#ifndef OPENSBLIBLOCK00\_KERNEL\_H

#define OPENSBLIBLOCK00\_KERNEL\_H

void opensbliblock00Kernel010(double \*rhou0\_B0, double \*rho\_B0, double \*rhoE\_B0, double \*x0\_B0, const int \*idx)

{

double u0 = 0.0;

double p = 0.0;

double d = 0.0;

x0\_B0[OPS\_ACC3(0)] = Delta0block0\*idx[0];

u0 = 0;

p = ((x0\_B0[OPS\_ACC3(0)] < 0.5) ? (

1.0

)

: (

0.1

));

d = ((x0\_B0[OPS\_ACC3(0)] < 0.5) ? (

1.0

)

: (

0.125

));

rho\_B0[OPS\_ACC1(0)] = d;

rhou0\_B0[OPS\_ACC0(0)] = d\*u0;

rhoE\_B0[OPS\_ACC2(0)] = 0.5\*d\*pow(u0, 2.0) + p/(gama - 1.0);

}

void opensbliblock00Kernel008(double \*rhou0\_B0, double \*rho\_B0, double \*rhoE\_B0)

{

double u0 = 0.0;

double p = 0.0;

double d = 0.0;

d = 1.00000000000000;

u0 = 0.0;

p = 1.00000000000000;

rho\_B0[OPS\_ACC1(0)] = d;

rhou0\_B0[OPS\_ACC0(0)] = d\*u0;

rhoE\_B0[OPS\_ACC2(0)] = 0.5\*d\*pow(u0, 2.0) + p/(gama - 1.0);

}

void opensbliblock00Kernel009(double \*rhou0\_B0, double \*rho\_B0, double \*rhoE\_B0)

{

double u0 = 0.0;

double p = 0.0;

double d = 0.0;

d = 0.125000000000000;

u0 = 0.0;

p = 0.100000000000000;

rho\_B0[OPS\_ACC1(0)] = d;

rhou0\_B0[OPS\_ACC0(0)] = d\*u0;

rhoE\_B0[OPS\_ACC2(0)] = 0.5\*d\*pow(u0, 2.0) + p/(gama - 1.0);

}

void opensbliblock00Kernel003(const double \*rhou0\_B0, const double \*rho\_B0, double \*u0\_B0)

{

u0\_B0[OPS\_ACC2(0)] = rhou0\_B0[OPS\_ACC0(0)]/rho\_B0[OPS\_ACC1(0)];

}

void opensbliblock00Kernel004(const double \*rho\_B0, const double \*u0\_B0, const double \*rhoE\_B0, double \*p\_B0)

{

p\_B0[OPS\_ACC3(0)] = (gama - 1)\*(rhoE\_B0[OPS\_ACC2(0)] - rc0\*rho\_B0[OPS\_ACC0(0)]\*pow(u0\_B0[OPS\_ACC1(0)], 2));

}

void opensbliblock00Kernel002(const double \*rho\_B0, const double \*p\_B0, double \*a\_B0)

{

a\_B0[OPS\_ACC2(0)] = sqrt(gama\*p\_B0[OPS\_ACC1(0)]/rho\_B0[OPS\_ACC0(0)]);

}

void opensbliblock00Kernel000(const double \*rhou0\_B0, const double \*p\_B0, const double \*a\_B0, const double \*rho\_B0,

const double \*u0\_B0, const double \*rhoE\_B0, double \*wk2\_B0, double \*wk0\_B0, double \*wk1\_B0)

{

double AVG\_0\_0\_LEV\_02 = 0.0;

double CS\_04 = 0.0;

double Recon\_1 = 0.0;

double beta\_2 = 0.0;

double CS\_24 = 0.0;

double CF\_11 = 0.0;

double AVG\_0\_0\_LEV\_12 = 0.0;

double CS\_20 = 0.0;

double AVG\_0\_0\_LEV\_20 = 0.0;

double CF\_15 = 0.0;

double CS\_01 = 0.0;

double CF\_25 = 0.0;

double CS\_05 = 0.0;

double inv\_omega\_sum = 0.0;

double CF\_00 = 0.0;

double beta\_1 = 0.0;

double CS\_03 = 0.0;

double max\_lambda\_0\_00 = 0.0;

double CF\_12 = 0.0;

double AVG\_0\_a = 0.0;

double CS\_14 = 0.0;

double CF\_21 = 0.0;

double AVG\_0\_0\_LEV\_01 = 0.0;

double AVG\_0\_rho = 0.0;

double AVG\_0\_0\_LEV\_21 = 0.0;

double AVG\_0\_0\_LEV\_11 = 0.0;

double AVG\_0\_inv\_rho = 0.0;

double CS\_10 = 0.0;

double alpha\_0 = 0.0;

double CF\_05 = 0.0;

double CF\_24 = 0.0;

double CS\_15 = 0.0;

double CF\_14 = 0.0;

double CF\_10 = 0.0;

double alpha\_1 = 0.0;

double CS\_12 = 0.0;

double delta\_2 = 0.0;

double Recon\_0 = 0.0;

double CF\_13 = 0.0;

double CS\_21 = 0.0;

double CF\_23 = 0.0;

double CS\_23 = 0.0;

double inv\_AVG\_rho = 0.0;

double AVG\_0\_0\_LEV\_00 = 0.0;

double inv\_alpha\_sum = 0.0;

double AVG\_0\_u0 = 0.0;

double CS\_11 = 0.0;

double beta\_0 = 0.0;

double CS\_02 = 0.0;

double CF\_01 = 0.0;

double CS\_00 = 0.0;

double AVG\_0\_0\_LEV\_22 = 0.0;

double max\_lambda\_0\_22 = 0.0;

double CF\_02 = 0.0;

double CS\_22 = 0.0;

double AVG\_0\_0\_LEV\_10 = 0.0;

double CF\_20 = 0.0;

double CS\_13 = 0.0;

double inv\_AVG\_a = 0.0;

double Recon\_2 = 0.0;

double CS\_25 = 0.0;

double delta\_0 = 0.0;

double CF\_04 = 0.0;

double delta\_1 = 0.0;

double CF\_22 = 0.0;

double CF\_03 = 0.0;

double max\_lambda\_0\_11 = 0.0;

double alpha\_2 = 0.0;

