

---

# **城市地下管线 PHM 系统使用文档**

2016-04-14

---

# 目 录

1. 软件安装运行说明.....	2
1.1. 初次运行配置.....	2
1.1.1. 配置系统环境变量 .....	2
1.1.2. 安装 MATLAB 库.....	2
1.1.3. 配置 oracle 数据库驱动.....	2
1.1.4. 测试数据库连接 .....	2
1.2. 软件目录结构.....	3
1.3. 软件运行说明.....	4
1.3.1. 安装并运行服务.....	4
1.3.2. 停止服务.....	4
1.3.3. 卸载服务.....	5
1.3.4. 功能模块运行日志.....	5
1.3.5. 服务运行日志.....	5
2. 软件扩展说明.....	5
2.1. 对象数目扩展.....	6
2.1.1. 管线数目扩展 .....	6
2.1.2. 传感器数目扩展 .....	6
2.2. CPD 表的配置.....	7
2.2.1. 故障与故障的 CPD.....	8
2.2.2. 故障与传感器的 CPD.....	9
2.3. 使用 xml 来存储 D 矩阵与 CPD 表（新） .....	10
2.3.1 xml 文件头 .....	12
2.3.2 故障结构节点<faultStructure> .....	12
2.3.3 传感器节点<sensor>.....	13
2.3.4 修改 xml 中的数据 .....	13
2.3.5 对 current_Water(or other)_info.xml 文件的说明 .....	13
2.4 健康度评估模块的参数配置.....	14
2.5 故障预警模块的参数配置.....	15
3. 其他 .....	15
3.1 .....	15

---

## 1. 软件安装运行说明

### 1.1. 初次运行配置

#### 1.1.1. 配置系统环境变量

解压缩安装包，以管理员方式运行根目录下的 AddPhmPath.bat，写入完成后可能需要注销或重启以使得环境变量生效。

可在命令提示符中输入 `echo %PHM_HOME%` 查看新的环境变量是否生效；注意设置完环境变量后，如果移动了软件文件夹所在位置，需要再次运行 AddPhmPath.bat 重新设置环境变量。

#### 1.1.2. 安装 MATLAB 库

运行 \MCR\_files 中的 MCRInstaller.exe，按步骤安装到某个位置（例如 D:\xxx\）。

#### 1.1.3. 配置 oracle 数据库驱动

将 \MCR 配置 中的 add\_classpath.bat 和 ojdbc6.jar 放到之前安装的 MATLAB 库的根目录下（例如 D:\xxx\v716\）

运行 add\_classpath.bat 将 oracle 驱动的地址写入 MATLAB 环境，运行成功后会有提示。

#### 1.1.4. 测试数据库连接

以上 3 步成功后，打开 \DBinfo.ini，修改其中的

service\_name\_oracle: 服务名/SID

username\_oracle: 用户名

password\_oracle: 密码

database\_url\_oracle: 将最后的 localhost:1521 改为目标 IP 和端口号

---

修改完成后，运行 \Precompiled EXE\linkDB\_test.bat 。初次运行等待时间较长，成功连接上数据库会有提示；如未成功连接也会有相应错误提示。

