重大危险源事故后果数值分析

模型说明

# 目 录

[目 录 2](#_Toc332379964)

[一、泄 漏 模 型 1](#_Toc332379965)

[1、 气体泄漏模型 1](#_Toc332379966)

[1.1 模型简介 1](#_Toc332379967)

[1.2 算法说明 1](#_Toc332379968)

[1.3 接口 2](#_Toc332379969)

[1.4 代码示例 2](#_Toc332379970)

[2、液体泄漏模型 4](#_Toc332379971)

[2.1 模型简介 4](#_Toc332379972)

[2.2 算法说明 4](#_Toc332379973)

[2.3 接口 5](#_Toc332379974)

[2.4 代码示例 5](#_Toc332379975)

[二、 火 灾 模 型 7](#_Toc332379976)

[1、固体火灾模型 8](#_Toc332379977)

[1.1 模型简介 8](#_Toc332379978)

[1.2 算法说明 8](#_Toc332379979)

[1.3 接口 8](#_Toc332379980)

[1.4 代码示例 9](#_Toc332379981)

[2、池火灾模型 11](#_Toc332379982)

[2.1 模型介绍 11](#_Toc332379983)

[2.2 算法说明 11](#_Toc332379984)

[2.3 接口 11](#_Toc332379985)

[2.4 代码示例 12](#_Toc332379986)

[3、喷射火模型 14](#_Toc332379987)

[3.1 模型简介 14](#_Toc332379988)

[3.2 算法说明 14](#_Toc332379989)

[3.3 接口 15](#_Toc332379990)

[3.4 代码示例 15](#_Toc332379991)

[4、沸腾液体扩展蒸气爆炸 17](#_Toc332379992)

[4.1 模型简介 17](#_Toc332379993)

[4.2 算法说明 18](#_Toc332379994)

[4.3 接口 18](#_Toc332379995)

[4.4 代码示例 18](#_Toc332379996)

[三、 爆 炸 模 型 20](#_Toc332379997)

[1、蒸汽云爆炸模型 20](#_Toc332379998)

[1.1 模型简介 20](#_Toc332379999)

[1.2 算法说明 21](#_Toc332380000)

[1.3 接口 22](#_Toc332380001)

[1.4 代码示例 22](#_Toc332380002)

[2、凝聚相含能材料爆炸模型 24](#_Toc332380003)

[2.1 模型简介 24](#_Toc332380004)

[2.2 算法说明 24](#_Toc332380005)

[2.3 接口 24](#_Toc332380006)

[2.4 代码示例 24](#_Toc332380007)

[四、 压 力 容 器 爆 破 模 型 26](#_Toc332380008)

[1、盛装气体压力容器爆破模型 26](#_Toc332380009)

[1.1 模型简介 26](#_Toc332380010)

[1.2 模型简介 27](#_Toc332380011)

[1.3 接口 27](#_Toc332380012)

[1.4 代码示例 27](#_Toc332380013)

[2、盛装液体压力容器爆破模型 29](#_Toc332380014)

[2.1 模型简介 29](#_Toc332380015)

[2.2 算法说明 29](#_Toc332380016)

[2.3 接口 30](#_Toc332380017)

[2.4 代码示例 30](#_Toc332380018)

[3、盛装液化气体容器爆破模型 32](#_Toc332380019)

[3.1 模型简介 32](#_Toc332380020)

[3.2 算法说明 32](#_Toc332380021)

[3.3 接口 33](#_Toc332380022)

[3.4 代码示例 33](#_Toc332380023)

[4、盛装饱和水容器爆破模型 36](#_Toc332380024)

[4.1 算法说明 36](#_Toc332380025)

[4.2 接口 36](#_Toc332380026)

[4.3 代码示例 36](#_Toc332380027)

[五、 毒 气 扩 散 模 型 38](#_Toc332380028)

[1、非重气云扩散模型 38](#_Toc332380029)

[1.1 模型简介 38](#_Toc332380030)

[1.2 算法说明 39](#_Toc332380031)

[1.3 接口 39](#_Toc332380032)

[1.4 代码示例 40](#_Toc332380033)

[2、重气云扩散模型 46](#_Toc332380034)

[2.1 模型简介 46](#_Toc332380035)

[2.1 算法说明 46](#_Toc332380036)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 题目 | | |  |  |  |  | |  |  |
|  | | **重大危险源事故后果数值分析模型建立** | | |  | | | | |  | |
| **修 改 履 历** | | | | | | | | | | | |
| 版本 | 日期 | | 修改内容 | | | | | | 修改人 | | |
| O | 2012-7-13 | | 出版发行 | | | | | | 栾兴桃 | | |
| 1 | 2012-7-20 | | 加沸腾液体扩展蒸气爆炸模型说明 | | | | | | 栾兴桃 | | |
| 2 | 2012-8-2 | | 增加模型说明 | | | | | | 栾兴桃 | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  |  | |  | | | | | |  | | |
|  | | | |  | | | | | | | |

# 一、泄 漏 模 型

由于设备损坏或操作失误引起泄漏从而大量释放易燃、易爆、有毒有害物质，将会导致火灾、爆炸、中毒等重大事故发生，因此后果分析首先要考虑泄漏情况。

当发生泄漏设备的裂口是规则的，而且裂口尺寸及泄漏物质的有关热力学、物理化学性质及参数已知时，可根据流体力学中的有关方程式计算泄漏量。当裂口不规则时，可采取等效尺寸代替。当遇到泄漏过程中压力变化等情况时，往往采用经验公式。

## 气体泄漏模型

1.1 模型简介

在工业生产中涉及到大量的易燃易爆的气体，如压缩天然气、煤制气等，一旦运输、贮存这些气体的管道、储罐或其他设备发生破裂，气体从裂口泄漏出去，和空气混合形成可燃气云，就有发生火灾、爆炸的危险；若泄漏出的气体有毒性，则后果更为严重。因此要预测或评价气体泄漏后造成事故的严重程度，必须对气体泄漏量进行定量计算。

气体从裂口泄漏的速度与其流动状态有关。因此，计算泄漏量时首先要判断泄漏时气体的流动属于声速还是亚音速流动，前者称为临界流，后者称为次临界流。

1.2 算法说明

当下式成立时，气体流动属声速流动。



（1）

当下式成立时，气体流动速度属亚音速流动。



（2）



（3）

式中：——环境压强，pa；

——容器内的绝对压力，pa；

——泄漏气体的绝热指数，定压热容/定容热容；

——泄漏系数；

——裂口面积，m^2；

——气体分子质量，kg/mol；

——气体常数，8.314J/mol\*k；

——气体的温度，k；

1.3 接口

调用类：LeakageGasClass

输入参数：GapArea 裂口面积，m^2

GasTemperature 气体的温度，k

hotmeltAtCP 定压热容

HotmeltAtCV 定容热容

LeakCoefficient 气体的泄漏系数，见表1

MolarMass 气体分子质量，kg/mol

VesselPressre 容器内介质压力，pa

输出参数：GasFlowType 气体的流动状态

LeakRate 气体泄漏速度，kg/s

表1

例：GapArea=0.000078m，GasTemperature=10K，hotmeltAtCP=1231，HotmeltAtCV=33，LeakCoefficient=1.00，MolarMass=1233，VesselPressre=123000

