```
/*************************************************************************
                                PFR&VF UDF
1   PFR
2   VF
*************************************************************************/
```

头文件，定义一些预定义宏，这里是将M、VOL、RHO分别赋值

```
#include "udf.h"
#define M 0.00001      //pollutant generation rate(kg/m^3*s)
#define VOL 50         //volume of object street space(m^3)
#define RHO 1.29       //density of air(kg/m^3)
```

//之后为行注释

```
DEFINE_ON_DEMAND(PFR_udf)
```

ON DEMAND宏，名字为PFR_udf

```
{
  Domain *domain;
  Thread *t;
  cell_t c;
```

ON DEMAND宏需要声明Domain和Thread，变量名分别为domain和t，这里*表示变量为线程（即指针）

```
  real cpt,cpa;                 //cpt,total concentration of each cell of object space(kg*m^3/kg);
cpa,average concentration of object space(kg/kg)
  real x[ND_ND];
  real PRF;
```

分别声明一些变量，这里变量类型均为real。其中cpt为某点处的示踪气体体积浓度；cpa为空间平均体积浓度；x[ND_ND]用来储存坐标值，是数组变量（具体含义之前已经解释过）；PRF为最后需要的PRF值（这里PRF和PFR搞混了，其实是一个意思）

通过Get_Domain()宏将得到的Domain线程返回到变量domain中

```
  domain=Get_Domain(1);

  thread_loop_c(t,domain)
    {
      begin_c_loop(c,t)
        {
          C_CENTROID(x,c,t);

          if(x[0]>0 && x[0]<1 && x[1]>0 && x[1]<1 && x[2]>0 && x[2]<1)      //coordinate
values of object space
            {
              cpt=cpt+C_YI(c,t,0)*C_VOLUME(c,t);
            }
            else
            {
              cpt=cpt;
            }
        }
      end_c_loop(c,t)
    }
  cpa=cpt/VOL;
```

遍历（loop）整个domain，将得到的Thread线程返回到变量t中

遍历线程t中的每一个单元c

每一次遍历会将c的重心坐标值返回到数组x中

目标区域的坐标值范围。这里的意思是：0<x<1且0<y<1且0<z<1（将0和1改为你想要求的区域的坐标值）；x[0]为x轴坐标，x[1]为y轴坐标，x[2]为z轴坐标。

每一次遍历会将目标区域的质量浓度转化为体积浓度，并相加得到目标区域的体积浓度总和（完全按照积分公式）

某个单元c的体积

第一种组分的浓度（软件默认为质量浓度，这里需要将其转化为体积浓度才能在公式中应用，因此与每个单元的体积相乘）

将得到的体积浓度总和除以目标区域的体积，即得到目标区域的空间平均体积浓度（这里VOL为目标区域的体积）

如果目标区域复杂，或者有多个目标区域，则预定义VOL会比较麻烦，这时可以采用C_VOLUME()宏将目标区域每个单元c的体积相加，即得到目标区域的整体体积

```
   PRF=(M*VOL)/(cpa*RHO);
   printf("PFR of object space is: %g\n",PRF);
}

/************************************************************************/

DEFINE_ON_DEMAND(VF_udf)

{
   Domain *domain;
   Thread *t;
   face_t f;

   real delta_qp,qp;                    //delta_qp,inflow flux of pollutants into the domain(kg/s);
qp,pollutants generation rate of the domain(kg/s)
   real x[ND_ND];
   real NV_VEC(A);
   real VF;

   domain=Get_Domain(1);

   thread_loop_f(t,domain)
     {
       begin_f_loop(f,t)
         {
             F_CENTROID(x,f,t);
             F_AREA(A,f,t);

             if(x[0]>0 && x[0]<1 && x[1]>0 && x[1]<1 && x[2]==1)       //z coordinate of the
face
             {

delta_qp=delta_qp+RHO*NV_MAG(A)*(-1*(F_W(f,t)-fabs(F_W(f,t)))/2)*F_YI(f,t,0);
             }
   else if(x[0]>0 && x[0]<1 && x[1]==0 && x[2]>0 && x[2]<1)
{
   delta_qp=delta_qp+RHO*NV_MAG(A)*((F_V(f,t)+fabs(F_V(f,t)))/2)*F_YI(f,t,0);
}
   else if(x[0]>0 && x[0]<1 && x[1]==1 && x[2]>0 && x[2]<1)
{
   delta_qp=delta_qp+RHO*NV_MAG(A)*(-1*(F_V(f,t)-fabs(F_V(f,t)))/2)*F_YI(f,t,0);
}
   else if(x[0]==0 && x[1]>0 && x[1]<1 && x[2]>0 && x[2]<1)
{
```

PRF的计算公式

将得到的结果输出到TUI中（具体语法参见C语言相关教程）

VF的udf类似，仅对不同之处进行说明

NV_VEC为定义向量的宏，这里是将变量A定义为向量（向量本质上是一个数组，在udf中将其单独称为向量）

通过F_AREA()宏将线程t中的面单元f的面积向量返回到变量A中（在udf中，面积包含面积大小与面法向向量两层概念，因此为向量类型，而不是简单的只包含面积大小）

通过NV_MAG()宏得到向量A的大小，这里也就是得到了面单元的面积大小（标量）

求得某个水平面上向下的入流速度（这里的操作是为除去向上的速度分量）

F_U(),F_V(),F_W()宏分别得到面f上u、v、w三个方向的速度分量大小

这里的操作的意思是得到目标区域的上前后左右五个方向（对应到xyz轴）上的面上的每个面单元f的入流速度，即inflow wind speed（详见公式），将其与浓度和面积和密度相乘后进行积分，得到入流污染物浓度流量。

```
                delta_qp=delta_qp+RHO*NV_MAG(A)*((F_U(f,t)+fabs(F_U(f,t)))/2)*F_YI(f,t,0);
            }
                else if(x[0]==1 && x[1]>0 && x[1]<1 && x[2]>0 && x[2]<1)
            {

delta_qp=delta_qp+RHO*NV_MAG(A)*(-1*(F_U(f,t)-fabs(F_U(f,t)))/2)*F_YI(f,t,0);
            }
                else
            {
                delta_qp=delta_qp;
            }
        }
      end_f_loop(f,t)
    }
  qp=VOL*M;
  VF=1+(delta_qp/qp);
  printf("VF of object space is: %g\n",VF);
}
```

VF的计算公式

输出得到的VF的大小