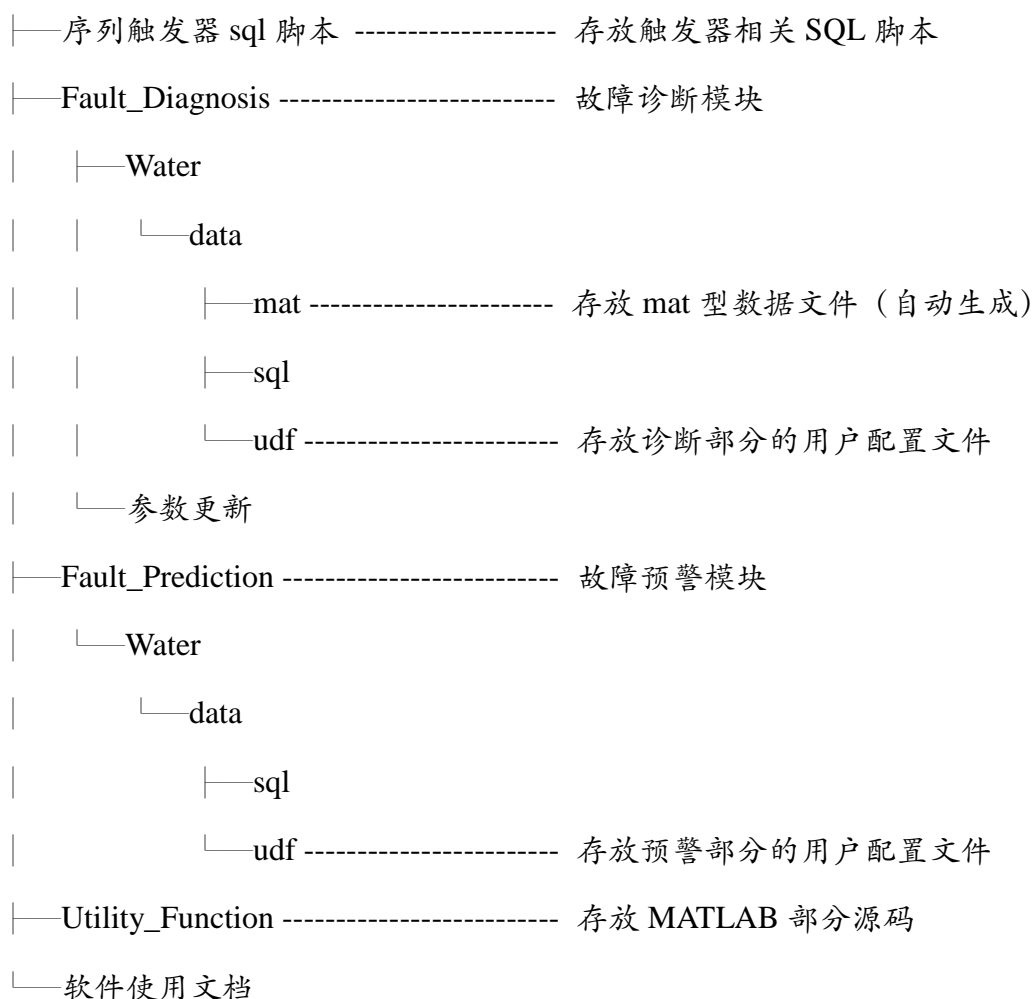
连接成功后，运行 \序列触发器 sql 脚本\create\_trigger.sql 给故障预警表 YJ\_WARNING\_FORECAST 添加触发器。

## 1.2. 软件目录结构

本软件目录结构及相关说明如下：

根目录.

- |—Csharp 代码
  - | |—PHM 运行失败日志 ----- 存放软件运行异常日志
  - | |—WindowsServicePHM ----- 存放 PHM 服务 C#源码及 exe 文件
- |—Precompiled EXE ----- 存放 Matlab 编译出来的 exe 文件
  - | |—PHMdiag ----- 故障诊断 exe 文件
  - | |—PHMpredict ----- 故障预测 exe 文件
- |—MCR files ----- 存放 MCR 配置文件
- |—Health\_Evaluation ----- 健康度评估模块
  - | |—Gas（燃气管线，暂无）
  - | |—Heat（热力管线，暂无）
  - | |—Rain（雨水管线，暂无）
  - | |—Sewage（污水管线，暂无）
  - | |—Water ----- 存放给水管线的代码及数据
    - | | |—data ----- 存放数据文件
    - | | |—sql ----- 存放相关的 SQL 文件
    - | | |—udf ----- UserDefineFile 存放用户配置文件
- |—RUL\_Prediction ----- 剩余寿命预测模块



### 1.3. 软件运行说明

本软件以 windows 服务的方式运行，可设置为开机自启，运行过程无需人工干预。运行完成后将数据直接写入数据库的相关表格，需要在数据库中查看运行结果。

#### 1.3.1. 安装并运行服务

以管理员方式运行 \Csharp 代码\InstallService.bat，执行服务的安装，安装完毕后服务自动开始执行，可在 \Csharp 代码\ServicePHM.log 中查看服务实时运行状态。

#### 1.3.2. 停止服务

在命令提示符 CMD 中输入 net stop PHM 可停止本服务的运行；输入 net

---

start PHM 开启服务；也可在系统的服务管理界面中对本服务进行操作。

### 1.3.3. 卸载服务

以管理员方式运行 \Csharp 代码\UninstallService.bat，卸载本服务。若服务正在运行，则会先关闭服务，再进行卸载。

### 1.3.4. 功能模块运行日志

\Csharp 代码\PHM.log 是软件各个模块的运行日志，每执行一次诊断、评估、或预警，都会在 PHM.log 中写入实时运行状态日志。

若该功能模块运行成功，日志会保留一段时间，直到被下一次的日志覆盖；若运行失败，日志会被转入到 \Csharp 代码\PHM 运行失败日志 中保存，日志名称为运行的时间。便于在以后进行集中查看处理。

