

大气传输计算库

简介

大气传输计算库（AirTransmission）是一个用于计算大气传输特性的 .NET 类库。它提供了一系列工具和模型，用于计算不同天气条件下的大气透过率和大气湍流影响。

主要功能

1. 大气透过率计算
 - 支持多种波长范围（紫外、可见光、红外、毫米波）
 - 考虑多种天气条件（晴天、多云、雨天等）
 - 计算不同传输距离的影响
2. 大气湍流模型
 - 计算大气湍流强度
 - 评估湍流对传输的影响
3. 天气条件模型
 - 温度、湿度、气压等参数的综合影响
 - 能见度和天气类型的影响

适用场景

- 激光传输系统设计
- 光电设备性能评估
- 大气光学研究
- 气象影响分析

技术特点

- 基于 .NET 8.0 开发
- 支持跨平台运行
- 提供 C# 原生支持
- 通过 C++/CLI 包装器支持 C++ 调用
- 完整的 API 文档和使用示例

版本信息

- 版本：1.0.0
- 发布日期：2024年10月

大气透过率计算库 API 文档

主要类和接口

AtmosphericTransmittanceCalculator

静态类，提供各种电磁波透过率的计算方法。

方法

CalcLaser

```
public static double CalcLaser(WeatherCondition weather, double distance)
```

计算激光（1.06μm）在给定天气条件和距离下的大气透过率。

- 参数：
 - weather: 天气条件
 - distance: 传输距离（米）
- 返回值：0-1之间的透过率值

CalcLaserWithSmoke

```
public static double CalcLaserWithSmoke(WeatherCondition weather, double distance, double smokeConcentration, double smokeThickness)
```

计算有烟雾条件下的激光透过率。

- 参数：
 - weather: 天气条件
 - distance: 传输距离（米）
 - smokeConcentration: 烟雾浓度（g/m³）
 - smokeThickness: 烟雾厚度（米）
- 返回值：0-1之间的透过率值

CalcTurbulenceEffect

```
public static double CalcTurbulenceEffect(WeatherCondition weather, double distance)
```

计算湍流效应对激光透过率的影响。

- 参数：
 - weather: 天气条件
 - distance: 传输距离（米）
- 返回值：0-1之间的透过率值

CalcIR

```
public static double CalcIR(WeatherCondition weather, double distance)
```

计算红外线（3-12μm）透过率。

- 参数：
 - weather: 天气条件
 - distance: 传输距离（米）
- 返回值：0-1之间的透过率值

CalcUV

```
public static double CalcUV(WeatherCondition weather, double distance)
```

计算紫外线 (0.2-0.4 μm) 透过率。

- 参数:
 - weather: 天气条件
 - distance: 传输距离 (米)
- 返回值: 0-1之间的透过率值

CalcMillimeterWave

```
public static double CalcMillimeterWave(WeatherCondition weather, double distance)
```

计算毫米波 (94GHz) 透过率。

- 参数:
 - weather: 天气条件
 - distance: 传输距离 (米)
- 返回值: 0-1之间的透过率值

CalculateSmokeScreenTransmittance

```
public static double CalculateSmokeScreenTransmittance(double smokeConcentration, double smokeThickness)
```

计算烟幕对电磁波的透过率。

- 参数:
 - smokeConcentration: 烟雾浓度 (g/m^3)
 - smokeThickness: 烟雾厚度 (米)
- 返回值: 0-1之间的透过率值

CalculateReceivedRadiationSinglePath

```
public static double CalculateReceivedRadiationSinglePath(double transmittance, double targetRadiation, double receiverDistance)
```

计算单程传输后接收到的辐射功率。

- 参数:
 - transmittance: 大气透过率
 - targetRadiation: 目标辐射 (W/Sr)
 - receiverDistance: 接收器距离 (米)
- 返回值: 接收到的辐射功率 (W/Sr)

CalculateReceivedRadiation

```
public static double CalculateReceivedRadiation(double transmittance, double laserEnergy, double pulseWidth, double targetDistance, double receiverDistance, double targetReflectivity)
```