AVG\_0\_rho = sqrt(rho\_B0[OPS\_ACC3(0)]\*rho\_B0[OPS\_ACC3(1)]);

AVG\_0\_inv\_rho = 1.0/(sqrt(rho\_B0[OPS\_ACC3(1)]) + sqrt(rho\_B0[OPS\_ACC3(0)]));

AVG\_0\_u0 = AVG\_0\_inv\_rho\*(sqrt(rho\_B0[OPS\_ACC3(1)])\*u0\_B0[OPS\_ACC4(1)] +

sqrt(rho\_B0[OPS\_ACC3(0)])\*u0\_B0[OPS\_ACC4(0)]);

AVG\_0\_a = sqrt(gamma\_m1\*(AVG\_0\_inv\_rho\*((p\_B0[OPS\_ACC1(0)] + rhoE\_B0[OPS\_ACC5(0)])/sqrt(rho\_B0[OPS\_ACC3(0)]) +

(p\_B0[OPS\_ACC1(1)] + rhoE\_B0[OPS\_ACC5(1)])/sqrt(rho\_B0[OPS\_ACC3(1)])) - rc0\*pow(AVG\_0\_u0, 2)));

inv\_AVG\_a = 1.0/AVG\_0\_a;

inv\_AVG\_rho = 1.0/AVG\_0\_rho;

AVG\_0\_0\_LEV\_00 = (rc1)\*AVG\_0\_u0\*inv\_AVG\_a\*(gama\*AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2) - AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a,

2) + 2);

AVG\_0\_0\_LEV\_01 = -rc0\*inv\_AVG\_a\*(gama\*AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2) - AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2) + 1);

AVG\_0\_0\_LEV\_02 = (rc0)\*gamma\_m1\*pow(inv\_AVG\_a, 2);

AVG\_0\_0\_LEV\_10 = -rc0\*(gama\*pow(AVG\_0\_u0, 2)\*pow(inv\_AVG\_a, 2) - pow(AVG\_0\_u0, 2)\*pow(inv\_AVG\_a, 2) - 2);

AVG\_0\_0\_LEV\_11 = gamma\_m1\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2);

AVG\_0\_0\_LEV\_12 = -gamma\_m1\*pow(inv\_AVG\_a, 2);

AVG\_0\_0\_LEV\_20 = (rc1)\*AVG\_0\_u0\*inv\_AVG\_a\*(gama\*AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2) - AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a,

2) - 2);

AVG\_0\_0\_LEV\_21 = -rc0\*inv\_AVG\_a\*(gama\*AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2) - AVG\_0\_a\*AVG\_0\_u0\*pow(inv\_AVG\_a, 2) - 1);

AVG\_0\_0\_LEV\_22 = (rc0)\*gamma\_m1\*pow(inv\_AVG\_a, 2);

CF\_00 = AVG\_0\_0\_LEV\_00\*rhou0\_B0[OPS\_ACC0(-2)] + AVG\_0\_0\_LEV\_01\*p\_B0[OPS\_ACC1(-2)] +

AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(-2)]\*u0\_B0[OPS\_ACC4(-2)] + AVG\_0\_0\_LEV\_02\*p\_B0[OPS\_ACC1(-2)]\*u0\_B0[OPS\_ACC4(-2)]

+ AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(-2)]\*u0\_B0[OPS\_ACC4(-2)];

CF\_10 = AVG\_0\_0\_LEV\_10\*rhou0\_B0[OPS\_ACC0(-2)] + AVG\_0\_0\_LEV\_11\*p\_B0[OPS\_ACC1(-2)] +

AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(-2)]\*u0\_B0[OPS\_ACC4(-2)] + AVG\_0\_0\_LEV\_12\*p\_B0[OPS\_ACC1(-2)]\*u0\_B0[OPS\_ACC4(-2)]

+ AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(-2)]\*u0\_B0[OPS\_ACC4(-2)];

CF\_20 = AVG\_0\_0\_LEV\_20\*rhou0\_B0[OPS\_ACC0(-2)] + AVG\_0\_0\_LEV\_21\*p\_B0[OPS\_ACC1(-2)] +

AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(-2)]\*u0\_B0[OPS\_ACC4(-2)] + AVG\_0\_0\_LEV\_22\*p\_B0[OPS\_ACC1(-2)]\*u0\_B0[OPS\_ACC4(-2)]

+ AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(-2)]\*u0\_B0[OPS\_ACC4(-2)];

CF\_01 = AVG\_0\_0\_LEV\_00\*rhou0\_B0[OPS\_ACC0(-1)] + AVG\_0\_0\_LEV\_01\*p\_B0[OPS\_ACC1(-1)] +

AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(-1)]\*u0\_B0[OPS\_ACC4(-1)] + AVG\_0\_0\_LEV\_02\*p\_B0[OPS\_ACC1(-1)]\*u0\_B0[OPS\_ACC4(-1)]

+ AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(-1)]\*u0\_B0[OPS\_ACC4(-1)];

CF\_11 = AVG\_0\_0\_LEV\_10\*rhou0\_B0[OPS\_ACC0(-1)] + AVG\_0\_0\_LEV\_11\*p\_B0[OPS\_ACC1(-1)] +

AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(-1)]\*u0\_B0[OPS\_ACC4(-1)] + AVG\_0\_0\_LEV\_12\*p\_B0[OPS\_ACC1(-1)]\*u0\_B0[OPS\_ACC4(-1)]

+ AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(-1)]\*u0\_B0[OPS\_ACC4(-1)];

CF\_21 = AVG\_0\_0\_LEV\_20\*rhou0\_B0[OPS\_ACC0(-1)] + AVG\_0\_0\_LEV\_21\*p\_B0[OPS\_ACC1(-1)] +

AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(-1)]\*u0\_B0[OPS\_ACC4(-1)] + AVG\_0\_0\_LEV\_22\*p\_B0[OPS\_ACC1(-1)]\*u0\_B0[OPS\_ACC4(-1)]

+ AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(-1)]\*u0\_B0[OPS\_ACC4(-1)];

CF\_02 = AVG\_0\_0\_LEV\_00\*rhou0\_B0[OPS\_ACC0(0)] + AVG\_0\_0\_LEV\_01\*p\_B0[OPS\_ACC1(0)] +

AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(0)]\*u0\_B0[OPS\_ACC4(0)] + AVG\_0\_0\_LEV\_02\*p\_B0[OPS\_ACC1(0)]\*u0\_B0[OPS\_ACC4(0)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(0)]\*u0\_B0[OPS\_ACC4(0)];

CF\_12 = AVG\_0\_0\_LEV\_10\*rhou0\_B0[OPS\_ACC0(0)] + AVG\_0\_0\_LEV\_11\*p\_B0[OPS\_ACC1(0)] +

AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(0)]\*u0\_B0[OPS\_ACC4(0)] + AVG\_0\_0\_LEV\_12\*p\_B0[OPS\_ACC1(0)]\*u0\_B0[OPS\_ACC4(0)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(0)]\*u0\_B0[OPS\_ACC4(0)];

CF\_22 = AVG\_0\_0\_LEV\_20\*rhou0\_B0[OPS\_ACC0(0)] + AVG\_0\_0\_LEV\_21\*p\_B0[OPS\_ACC1(0)] +

AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(0)]\*u0\_B0[OPS\_ACC4(0)] + AVG\_0\_0\_LEV\_22\*p\_B0[OPS\_ACC1(0)]\*u0\_B0[OPS\_ACC4(0)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(0)]\*u0\_B0[OPS\_ACC4(0)];

CF\_03 = AVG\_0\_0\_LEV\_00\*rhou0\_B0[OPS\_ACC0(1)] + AVG\_0\_0\_LEV\_01\*p\_B0[OPS\_ACC1(1)] +

AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(1)]\*u0\_B0[OPS\_ACC4(1)] + AVG\_0\_0\_LEV\_02\*p\_B0[OPS\_ACC1(1)]\*u0\_B0[OPS\_ACC4(1)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(1)]\*u0\_B0[OPS\_ACC4(1)];

CF\_13 = AVG\_0\_0\_LEV\_10\*rhou0\_B0[OPS\_ACC0(1)] + AVG\_0\_0\_LEV\_11\*p\_B0[OPS\_ACC1(1)] +

AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(1)]\*u0\_B0[OPS\_ACC4(1)] + AVG\_0\_0\_LEV\_12\*p\_B0[OPS\_ACC1(1)]\*u0\_B0[OPS\_ACC4(1)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(1)]\*u0\_B0[OPS\_ACC4(1)];

CF\_23 = AVG\_0\_0\_LEV\_20\*rhou0\_B0[OPS\_ACC0(1)] + AVG\_0\_0\_LEV\_21\*p\_B0[OPS\_ACC1(1)] +

AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(1)]\*u0\_B0[OPS\_ACC4(1)] + AVG\_0\_0\_LEV\_22\*p\_B0[OPS\_ACC1(1)]\*u0\_B0[OPS\_ACC4(1)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(1)]\*u0\_B0[OPS\_ACC4(1)];

CF\_04 = AVG\_0\_0\_LEV\_00\*rhou0\_B0[OPS\_ACC0(2)] + AVG\_0\_0\_LEV\_01\*p\_B0[OPS\_ACC1(2)] +

AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(2)]\*u0\_B0[OPS\_ACC4(2)] + AVG\_0\_0\_LEV\_02\*p\_B0[OPS\_ACC1(2)]\*u0\_B0[OPS\_ACC4(2)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(2)]\*u0\_B0[OPS\_ACC4(2)];

CF\_14 = AVG\_0\_0\_LEV\_10\*rhou0\_B0[OPS\_ACC0(2)] + AVG\_0\_0\_LEV\_11\*p\_B0[OPS\_ACC1(2)] +

AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(2)]\*u0\_B0[OPS\_ACC4(2)] + AVG\_0\_0\_LEV\_12\*p\_B0[OPS\_ACC1(2)]\*u0\_B0[OPS\_ACC4(2)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(2)]\*u0\_B0[OPS\_ACC4(2)];

CF\_24 = AVG\_0\_0\_LEV\_20\*rhou0\_B0[OPS\_ACC0(2)] + AVG\_0\_0\_LEV\_21\*p\_B0[OPS\_ACC1(2)] +

AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(2)]\*u0\_B0[OPS\_ACC4(2)] + AVG\_0\_0\_LEV\_22\*p\_B0[OPS\_ACC1(2)]\*u0\_B0[OPS\_ACC4(2)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(2)]\*u0\_B0[OPS\_ACC4(2)];

CF\_05 = AVG\_0\_0\_LEV\_00\*rhou0\_B0[OPS\_ACC0(3)] + AVG\_0\_0\_LEV\_01\*p\_B0[OPS\_ACC1(3)] +

AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(3)]\*u0\_B0[OPS\_ACC4(3)] + AVG\_0\_0\_LEV\_02\*p\_B0[OPS\_ACC1(3)]\*u0\_B0[OPS\_ACC4(3)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(3)]\*u0\_B0[OPS\_ACC4(3)];