1.4 代码示例

1、.NET示例

ILeakageGas em = new LeakageGasClass();

em.VesselPressure = Convert.ToDouble(txtVesselPressure.Text.Trim());

em.hotmeltAtCP = Convert.ToDouble(txthotmeltAtCP.Text.Trim());

em.HotmeltAtCV = Convert.ToDouble(txthotmeltAtCV.Text.Trim());

em.MolarMass = Convert.ToDouble(txtMolarMass.Text.Trim());

em.GasTemperature = Convert.ToDouble(txtGasTemperature.Text.Trim());

em.GapArea = Convert.ToDouble(txtGapArea.Text.Trim());

em.LeakCoefficient = Convert.ToDouble(txtLeakCoefficient.Text.Trim());

em.CalculateGasLeakage();

txtGasFlowType.Text = em.GasFlowType.ToString();

txtLeakRate.Value = Convert.ToDecimal(em.LeakRate.ToString(".00"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

1. JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.LeakageGas");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"VesselPressure",213100);

Dispatch.put(msg,"hotmeltAtCP",1231);

Dispatch.put(msg,"hotmeltAtCV",33);

Dispatch.put(msg,"MolarMass",1233);

Dispatch.put(msg,"GasTemperature",30);

Dispatch.put(msg,"GapArea",0.00058);

Dispatch.put(msg,"LeakCoefficient",1);

Dispatch.call(msg, "CalculateGasLeakage");

System.out.println("气体泄漏模型");

System.out.println("泄漏类型： "+Dispatch.get(msg, "GasFlowType"));

System.out.println("泄漏速率或泄露量： "+Dispatch.get(msg, "LeakRate")+" kg/s");

1. JavaScript

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>气体泄漏模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.LeakageGas");

ver.VesselPressure = 213100;

ver.hotmeltAtCP = 1231;

ver.hotmeltAtCV = 33;

ver.MolarMass = 1233;

ver.GasTemperature = 30;

ver.GapArea = 0.00058;

ver.LeakCoefficient = 1;

ver.CalculateGasLeakage();

var outPut = "<气体泄漏模型>" + "\n \n";

outPut += "气体流动类型：" + ver.GasFlowType + "\n";

outPut += "泄漏速率=" + ver.LeakRate;

alert(outPut);

</script>

</body>

</html>

## 2、液体泄漏模型

2.1 模型简介

对于常压下的液体泄漏速度，取决于裂口之上的液位高低；对于非常压下的液体泄漏速度，主要取决于容器内的介质压力与环境压力只差和液位高低。

2.2 算法说明

计算液体连续泄漏速度：



（4）

计算液体瞬时泄漏速度：



（5）

式中：——液体泄漏的质量速率，kg/s

——泄漏系数

——裂口面积，m^2

——泄漏液体密度，kg/m^3

——重力加速度，9.8m/s^2

——裂口之上液位高度，m

——容器的横截面积

——设备内物质压力（表压），pa

——泄漏时间，s

2.3 接口

调用类：

输入参数：GapArea 裂口面积，m^2

HeightDifference 裂口之上液位高度，m

LeakageCoefficient 泄漏系数，按表2选取

LeakTime 泄漏时间，s

LiquidDensity 泄漏液体密度，kg/m^3

VesselCrossSectionalArea 容器的横截面积，m^2

VesselPressure 设备内物质压力，pa

输出参数：LeakType 液体是顺势泄漏还是连续泄漏

LeakRateOrAmount 泄漏速度还是泄漏量

表2

例：

2.4 代码示例

1、.NET示例

ILeakageLiquid em = new LeakageLiquidClass();

em.LeakageCoefficient = Convert.ToDouble(txtLeakageCoefficient.Text.Trim());

em.GapArea = Convert.ToDouble(txtGapArea.Text.Trim());

em.LiquidDensity = Convert.ToDouble(txtLiquidDensity.Text.Trim());

em.VesselPressure = Convert.ToDouble(txtVesselPressure.Text.Trim());

em.HeightDifference = Convert.ToDouble(txtHeightDifference.Text.Trim());

em.VesselCrossSectionalArea = Convert.ToDouble(txtVesselCrossSectionalArea.Text.Trim());

em.LeakTime = Convert.ToDouble(txtLeakTime.Text.Trim());

em.CalculateLiquidLeakage();

txtLeakType.Text = em.LeakType.ToString();

if (txtLeakType.Text=="连续泄露")

{

txtLeakRate.Value = Convert.ToDecimal(em.LeakRateOrAmount.ToString(".00"));

}

else

{

txtLeakAmount.Value = Convert.ToDecimal(em.LeakRateOrAmount.ToString(".0000"));

txtLeakRate.Value = Convert.ToDecimal(em.LeakRateOrAmount.ToString(".0000"))/txtLeakTime.Value;

}

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.LeakageLiquid");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"LeakageCoefficient",0.8);

Dispatch.put(msg,"GapArea",0.0058);

Dispatch.put(msg,"LiquidDensity",1.98);

Dispatch.put(msg,"VesselPressure",213100);

Dispatch.put(msg,"HeightDifference",1.2);

Dispatch.put(msg,"VesselCrossSectionalArea",0.8);

Dispatch.put(msg,"LeakTime",29);

Dispatch.call(msg, "CalculateLiquidLeakage");

System.out.println("液体泄漏模型");

System.out.println("泄漏类型： "+Dispatch.get(msg, "LeakType"));

System.out.println("泄漏速率或泄露量： "+Dispatch.get(msg, "LeakRateOrAmount")+" m");

1. JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>液体泄漏模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.LeakageLiquid");

ver.LeakageCoefficient = 0.8;

ver.GapArea = 0.0058;

ver.LiquidDensity = 1.98;

ver.VesselPressure = 213100;

ver.HeightDifference = 1.2;

ver.VesselCrossSectionalArea = 0.8;

ver.LeakTime =29;

ver.CalculateLiquidLeakage();

var outPut = "<液体泄漏模型>" + "\n \n";

outPut += "泄露类型：" + ver.LeakType + "\n";

outPut += "泄漏速率或泄露量=" + ver.LeakRateOrAmount;

alert(outPut);

</script>

</body>

</html>

# 火 灾 模 型

易燃、易爆的气体、液体泄漏后遇到引火源就会被点燃而着火。火灾对周围环境的影响主要在于其热辐射，火灾辐射热的影响范围一般均在200m左右的近火源区域。火灾主要有固体火灾、池火灾、喷射火和沸腾液体扩展蒸气爆炸四种。

## 1、固体火灾模型

1.1 模型简介

固体火灾的热辐射参数按点源模型估计。此模型认为火焰射出的能量为燃烧的一部分，并且辐射强度与目标至火源中心距离的平方成反比。

1.2 算法说明

目标接受到的热辐射通量：



（6）

皮肤裸露时的死亡几率：



（7）

一度烧伤几率：



（8）

二度烧伤几率：



（9）

财产损失半径计算公式：



（10）

式中：、、、、——目标接受到的热通量，kw/m^2；

——目标离火源的距离，m；

——物质质量，kg；

——伤害几率（如=5时，人员伤害的百分数为50%）；

——物质燃烧时间，s；

——燃烧热，KJ/kg；

——热辐射系数，取0.25

1.3 接口

调用类：FireSolidClass

输入参数：CombustionHeat 固体物质的燃烧放热，J/kg

CombustionRate 固体物质的燃烧速度，kg/s

MaterialWeight 固体燃烧物的质量，kg

输出参数：Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：CombustionHeat=4250 J/kg，CombustionRate=50 kg/s，MaterialWeight=500 kg

1.4 代码示例

1、.NET示例

IFireSolid fs = new FireSolidClass();

fs.MaterialWeight = Convert.ToDouble(txtMaterialWeight.Text.Trim());

fs.CombustionHeat = Convert.ToDouble(txtCombustionHeat.Text.Trim());

fs.CombustionRate = Convert.ToDouble(txtCombustionRate.Text.Trim());

fs.CalculateSolidFire();

spnDeathR.Value = Convert.ToDecimal(fs.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(fs.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(fs.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(fs.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.FireSolid");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"MaterialWeight",10000);