### 1.3.5. 服务运行日志

\Csharp 代码\ServicePHM.log 是 PHM 服务的运行日志，用于监控 PHM 服务的运行状态，主要包括服务的开启时刻，关闭时刻，每次执行诊断、评估、预警等功能模块的时间，执行是否成功，服务总运行时间，总运行次数等信息。

若服务启动失败或运行状态异常，能迅速在 ServicePHM.log 反映出来，便于查找异常原因，进行相应的软件维护措施。

## 2.软件扩展说明

本软件部分内容（核心算法除外）支持用户的自行定义和配置，主要包括对象数目的扩展、条件概率表 CPD 的配置、健康度评估模块中不同影响因素的权重配置、故障预警模块的相关配置等。

## 2.1. 对象数目扩展

对于同一类型的管线，本软件支持模型结构的横向扩展，即传感器数目和管线数目的扩展。但不支持纵向扩展（如增加故障种类、改变故障的因果关系等）。

以给水管线为例，数目扩展主要是通过修改传感器与管线的 D 矩阵文件——\\Fault\_Diagnosis\\Water\\data\\udf\\Dmatrix\_Water.txt 实现。打开该文件，可看到传感器名称以及它能监测到的管线编号，如图 1 所示：



212015090057	:GX_JSL_3000_JYJ_254	:GX_JSL_3000_JYJ_257	:GX_JSL_3000_JYJ_255,GX_JSL_3000_JYJ_256
212015090004	:GX_JSL_3000_JYJ_258	:GX_JSL_3000_JYJ_261	:GX_JSL_3000_JYJ_259,GX_JSL_3000_JYJ_260
212015090002	:GX_JSL_3000_JYJ_261	:GX_JSL_3000_JYJ_266	:GX_JSL_3000_JYJ_265,GX_JSL_3000_JYJ_264,GX_
212015090044	:GX_JSL_3000_JYJ_267	:GX_JSL_3000_JYJ_270	:GX_JSL_3000_JYJ_268,GX_JSL_3000_JYJ_269
212015090059	:GX_JSL_3000_JYJ_231	:GX_JSL_3000_JYJ_234	:GX_JSL_3000_JYJ_232,GX_JSL_3000_JYJ_233
212015090032	:GX_JSL_3000_JYJ_243	:GX_JSL_3000_JYJ_244	

图 1. Dmatrix\_Water.txt 示意图

第一行中，212015090057 代表传感器编号，英文字符冒号“:”作为分隔符，其可以检测到的管线组为后面 3 组：

- 1、GX\_JSL\_3000\_JYJ\_254
- 2、GX\_JSL\_3000\_JYJ\_257
- 3、GX\_JSL\_3000\_JYJ\_255,GX\_JSL\_3000\_JYJ\_256（这组无法进一步细分）

### 2.1.1. 管线数目扩展

若某一条对应关系为 BG1 : 358: 371；此时若新增的管线编号 900 也可以被 BG1 监测到，只需在 371 后面添加字符“: 900”即可，变为“BG1: 358: 371: 900”（注意冒号必须为英文字符）。

如果新增的管线无法被当前任意一个传感器监测到，则该管线属于不可探测管线，无需添加对应关系。

### 2.1.2. 传感器数目扩展

如果现在新增了传感器 T800，则需要根据传感器的安装位置找出该传感器

能够监测到的管线编号，如 100, 200, 300 和 400。然后在 txt 文件的末尾添加一行“T800 : 100: 200: 300: 400”即可。

通过以上 2 步，就可以实现对管线数目和传感器数目的扩展，但还需要在 CPD 表中添加对应的 CPD 值才能够使软件正常运行（见 2.2. CPD 表的修改）。注意所有修改完成后，需要运行\Fault\_Diagnosis\删除原配置-给水.bat，以删除之前的缓存数据，确保修改生效。