CF\_15 = AVG\_0\_0\_LEV\_10\*rhou0\_B0[OPS\_ACC0(3)] + AVG\_0\_0\_LEV\_11\*p\_B0[OPS\_ACC1(3)] +

AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(3)]\*u0\_B0[OPS\_ACC4(3)] + AVG\_0\_0\_LEV\_12\*p\_B0[OPS\_ACC1(3)]\*u0\_B0[OPS\_ACC4(3)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(3)]\*u0\_B0[OPS\_ACC4(3)];

CF\_25 = AVG\_0\_0\_LEV\_20\*rhou0\_B0[OPS\_ACC0(3)] + AVG\_0\_0\_LEV\_21\*p\_B0[OPS\_ACC1(3)] +

AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(3)]\*u0\_B0[OPS\_ACC4(3)] + AVG\_0\_0\_LEV\_22\*p\_B0[OPS\_ACC1(3)]\*u0\_B0[OPS\_ACC4(3)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(3)]\*u0\_B0[OPS\_ACC4(3)];

CS\_00 = AVG\_0\_0\_LEV\_00\*rho\_B0[OPS\_ACC3(-2)] + AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(-2)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(-2)];

CS\_01 = AVG\_0\_0\_LEV\_00\*rho\_B0[OPS\_ACC3(-1)] + AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(-1)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(-1)];

CS\_02 = AVG\_0\_0\_LEV\_00\*rho\_B0[OPS\_ACC3(0)] + AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(0)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(0)];

CS\_03 = AVG\_0\_0\_LEV\_00\*rho\_B0[OPS\_ACC3(1)] + AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(1)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(1)];

CS\_04 = AVG\_0\_0\_LEV\_00\*rho\_B0[OPS\_ACC3(2)] + AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(2)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(2)];

CS\_05 = AVG\_0\_0\_LEV\_00\*rho\_B0[OPS\_ACC3(3)] + AVG\_0\_0\_LEV\_01\*rhou0\_B0[OPS\_ACC0(3)] +

AVG\_0\_0\_LEV\_02\*rhoE\_B0[OPS\_ACC5(3)];

CS\_10 = AVG\_0\_0\_LEV\_10\*rho\_B0[OPS\_ACC3(-2)] + AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(-2)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(-2)];

CS\_11 = AVG\_0\_0\_LEV\_10\*rho\_B0[OPS\_ACC3(-1)] + AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(-1)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(-1)];

CS\_12 = AVG\_0\_0\_LEV\_10\*rho\_B0[OPS\_ACC3(0)] + AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(0)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(0)];

CS\_13 = AVG\_0\_0\_LEV\_10\*rho\_B0[OPS\_ACC3(1)] + AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(1)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(1)];

CS\_14 = AVG\_0\_0\_LEV\_10\*rho\_B0[OPS\_ACC3(2)] + AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(2)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(2)];

CS\_15 = AVG\_0\_0\_LEV\_10\*rho\_B0[OPS\_ACC3(3)] + AVG\_0\_0\_LEV\_11\*rhou0\_B0[OPS\_ACC0(3)] +

AVG\_0\_0\_LEV\_12\*rhoE\_B0[OPS\_ACC5(3)];

CS\_20 = AVG\_0\_0\_LEV\_20\*rho\_B0[OPS\_ACC3(-2)] + AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(-2)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(-2)];

CS\_21 = AVG\_0\_0\_LEV\_20\*rho\_B0[OPS\_ACC3(-1)] + AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(-1)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(-1)];

CS\_22 = AVG\_0\_0\_LEV\_20\*rho\_B0[OPS\_ACC3(0)] + AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(0)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(0)];

CS\_23 = AVG\_0\_0\_LEV\_20\*rho\_B0[OPS\_ACC3(1)] + AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(1)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(1)];

CS\_24 = AVG\_0\_0\_LEV\_20\*rho\_B0[OPS\_ACC3(2)] + AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(2)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(2)];

CS\_25 = AVG\_0\_0\_LEV\_20\*rho\_B0[OPS\_ACC3(3)] + AVG\_0\_0\_LEV\_21\*rhou0\_B0[OPS\_ACC0(3)] +

AVG\_0\_0\_LEV\_22\*rhoE\_B0[OPS\_ACC5(3)];

max\_lambda\_0\_00 = fmax(fabs(a\_B0[OPS\_ACC2(3)] - u0\_B0[OPS\_ACC4(3)]), fmax(fabs(a\_B0[OPS\_ACC2(-2)] -

u0\_B0[OPS\_ACC4(-2)]), fmax(fabs(a\_B0[OPS\_ACC2(-1)] - u0\_B0[OPS\_ACC4(-1)]), fmax(fabs(a\_B0[OPS\_ACC2(0)] -

u0\_B0[OPS\_ACC4(0)]), fmax(fabs(a\_B0[OPS\_ACC2(1)] - u0\_B0[OPS\_ACC4(1)]), fabs(a\_B0[OPS\_ACC2(2)] -

u0\_B0[OPS\_ACC4(2)]))))));

max\_lambda\_0\_11 = fmax(fabs(u0\_B0[OPS\_ACC4(2)]), fmax(fabs(u0\_B0[OPS\_ACC4(3)]), fmax(fabs(u0\_B0[OPS\_ACC4(0)]),

fmax(fabs(u0\_B0[OPS\_ACC4(-2)]), fmax(fabs(u0\_B0[OPS\_ACC4(-1)]), fabs(u0\_B0[OPS\_ACC4(1)]))))));

max\_lambda\_0\_22 = fmax(fabs(a\_B0[OPS\_ACC2(-2)] + u0\_B0[OPS\_ACC4(-2)]), fmax(fabs(a\_B0[OPS\_ACC2(-1)] +

u0\_B0[OPS\_ACC4(-1)]), fmax(fabs(a\_B0[OPS\_ACC2(1)] + u0\_B0[OPS\_ACC4(1)]), fmax(fabs(a\_B0[OPS\_ACC2(3)] +

u0\_B0[OPS\_ACC4(3)]), fmax(fabs(a\_B0[OPS\_ACC2(0)] + u0\_B0[OPS\_ACC4(0)]), fabs(a\_B0[OPS\_ACC2(2)] +

u0\_B0[OPS\_ACC4(2)]))))));

beta\_0 = (rc1)\*pow((rc0)\*(CF\_01 + CS\_01\*max\_lambda\_0\_00) - rc0\*(CF\_03 + CS\_03\*max\_lambda\_0\_00), 2) +