Dispatch.put(msg,"CombustionRate",100);

Dispatch.put(msg,"CombustionHeat",1000);

Dispatch.call(msg, "CalculateSolidFire");

System.out.println("固体火灾模型");

System.out.println("物质质量： "+Dispatch.get(msg, "MaterialWeight")+" kg");

System.out.println("燃烧速率： "+Dispatch.get(msg, "CombustionRate")+" kg/s");

System.out.println("燃烧热： "+Dispatch.get(msg, "CombustionHeat")+" kj/kg");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>固体火灾模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.FireSolid");

ver.MaterialWeight = 10000;

ver.CombustionRate = 100;

ver.CombustionHeat = 1000;

ver.CalculateSolidFire();

var outPut = "<固体火灾模型>" + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage;

alert(outPut);

</script>

</body>

</html>

## 2、池火灾模型

2.1 模型介绍

可燃液体（如汽油、柴油等）泄漏后流到地面形成液池，或流到水面覆盖水面，遇到点火源燃烧而成池火灾。

2.2 算法说明

计算池直径：

（11）



计算火焰高度：



（12）

计算火焰表面的热通量：



（13）

计算目标接受的热辐射通量：



（14）

式中：——池面积，m^2；

——液池半径，m；

——火焰高度，m；

——燃烧速率，kg/(m^2\*s)；

——空气密度，kg/m^3；

——引力常数，9.8,/s^2；

——火焰表面的热通量，kw/m^2；

——燃烧热，kJ/kg；

2.3 接口

调用类：FirePool

输入参数：CombustionHeat 燃烧放热，J/kg

CombustionRate 液体的燃烧速度，kg/s

LiquidWeight 泄漏物质量，kg

PoolArea 液池面积，m^2

输出参数：Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：CombustionHeat=5200 J/kg，CombustionRate=5200 kg/s，LiquidWeight=5000 kg，PoolArea=100 m^2

2.4 代码示例

1、.NET示例

IFirePool fp = new FirePoolClass();

fp.PoolArea = Convert.ToDouble(txtPoolArea.Text.Trim());

fp.LiquidWeight = Convert.ToDouble(txtMaterialWeight.Text.Trim());

fp.CombustionHeat = Convert.ToDouble(txtCombustionHeat.Text.Trim());

fp.CombustionRate = Convert.ToDouble(txtCombustionRate.Text.Trim());

fp.CalculatePoolFire();

spnDeathR.Value = Convert.ToDecimal(fp.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(fp.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(fp.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(fp.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.FirePool");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"PoolArea",10000);

Dispatch.put(msg,"CombustionRate",10000);

Dispatch.put(msg,"CombustionHeat",100000);

Dispatch.put(msg,"LiquidWeight",1000);

Dispatch.call(msg, "CalculatePoolFire");

System.out.println("池火灾模型");

System.out.println("液池面积： "+Dispatch.get(msg, "PoolArea")+" m^2");

System.out.println("燃烧速率： "+Dispatch.get(msg, "CombustionRate")+" m^2\*s");

System.out.println("燃烧热： "+Dispatch.get(msg, "CombustionHeat")+" kJ/kg");

System.out.println("燃烧质量： "+Dispatch.get(msg, "LiquidWeight")+" kg");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>池火灾模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.FirePool");

ver.PoolArea = 10000;

ver.CombustionRate = 10000;

ver.CombustionHeat = 100000;

ver.LiquidWeight = 1000;

ver.CalculatePoolFire();

var outPut = "<池火灾模型>" + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage;

alert(outPut);

</script>

</body>

</html>

## 3、喷射火模型

3.1 模型简介

加压的可燃物质泄漏时形成射流，如果在泄漏裂口处被点燃，则形成喷射火。这里所用的喷射火辐射热计算方法是一种包括气流在内的喷射扩散模式的扩展。把整个喷射火看出是有沿喷射中心线上的全部点源组成，每个点源的热辐射通量相等。

3.2 算法说明

气体泄漏速度：



（15）



（16）



（17）

计算点热源的热辐射通量，一般取点源n=5：



（18）

目标接受到热辐射通量：



（19）

式中：——环境压强，pa；

——容器内的绝对压力，pa；

——泄漏气体的绝热指数，定压热容/定容热容；

——裂口形状，圆形取1.00，三角形取0.95，长方形取0.90；

——裂口面积，m^2；

——气体分子质量，kg/mol；

——气体常数，8.314J/mol\*k；

——气体的温度，k；

3.3 接口

调用类：FireJetClass

输入参数：CombustionHeat 物质的燃烧热，J/kg

CombustionTime 燃烧时间，s

Diameter 小口直径，m

HotmeltAtConstantPressure 定压比热容

HotmeltAtConstantVolume 定容比热容

InitialPressure 容器的初始压力，pa

MolecularWeight 物质的分子质量，kg/mol

Temperature 容器的工作温度，K

输出参数：Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：CombustionHeat=12455 J/kg，CombustionTime=100 s，Diameter=0.0123 m，HotmeltAtConstantPressure=1231，HotmeltAtConstantVolume=33，InitialPressure=123000 pa，MolecularWeight=1233 kg/mol，Temperature=45 K

3.4 代码示例

1、.NET示例

IFireJet fj = new FireJetClass();

fj.MolecularWeight = Convert.ToDouble(txtMolecularWeight.Text.Trim());

fj.InitialPressure = Convert.ToDouble(txtInitialPressure.Text.Trim());

fj.HotmeltAtConstantPressure = Convert.ToDouble(txtHotmeltAtCP.Text.Trim());

fj.HotmeltAtConstantVolume = Convert.ToDouble(txtHotmeltAtCV.Text.Trim());

fj.Diameter = Convert.ToDouble(txtDiameter.Text.Trim());

fj.Temperature = Convert.ToDouble(txtTeperature.Text.Trim());

fj.CombustionTime = Convert.ToDouble(txtCombustionTime.Text.Trim());

fj.CombustionHeat = Convert.ToDouble(txtCombustionHeat.Text.Trim());

fj.CalculateJetFire();

spnDeathR.Value = Convert.ToDecimal(fj.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(fj.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(fj.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(fj.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.FireJet");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"MolecularWeight",10);

Dispatch.put(msg,"InitialPressure",1000);

Dispatch.put(msg,"HotmeltAtConstantPressure",100);

Dispatch.put(msg,"HotmeltAtConstantVolume",100);

Dispatch.put(msg,"Temperature",100);

Dispatch.put(msg,"Diameter",100);

Dispatch.put(msg,"CombustionHeat",1000);

Dispatch.put(msg,"CombustionTime",100);

Dispatch.call(msg, "CalculateJetFire");

System.out.println("喷射火灾模型");

System.out.println("分子量： "+Dispatch.get(msg, "MolecularWeight")+" ");

System.out.println("初始压力： "+Dispatch.get(msg, "InitialPressure")+" pa");

System.out.println("定压比热容： "+Dispatch.get(msg, "HotmeltAtConstantPressure")+" ");

System.out.println("定容比热容： "+Dispatch.get(msg, "HotmeltAtConstantVolume")+" ");

System.out.println("容器的工作温度： "+Dispatch.get(msg, "Temperature")+" 摄氏度");

System.out.println("泄漏口直径： "+Dispatch.get(msg, "Diameter")+" m");