下一次执行时，软件就会读入 Dmatrix\_Water.txt 中的新型结构，并自动搭建新型贝叶斯推理网络，完成后续的各个功能步骤，无需修改源码重新编译。

## 2.2. CPD 表的配置

条件概率表（CPD）的初始值需要由人工设定，它来自于先验知识，可看作是从以往大量的历史数据中统计出来的结果，体现了：

1. 各故障之间的因果关系大小。
2. 管线运行状态与传感器数值之间的关系。

以给水管线为例，打开 \故障诊断\Water\data\udf\CPD\_Water.txt，如下图 2：

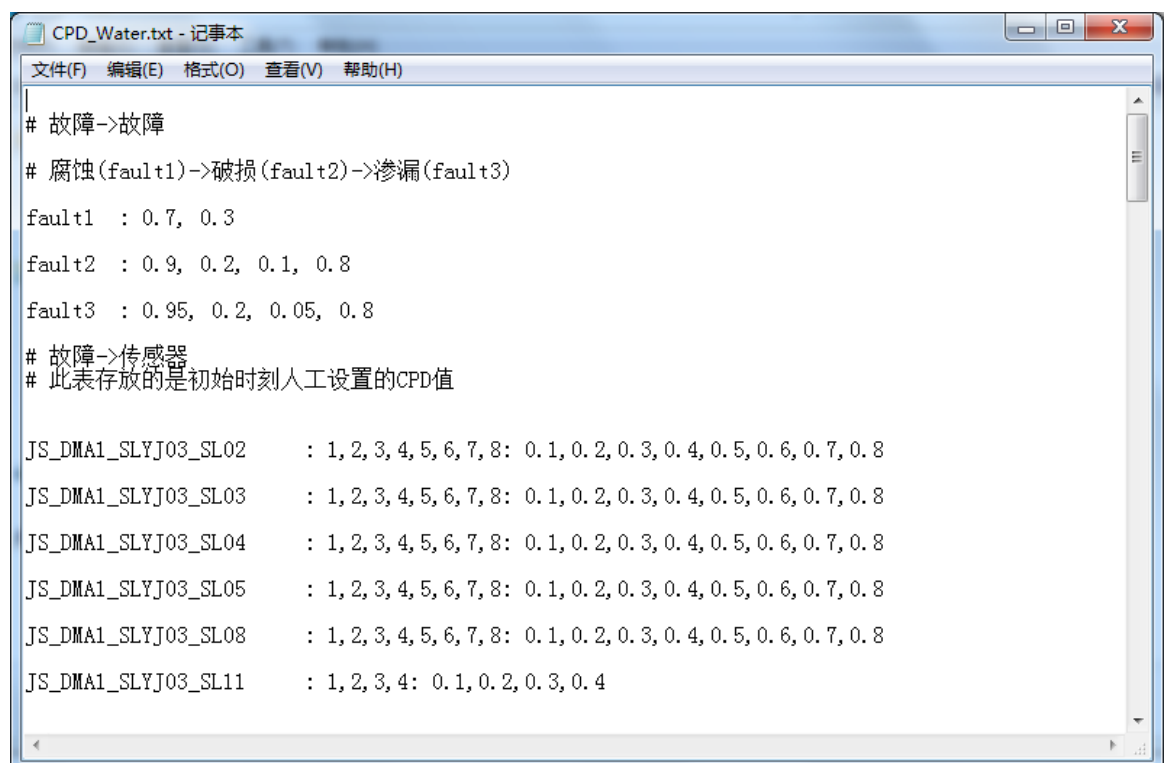




图 2. CPD\_Water.txt 示意图

### 2.2.1. 故障与故障的 CPD

给水管线的故障结构为：腐蚀->破损->渗漏。每个节点都是离散节点，只有 2 种状态，发生/不发生，因此可用 0/1 表示。若某离散节点有  $n$  个父节点，则其 CPD 值有  $2^{n+1}$  个。因此此处 3 个节点的 CPD 值分别为 2、4、4 个。

这里分别用 C、B、L 代表 3 种故障，则第一行腐蚀故障的 0.7, 0.3 代表  $P(C=0)=0.7$ ,  $P(C=1)=0.3$ ，即 C 不发生的概率为 0.7，发生的概率为 0.3。

第二行破损故障的 0.9, 0.2, 0.1, 0.8 代表  $P(B=0|C=0)=0.9$ ,  $P(B=0|C=1)=0.2$ ,  $P(B=1|C=0)=0.1$ ,  $P(B=1|C=1)=0.8$ 。

第三行渗漏故障的 0.95, 0.2, 0.05, 0.8 代表  $P(L=0|B=0)=0.95$ ,  $P(L=0|B=1)=0.2$ ,  $P(L=1|B=0)=0.05$ ,  $P(L=1|B=1)=0.8$ 。