(rc2)\*pow((rc0)\*(CF\_01 + CS\_01\*max\_lambda\_0\_00) - (CF\_02 + CS\_02\*max\_lambda\_0\_00) + (rc0)\*(CF\_03 +

CS\_03\*max\_lambda\_0\_00), 2);

beta\_1 = (rc2)\*pow((rc0)\*(CF\_02 + CS\_02\*max\_lambda\_0\_00) - (CF\_03 + CS\_03\*max\_lambda\_0\_00) + (rc0)\*(CF\_04 +

CS\_04\*max\_lambda\_0\_00), 2) + (rc1)\*pow((rc3)\*(CF\_02 + CS\_02\*max\_lambda\_0\_00) - 2\*(CF\_03 + CS\_03\*max\_lambda\_0\_00) +

(rc0)\*(CF\_04 + CS\_04\*max\_lambda\_0\_00), 2);

beta\_2 = (rc1)\*pow((rc0)\*(CF\_00 + CS\_00\*max\_lambda\_0\_00) - 2\*(CF\_01 + CS\_01\*max\_lambda\_0\_00) + (rc3)\*(CF\_02 +

CS\_02\*max\_lambda\_0\_00), 2) + (rc2)\*pow((rc0)\*(CF\_00 + CS\_00\*max\_lambda\_0\_00) - (CF\_01 + CS\_01\*max\_lambda\_0\_00) +

(rc0)\*(CF\_02 + CS\_02\*max\_lambda\_0\_00), 2);

alpha\_0 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_0), 6);

alpha\_1 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_1), 6);

alpha\_2 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_2), 6);

inv\_alpha\_sum = 1.0/(alpha\_0 + alpha\_1 + alpha\_2);

delta\_0 = ((alpha\_0\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_1 = ((alpha\_1\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_2 = ((alpha\_2\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

inv\_omega\_sum = 1.0/((rc4)\*delta\_0 + (rc5)\*delta\_1 + (rc6)\*delta\_2);

Recon\_0 = Recon\_0 + (rc4)\*delta\_0\*inv\_omega\_sum\*(-rc7\*(CF\_01 + CS\_01\*max\_lambda\_0\_00) + (rc8)\*(CF\_02 +

CS\_02\*max\_lambda\_0\_00) + (rc9)\*(CF\_03 + CS\_03\*max\_lambda\_0\_00)) + (rc5)\*delta\_1\*inv\_omega\_sum\*((rc9)\*(CF\_02 +

CS\_02\*max\_lambda\_0\_00) + (rc8)\*(CF\_03 + CS\_03\*max\_lambda\_0\_00) - rc7\*(CF\_04 + CS\_04\*max\_lambda\_0\_00)) +

(rc6)\*delta\_2\*inv\_omega\_sum\*((rc9)\*(CF\_00 + CS\_00\*max\_lambda\_0\_00) - rc10\*(CF\_01 + CS\_01\*max\_lambda\_0\_00) +

(rc11)\*(CF\_02 + CS\_02\*max\_lambda\_0\_00));

beta\_0 = (rc1)\*pow((rc0)\*(CF\_02 - CS\_02\*max\_lambda\_0\_00) - rc0\*(CF\_04 - CS\_04\*max\_lambda\_0\_00), 2) +

(rc2)\*pow((rc0)\*(CF\_02 - CS\_02\*max\_lambda\_0\_00) - (CF\_03 - CS\_03\*max\_lambda\_0\_00) + (rc0)\*(CF\_04 -

CS\_04\*max\_lambda\_0\_00), 2);

beta\_1 = (rc1)\*pow((rc0)\*(CF\_01 - CS\_01\*max\_lambda\_0\_00) - 2\*(CF\_02 - CS\_02\*max\_lambda\_0\_00) + (rc3)\*(CF\_03 -

CS\_03\*max\_lambda\_0\_00), 2) + (rc2)\*pow((rc0)\*(CF\_01 - CS\_01\*max\_lambda\_0\_00) - (CF\_02 - CS\_02\*max\_lambda\_0\_00) +

(rc0)\*(CF\_03 - CS\_03\*max\_lambda\_0\_00), 2);

beta\_2 = (rc2)\*pow((rc0)\*(CF\_03 - CS\_03\*max\_lambda\_0\_00) - (CF\_04 - CS\_04\*max\_lambda\_0\_00) + (rc0)\*(CF\_05 -

CS\_05\*max\_lambda\_0\_00), 2) + (rc1)\*pow((rc3)\*(CF\_03 - CS\_03\*max\_lambda\_0\_00) - 2\*(CF\_04 - CS\_04\*max\_lambda\_0\_00) +

(rc0)\*(CF\_05 - CS\_05\*max\_lambda\_0\_00), 2);

alpha\_0 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_0), 6);

alpha\_1 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_1), 6);

alpha\_2 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_2), 6);

inv\_alpha\_sum = 1.0/(alpha\_0 + alpha\_1 + alpha\_2);

delta\_0 = ((alpha\_0\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_1 = ((alpha\_1\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_2 = ((alpha\_2\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

inv\_omega\_sum = 1.0/((rc4)\*delta\_0 + (rc5)\*delta\_1 + (rc6)\*delta\_2);