System.out.println("燃烧热： "+Dispatch.get(msg, "CombustionHeat")+" kJ/kg");

System.out.println("燃烧时间： "+Dispatch.get(msg, "CombustionTime")+" s");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>喷射火灾模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.FireJet");

ver.MolecularWeight = 10;

ver.InitialPressure = 1000;

ver.HotmeltAtConstantPressure = 100;

ver.HotmeltAtConstantVolume = 100;

ver.Temperature = 100;

ver.Diameter =100;

ver.CombustionHeat = 1000;

ver.CombustionTime = 100;

ver.CalculateJetFire();

var outPut = "<喷射火灾模型>" + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage;

alert(outPut);

</script>

</body>

</html>

## 4、沸腾液体扩展蒸气爆炸

4.1 模型简介

沸腾液体扩展蒸气爆炸（BLEVE）的经典模型有：ILO模型、H.T.Greeberg&J.J.Cramer模型和A.F.Roberts模型等。BLEVE主要危害是火球产生的强烈热辐射伤害，因而采用瞬态火灾作用下的热剂量准则确定人员的伤亡和财产损失的区域。

4.2 算法说明

计算火球半径：



（20）

计算火球持续时间



（21）

计算目标接受到的热辐射通量：



（22）

式中：——火球半径，m

——火球中消耗的可燃物质量，kg

——火球持续时间，s

——火球表面的热辐射通量，对柱形罐可取270kW/m^2,；对球形罐可取200kW/m^2

——目标到火球中心的水平距离，m

4.3 接口

调用类：BLEVEClass

输入参数：CombusitibleMaterialWeight 火球中消耗的可燃物质量，kg

RadiantFlux 火球表面的热辐射通量，对柱形罐可取270kW/m^2；对球形罐可取200kW/m^2

输出参数：CombustionTime 火球持续时间，s

FireballRadius 火球半径，m

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：CombusitibleMaterialWeight=55650.6 kg，RadiantFlux=270000 W/m^2

4.4 代码示例

1、.NET示例

IBLEVE em = new BLEVEClass();

em.CombustibleMaterialWeight = Convert.ToDouble(txtMaterialWeight.Text.Trim());

em.RadiantFlux = Convert.ToDouble(cmbRadiantFlux.Text.Trim());

em.CalculateBLEVE();

spnFireballRadius.Value = Convert.ToDecimal(em.FireballRadius.ToString(".00"));

spnCombustionTime.Value = Convert.ToDecimal(em.CombustionTime.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.BLEVE");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"CombustibleMaterialWeight",55650);

Dispatch.put(msg,"RadiantFlux",270000);

Dispatch.call(msg, "CalculateBLEVE");

System.out.println("火球爆炸模型");

System.out.println("物质质量： "+Dispatch.get(msg, "CombustibleMaterialWeight")+" kg");

System.out.println("火球表面辐射通量： "+Dispatch.get(msg, "RadiantFlux")+" kw/m^2");

System.out.println("火球半径： "+Dispatch.get(msg, "FireballRadius")+" m");

System.out.println("燃烧时间： "+Dispatch.get(msg, "CombustionTime")+" s");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>火球爆炸模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.BLEVE");

ver.CombustibleMaterialWeight = 55650;

ver.RadiantFlux = 270000;

ver.CalculateBLEVE();

var outPut = "<火球爆炸模型>" + "\n \n";

outPut += "火球半径=" + ver.FireballRadius + "\n";

outPut += "燃烧时间=" + ver.CombustionTime + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

# 爆 炸 模 型

## 1、蒸汽云爆炸模型

1.1 模型简介

本方法采用TNT当量法估计蒸汽云爆炸的严重度。

如果某次事故造成的破坏状况与 kgTNT爆炸造成的破坏状况相当，则称此次爆炸的威力为 kgTNT当量。

用TNT当量法来预测蒸汽云爆炸严重度得原理是：假定一定百分比的蒸汽云参与了爆炸，对形成冲击波有实际的贡献，并以TNT当量来表示蒸汽云爆炸的威力。

1.2 算法说明

计算爆炸总能量：



（23）

蒸汽云爆炸当量：

（24）



计算死亡半径：



（25）

计算重（轻）伤半径：



（26）



计算财产损失半径：

（27）



式中：——爆炸总能量，KJ；

——可燃气体蒸汽云的当量系数，取0.04；

——地面爆炸系数；

——气体泄漏量kg；

——气体燃烧热，KJ/kg；

——蒸汽云TNT当量，kg；

——TNT爆炸热，取4520KJ/kg；

——大气压强，pa

1.3 接口

调用类：ExplosionSCClass

输入参数：CombustionHeat 气体的燃烧放热，KJ/kg

MaterialWeight 可燃气体质量，kg

输出参数：BlastingTotalEnergy 爆破总能量，KJ

TNTEquivalent TNT当量，kg

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：CombustionHeat=4520 KJ/kg，MaterialWeight=50 kg

1.4 代码示例

1、.NET示例

IExplosionSC em = new ExplosionSCClass();

em.MaterialWeight = Convert.ToDouble(txtMaterialWeight.Text.Trim());

em.CombustionHeat = Convert.ToDouble(txtCombustionHeat.Text.Trim());

em.CalculateStramCloudExplosion();

spnTotalEnergy.Value = Convert.ToDecimal(em.ExplosionTotalEnergy.ToString(".00"));

spnTNT.Value = Convert.ToDecimal(em.TNTEquivalent.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.ExplosionSC");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"MaterialWeight",10000);

Dispatch.put(msg,"CombustionHeat",100);

Dispatch.call(msg, "CalculateStramCloudExplosion");

System.out.println("蒸气云爆炸模型");

System.out.println("物质质量： "+Dispatch.get(msg, "MaterialWeight")+" kg");

System.out.println("燃烧热： "+Dispatch.get(msg, "CombustionHeat")+" kj/kg");

System.out.println("爆炸总能量： "+Dispatch.get(msg, "ExplosionTotalEnergy")+" kj");

System.out.println("爆炸TNT当量： "+Dispatch.get(msg, "TNTEquivalent"));

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>蒸气云爆炸模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.ExplosionSC");

ver.MaterialWeight = 10000;

ver.CombustionHeat = 100;

ver.CalculateStramCloudExplosion();

var outPut = "<蒸气云爆炸模型>" + "\n \n";

outPut += "爆炸总能量=" + ver.ExplosionTotalEnergy + "\n ";

outPut += "TNT当量=" + ver.TNTEquivalent + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

## 2、凝聚相含能材料爆炸模型

2.1 模型简介

凝聚相含能材料爆炸能产生多种破坏效应，如热辐射、一次破片作用、有毒气体产物的致命效应，但破坏力最强，破坏区域最大的是冲击波的破坏效应，因此，凝聚相爆炸模型主要考虑冲击波得伤害作用。

2.2 算法说明

计算含能材料爆炸释放的能量：



（28）

式中：——爆炸能量，KJ；

——爆炸物质质量，kg；

——物质的爆热，KJ/kg

2.3 接口

调用类：ExplosionEMClass

输入参数：CombustionHeat 气体的燃烧放热，KJ/kg

MaterialWeight 可燃气体质量，kg

输出参数：BlastingTotalEnergy 爆破总能量，KJ

TNTEquivalent TNT当量，kg

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：CombustionHeat=4520 KJ/kg，MaterialWeight=50 kg