注意设置数值时必须满足  $P(X=0|Y=0) + P(X=1|Y=0) = 1$ ，即在其他条件相同时，故障发生与不发生的概率之和必须为 1。

#### 1、故障之间的CPD（示例如下）

故障关系描述：腐蚀(C)->破损(B)->渗漏(L)

```

C
0  x1 = P(C=0)
1  x2 = P(C=1)
-----
B  C
0  0  y1 = P(B=0 | C=0)
0  1  y2 = P(B=0 | C=1)
1  0  y3 = P(B=1 | C=0)
1  1  y4 = P(B=1 | C=1)
-----
L  B
0  0  z1 = P(L=0 | B=0)
0  1  z2 = P(L=0 | B=1)
1  0  z3 = P(L=1 | B=0)
1  1  z4 = P(L=1 | B=1)
-----

```

结果节点写前面，原因节点写后面，按二进制数递增排列。

最终记录为：

```

C      : x1, x2
B      : y1, y2, y3, y4
L      : z1, z2, z3, z4

```

图 3. 故障之间的 CPD 示意图

## 2.2.2. 故障与传感器的 CPD

见图 2 的后半部分，传感器的输出是一个连续值，在同一状态下的输出应服从高斯分布  $N(\text{mean}, \text{cov})$ ，因此需要给出管线位于不同状态时传感器输出的均值  $\text{mean}$  和方差  $\text{cov}$ ，这也可以从历史数据中学习得到。

注意，CPD\_Water.txt 中传感器的数目、名称及排序必须与 Dmatrix\_Water.txt 中的完全相同，否则会导致某些传感器找不到对应的 CPD 值，引起错误。因此设置 CPD\_Water.txt 时必须对照着 Dmatrix\_Water.txt 进行。

例如：对于 Dmatrix 的第一条记录“BG1 : 358: 371”，其对应的 CPD 记录为“BG1 : 1,2,3,4: 0.1,0.2,0.3,0.4”。这里用 A 代表管线 358 的渗漏故障，B 代表管线 371 的故障，则 CPD 含义如下表：

A (发生为 1)    B (不发生为 0)	高斯分布均值 mean	高斯分布方差 cov
0                      0	1	0.1
1                      0	2	0.2
0                      1	3	0.3
1                      1	4	0.4

即：当 A，B 均不发生时，传感器数值在 1 左右，方差为 0.1；

当 A 发生而 B 不发生时，传感器数值在 2 左右，方差为 0.2；

当 A 不发生而 B 发生时，传感器数值在 3 左右，方差为 0.3；

当 A 发生而 B 也发生时，传感器数值在 4 左右，方差为 0.4；

以此类推，若某传感器可以监测 3 个管线的渗漏状态，则其 CPD 值应为 8 个。即对于传感器高斯节点来说， $n$  个管线对应  $2^n$  个 CPD 值，配置时一定要保证数目的匹配，否则会报错。

## 2、故障与传感器之间的CPD（示例如下）

注意：CPD表需要与D矩阵一一对应！

假设D矩阵为：

T1 : pipe1: pipe2

（1）传感器节点为离散节点，其取值为0/1

则CPD表为：

T1	pipe2	pipe1	Prob（注意pipe的顺序要反过来）
0	0	0	x1
0	0	1	x2
0	1	0	x3
0	1	1	x4
1	0	0	x5
1	0	1	x6
1	1	0	x7
1	1	1	x8

最终记录为：

T1 : x1, x2, x3, x4, x5, x6, x7, x8（共有 $2^{(n+1)}$ 个——n为连接的管线数目）

（2）传感器节点为高斯节点，其取值为（均值/方差）

则CPD表为：

pipe2	pipe1	Mean	Cov（注意pipe的顺序要反过来）
0	0	m1	c1
0	1	m2	c2
1	0	m3	c3
1	1	m4	c4

最终记录为：

T1 : m1, m2, m3, m4: c1, c2, c3, c4（共有 $2^n$ 组——n为连接的管线数目）

（注意所有字符均为英文）

图 4. 故障与传感器之间的 CPD 示意图

因此，如果在 Dmatrix\_Water.txt 中增加或修改了传感器数目和管线数目，必须要在 CPD\_Water.txt 进行相应的修改或增加，并保证相关的匹配；否则会导致传感器找不到对应的 CPD，程序便无法运行。