Recon\_0 = Recon\_0 + (rc4)\*delta\_0\*inv\_omega\_sum\*((rc9)\*(CF\_02 - CS\_02\*max\_lambda\_0\_00) + (rc8)\*(CF\_03 -

CS\_03\*max\_lambda\_0\_00) - rc7\*(CF\_04 - CS\_04\*max\_lambda\_0\_00)) + (rc5)\*delta\_1\*inv\_omega\_sum\*(-rc7\*(CF\_01 -

CS\_01\*max\_lambda\_0\_00) + (rc8)\*(CF\_02 - CS\_02\*max\_lambda\_0\_00) + (rc9)\*(CF\_03 - CS\_03\*max\_lambda\_0\_00)) +

(rc6)\*delta\_2\*inv\_omega\_sum\*((rc11)\*(CF\_03 - CS\_03\*max\_lambda\_0\_00) - rc10\*(CF\_04 - CS\_04\*max\_lambda\_0\_00) +

(rc9)\*(CF\_05 - CS\_05\*max\_lambda\_0\_00));

beta\_0 = (rc1)\*pow((rc0)\*(CF\_11 + CS\_11\*max\_lambda\_0\_11) - rc0\*(CF\_13 + CS\_13\*max\_lambda\_0\_11), 2) +

(rc2)\*pow((rc0)\*(CF\_11 + CS\_11\*max\_lambda\_0\_11) - (CF\_12 + CS\_12\*max\_lambda\_0\_11) + (rc0)\*(CF\_13 +

CS\_13\*max\_lambda\_0\_11), 2);

beta\_1 = (rc2)\*pow((rc0)\*(CF\_12 + CS\_12\*max\_lambda\_0\_11) - (CF\_13 + CS\_13\*max\_lambda\_0\_11) + (rc0)\*(CF\_14 +

CS\_14\*max\_lambda\_0\_11), 2) + (rc1)\*pow((rc3)\*(CF\_12 + CS\_12\*max\_lambda\_0\_11) - 2\*(CF\_13 + CS\_13\*max\_lambda\_0\_11) +

(rc0)\*(CF\_14 + CS\_14\*max\_lambda\_0\_11), 2);

beta\_2 = (rc1)\*pow((rc0)\*(CF\_10 + CS\_10\*max\_lambda\_0\_11) - 2\*(CF\_11 + CS\_11\*max\_lambda\_0\_11) + (rc3)\*(CF\_12 +

CS\_12\*max\_lambda\_0\_11), 2) + (rc2)\*pow((rc0)\*(CF\_10 + CS\_10\*max\_lambda\_0\_11) - (CF\_11 + CS\_11\*max\_lambda\_0\_11) +

(rc0)\*(CF\_12 + CS\_12\*max\_lambda\_0\_11), 2);

alpha\_0 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_0), 6);

alpha\_1 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_1), 6);

alpha\_2 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_2), 6);

inv\_alpha\_sum = 1.0/(alpha\_0 + alpha\_1 + alpha\_2);

delta\_0 = ((alpha\_0\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_1 = ((alpha\_1\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_2 = ((alpha\_2\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

inv\_omega\_sum = 1.0/((rc4)\*delta\_0 + (rc5)\*delta\_1 + (rc6)\*delta\_2);

Recon\_1 = Recon\_1 + (rc4)\*delta\_0\*inv\_omega\_sum\*(-rc7\*(CF\_11 + CS\_11\*max\_lambda\_0\_11) + (rc8)\*(CF\_12 +

CS\_12\*max\_lambda\_0\_11) + (rc9)\*(CF\_13 + CS\_13\*max\_lambda\_0\_11)) + (rc5)\*delta\_1\*inv\_omega\_sum\*((rc9)\*(CF\_12 +

CS\_12\*max\_lambda\_0\_11) + (rc8)\*(CF\_13 + CS\_13\*max\_lambda\_0\_11) - rc7\*(CF\_14 + CS\_14\*max\_lambda\_0\_11)) +

(rc6)\*delta\_2\*inv\_omega\_sum\*((rc9)\*(CF\_10 + CS\_10\*max\_lambda\_0\_11) - rc10\*(CF\_11 + CS\_11\*max\_lambda\_0\_11) +

(rc11)\*(CF\_12 + CS\_12\*max\_lambda\_0\_11));

beta\_0 = (rc1)\*pow((rc0)\*(CF\_12 - CS\_12\*max\_lambda\_0\_11) - rc0\*(CF\_14 - CS\_14\*max\_lambda\_0\_11), 2) +

(rc2)\*pow((rc0)\*(CF\_12 - CS\_12\*max\_lambda\_0\_11) - (CF\_13 - CS\_13\*max\_lambda\_0\_11) + (rc0)\*(CF\_14 -

CS\_14\*max\_lambda\_0\_11), 2);

beta\_1 = (rc1)\*pow((rc0)\*(CF\_11 - CS\_11\*max\_lambda\_0\_11) - 2\*(CF\_12 - CS\_12\*max\_lambda\_0\_11) + (rc3)\*(CF\_13 -

CS\_13\*max\_lambda\_0\_11), 2) + (rc2)\*pow((rc0)\*(CF\_11 - CS\_11\*max\_lambda\_0\_11) - (CF\_12 - CS\_12\*max\_lambda\_0\_11) +

(rc0)\*(CF\_13 - CS\_13\*max\_lambda\_0\_11), 2);

beta\_2 = (rc2)\*pow((rc0)\*(CF\_13 - CS\_13\*max\_lambda\_0\_11) - (CF\_14 - CS\_14\*max\_lambda\_0\_11) + (rc0)\*(CF\_15 -

CS\_15\*max\_lambda\_0\_11), 2) + (rc1)\*pow((rc3)\*(CF\_13 - CS\_13\*max\_lambda\_0\_11) - 2\*(CF\_14 - CS\_14\*max\_lambda\_0\_11) +