计算双程传输后接收到的辐射功率。

- 参数:
 - transmittance: 大气透过率
 - laserEnergy: 激光能量 (焦耳)
 - pulseWidth: 脉冲宽度 (纳秒)
 - targetDistance: 目标距离 (米)
 - receiverDistance: 接收器距离 (米)
 - targetReflectivity: 目标反射率
- 返回值: 接收到的辐射功率 (瓦特)

WeatherCondition

表示大气环境条件的类。

构造函数

```
public WeatherCondition(
    WeatherType type,
    double temperature,
    double relativeHumidity,
    double visibility,
    double? precipitation = null,
    double co2Concentration = 415)
```

- 参数：
 - type: 天气类型
 - temperature: 温度（摄氏度）
 - relativeHumidity: 相对湿度（0-1）
 - visibility: 能见度（米）
 - precipitation: 降水量（mm/h），可选
 - co2Concentration: 二氧化碳浓度（ppm），默认415

属性

- Type: 天气类型（WeatherType 枚举）
- Temperature: 温度（摄氏度）
- RelativeHumidity: 相对湿度（0-1）
- Visibility: 能见度（米）
- Precipitation: 降水量（mm/h）
- CO2Concentration: 二氧化碳浓度（ppm）

WeatherType

天气类型枚举。

```
public enum WeatherType
{
    晴天, // 晴朗天气
    雨天, // 雨天
    雪天, // 雪天
    雾天, // 雾天
    沙尘 // 沙尘天气
}
```

使用示例

基本使用

```
// 创建天气条件
var weather = new WeatherCondition(
    type: WeatherType.晴天,
    temperature: 20, // 20℃
    relativeHumidity: 0.45, // 45%
    visibility: 23000 // 23km
);
// 计算1000米距离的激光透过率
double transmittance = AtmosphericTransmittanceCalculator.CalcLaser(weather, 1000);
```

计算有烟雾条件下的透过率

```
// 计算有烟雾条件下的透过率
double smokeTransmittance = AtmosphericTransmittanceCalculator.CalcLaserWithSmoke(
    weather,
    distance: 1000, // 1000米
    smokeConcentration: 0.1, // 0.1 g/m³
    smokeThickness: 100 // 100米
);
```

注意事项

1. 所有透过率计算方法返回值范围都在0到1之间
2. 距离单位统一使用米 (m)
3. 温度使用摄氏度 (°C)
4. 相对湿度使用0-1的小数表示
5. 能见度使用米 (m) 表示
6. 降水量使用毫米/小时 (mm/h) 表示

性能考虑

- 所有计算方法都是线程安全的
- 计算过程中会考虑多种大气效应，包括分子散射、气溶胶散射、大气湍流等
- 对于大量计算，建议复用 `WeatherCondition` 对象以提高性能



[Show / Hide Table of Contents](#)

Class AtmosphericTransmittanceCalculator

大气透过率计算器，提供各种电磁波在大气中传输的透过率计算方法

Inheritance

↳ [object](#)
↳ AtmosphericTransmittanceCalculator

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [AirTransmission](#)

Assembly: AirTransmission.dll

Syntax

```
public static class AtmosphericTransmittanceCalculator
```

Remarks

支持以下电磁波类型的透过率计算：

- 激光（包含湍流效应）
- 红外线
- 紫外线
- 毫米波 可以处理各种天气条件和烟雾环境

Methods

CalcIR(WeatherCondition, double, double, double)

计算红外线在给定条件下的大气透过率

Declaration

```
public static double CalcIR(WeatherCondition weather, double distance, double smokeConcentration = 0, double smokeThickness = 0)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件
double	<i>distance</i>	传输距离（米）
double	<i>smokeConcentration</i>	烟雾浓度
double	<i>smokeThickness</i>	烟雾厚度（米）

Returns

Type	Description
double	大气透过率

CalcLaser(WeatherCondition, double)

计算激光在给定天气条件和距离下的大气透过率

Declaration

```
public static double CalcLaser(WeatherCondition weather, double distance)
```

Parameters

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件
double	<i>distance</i>	传输距离（米）

Returns

Type	Description
double	大气透过率

CalcLaserWithSmoke(WeatherCondition, double, double, double)

计算激光在有烟雾条件下的大气透过率

Declaration