## 2.3. 使用 xml 来存储 D 矩阵与 CPD 表（新）

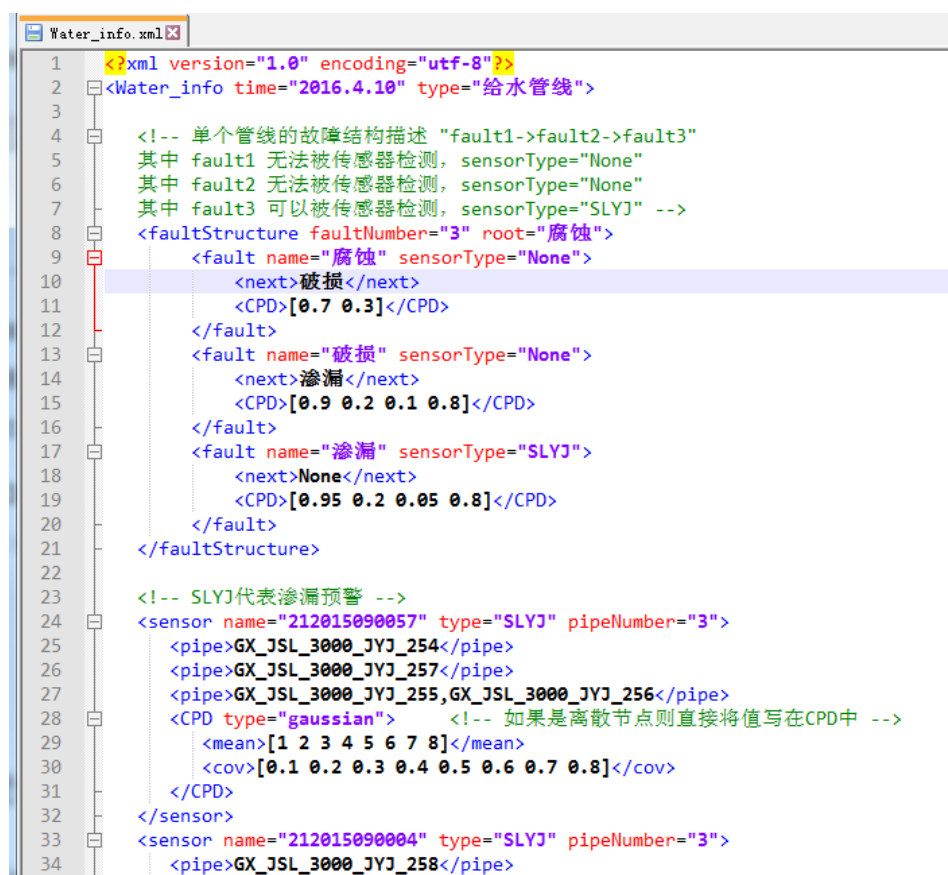
在 2.1 和 2.2 节中，我们介绍了如何使用 Dmatrix\_Water.txt 和 CPD\_Water.txt 以进行相关的配置操作。这种方法虽然能实现所需的功能，但存在以下几个不足：

- 1、故障与故障之间的关系无法体现：由图 2 可知，CPD\_Water.txt 中仅提供了故障的名称与条件概率，而各故障之间的因果关系实际上是固定在代码中的，无法更改。
- 2、当故障有多个时，传感器无法指定监控哪个故障：由图 1 的数据形式可知，传感器实际上只对应到管线名称，而具体对应到该管线的哪种故障是固定在代码中的。例如管线 p 有故障 f1, f2，其中 f1 可被传感器 t1 监测，f2 可被传感器 t2 监测，按照 Dmatrix\_Water.txt 中的记录形式为 t1: p, t2: p。实际

上这种记录方式比较混乱，无法体现出 t1 对应 f1 而 t2 对应 f2。

- 3、**D 矩阵和 CPD 表实行分开记录**：之前的方法将 D 矩阵和 CPD 表分开到 2 个文件中，但实际上这 2 者是紧密相关的；我们在记录信息时需 2 者对照进行配置，以保证每条记录能够一一对应，如果漏记或者多写，就会引起错误，不利于人工校对。

因此，在 2016.4.14 更新的版本中，我们摒弃了原先的 txt 配置方式，采用更为规范的 xml 格式，并将 D 矩阵和 CPD 合并到了一个表中。如下图所示：



