Matrix classes

Matrix — general m x n matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Diagonal — square matrix with non-zero leading diagonal, elsewhere zero

$$\begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_n \end{pmatrix}$$

Vector — column matrix

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix}$$

Symm_Band — square symmetric sparse matrix, consisting of a non zero leading diagonal and non zero bands at fixed distance from the leading diagonal

$$\begin{pmatrix}
2 & -1 & & & -1 & & & & & & & & & & & \\
-1 & 3 & -1 & & & -1 & & & & & & & & & & \\
 & -1 & 3 & -1 & & & -1 & & & & & & & & & \\
 & & -1 & 2 & 0 & & & -1 & & & & & & & & \\
-1 & & & 0 & 3 & -1 & & & -1 & & & & & & & \\
 & -1 & & & -1 & 4 & -1 & & & -1 & & & & & & \\
 & & -1 & & & -1 & 4 & -1 & & & -1 & & & & & \\
 & & & -1 & & & -1 & 3 & 0 & & & -1 & & & & \\
 & & & & -1 & & & 0 & 3 & -1 & & & -1 & & & \\
 & & & & & -1 & & & -1 & 4 & -1 & & & -1 & & \\
 & & & & & & -1 & & & -1 & 4 & -1 & & & -1 & \\
 & & & & & & & -1 & & & -1 & 3 & 0 & & & -1 \\
 & & & & & & & & -1 & & & 0 & 2 & -1 & & \\
 & & & & & & & & & -1 & & & -1 & 3 & -1 & \\
 & & & & & & & & & & -1 & & & -1 & 3 & -1 \\
 & & & & & & & & & & & -1 & & & -1 & 2
\end{pmatrix}$$

Operators

*, +, = between these matrix classes are overloaded, to allow more readable code.

Solvers

Although quite different, these all solve the same matrix equation Ap = b, which can also be thought of as a set of N linear equations expressed in matrix form.

- Gauss-Seidel
- Conjugate Gradients
- Pre-conditioned Conjugate Gradients

They all start with an initial guess p0, and they all continue iteratively until the residual error is less than a supplied tolerance, or a maximum number of iterations is reached.

Grid Classes

FluidQuantity A vector, the same size as the no. of cells in the simulation N, representing a discrete grid of samples of a continuous fluid quantity such as velocity, temperature, smoke density etc.

The FluidQuantity class contains a pointer to the FluidGrid class, below, and obtains information about the grid, such as the time, by asking it directly.

Important functions are:

InterpolateLinear - linearly interpolates the value of the quantity at any point in the grid

Advect - Transports the quantity around the grid under the influence of the velocity field

FluidGrid

This class represents the discrete grid on which the simulation occurs. Contains pointers to all the FluidQuantity objects which exist on the grid:

- *Velocity* (u,v, and w components)
- *Density*
- *Temperature*

Also stores important simulation attributes like the current time, the time of the next cache, the dimensions of the grid, density of the fluid etc.

The grid also defines a vector to store *pressure*, and other matrix objects which relate to the solver.

Important FluidGrid functions include:

BuildLinearSystem() – Sets up the equation Ap = b for the *Project* function

setDeltaT - Calculate the time step to be used in the simulation

addSmoke - injects *Density* and *Temperature* locally into the grid

Project - Iteratively solves the *pressure* necessary to satisfy the incompressibility condition, then updates the *velocity* using these *pressure* values.

Finally, the most important function defined by FluidGrid is Update, which performs the following steps:

```
void FluidGrid::Update(){

      setDeltaT();

      Advect();

      AddForces();
```

```
        addSmoke(0.25, 0.1, 0.45, 0.1, 0.05, 0.1,
                    0.5, 450);

        CalculateVolumes();

        BuildLinearSystem();
        MIC0precon(*A, *Ei);

        Project();

        currtime += deltaT;

        if (writetocache){
        WriteToCache();
        writetocache = false;
        }

}
```