2.4 代码示例

1、.NET示例

IExplosionEM em = new ExplosionEMClass();

em.MaterialWeight = Convert.ToDouble(txtMaterialWeight.Text.Trim());

em.CombustionHeat = Convert.ToDouble(txtCombustionHeat.Text.Trim());

em.CalculateEnergeticMaterialExplosion();

spnTotalEnergy.Value = Convert.ToDecimal(em.ExplosionTotalEnergy.ToString(".00"));

spnTNT.Value = Convert.ToDecimal(em.TNTEquivalent.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.ExplosionEM");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"MaterialWeight",10000);

Dispatch.put(msg,"CombustionHeat",100);

Dispatch.call(msg, "CalculateEnergeticMaterialExplosion");

System.out.println("凝聚态含能物质爆炸模型");

System.out.println("物质质量： "+Dispatch.get(msg, "MaterialWeight")+" kg");

System.out.println("燃烧热： "+Dispatch.get(msg, "CombustionHeat")+" kj/kg");

System.out.println("爆炸总能量： "+Dispatch.get(msg, "ExplosionTotalEnergy")+" kj");

System.out.println("爆炸TNT当量： "+Dispatch.get(msg, "TNTEquivalent"));

System.out.println("");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>凝聚态含能物质爆炸模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.ExplosionEM");

ver.MaterialWeight = 10000;

ver.CombustionHeat = 100;

ver.CalculateEnergeticMaterialExplosion();

var outPut = "<凝聚态含能物质爆炸模型>" + "\n \n";

outPut += "爆炸总能量=" + ver.ExplosionTotalEnergy + "\n ";

outPut += "TNT当量=" + ver.TNTEquivalent + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

# 压 力 容 器 爆 破 模 型

## 1、盛装气体压力容器爆破模型

1.1 模型简介

盛装气体的压力容器在爆破时，气体膨胀所释放的能量（即爆破能量），与压力容器的容积有关。其爆破过程是容器内的气体由容器破裂前的压力降至大气压力的一个简单膨胀过程，所以历时一般都很短，不管容器内介质的温度与周围大气存在多大的温差，都可以认为容器内的气体与大气无热量交换，即此时气体介质的膨胀是一个绝热膨胀过程，因此其爆破能量亦即为气体介质膨胀所做的功，可按理想气体绝热膨胀做功计算气体的爆破能量。

1.2 模型简介

计算气体的爆破能量：

（29）



式中：——气体的爆破能量，KJ；

——气体的绝对压力，Mpa；

——气体容器的体积，m^3

——气体绝热指数，双原子分子取1.4，三原子分子和四原子分子取1.2-1.3

1.3 接口

调用类：BlastingGasContainerClass

输入参数：AbsolutePressure 绝对压力Mpa

AdiabaticIndex 气体绝热指数

ContainerVolume 容器体积m^3

输出参数：BlastingTotalEnergy 爆破总能量

TNTEquivalent TNT当量

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：AbsolutePressure=30 Mpa，AdiabaticIndex=1.4，ContainerVolume=30 m^3

1.4 代码示例

1、.NET示例

IBlastingGasContainer em = new BlastingGasContainerClass();

em.AbsolutePressure = Convert.ToDouble(txtAbsolutePressure.Text.Trim());

em.ContainerVolume = Convert.ToDouble(txtContainerVolume.Text.Trim());

em.AdiabaticIndex = Convert.ToDouble(txtAdiabaticIndex.Text.Trim());

em.CalculateGasContainerBlasting();

spnTotalEnergy.Value = Convert.ToDecimal(em.BlastingTotalEnergy.ToString(".00"));

spnTNT.Value = Convert.ToDecimal(em.TNTEquivalent.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.BlastingGasContainer");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"AbsolutePressure",1000);

Dispatch.put(msg,"ContainerVolume",1000);

Dispatch.put(msg,"AdiabaticIndex",100);

Dispatch.call(msg, "CalculateGasContainerBlasting");

System.out.println("气体压力容器爆破模型");

System.out.println("绝对压力： "+Dispatch.get(msg, "AbsolutePressure")+" MPa");

System.out.println("容器体积： "+Dispatch.get(msg, "ContainerVolume")+" m^3");

System.out.println("气体绝热指数： "+Dispatch.get(msg, "AdiabaticIndex")+" Mpa^(-1)");

System.out.println("爆炸总能量： "+Dispatch.get(msg, "BlastingTotalEnergy")+" kj");

System.out.println("爆炸TNT当量： "+Dispatch.get(msg, "TNTEquivalent"));

System.out.println("");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>气体压力容器爆破模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.BlastingGasContainer");

ver.AbsolutePressure = 1000;

ver.ContainerVolume = 1000;

ver.AdiabaticIndex = 100;

ver.CalculateGasContainerBlasting();

var outPut = "<气体压力容器爆破模型>" + "\n \n";

outPut += "爆炸总能量=" + ver.BlastingTotalEnergy + "\n";

outPut += "TNT当量=" + ver.TNTEquivalent + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

## 2、盛装液体压力容器爆破模型

2.1 模型简介

当压缩液体盛装在容器内超压或容器受损发生爆破时，所释放出的能量为压缩液体压力、体积变化的函数。

2.2 算法说明

计算液体压力容器爆炸释放的能量：



（30）

式中：——爆炸能量，KJ；

——液体的绝对压力，Mpa；

——大气压力，0.101300Mpa

——容器的体积，m^3；

——液体压缩系数，Mpa^(-1) 在常温和MPa以内的水，取0.000452MPa(-1)，在常温和MPa以内的水，取.00044MPa(-1)

2.3 接口

调用类：BlastingLiquidContainerClass

输入参数：AbsolutePressure 容器的内部压力Mpa

CompressionFactor 液体的压缩系数Mpa^(-1)

ContainerVolume 液体的体积m^3

输出参数：BlastingTotalEnergy 爆破总能量J

TNTEquivalent TNT当量kg

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：AbsolutePressure=20 Mpa，CompressionFactor=0.00044，ContainerVolume=20 m^3

2.4 代码示例

1、.NET示例

IBlastingLiquidContainer em = new BlastingLiquidContainerClass();

em.AbsolutePressure = Convert.ToDouble(txtAbsolutePressure.Text.Trim());

em.ContainerVolume = Convert.ToDouble(txtContainerVolume.Text.Trim());

em.CompressionFactor = Convert.ToDouble(txtCompressionFactor.Text.Trim());

em.CalculateLiquidContainerBlasting();

spnTotalEnergy.Value = Convert.ToDecimal(em.BlastingTotalEnergy.ToString(".00"));

spnTNT.Value = Convert.ToDecimal(em.TNTEquivalent.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.BlastingLiquidContainer");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"AbsolutePressure",150);

Dispatch.put(msg,"ContainerVolume",30);

Dispatch.put(msg,"CompressionFactor",0.44);

Dispatch.call(msg, "CalculateLiquidContainerBlasting");

System.out.println("液体压力容器爆破模型");

System.out.println("绝对压力： "+Dispatch.get(msg, "AbsolutePressure")+" Mpa");

System.out.println("容器体积： "+Dispatch.get(msg, "ContainerVolume")+" m^3");

System.out.println("压缩系数： "+Dispatch.get(msg, "CompressionFactor"));

System.out.println("爆炸总能量： "+Dispatch.get(msg, "BlastingTotalEnergy"));

System.out.println("爆炸TNT当量： "+Dispatch.get(msg, "TNTEquivalent"));

System.out.println("");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>液体压力容器爆破模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.BlastingLiquidContainer");

ver.AbsolutePressure = 150;

ver.ContainerVolume = 30;

ver.CompressionFactor = 0.44;

ver.CalculateLiquidContainerBlasting();

var outPut = "<液体压力容器爆破模型>" + "\n \n";

outPut += "爆炸总能量=" + ver.BlastingTotalEnergy + "\n";

outPut += "TNT当量=" + ver.TNTEquivalent + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