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Water_info time="2016.4.10" type="给水管线">
3
4   <!-- 单个管线的故障结构描述 "fault1->fault2->fault3"
5   其中 fault1 无法被传感器检测, sensorType="None"
6   其中 fault2 无法被传感器检测, sensorType="None"
7   其中 fault3 可以被传感器检测, sensorType="SLYJ" -->
8   <faultStructure faultNumber="3" root="腐蚀">
9     <fault name="腐蚀" sensorType="None">
10       <next>破损</next>
11       <CPD>[0.7 0.3]</CPD>
12     </fault>
13     <fault name="破损" sensorType="None">
14       <next>渗漏</next>
15       <CPD>[0.9 0.2 0.1 0.8]</CPD>
16     </fault>
17     <fault name="渗漏" sensorType="SLYJ">
18       <next>None</next>
19       <CPD>[0.95 0.2 0.05 0.8]</CPD>
20     </fault>
21   </faultStructure>
22
23   <!-- SLYJ代表渗漏预警 -->
24   <sensor name="212015090057" type="SLYJ" pipeNumber="3">
25     <pipe>GX_JSL_3000_JYJ_254</pipe>
26     <pipe>GX_JSL_3000_JYJ_257</pipe>
27     <pipe>GX_JSL_3000_JYJ_255,GX_JSL_3000_JYJ_256</pipe>
28     <CPD type="gaussian"> <!-- 如果是离散节点则直接将值写在CPD中 -->
29       <mean>[1 2 3 4 5 6 7 8]</mean>
30       <cov>[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8]</cov>
31     </CPD>
32   </sensor>
33   <sensor name="212015090004" type="SLYJ" pipeNumber="3">
34     <pipe>GX_JSL_3000_JYJ_258</pipe>
```

图 5. Water\_info.xml 示意图

以给水管线的 Water\_info.xml 文件为例，该文件与原先的 txt 配置文件同位于 \PHM PIPE\Fault Diagnosis\Water\data\udf 中（其他类型的 xml 文件地址只需将 Water 替换为该类型名称即可）。下面对该 xml 中的内容进行分析，请参考图 3。注意：使用 windows 自带的文本编辑器打开 xml 会无法显示换行，推荐使用 **Notepad++**：（百度搜索第一个就是下载链接）

### 2.3.1 xml 文件头

按照 xml 的规范，以 `<?xml version="1.0" encoding="utf-8"?>` 开头；第二行 `<Water_info time="2016.3.23" type="给水管线">` 包含了文件信息，在雨水的 xml 中，该行为 `<Rain_info time="2016.4.7" type="雨水管线">`，其中 Rain\_info 为节点，`<>` 中的内容除节点外都为属性，如 time 属性为 "2016.4.7"，type 属性为 "雨水管线"，属性的值必须用英文双引号包含，否则读写时会出错！

第 4-7 行以 `<!-- -->` 包含的内容为注释，可填写对文档内容等的说明。可自行添加相关注释。

### 2.3.2 故障结构节点<faultStructure>

第 8-21 行包含在 `<faultStructure /faultStructure>` 中的为故障结构节点，该段内容描述了故障间的连接关系。第 8 行 `faultNumber="3"` 表示故障类型为 3 种，`root="腐蚀"` 表示故障树的根节点为腐蚀。

第 9 到 20 行为 3 个故障节点的详细描述，如第 9 行属性中，`name="腐蚀"` 表明该节点为腐蚀故障，而 `sensorType="None"` 表示该故障无法被下面的传感器监控到。

第 10-11 行描述了 `<fault>` 节点中的 2 个子节点，`next` 表示该故障节点指向的节点，如果为 None，则表示该故障为叶节点。CPD 表示该故障的条件概率（先验概率），CPD 的书写规则与 2.2 节中相同（详见图 3）。

第 17 行中，由于渗漏节点可以直接被渗漏预警传感器监测到，因此其 `sensorType="SLYJ"`，"SLYJ" 是对传感器类型的标识，可以自行定义别的名称，但需与第 24 行以后的 `<sensor>` 节点中的 type 属性保持一致！

因此，通过对 faultStructure 节点的分析，我们可以得到给水管线的故障连接方式如下：腐蚀（根节点）->破损->渗漏（叶节点），其中只有渗漏可以被 "SLYJ" 类型的传感器监测到。

通过以上方式，如果以后出现新的管线，只需配置其 xml 文件，无需修改代码，就能实现该类型贝叶斯网络结构的搭建。

### 2.3.3 传感器节点<sensor>

图 3 中的第 24-31 行描述了一个典型的<sensor>节点，其属性 name="212015090057"表示了传感器名称，type="SLYJ"表示传感器类型，此属性需要与 2.3.2 节中<fault>的 sensorType 属性一致，以表明传感器对故障的监测关系。pipeNumber="3"表示其能够监测到的管线组的个数（注意不是管线个数），该数目需要与<sensor>节点下的<pipe>节点数目保持一致，否则会出错！

第 25-27 行描述一个<pipe>节点的信息，为管线的名称，若有多条管线同属该组，则按照第 27 行的方式填写（注意逗号为英文字符，且管线名称前后中间均不能有空格！）。这部分实际上与原 Dmatrix\_Water.txt 中（见图 1）的描述方式相同。

第 28-31 行是<CPD>节点，属性 type="gaussian"表明 CPD 表类型为高斯，因此其含有<mean>和<cov>2 个子节点，若为离散节点则直接按<CPD>[1 2 ... n]<CPD>的格式进行书写。<mean>和<cov>节点中分别填写了均值和方差信息，其填写规则与 2.2 节中相同（详见图 4）。

通过以上的描述方法，一个<sensor>节点内包含了传感器名称 name、类型 type、能够监测到的管线组数目 pipeNumber，其下有具体的<pipe>节点表明管线名称、<CPD>节点填入 CPD 表数据。实现了原 Dmatrix\_Water.txt 和 CPD\_Water.txt 的融合。

### 2.3.4 修改 xml 中的数据

如需增加传感器节点<sensor>，只需按照 2.3.3 节中描述的格式进行添加，若要删除传感器节点，直接删去或将该节点注释即可。如需修改<CPD>节点下的<mean>和<cov>数据，则直接进行修改即可，注意<mean>中的数据个数和<cov>中的必须相等，且必须为 2 的 pipeNumber 次方（详见 2.2 节）。

注意，任何修改之后都必须运行\PHM\_PIPE\Fault\_Diagnosis\删除原配置-给水.bat（根据管线种类选择），以删除之前的缓存数据，确保修改生效。

### 2.3.5 对 current\_Water(or other)\_info.xml 文件的说明

以给水管线为例，Water\_info.xml 中包含了对原始人为设定的 CPD 表的描述，

具有较大的随意性；而在软件第一次运行（或以后的某时刻）时，会从传感器数据库中提取相应的数据计算均值和方差，因此会生成 current Water info.xml 文件，该文件作用仅为直观体现当前的 CPD 值（文件中包含时间属性，用户可查看该 CPD 被计算出来的时间），修改该文件对软件运行不造成影响。因此用户的所有操作都应在 Water info.xml 上进行！

## 2.4 健康度评估模块的参数配置

在健康度评估模块中，我们设置了影响健康度的 3 个要素：相对维修费用，相对破坏程度和专家建议，并依次分配影响权值为 0.4，0.4 和 0.2。以给水管线为例，打开 \Health\_Evaluation\Water\data\udf\健康度权值表.txt，如下图：