```
public static double CalcLaserWithSmoke(WeatherCondition weather, double distance, double smokeConcentration = 0, double smokeThickness = 0)
```

Parameters

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件
double	<i>distance</i>	传输距离（米）
double	<i>smokeConcentration</i>	烟雾浓度
double	<i>smokeThickness</i>	烟雾厚度（米）

Returns

Type	Description
double	大气透过率

CalcMillimeterWave(WeatherCondition, double, double, double)

计算毫米波在给定条件下的大气透过率

Declaration

```
public static double CalcMillimeterWave(WeatherCondition weather, double distance, double smokeConcentration = 0, double smokeThickness = 0)
```

Parameters

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件
double	<i>distance</i>	传输距离（米）
double	<i>smokeConcentration</i>	烟雾浓度
double	<i>smokeThickness</i>	烟雾厚度（米）

Returns

Type	Description
double	大气透过率

CalcTurbulenceEffect(WeatherCondition, double)

计算湍流效应对激光透过率的影响

Declaration

```
public static double CalcTurbulenceEffect(WeatherCondition weather, double distance)
```

Parameters

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件
double	<i>distance</i>	传输距离（米）

Returns

Type	Description
double	大气透过率

CalcUV(WeatherCondition, double)

计算紫外线在给定天气条件和距离下的大气透过率

Declaration

```
public static double CalcUV(WeatherCondition weather, double distance)
```

Parameters

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件
double	<i>distance</i>	传输距离（米）

Returns

Type	Description
------	-------------

Type	Description
double	大气透过率

CalculateAtmosphericTurbulenceExport(double, double, double, double, double, double, int)

导出函数：计算大气湍流影响

Declaration

```
[DllImport("CalculateAtmosphericTurbulence", CallingConvention = CallingConvention.Cdecl)]
public static double CalculateAtmosphericTurbulenceExport(double wavelength, double distance, double temperature, double relativeHumidity, double pressure, double visibility, int weatherType)
```

Parameters

Type	Name	Description
double	<i>wavelength</i>	
double	<i>distance</i>	
double	<i>temperature</i>	
double	<i>relativeHumidity</i>	
double	<i>pressure</i>	
double	<i>visibility</i>	
int	<i>weatherType</i>	

Returns

Type	Description
double	

CalculateReceivedRadiation(double, double, double, double, double, double)

计算双程传输后接收到的辐射功率

Declaration

```
public static double CalculateReceivedRadiation(double transmittance, double laserEnergy, double pulseWidth, double targetDistance, double receiverDistance, double targetReflectivity)
```

Parameters

Type	Name	Description
double	<i>transmittance</i>	大气透过率
double	<i>laserEnergy</i>	激光能量（焦耳）
double	<i>pulseWidth</i>	脉冲宽度（纳秒）
double	<i>targetDistance</i>	目标距离（米）
double	<i>receiverDistance</i>	接收器距离（米）
double	<i>targetReflectivity</i>	目标反射率

Returns

Type	Description
double	接收到的辐射功率（瓦特）

CalculateReceivedRadiationSinglePath(double, double, double)

计算单程传输后接收到的辐射功率

Declaration

```
public static double CalculateReceivedRadiationSinglePath(double transmittance, double targetRadiation, double receiverDistance)
```

Parameters

Type	Name	Description
double	<i>transmittance</i>	大气透过率
double	<i>targetRadiation</i>	目标辐射（W/Sr）
double	<i>receiverDistance</i>	接收器距离（米）

Returns

Type	Description
double	接收到的辐射功率（W/Sr）

CalculateSmokeScreenTransmittance(double, double)

计算烟幕对电磁波的透过率

Declaration

```
public static double CalculateSmokeScreenTransmittance(double smokeConcentration, double smokeThickness)
```

Parameters

Type	Name	Description
double	<i>smokeConcentration</i>	烟幕浓度（g/m ³ ）
double	<i>smokeThickness</i>	烟幕厚度（米）

Returns

Type	Description
double	烟幕透过率（0到1之间的值）

CalculateTransmittanceExport(double, double, double, double, double, double, int)

导出函数：计算大气透过率

Declaration