## 3、盛装液化气体容器爆破模型

3.1 模型简介

液氯、液氨储罐及锅炉汽包等压力容器内以气、液两态存在，工作介质的压力大于大气压，介质温度大于其在大气压力下的沸点。当容器爆裂时，气体迅速膨胀，液体迅速沸腾，剧烈蒸发，产生暴沸或水蒸气爆炸。

容器爆破所释放出的能量为气体的能量和饱和液体的能量，由于前者量很小，往往可忽略不计，因为暴沸或水蒸气爆炸在瞬间完成，所以是一个绝热过程。

3.2 算法说明

计算爆炸能量：



（31）

式中：——过热状态下液体的爆炸能量，kJ

——爆炸前液化气体的焓，kJ/kg

——在大气压力下饱和液体的焓，kJ/kg

——爆炸前饱和液体的熵，kJ/(kg·K)

——在大气压力下饱和液体的熵，kJ/(kg·K)

——介质在大气压力下的沸点，K

——饱和液体的质量，kg

3.3 接口

调用类：BlastingLiqGasContainerClass

输入参数：BoilingPoint 介质在大气压力下的沸点，K

LiquefiedGasEthalpy 爆炸前液化气体的焓，kJ/kg

LiquefiedEhtropy 爆炸前饱和液体的熵，kJ/(kg.K)

SaturatedLiquidEnthalpy 在大气压力下饱和液体的焓，kJ/kg

SaturatedLiquidEntropy 在大气压力下饱和液体的熵，kJ/(kg.K)

SaturatedLiquidWeight 饱和液体的质量，kg

输出参数：BlastingTotalEnergy 爆破总能量

TNTEquivalent TNT当量

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：BoilingPoint=90.05 K，LiquefiedGasEthalpy=-79.84 kJ/kg，LiquefiedEhtropy=3.44 kJ/(kg.K)，SaturatedLiquidEnthalpy=-133.69 kJ/kg，SaturatedLiquidEntropy=2.94 kJ/(kg.K)，SaturatedLiquidWeight=3892.48 kg

3.4 代码示例

1、.NET示例

IBlastingLiqGasContainer em = new BlastingLiqGasContainerClass();

em.LiquefiedGasEnthalpy = Convert.ToDouble(txtLiquefiedGasEnthalpy.Text.Trim());

em.SaturatedLiquidEnthalpy = Convert.ToDouble(txtSaturatedLiquidEnthalpy.Text.Trim());

em.LiquefiedGasEntropy = Convert.ToDouble(txtLiquefiedGasEntropy.Text.Trim());

em.SaturatedLiquidEntropy = Convert.ToDouble(txtSaturatedLiquidEntropy.Text.Trim());

em.SaturatedLiquidWeight = Convert.ToDouble(txtSaturatedLiquidWeight.Text.Trim());

em.BoilingPoint = Convert.ToDouble(txtBoilingPoint.Text.Trim());

em.CalculateLiquefiedGasContainerBlasting();

spnTotalEnergy.Value = Convert.ToDecimal(em.BlastingTotalEnergy.ToString(".00"));

spnTNT.Value = Convert.ToDecimal(em.TNTEquivalent.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.BlastingLiqGasContainer");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"LiquefiedGasEnthalpy",-79.84);

Dispatch.put(msg,"SaturatedLiquidEnthalpy",-133.69);

Dispatch.put(msg,"LiquefiedGasEntropy",3.44);

Dispatch.put(msg,"SaturatedLiquidEntropy",2.94);

Dispatch.put(msg,"SaturatedLiquidWeight",3892.48);

Dispatch.put(msg,"BoilingPoint",90.05);

Dispatch.call(msg, "CalculateLiquefiedGasContainerBlasting");

System.out.println("液化气体容器爆破模型");

//System.out.println("容器体积： "+Dispatch.get(msg, "Volume")+" m^3");

//System.out.println("爆炸能量系数： "+Dispatch.get(msg, "Coefficient")+" Mpa^(-1)");

System.out.println("爆炸总能量： "+Dispatch.get(msg, "BlastingTotalEnergy")+" kj");

System.out.println("爆炸TNT当量： "+Dispatch.get(msg, "TNTEquivalent"));

System.out.println("");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>液化气体容器爆破模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.BlastingLiqGasContainer");

ver.LiquefiedGasEnthalpy = -79.84;

ver.SaturatedLiquidEnthalpy = -133.69;

ver.LiquefiedGasEntropy = 3.44;

ver.SaturatedLiquidEntropy = 2.94;

ver.SaturatedLiquidWeight = 3892.48;

ver.BoilingPoint = 90.05;

ver.CalculateLiquefiedGasContainerBlasting();

var outPut = "<液化气体容器爆破模型>" + "\n \n";

outPut += "爆炸总能量=" + ver.BlastingTotalEnergy + "\n";

outPut += "TNT当量=" + ver.TNTEquivalent + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

## 4、盛装饱和水容器爆破模型

4.1 算法说明

计算饱和水容器的爆炸能量：

（32）



式中：——饱和水容器爆炸释放的能量，kJ

——饱和水爆破能量系数，kJ/m^3

——容器内饱和水所占容积，m^3

4.2 接口

调用类：BlastingSatWaterContainerClass

输入参数：Coefficient 饱和水爆破能量系数，kJ/m^3

Volume 容器内饱和水所占容积m^3

输出参数：BlastingTotalEnergy 爆破总能量，KJ

TNTEquivalent TNT当量，kg

Radius\_Death 死亡半径

Radius\_SeriouslyInjured 重伤半径

Radius\_SlightlyInjured 轻伤半径

Radius\_PropertyDamage 财产损失半径

例：Coefficient=32500 kJ/m^3，Volume=30 m^3

4.3 代码示例

1、.NET示例

IBlastingSatWaterContainer em = new BlastingSatWaterContainerClass();

em.Volume = Convert.ToDouble(txtVolume.Text.Trim());

em.Coefficient = Convert.ToDouble(txtCoefficient.Text.Trim());

em.CalculateSaturatedWaterContainerBlasting();

spnTotalEnergy.Value = Convert.ToDecimal(em.BlastingTotalEnergy.ToString(".00"));

spnTNT.Value = Convert.ToDecimal(em.TNTEquivalent.ToString(".00"));

spnDeathR.Value = Convert.ToDecimal(em.Radius\_Death.ToString(".00"));

spnSerieslyInjuredR.Value = Convert.ToDecimal(em.Radius\_SeriouslyInjured.ToString(".0000"));

spnSlightlyInjuredR.Value = Convert.ToDecimal(em.Radius\_SlightlyInjured.ToString(".0000"));

spnPropertyDamageR.Value = Convert.ToDecimal(em.Radius\_PropertyDamage.ToString(".0000"));

}

}

catch (System.Exception ex)

{

XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.BlastingSatWaterContainer");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"Volume",30);

Dispatch.put(msg,"Coefficient",32500);

Dispatch.call(msg, "CalculateSaturatedWaterContainerBlasting");

System.out.println("饱和水容器爆破模型");

System.out.println("容器体积： "+Dispatch.get(msg, "Volume")+" m^3");

System.out.println("爆炸能量系数： "+Dispatch.get(msg, "Coefficient")+" Mpa^(-1)");

System.out.println("爆炸总能量： "+Dispatch.get(msg, "BlastingTotalEnergy")+" kj");

System.out.println("爆炸TNT当量： "+Dispatch.get(msg, "TNTEquivalent"));

