin(), m_errorValues.end());
59      double min_errorValue = *min_error;
60      double max_errorValue = *max_error;
61
62      // x和y方向数据和绘图位置的比例关系，留出两侧空白
63      dXInter = (ptRight.x - ptLeft.x - m_interval * 2) / data_size;
```

```
64      dYInter = (ptLeft.y - ptRight.y - m_interval * 2) / (max_errorValue -
            min_errorValue);
65
66      //绘制误差图像
67      CPen m_bluePen;
68      m_bluePen.CreatePen(PS_SOLID, 3, RGB(0, 0, 255));
69      dc.SelectObject(m_bluePen);
70
71      x = (int)(m_num * dXInter);
72      y = (int)((m_errorValues[m_num] - floor(min_errorValue)) * dYInter);
73      dc.MoveTo(x, y);
74      x = (int)((m_num + 1) * dXInter);
75      y = (int)((m_errorValues[m_num + 1] - floor(min_errorValue)) * dYInter);
76      dc.LineTo(x, y);
77
78      CString str2;
79      str2.Format(_T("当前误差: %.2f m"), m_errorValues[m_num]);
80      dc.TextOut(data_size * dXInter - 100, 300, str2);
81
82      // 恢复坐标系
83      dc.SetMapMode(MM_TEXT);
84      dc.SetWindowOrg(0, 0);
85
86      // 液位 (左下角)
87      dc.SetMapMode(MM_LOMETRIC);
88      ptLeft = CPoint(rect.left, rect.bottom / 2);
89      ptRight = CPoint(rect.right / 2, rect.bottom);
90      dc.DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
91      dc.DPtoLP(&ptRight);
92      dc.SetWindowOrg(-m_interval - ptLeft.x,
93         (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
94
95      // 误差的最值
96      auto min_h = std::min_element(m_hValues.begin(), m_hValues.end());
97      auto max_h = std::max_element(m_hValues.begin(), m_hValues.end());
98      double min_hValue = *min_h;
99      double max_hValue = *max_h;
100
101     auto min_i = std::min_element(m_InputValue.begin(), m_InputValue.end());
102     auto max_i = std::max_element(m_InputValue.begin(), m_InputValue.end());
103     double min_iValue = *min_i;
104     double max_iValue = *max_i;
```

```
105
106     if (max_iValue > max_hValue)
107        max_hValue = max_iValue;
108
109     // x和y方向数据和绘图位置的比例关系，留出两侧空白
110     dXInter = (ptRight.x - ptLeft.x - m_interval * 2) / data_size;
111     dYInter = (ptLeft.y - ptRight.y - m_interval * 2) / (max_hValue - 0);
112
113     //绘制液位图像
114     CPen m_redPen;
115     m_redPen.CreatePen(PS_SOLID, 3, RGB(255, 0, 0));
116     dc.SelectObject(m_redPen);
117
118     x = (int)(m_num * dXInter);
119     y = (int)(m_hValues[m_num] * dYInter);
120     dc.MoveTo(x, y);
121     x = (int)((m_num + 1) * dXInter);
122     y = (int)(m_hValues[m_num + 1] * dYInter);
123     dc.LineTo(x, y);
124
125     // 绘制期望液位图像
126     CPen m_greenPen;
127     m_greenPen.CreatePen(PS_SOLID, 3, RGB(0, 255, 0));
128     dc.SelectObject(m_greenPen);
129
130     x = (int)(m_num * dXInter) + 10;
131     y = (int)(m_InputValue[m_num] * dYInter);
132     dc.MoveTo(x, y);
133     x = (int)((m_num + 1) * dXInter) + 10;
134     y = (int)(m_InputValue[m_num + 1] * dYInter);
135     dc.LineTo(x, y);
136
137     // 恢复坐标系
138     dc.SetMapMode(MM_TEXT);
139     dc.SetWindowOrg(0, 0);
140
141     // 绘制水槽
142     dc.SetMapMode(MM_LOMETRIC);
143     ptLeft = CPoint(rect.right / 2, rect.bottom / 2);
144     ptRight = CPoint(rect.right, rect.bottom);
145     dc.DPtoLP(&ptLeft); // 设备坐标转换为逻辑坐标
146     dc.DPtoLP(&ptRight);
```

```
147         dc.SetWindowOrg(-m_interval - ptLeft.x,
148             (int)(-ptRight.y - m_interval)); // 重新设定坐标原点
149
150         // 水槽区域
151         CRect m_bucketRect = CRect(rect.right / 2, rect.bottom / 2, rect.right,
                rect.bottom); // 右下角区域
152         CRect m_waterRect = CRect(0, dYInter * (m_hValues[m_num + 1] - 0), 600, 0); //
                水的区域
153         //InvalidateRect(&m_bucketRect); // 重绘
154         RedrawWindow(&m_bucketRect);
155
156         // 水
157         dc.FillSolidRect(m_waterRect, RGB(0, 0, 255));
158
159         // 期望液位
160         dc.MoveTo(0, dYInter * (m_InputValue[m_num + 1] - 0));
161         dc.LineTo(600, dYInter * (m_InputValue[m_num + 1] - 0));
162
163         m_num++;
164         if (m_num == data_size - 1) KillTimer(1);
165     }
166
167     CView::OnTimer(nIDEvent);
168
169 }
```

对工具条中对应的按钮添加消息响应函数：按下开始按钮时，创建一个计时器；按下暂停按钮时，将判断暂停和运行的状态，创建或关闭计时器；按下停止按钮时，关闭计时器，并重置变量 m_num。代码如下所示：

