Métodos Computacionales Avanzados
EulerCartesian3D

init.h

```
void init_problem(physics_grid *P, U_grid *U, F_grid *F, int problem);
physics_grid * create_physics_grid(void);
U_grid * create_U_grid(void);
F_grid * create_F_grid(void);
```

io.h

```
void print_L(physics_grid *G);
```

problem.h

```
void init_cond(U_grid *U);
void prob_solve(U_grid *U, F_grid *F, physics_grid *P, FLOAT T);
```

solver.h

```
int transform_U(U_grid *U, int pos_x, int pos_y, int pos_z, int prop);
FLOAT extract_rho(U_grid *U, int pos_x, int pos_y, int pos_z);
FLOAT extract_u(U_grid *U, int pos_x, int pos_y, int pos_z, FLOAT rho);
FLOAT extract_v(U_grid *U, int pos_x, int pos_y, int pos_z, FLOAT rho);
FLOAT extract_w(U_grid *U, int pos_x, int pos_y, int pos_z, FLOAT rho);
FLOAT extract_E(U_grid *U, int pos_x, int pos_y, int pos_z, FLOAT rho);
FLOAT calce(FLOAT E, FLOAT u, FLOAT v, FLOAT w);
FLOAT calcp(FLOAT rho, FLOAT e);
FLOAT calch(FLOAT E, FLOAT p, FLOAT rho);
FLOAT calcs(FLOAT h);
FLOAT calcsps(U_grid *U);
FLOAT calcdt(physics_grid *P, FLOAT sps);
int transform_F(F_grid *F, int pos_x, int pos_y, int pos_z, int pos_g, int prop);
void calcF(F_grid *F, U_grid *U, int pos_x, int pos_y, int pos_z);
void newU(U_grid *U, F_grid *F, physics_grid *P, int pos_x, int pos_y, int pos_z, FLOAT dt);
```

struct.h

```
#ifndef STRUCT_H
#define STRUCT_H

#define GAMMA 1.4

#define SEDOV 1
#define NDIM 3

#define PRESSURE 1.0
#define RHO 10.0
#define VX 2.0
#define VY 3.0
#define VZ 4.0

#define FLOAT double
typedef struct physics_grid_str
{
  FLOAT L_x;
  FLOAT L_y;
  FLOAT L_z;
  FLOAT delta_x;
  FLOAT delta_y;
  FLOAT delta_z;
  int N_x;
  int N_y;
  int N_z;
  int N_cells;
  FLOAT *P;
} physics_grid;

typedef struct U_grid_str{
  int N_x;
  int N_y;
  int N_z;
  int N_cells;
  FLOAT *U;
} U_grid;


typedef struct F_grid_str{
  int N_x;
  int N_y;
  int N_z;
  int N_cells;
  FLOAT *F;
} F_grid;


#endif
```