(rc0)\*(CF\_15 - CS\_15\*max\_lambda\_0\_11), 2);

alpha\_0 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_0), 6);

alpha\_1 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_1), 6);

alpha\_2 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_2), 6);

inv\_alpha\_sum = 1.0/(alpha\_0 + alpha\_1 + alpha\_2);

delta\_0 = ((alpha\_0\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_1 = ((alpha\_1\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_2 = ((alpha\_2\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

inv\_omega\_sum = 1.0/((rc4)\*delta\_0 + (rc5)\*delta\_1 + (rc6)\*delta\_2);

Recon\_1 = Recon\_1 + (rc4)\*delta\_0\*inv\_omega\_sum\*((rc9)\*(CF\_12 - CS\_12\*max\_lambda\_0\_11) + (rc8)\*(CF\_13 -

CS\_13\*max\_lambda\_0\_11) - rc7\*(CF\_14 - CS\_14\*max\_lambda\_0\_11)) + (rc5)\*delta\_1\*inv\_omega\_sum\*(-rc7\*(CF\_11 -

CS\_11\*max\_lambda\_0\_11) + (rc8)\*(CF\_12 - CS\_12\*max\_lambda\_0\_11) + (rc9)\*(CF\_13 - CS\_13\*max\_lambda\_0\_11)) +

(rc6)\*delta\_2\*inv\_omega\_sum\*((rc11)\*(CF\_13 - CS\_13\*max\_lambda\_0\_11) - rc10\*(CF\_14 - CS\_14\*max\_lambda\_0\_11) +

(rc9)\*(CF\_15 - CS\_15\*max\_lambda\_0\_11));

beta\_0 = (rc1)\*pow((rc0)\*(CF\_21 + CS\_21\*max\_lambda\_0\_22) - rc0\*(CF\_23 + CS\_23\*max\_lambda\_0\_22), 2) +

(rc2)\*pow((rc0)\*(CF\_21 + CS\_21\*max\_lambda\_0\_22) - (CF\_22 + CS\_22\*max\_lambda\_0\_22) + (rc0)\*(CF\_23 +

CS\_23\*max\_lambda\_0\_22), 2);

beta\_1 = (rc2)\*pow((rc0)\*(CF\_22 + CS\_22\*max\_lambda\_0\_22) - (CF\_23 + CS\_23\*max\_lambda\_0\_22) + (rc0)\*(CF\_24 +

CS\_24\*max\_lambda\_0\_22), 2) + (rc1)\*pow((rc3)\*(CF\_22 + CS\_22\*max\_lambda\_0\_22) - 2\*(CF\_23 + CS\_23\*max\_lambda\_0\_22) +

(rc0)\*(CF\_24 + CS\_24\*max\_lambda\_0\_22), 2);

beta\_2 = (rc1)\*pow((rc0)\*(CF\_20 + CS\_20\*max\_lambda\_0\_22) - 2\*(CF\_21 + CS\_21\*max\_lambda\_0\_22) + (rc3)\*(CF\_22 +

CS\_22\*max\_lambda\_0\_22), 2) + (rc2)\*pow((rc0)\*(CF\_20 + CS\_20\*max\_lambda\_0\_22) - (CF\_21 + CS\_21\*max\_lambda\_0\_22) +

(rc0)\*(CF\_22 + CS\_22\*max\_lambda\_0\_22), 2);

alpha\_0 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_0), 6);

alpha\_1 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_1), 6);

alpha\_2 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_2), 6);

inv\_alpha\_sum = 1.0/(alpha\_0 + alpha\_1 + alpha\_2);

delta\_0 = ((alpha\_0\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_1 = ((alpha\_1\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_2 = ((alpha\_2\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

inv\_omega\_sum = 1.0/((rc4)\*delta\_0 + (rc5)\*delta\_1 + (rc6)\*delta\_2);

Recon\_2 = Recon\_2 + (rc4)\*delta\_0\*inv\_omega\_sum\*(-rc7\*(CF\_21 + CS\_21\*max\_lambda\_0\_22) + (rc8)\*(CF\_22 +

CS\_22\*max\_lambda\_0\_22) + (rc9)\*(CF\_23 + CS\_23\*max\_lambda\_0\_22)) + (rc5)\*delta\_1\*inv\_omega\_sum\*((rc9)\*(CF\_22 +

CS\_22\*max\_lambda\_0\_22) + (rc8)\*(CF\_23 + CS\_23\*max\_lambda\_0\_22) - rc7\*(CF\_24 + CS\_24\*max\_lambda\_0\_22)) +

(rc6)\*delta\_2\*inv\_omega\_sum\*((rc9)\*(CF\_20 + CS\_20\*max\_lambda\_0\_22) - rc10\*(CF\_21 + CS\_21\*max\_lambda\_0\_22) +

(rc11)\*(CF\_22 + CS\_22\*max\_lambda\_0\_22));

beta\_0 = (rc1)\*pow((rc0)\*(CF\_22 - CS\_22\*max\_lambda\_0\_22) - rc0\*(CF\_24 - CS\_24\*max\_lambda\_0\_22), 2) +

(rc2)\*pow((rc0)\*(CF\_22 - CS\_22\*max\_lambda\_0\_22) - (CF\_23 - CS\_23\*max\_lambda\_0\_22) + (rc0)\*(CF\_24 -

CS\_24\*max\_lambda\_0\_22), 2);

beta\_1 = (rc1)\*pow((rc0)\*(CF\_21 - CS\_21\*max\_lambda\_0\_22) - 2\*(CF\_22 - CS\_22\*max\_lambda\_0\_22) + (rc3)\*(CF\_23 -

CS\_23\*max\_lambda\_0\_22), 2) + (rc2)\*pow((rc0)\*(CF\_21 - CS\_21\*max\_lambda\_0\_22) - (CF\_22 - CS\_22\*max\_lambda\_0\_22) +