```
1  void CFinalHomeworkView::OnStart()
2  {
3      // TODO: 在此添加命令处理程序代码
4      SetTimer(1, 1000 * m_TimeStep, NULL);
5  }
6
7  void CFinalHomeworkView::OnPause()
8  {
9      // TODO: 在此添加命令处理程序代码
10     if (m_isPause) {  // 现在暂定，开启定时器
11         SetTimer(1, 1000*m_TimeStep, NULL); m_isPause = false;
12     }
13     else {       // 正在运行，关闭定时器
```

```
14      KillTimer(1); m_isPause = true;
15    }
16
17  }
18
19  void CFinalHomeworkView::OnStop()
20  {
21    // TODO: 在此添加命令处理程序代码
22    KillTimer(1);
23    m_num = 0;
24    Invalidate();
25  }
```

3.3.5 文件的保存与加载函数

对“保存”和“打开”按钮添加消息响应函数，利用 Doc 类中的 Serialize 函数、Get 函数和 Set 函数完成文件的保存和加载，代码如下：

```
1   void CFinalHomeworkView::OnFileOpen()
2   {
3     // TODO: 在此添加命令处理程序代码
4     m_hValues.clear();
5     m_uValues.clear();
6     m_QValues.clear();
7     m_errorValues.clear();
8     m_timePoints.clear();
9     m_InputValue.clear();
10    m_num = 0;
11
12    CFileDialog fileDlg(TRUE, NULL, NULL, OFN_FILEMUSTEXIST | OFN_HIDEREADONLY,
          _T("All Files (*.*)|*.*||"));
13    if (fileDlg.DoModal() == IDOK)
14    {
15      CString filePath = fileDlg.GetPathName();
16      CFile file;
17      if (file.Open(filePath, CFile::modeRead))
18      {
19        CArchive ar(&file, CArchive::load);
20        CFinalHomeworkDoc* pDoc = GetDocument();
21        ASSERT_VALID(pDoc);
22        pDoc->Serialize(ar);
23        pDoc->Get(m_A, m_h0, m_A0, m_Ku, m_Max, m_T, m_TimeLength, m_TimeStep,
```

```
            m_Kp, m_Ki, m_Kd, m_InputType);
        ar.Close();
        file.Close();
        Invalidate();
    }
    else
    {
        AfxMessageBox(_T("无法打开文件"), MB_ICONERROR | MB_OK);
    }
}

if (m_InputType == 0) {
    if (m_Max < 0 || m_Max>10) {
        AfxMessageBox(_T("数值超出范围"), MB_ICONERROR | MB_OK);
            return;
    }
    for (double t = 0.0; t <= m_TimeLength; t += m_TimeStep)
    {
        m_timePoints.push_back(t);
        m_InputValue.push_back(m_Max);
    }
}
else {
    if (m_Max < 0 || m_Max>5 || m_T < 5 || m_T > 10) {
        AfxMessageBox(_T("数值超出范围"), MB_ICONERROR | MB_OK);
            return;
    }
    for (double t = 0.0; t <= m_TimeLength; t += m_TimeStep)
    {
        double h = m_Max * sin(2 * 3.14159265 / m_TimeLength * t) + m_h0;
        m_timePoints.push_back(t);
        m_InputValue.push_back(h);
    }
}
RungeKutta4();
m_Draw = true;
Invalidate();
}