```
# 三个要素：分别占据一定的权重

# 相对维修费用
repair_cost_weight = 0.4
repair_cost_corrosion = 1
repair_cost_broken = 50
repair_cost_leak = 100

# 相对破坏程度
damage_weight = 0.4
damage_corrosion = 1
damage_broken = 20
damage_leak = 100

# 专家建议
expert_weight = 0.2
expert_corrosion = 1
expert_broken = 10
expert_leak = 100

# 分级标准 满分100分 数字表示多少分以上才能得到该评价
very_healthy = 80
healthy = 60
semi_healthy = 40
ill = 20
badly_ill = 0
```

图 4. 健康度权值表.txt 示意图

用户可根据实际情况修改上面三个因素的权值，一般确定后就不需要改动了。而本软件通过健康度评估算法对当前诊断结果进行处理后，会得到一个健康度分数，位于 0-100 之间。

最后一项分级标准，如 `very_healthy = 80` 表示分数在 80 以上才认为是“非常健康”；同理，`ill = 20` 表示分数在 20~40 之间的为“疾病”状态，而低于 20 为“严重疾病”状态。

---

用户可通过修改分级标准，来改变属于不同级别的阈值。

## 2.5 故障预警模块的参数配置

故障预警模块的用户可定义参数位于 `\Fault_Prediction\predict_config.ini` 中，包括 `n_train`、`n_predict` 和 `diag_trigger_time`：`n_train` 代表预测所用到的诊断记录数目，`n_predict` 代表要预测的数目，`diag_trigger_time` 表示诊断触发的时间（分钟）。

故障预警即是利用最近的 `n_train` 条诊断记录来预测未来的 `n_predict` 次诊断结果。设置 `n_train` 是为了舍弃较早之前的数据，避免很久以前的数据对近期预测的影响，用户可根据实际情况调整 `n_train` 的值。

而 `n_predict` 则是根据故障诊断的执行频率和故障预警的执行频率相除得到。即若一天执行 24 次诊断（每小时一次），并执行 1 次预警，则 `n_predict` 应该设置为 24，这样可以保证每次预警可以预测出未来一天内所有的诊断结果。因此有  $n\_predict \times diag\_trigger\_time = predict\_trigger\_time$ 。

## 3.其他

### 3.1