```
[DllImport("CalculateTransmittance", CallingConvention = CallingConvention.Cdecl)]
public static double CalculateTransmittanceExport(double wavelength, double distance, double temperature, double relativeHumidity, double pressure, double visibility, int weatherType)
```

Parameters

Type	Name	Description
double	<i>wavelength</i>	
double	<i>distance</i>	
double	<i>temperature</i>	
double	<i>relativeHumidity</i>	
double	<i>pressure</i>	
double	<i>visibility</i>	
int	<i>weatherType</i>	

Returns

Type	Description
double	

[Show / Hide Table of Contents](#)

Class WeatherCondition

天气条件类，用于描述大气传输计算所需的天气参数

Inheritance

↳ [object](#)
↳ WeatherCondition

Inherited Members

[object.Equals\(object\)](#)
[object.Equals\(object, object\)](#)
[object.GetHashCode\(\)](#)
[object.GetType\(\)](#)
[object.MemberwiseClone\(\)](#)
[object.ReferenceEquals\(object, object\)](#)
[object.ToString\(\)](#)

Namespace: [AirTransmission](#)

Assembly: AirTransmission.dll

Syntax

```
public class WeatherCondition
```

Remarks

包含以下主要天气参数：

- 天气类型（晴天、雨天、雪天等）
- 温度
- 相对湿度
- 能见度
- 降水量
- CO2浓度

Constructors

WeatherCondition(WeatherType, double, double, double, double?, double)

天气条件类，用于描述大气传输计算所需的天气参数

Declaration

```
public WeatherCondition(WeatherType type, double temperature, double relativeHumidity, double visibility, double? precipitation = null, double co2Concentration = 415)
```

Parameters

Type	Name	Description
------	------	-------------

Type	Name	Description
WeatherType	<i>type</i>	天气类型
double	<i>temperature</i>	温度（摄氏度）
double	<i>relativeHumidity</i>	相对湿度（百分比）
double	<i>visibility</i>	能见度（公里）
double?	<i>precipitation</i>	降水量（毫米/小时），可选
double	<i>co2Concentration</i>	二氧化碳浓度（ppm），默认415ppm

Remarks

包含以下主要天气参数：

- 天气类型（晴天、雨天、雪天等）
- 温度
- 相对湿度
- 能见度
- 降水量
- CO2浓度

Properties

CO2Concentration

二氧化碳浓度（ppm）

Declaration

```
public double CO2Concentration { get; set; }
```

Property Value

Type	Description
double	

Precipitation

降水量（毫米/小时）

Declaration

```
public double? Precipitation { get; set; }
```

Property Value

Type	Description
double?	

RelativeHumidity

相对湿度（百分比）

Declaration

```
public double RelativeHumidity { get; set; }
```

Property Value

Type	Description
double	

Temperature

温度（摄氏度）

Declaration

```
public double Temperature { get; set; }
```

Property Value

Type	Description
double	

Type

天气类型

Declaration

```
public WeatherType Type { get; set; }
```

Property Value

Type	Description
WeatherType	

Visibility

能见度（公里）

Declaration

```
public double Visibility { get; set; }
```

Property Value

Type	Description
double	

Methods

PrintWeatherInfo(WeatherCondition)

打印天气信息

Declaration

```
public static void PrintWeatherInfo(WeatherCondition weather)
```

Parameters

Type	Name	Description
WeatherCondition	<i>weather</i>	天气条件对象

Remarks

输出格式： 天气类型: [类型], 温度: [温度]°C, 相对湿度: [湿度]%, 能见度: [能见度]km, 降水量: [降水量]mm/h

[Show / Hide Table of Contents](#)

Enum WeatherType

天气类型枚举

Namespace: [AirTransmission](#)

Assembly: AirTransmission.dll

Syntax