void CFinalHomeworkView::OnFileSave()
{
```

```
65    // TODO: 在此添加命令处理程序代码
66    CFileDialog fileDlg(FALSE, NULL, NULL, OFN_OVERWRITEPROMPT, _T("All Files
          (*.*)|*.*||"));
67    if (fileDlg.DoModal() == IDOK)
68    {
69       CString filePath = fileDlg.GetPathName();
70       CFile file;
71       if (file.Open(filePath, CFile::modeCreate | CFile::modeWrite))
72       {
73          CArchive ar(&file, CArchive::store);
74          CFinalHomeworkDoc* pDoc = GetDocument();
75          ASSERT_VALID(pDoc);
76          pDoc->Set(m_A, m_h0, m_A0, m_Ku, m_Max, m_T, m_TimeLength, m_TimeStep,
                m_Kp, m_Ki, m_Kd, m_InputType);
77          pDoc->Serialize(ar);
78          ar.Close();
79          file.Close();
80       }
81       else
82       {
83          AfxMessageBox(_T("无法保存文件"), MB_ICONERROR | MB_OK);
84       }
85    }
86 }
```

4 实验结果

点击菜单栏“操作-对话框”将显示默认对话框，如图8所示。当输入信号不满足要求时，会弹出报错信息，如图9所示。

图 8: 默认对话框

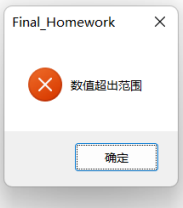

图 9: 报错信息

阶跃输入和正弦输入绘制得到的图像分别如图10和图11所示。结果显示程序能够完成对于两种输入模式的绘图。

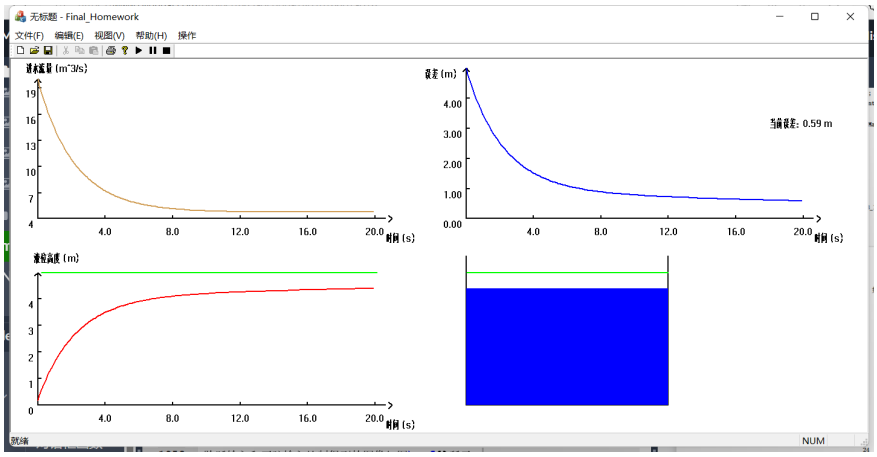

图 10: 阶跃输入结果

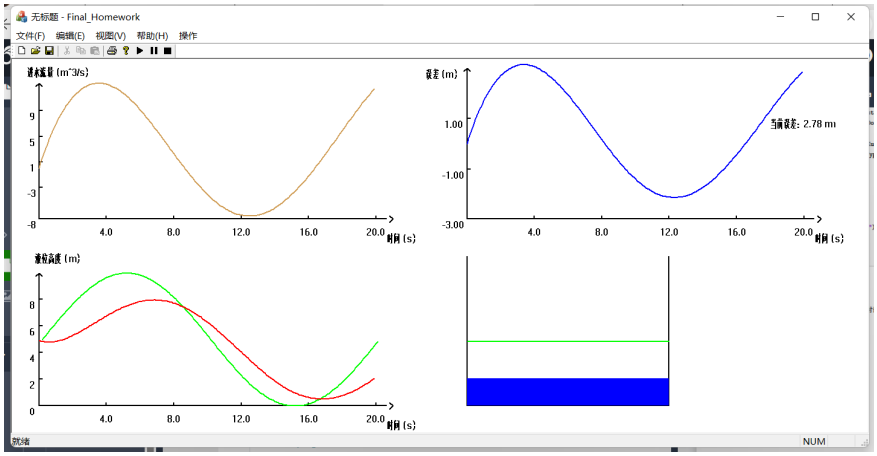

图 11: 正弦输入结果

动态的实验结果见录屏文件，其中展示了绘图的动画过程、“开始”“暂停”和“停止”键的使用、参数的保存和加载等。

5 实验总结

在本次实验中，我成功设计并实现了一个用于水槽液位高度控制的智能系统。通过采用 PID 控制器，结合数值求解方法，对单容对象的液位高度进行了有效控制。具体来说，实验过程中主要使用了以下方法：

1. 物料平衡方程的建立：通过分析进水流量 Q_i 和出水流量 Q_0 的关系，建立了液位高度 $h(t)$ 的动态模型。

2. PID 控制器设计：设计了一个 PID 控制器来调节进水阀门开度 $u(t)$，使液位高度 $h(t)$ 能够达到并保持在期望值。

3. 数值求解方法：为了对所建立的微分方程进行求解，实验中采用了四阶龙格-库塔法。这种方法能够在保证计算精度的同时提高计算效率，适用于求解复杂的非线性微分方程。

4. MFC 编程：在程序实现上，利用 MFC 框架设计了实验界面。包括：

   - CMyDlg 类：用于获取和设置对话框中的控制参数。
   - Doc 类：用于管理控制参数的保存与加载。
   - View 类：用于绘制仿真图形，展示液位高度、进水流量和误差曲线。

通过本次实验，我不仅巩固了对 PID 控制理论的理解，还掌握了数值求解方法在实际控制系统中的应用。此外，我掌握了 MFC 编程中绘图、文档、资源、对话框、消息响应、计时器、快捷键等方面的应用，增强了对复杂界面程序设计的能力。