System.out.println("");

System.out.println("死亡半径： "+Dispatch.get(msg, "Radius\_Death")+" m");

System.out.println("重伤半径： "+Dispatch.get(msg, "Radius\_SeriouslyInjured")+" m");

System.out.println("轻伤半径： "+Dispatch.get(msg, "Radius\_SlightlyInjured")+" m");

System.out.println("财产损失半径： "+Dispatch.get(msg, "Radius\_PropertyDamage")+" m");

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>饱和水容器爆破模型</title>

</head>

<body>

<script language="javascript">

var ver = new ActiveXObject("AccidentModel.BlastingSatWaterContainer");

ver.Volume = 30;

ver.Coefficient = 32500;

ver.CalculateSaturatedWaterContainerBlasting();

var outPut = "<饱和水容器爆破模型>" + "\n \n";

outPut += "爆炸总能量=" + ver.BlastingTotalEnergy + "\n";

outPut += "TNT当量=" + ver.TNTEquivalent + "\n \n";

outPut += "死亡半径=" + ver.Radius\_Death + "\n";

outPut += "重伤半径=" + ver.Radius\_SeriouslyInjured + "\n";

outPut += "轻伤半径=" + ver.Radius\_SlightlyInjured + "\n";

outPut += "财产损失半径=" + ver.Radius\_PropertyDamage + "\n";

alert(outPut);

</script>

</body>

</html>

# 毒 气 扩 散 模 型

根据气云密度与空气密度的相对大小，将气云分成重气云、中性气云和轻气云三类。如果气云密度显著大于空气密度，气云将受到方向向下的重力作用，这样的气云称为重气云。如果气云密度明显小于空气密度，气云将受到方向向上的浮力作用，这样的气云称为轻气云。如果气云密度与空气密度相当，气云将不受明显的浮力作用，这样的气云称为中性气云。轻气云和中性气云统称为非重气云非重气云的空中扩散可用众所周知的高斯模型描述，重气云得空中扩散过程应该用20世纪70年代以后陆续提出的重气扩散模型描述。

## 1、非重气云扩散模型

1.1 模型简介

高斯模型用来描述危险物质泄漏形成的非重气云扩散行为，或描述重气云在重力作用消失后的远场扩散行为。为了方便分析，建立OXYZ：其中原点O是泄漏点在地面上的正投影，X轴沿下风向水平延伸，Y轴在水平面上垂直于X轴，Z轴垂直向上延伸。

高斯模型使用了如下假设：

1. 气云密度与环境空气密度相当，气云不受浮力作用。
2. 云团中心的移动速度或云羽轴向蔓延速度等于环境风速。
3. 云团内部或云羽横截面上浓度、密度等参数服从高斯分布（即正态分布）。

1.2 算法说明

根据高斯模型，连续泄漏源下风向某点（x,y,z）在t时刻的浓度为：



（33）

瞬时泄漏源下风向某点（x,y,z）在t时刻的浓度为：



（34）

式中：——给定气体浓度，kg/m^3；

——泄漏速度（连续）kg/s，泄漏量（瞬时）kg；

——风速，m/s；

——扩散时间，s；

——泄漏源距地面高度m

1.3 接口

调用类：DiffusionNonHeavyGasCloudClass

输入参数：AtmosphericStability 大气稳定度

GasConcentration 给定的气体浓度，kg/m^3

Hight 气体离地面高度，m

WindSpeed 风速，m/s

WindDirection 风的方向

LeakRate 气体的泄漏速度，kg/s（连续泄漏）

LeakTime 气体泄漏时间，s

LeakAmount 气体的泄漏量，kg

isContinuous 1为连续泄漏扩散，0为瞬时泄漏扩散

输出参数：OutDrawPoints 绘图的点

OutPointsNumber 点的个数

LeakPoint\_X x轴的点

LeakPoint\_Y y轴的点

例：AtmosphericStability=3，GasConcentration=0.000023 kg/m^3，Hight=23 m，WindSpeed=23 m/s，LeakRate=1000 kg/s，LeakTime=10 s，LeakAmount=1000 kg，isContinuous=1

1.4 代码示例

1、.NET示例

AccidentModelLib.IDiffusionNonHeavyGasCloud dnh = new AccidentModelLib.DiffusionNonHeavyGasCloudClass();

dnh.AtmosphericStability = ResolveAtmosphericStability(cmbAtmosphericStability.Text.Trim());

dnh.LeakAmount = Convert.ToDouble(txtLeakAmount.Text.Trim());

dnh.LeakRate = Convert.ToDouble(txtLeakRate.Text.Trim());

dnh.WindSpeed = Convert.ToDouble(txtWindSpeed.Text.Trim());

dnh.LeakTime = Convert.ToDouble(txtLeakTime.Text.Trim());

dnh.Hight = Convert.ToDouble(txtLeakHeight.Text.Trim());

dnh.LeakPoint\_X = Convert.ToDouble(txtLeakPoint\_X.Text.Trim());

dnh.LeakPoint\_Y = Convert.ToDouble(txtLeakPoint\_Y.Text.Trim());

dnh.WindDirection = Convert.ToDouble(txtWindDirection.Text.Trim());

dnh.GasConcentration = Convert.ToDouble(txtGasConcentration.Text.Trim());

object objects;

if (cmbLeakType.SelectedIndex == 0)

{

dnh.CalculateGasDiffusion(true);

objects = dnh.OutDrawPoints;

}

else

{

dnh.CalculateGasDiffusion(false);

objects = dnh.OutDrawPoints;

}

int number = dnh.OutPointsNumber;

txtPointsNumber.Text = "绘图点数：" + number + "个";

DrawChart(objects, number);

}

}

catch (System.Exception ex)

{

//XtraMessageBox.Show("参数输入有误，请重新输入参数进行计算", "错误", MessageBoxButtons.OK, MessageBoxIcon.Information);

XtraMessageBox.Show(ex.Message);

}

}

/// <summary>

/// 绘制扩散图形

/// </summary>

/// <param name="objects">输入点数组</param>

/// <param name="number">输入点个数</param>

private void DrawChart(object objects, int number)

{

double[,] a = new double[number, 2];

a = (double[,])objects;

//double[,] b = new double[number, 2];

//b = (double[,])objects;

//double[,] a = new double[2 \* number, 2];

//for (int p = 0; p < number; p++)

//{

// a[p, 0] = b[p, 0];

// a[p, 1] = b[p, 1];

//}

//for (int q = 0; q < number; q++)

//{

// a[number + q, 0] = b[number - q - 1, 0];

// a[number + q, 1] = -b[number - q - 1, 1];

//}

Series serie = new Series("aa", ViewType.ScatterLine);

serie.Visible = true;

serie.SeriesPointsSortingKey = SeriesPointKey.Argument;

serie.ShowInLegend = true;

serie.ArgumentScaleType = ScaleType.Qualitative;//.DateTime;

serie.ValueScaleType = ScaleType.Numerical;

serie.View.Color = GetColor();

////serie.LegendPointOptions.ArgumentDateTimeOptions.Format = DateTimeFormat.Custom;

////serie.LegendPointOptions.ArgumentDateTimeOptions.FormatString = "hh";

serie.LegendPointOptions.PointView = PointView.Undefined;

//serie.LegendPointOptions.ValueNumericOptions.Format = NumericFormat.FixedPoint;

//serie.LegendPointOptions.ValueNumericOptions.Precision = 2;

serie.Label.Visible = false;

LineSeriesView view = serie.View as LineSeriesView;

view.LineMarkerOptions.Visible = false;

