有关浮点数的比较

黄永峰 2011-10-10

在数学运算当中经常会涉及到判断两个数是否相等的情况.对于整数很好处理 A==B 这样的一个语句就可以解决全部的问题,但是对于浮点数是不同的。

首先,浮点数在计算机当中的二进制表达方式就决定了大多数浮点数都是无法精确的表达的。现在的计算机大部分都是数字计算机，不是模拟机，数字机的离散化的数据表示方法自然无法精确表达大部分的数据量的。

其次计算机浮点数的精度在单精度 float 类型下，只有 7 位，在进行浮点运算的时候，这个精度往往会导致运算的结果和实际期望的结果之间有误差.因为前两个原因，我们很难用 A==B 来判定两个浮点数是否相同.很自然，我们可以想到 fabs(A-B)<epsilon 这样的一种判别方法（具体见第 3 次讲义中“浮点数陷阱”）。这种方法只适用于在实际应用下具体确定不同的 epsilon 这个绝对的数据。例如，如果 A 和 B 变量是表示人民币（单位是元），则这个 epsilon 就应该是 0.001.但是，在不考虑实际应用情况，要确定一个普遍使用的 epsilon 绝对数据，这种判别方法稳也就不稳妥了。

首先，epsilon 是一个绝对的数据，也就是误差分析当中说的绝对误差,使用一个固定的数值,对于 float 类型可以表达的整个数域来说是不可以的。比如 epsilon 取值为 0.0001，而 a 和 b 的数值大小也是 0.0001 附近的，那么显然不合适.另外对于 a 和 b 大小是 10000 这样的数据的时候，它也不合适，因为 10000 和 10001 也可以认为是相等的呢?适合它的情况只是 a 或者 b 在 1 或者 0 附近的时候.既然绝对误差不可以，那么自然的我们就会想到了相对误差。

```
bool IsEqual(float a, float b, float relError )
{return ( fabs ( (a-b)/a ) < relError )?true:false;
}
```

这样写还不完善，因为是拿固定的第一个参数做比较的，那么在调用 IsEqual(a, b, relError) 和 IsEqual(b, a, relError) 的时候，可能得到不同的结果.同时如果第一个参数是 0 的话,就有可能是除 0 溢出.这个可以改造,把除数选取为 a 和 b 当中绝对数值较大的即可

```
bool IsEqual(float a, float b, relError )
{if (fabs(a)<fabs(b)) return( fabs((a-b)/a) > relError )?true:false;
return(fabs( (a-b)/b) > relError )?true:false;};
```

使用相对误差就很完善吗？也不是，在某些特殊情况下，相对误差也不能代表全部。比如在判断空间三点是否共线的时候,使用判断点到另外两个点形成的线段的距离的方法的时候.只用相对误差是不够的，应为线段距离可能很段，也可能很长，点到线段的距离，以及线段的长度做综合比较的时候，需要相对误差和绝对误差结合的方式才可以。相对完整的比较算法应该如下：

```
bool IsEqual(float a, float b, float absError, float relError )
```

```
{if (a==b) return true;
if (fabs(a-b)<absError)return true;
if (fabs(a>b)return(fabs((a-b)/a>relError )?true:false;
return(fabs((a-b)/b>relError )?true:false;}
```

这样才相对完整.这仅仅是浮点数之间最初级的比较方法.高级的方法看 浮点数比较（二）