(rc0)\*(CF\_23 - CS\_23\*max\_lambda\_0\_22), 2);

beta\_2 = (rc2)\*pow((rc0)\*(CF\_23 - CS\_23\*max\_lambda\_0\_22) - (CF\_24 - CS\_24\*max\_lambda\_0\_22) + (rc0)\*(CF\_25 -

CS\_25\*max\_lambda\_0\_22), 2) + (rc1)\*pow((rc3)\*(CF\_23 - CS\_23\*max\_lambda\_0\_22) - 2\*(CF\_24 - CS\_24\*max\_lambda\_0\_22) +

(rc0)\*(CF\_25 - CS\_25\*max\_lambda\_0\_22), 2);

alpha\_0 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_0), 6);

alpha\_1 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_1), 6);

alpha\_2 = pow(1 + fabs(beta\_0 - beta\_2)/(eps + beta\_2), 6);

inv\_alpha\_sum = 1.0/(alpha\_0 + alpha\_1 + alpha\_2);

delta\_0 = ((alpha\_0\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_1 = ((alpha\_1\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

delta\_2 = ((alpha\_2\*inv\_alpha\_sum < TENO\_CT) ? (

0.0

)

: (

1.0

));

inv\_omega\_sum = 1.0/((rc4)\*delta\_0 + (rc5)\*delta\_1 + (rc6)\*delta\_2);

Recon\_2 = Recon\_2 + (rc4)\*delta\_0\*inv\_omega\_sum\*((rc9)\*(CF\_22 - CS\_22\*max\_lambda\_0\_22) + (rc8)\*(CF\_23 -

CS\_23\*max\_lambda\_0\_22) - rc7\*(CF\_24 - CS\_24\*max\_lambda\_0\_22)) + (rc5)\*delta\_1\*inv\_omega\_sum\*(-rc7\*(CF\_21 -

CS\_21\*max\_lambda\_0\_22) + (rc8)\*(CF\_22 - CS\_22\*max\_lambda\_0\_22) + (rc9)\*(CF\_23 - CS\_23\*max\_lambda\_0\_22)) +

(rc6)\*delta\_2\*inv\_omega\_sum\*((rc11)\*(CF\_23 - CS\_23\*max\_lambda\_0\_22) - rc10\*(CF\_24 - CS\_24\*max\_lambda\_0\_22) +

(rc9)\*(CF\_25 - CS\_25\*max\_lambda\_0\_22));

wk0\_B0[OPS\_ACC7(0)] = Recon\_0 + Recon\_1 + Recon\_2;

wk1\_B0[OPS\_ACC8(0)] = AVG\_0\_u0\*Recon\_1 + Recon\_0\*(-AVG\_0\_a + AVG\_0\_u0) + Recon\_2\*(AVG\_0\_a + AVG\_0\_u0);

wk2\_B0[OPS\_ACC6(0)] = (rc0)\*pow(AVG\_0\_u0, 2)\*Recon\_1 + Recon\_0\*(rcinv12\*pow(AVG\_0\_a, 2) - AVG\_0\_a\*AVG\_0\_u0 +

(rc0)\*pow(AVG\_0\_u0, 2)) + Recon\_2\*(rcinv12\*pow(AVG\_0\_a, 2) + AVG\_0\_a\*AVG\_0\_u0 + (rc0)\*pow(AVG\_0\_u0, 2));

}

void opensbliblock00Kernel006(const double \*wk2\_B0, const double \*wk0\_B0, const double \*wk1\_B0, double \*Residual1\_B0,

double \*Residual0\_B0, double \*Residual2\_B0)

{

Residual0\_B0[OPS\_ACC4(0)] = -rcinv13\*(-wk0\_B0[OPS\_ACC1(-1)] + wk0\_B0[OPS\_ACC1(0)]);

Residual1\_B0[OPS\_ACC3(0)] = -rcinv13\*(-wk1\_B0[OPS\_ACC2(-1)] + wk1\_B0[OPS\_ACC2(0)]);

Residual2\_B0[OPS\_ACC5(0)] = -rcinv13\*(wk2\_B0[OPS\_ACC0(0)] - wk2\_B0[OPS\_ACC0(-1)]);

}

void opensbliblock00Kernel011(const double \*Residual2\_B0, const double \*Residual0\_B0, const double \*Residual1\_B0,

double \*rho\_B0, double \*tempRK\_rhou0\_B0, double \*rhou0\_B0, double \*tempRK\_rhoE\_B0, double \*tempRK\_rho\_B0, double

\*rhoE\_B0, const double \*rkB, const double \*rkA)

{

tempRK\_rho\_B0[OPS\_ACC7(0)] = dt\*Residual0\_B0[OPS\_ACC1(0)] + rkA[0]\*tempRK\_rho\_B0[OPS\_ACC7(0)];

rho\_B0[OPS\_ACC3(0)] = rkB[0]\*tempRK\_rho\_B0[OPS\_ACC7(0)] + rho\_B0[OPS\_ACC3(0)];

tempRK\_rhou0\_B0[OPS\_ACC4(0)] = dt\*Residual1\_B0[OPS\_ACC2(0)] + rkA[0]\*tempRK\_rhou0\_B0[OPS\_ACC4(0)];

rhou0\_B0[OPS\_ACC5(0)] = rkB[0]\*tempRK\_rhou0\_B0[OPS\_ACC4(0)] + rhou0\_B0[OPS\_ACC5(0)];

tempRK\_rhoE\_B0[OPS\_ACC6(0)] = dt\*Residual2\_B0[OPS\_ACC0(0)] + rkA[0]\*tempRK\_rhoE\_B0[OPS\_ACC6(0)];

rhoE\_B0[OPS\_ACC8(0)] = rkB[0]\*tempRK\_rhoE\_B0[OPS\_ACC6(0)] + rhoE\_B0[OPS\_ACC8(0)];

}

#endif