//serie.PointOptions.ArgumentDateTimeOptions.Format = DateTimeFormat.Custom;

// serie.PointOptions.ArgumentDateTimeOptions.FormatString = "hh";

serie.PointOptions.PointView = PointView.Undefined;

serie.PointOptions.ValueNumericOptions.Precision = 0;

//serie.PointOptions.ValueNumericOptions.Format = NumericFormat.FixedPoint;

serie.TopNOptions.Enabled = false;

for (int k = 0; k <number; k++)

{

//DateTime dt = Convert.ToDateTime(dt.Rows[0]["measure\_time"]);

double xLable = a[k, 0];

double[] value = new double[] { a[k, 1] };

SeriesPoint pt = new SeriesPoint(xLable, value);

serie.Points.Add(pt);

}

//chartControl1.Series.Clear();

chartControl1.Series.Add(serie);

}

/// <summary>

/// 解析大气稳定度输入

/// </summary>

/// <param name="p">大气稳定度级别</param>

/// <returns>数值表示方式</returns>

private short ResolveAtmosphericStability(string atmosphericStability)

{

short a=-1;

switch (atmosphericStability)

{

case "A":

a = 0;

break;

case "B":

a = 1;

break;

case "C":

a = 2;

break;

case "D":

a = 3;

break;

case "E":

a = 4;

break;

case "F":

a = 5;

break;

}

return a;

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

private void cmbLeakType\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cmbLeakType.SelectedIndex==0)

{

txtLeakAmount.Enabled = false;

txtLeakAmount.Value = 1;

txtLeakAmount.TabStop = false;

txtLeakTime.Enabled = false;

txtLeakTime.Value = 1;

txtLeakTime.TabStop = false;

txtLeakRate.Enabled = true;

txtLeakRate.Value = 0;

txtLeakRate.TabStop = true;

}

if (cmbLeakType.SelectedIndex == 1)

{

txtLeakAmount.Enabled = true;

txtLeakAmount.Value = 0;

txtLeakAmount.TabStop = true;

txtLeakTime.Enabled = true;

txtLeakTime.Value = 0;

txtLeakTime.TabStop = true;

txtLeakRate.Enabled = false;

txtLeakRate.Value = 1;

txtLeakRate.TabStop = false;

}

}

private void btnClearChart\_Click(object sender, EventArgs e)

{

chartControl1.Series.Clear();

}

static Color GetColor()

{

//Random random = new Random();

//int number = random.Next(1, 4);

//switch (number)

//{

// case 1:

// return Color.Red;

// case 2:

// return Color.Black;

// default:

// return Color.Yellow;

//}

Random r = new Random();

Color c = Color.FromKnownColor((KnownColor)r.Next(1, 167));

return c;

}

}

}

2、JAVA示例

ActiveXComponent smsg=new ActiveXComponent("AccidentModel.DiffusionNonHeavyGasCloud");

Dispatch msg = smsg.getObject();

Dispatch.put(msg,"AtmosphericStability",2);

Dispatch.put(msg,"LeakAmount",451350);

Dispatch.put(msg,"LeakRate",10);

Dispatch.put(msg,"WindSpeed",1);

Dispatch.put(msg,"LeakTime",30);

Dispatch.put(msg,"Hight",5);

Dispatch.put(msg,"GasConcentration",0.0035);

Dispatch.put(msg,"LeakPoint\_X",1000);

Dispatch.put(msg,"LeakPoint\_Y",1000);

Dispatch.put(msg,"WindDirection",45);

Dispatch.put(msg,"isContinuous",true);

Dispatch.call(msg, "CalculateGasDiffusion",false);

Variant var=Dispatch.get(msg, "OutDrawPoints");

SafeArray sa=var.toSafeArray();

int i=sa.getUBound();

int j=sa.getNumDim();

double[][] outPoints=new double[i+1][j];

for(int m=0;m<=i;m++)

{

for(int n=0;n<j;n++)

{

outPoints[m][n]=sa.getDouble(m, n);

System.out.println(outPoints[m][n]);

}

}

System.out.println(Dispatch.get(msg, "OutPointsNumber"));

3、JavaScript示例

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<head>

<title>非重气云扩散模型</title>

</head>

<body>

<script language="JavaScript">

var ver = new ActiveXObject("AccidentModel.DiffusionNonHeavyGasCloud");

ver.AtmosphericStability = 2;

ver.LeakAmount = 451350;

ver.LeakRate = 10;

ver.WindSpeed = 1;

ver.LeakTime = 30;

ver.Hight = 5;

ver.GasConcentration = 0.0035

ver.LeakPoint\_X = 1000;

ver.LeakPoint\_Y = 1000;

ver.WindDirection = 45;

ver.isContinuous = false;

var array =ver.CalculateGasDiffusionN();

var outPut = "<非重气云扩散模型>" + "\n \n";

outPut += "输出绘图点数=" + ver.OutPointsNumber + "\n";

alert(outPut);

//var vbArray = new VBArray(array);

//Array = ver.CalculateGasDiffusionN();

//alert(vbArray.dimensions());

// alert(vbArray.dimensions());

// alert(Array.getItem(0, 0));

//var vbArr = new VBArray(Array);

// var vbArray = (VBArray)ver.OutDrawPoints;

////ver.CalculateGasDiffusionN(vbArray)

////var SafeArray = ver.OutDrawPoints;

//var jsArray = vbArr.toArray();

// alert(jsArray[0]);

// alert(ver.OutPointsNumber);

</script>

</body>

</html>

## 2、重气云扩散模型

2.1 模型简介

重气云连续泄漏扩散模型：

为了进行危险气体近地面连续泄漏形成的重气云羽的扩散分析，研究人员提出了“SLAB”模型。在“SLAB”模型中使用了如下假设：

1. 重气云羽横截面为矩形，横风向半宽b（m），垂直方向高度h（m）。在泄漏源点，云羽半宽为高度的两倍。
2. 重气云羽横截面内，浓度、温度、密度等参数均匀分布。
3. 重气云羽的轴向蔓延速度等于风速。

2.1 算法说明

计算重气云羽的横向半宽：



（35）

计算重气云羽的高度：



（36）

计算重气云羽的浓度：



（37）

式中：——烟羽的浓度，kg/m^3

——重气云羽的高度，m

——重气云羽的横向半宽，m

——常数，这里取1.0

——泄漏源点重气云羽的横向半宽，m

——初始重气云羽的高度，，m

——重气云羽的轴向蔓延速度，在这里等于风速，m/s

——摩擦风速，一般取，其中为10m处得平均风速，m/s

——重气云羽的初始密度，kg/m^3

——下风向距离，m

——空气的密度，kg/m^3

重气云瞬间泄漏扩散模型：

为了分析危险气体近地面瞬间泄漏形成的重气云团在空中的扩散过程，研究人员提出了“BOX”模型。为了分析方便，在“BOX”模型中使用如下假设：

1. 重气云团为正立的坍塌圆柱体，圆柱体初始高度等于初始半径的一半。
2. 在重气云团内部，温度、密度和危险物质浓度等参数均匀分布。
3. 重气云团中心的移动速度等于风速。

计算重气云团半径：



（38）

计算重气云团的高度：



（39）

计算重气云团内密度：



（40）

式中：——重气云团的半径，m

——重气云团初始半径，m

——重气云团的初始密度，kg/m^3

——空气密度，kg/m^3

——重气云团的初始体积，m^3

——重气云团扩散时间，s

——重气云团高度，m

——下风向距离，m

——给定重气云团内密度，kg/m^3

——重气云团的初始密度，kg/m^3