```
public enum WeatherType
```

Fields

Name	Description
晴天	晴朗天气
沙尘	沙尘天气
雨天	雨天
雪天	雪天
雾天	雾天

使用示例

本文档提供了在 C# 和 C++ 项目中使用大气传输计算库的示例。

C# 调用示例

项目配置

1. 在 Visual Studio 中创建新的 .NET 项目
2. 将 AirTransmission.dll 复制到项目目录
3. 在项目中添加 DLL 引用：
 - 右键点击项目 -> 添加 -> 引用
 - 浏览 -> 选择 AirTransmission.dll

代码示例

```
using AirTransmission;

// 创建天气条件
var weather = new WeatherCondition
{
    Temperature = 20,      // 温度 (摄氏度)
    Humidity = 0.65,       // 相对湿度 (0-1)
    Pressure = 101.325,    // 大气压力 (kPa)
    Visibility = 10000,    // 能见度 (米)
    WeatherType = WeatherType.Clear // 天气类型
};

// 创建大气透过率计算器
var calculator = new AtmosphericTransmittanceCalculator();

// 计算激光透过率
double wavelength = 1.064; // 波长 (微米)
double distance = 1000;    // 传输距离 (米)
double transmittance = calculator.CalculateTransmittance(wavelength, distance, weather);

Console.WriteLine($"激光透过率: {transmittance:P2}");
```

高级用法

```
// 计算不同天气条件下的透过率
var weatherConditions = new[]
{
    new WeatherCondition { WeatherType = WeatherType.Clear, Temperature = 20, Humidity = 0.65 },
    new WeatherCondition { WeatherType = WeatherType.Cloudy, Temperature = 18, Humidity = 0.8 },
    new WeatherCondition { WeatherType = WeatherType.Rain, Temperature = 15, Humidity = 0.95 }
};

var calculator = new AtmosphericTransmittanceCalculator();
double wavelength = 1.064; // 微米
double distance = 1000;    // 米

foreach (var weather in weatherConditions)
{
    var transmittance = calculator.CalculateTransmittance(wavelength, distance, weather);
    Console.WriteLine($"天气: {weather.WeatherType}, 透过率: {transmittance:P2}");
}

// 计算大气湍流影响
var turbulence = calculator.CalculateAtmosphericTurbulence(wavelength, distance, weather);
Console.WriteLine($"大气湍流强度: {turbulence:E2}");
```

C++ 调用示例

项目配置

1. 在 Visual Studio 中创建新的 C++ 项目
2. 将以下文件复制到项目目录：
 - `AirTransmission.dll` - 动态链接库文件
 - `include/AirTransmission.h` - 头文件
3. 在项目设置中添加 `include` 目录
4. 在代码中包含头文件并使用导出函数

代码示例

```

#include "AirTransmission.h"
#include <iostream>

int main() {
    // 设置参数
    double wavelength = 1.064;    // 波长 (微米)
    double distance = 1000.0;      // 距离 (米)
    double temperature = 20.0;     // 温度 (摄氏度)
    double relativeHumidity = 0.65; // 相对湿度 (0-1)
    double pressure = 101.325;     // 大气压力 (kPa)
    double visibility = 10000.0;   // 能见度 (米)
    int weatherType = WeatherType::Clear; // 天气类型 (晴天)

    // 计算透过率
    double transmittance = CalculateTransmittance(
        wavelength, distance, temperature, relativeHumidity,
        pressure, visibility, weatherType
    );
    std::cout << "透过率: " << transmittance * 100.0 << "%" << std::endl;

    // 计算湍流影响
    double turbulence = CalculateAtmosphericTurbulence(
        wavelength, distance, temperature, relativeHumidity,
        pressure, visibility, weatherType
    );
    std::cout << "湍流强度: " << turbulence << std::endl;

    return 0;
}

```

注意事项

1. 确保 AirTransmission.dll 在程序运行目录下
2. 头文件中已定义了所有必要的类型和函数
3. 使用 WeatherType 枚举来指定天气类型
4. 所有浮点数参数都使用 double 类型
5. 参数单位说明:
 - 波长: 微米 (μm)
 - 距离: 米 (m)
 - 温度: 摄氏度 (°C)
 - 相对湿度: 0-1
 - 压力: 千帕 (kPa)
 - 能见度: 米 (m)