transCnt: Tutorial

Getting Started

How to compile this library

To use this library, you need to install a few packages:

```
sudo apt update && sudo apt install libboost-all-dev gnuplot -y
```

Assume you have the all the code in `transcnt` directory and `g++ 6.2+` installed:

```
cd transcnt
git clone --recursive \
git@github.com:dstahlke/gnuplot-iostream.git
make all
```

How to run test program

To add the shared library to your PATH:

```
export LD_LIBRARY_PATH=$(pwd):$LD_LIBRARY_PATH
```

To run the test program:

```
make test
./test
```

How to use this library in user program

Include `#include "calculation.h"` in your files.

And compile with:

```
$(CXX) $(CPPFLAGS) -L. -ltranscnt app.cpp -o app \
-lboost_iostreams -lboost_system -lboost_filesystem
```

Or customize the `Makefile`.

Calculate Temperature at a Specific Point

1. Specify a `Geometry`

To create a geometry, you need to provide the dimension of the geometry, the material name, and the initial temperature of the geometry in Kelvin. The dimension could be different for different geometries. For example, the constructors for `Sphere` and `RectBar` are respectively:

```
Sphere(Dim radius, string mat, Kelvin t_init);
RectBar(Dim L_1, Dim L_2, Dim L_3, string mat, Kelvin t_init);
```

`Dim` and `Kelvin` are type aliases to `float`. Note that currently our library only supports three materials: `st` for steel, `al` for aluminum, and `egg` for egg. If you want to use a custom material, there is another constructor for each geometry that requires `p` (mass density, in kg/m^3), `c` (heat capacity, in J/K), `k` (conduction coefficient, in W/m*K) values instead of the material name. For example:

```
Sphere(Dim radius, float k, float c, float p, Kelvin t_init);
```

2. Specify a `Point`

The second step is to create a `Point`. You need to provide the location of the point and the time. The location should correspond to the dimension of the geometry where the point resides. For example, the constructors for sphere point and rectangular bar point are respectively:

```
SpherePoint(Loc sphere_loc, Secs secs);
RectBarPoint(Loc rect_loc1, Loc rect_loc2, Loc rect_loc3, Secs
secs);
```

3. Specify an `EnvMat` (environment)

To create an environment, you should provide the name of the environment material and the temperature. Our library currently supports two environment materials: `air` and `water`. If you want your own environment, you can use the other constructor, and you will need to provide `h` (heat transfer coefficient, in W/m^2*K) instead of the material name. The constructors for `EnvMat` are:

```
EnvMat(string envmat, Kelvin t_inf);
EnvMat(float h, Kelvin t_inf);
```

After you have a geometry, a point and an environment, you are ready to use the `temp_at_point` function! This function will calculate the temperature of the point at given conditions, and it will store the temperature inside the point you give. The function declarations are:

```
void temp_at_point(PlaneWall &w, PlaneWallPoint &p, EnvMat &envmat);
void temp_at_point(Sphere &s, SpherePoint &p, EnvMat &envmat);
void temp_at_point(InfCylinder &icyl, InfCylinderPoint &p, EnvMat
&envmat);
void temp_at_point(RectBar &rb, RectBarPoint &p, EnvMat &envmat);
void temp_at_point(Cylinder &cyl, CylinderPoint &p, EnvMat &envmat);
void temp_at_point(InfRectBar &irb, InfRectBarPoint &p, EnvMat
&envmat);
```

To access the temperature afterwards, simply call `p.temp()`.

USE CASE:

```
PlaneWall w = PlaneWall(0.1, "st", 500);
PlaneWallPoint p = PlaneWallPoint(0.05, 100);
Envmat envmat = EnvMat("water", 300);
temp_at_point(w, p, envmat);
cout << p.temp() << endl;
```

Calculate Temperature on Points Lying on Mesh

To calculate temperature on a mesh, you need to provide a geometry, time, mesh density and environment. The function will divide each dimension of your geometry into (`mesh_density + 1`) regions along each dimension. For example, a mesh density of 10 on a `RectBar` will result in 1,000 regions and a total of 1,331 points (11^3). A `.csv` file named with an abbreviation for the geometry used will be generated in your working directory. The function declarations are:

```
void temp_on_mesh(PlaneWall &w, Secs secs, int mesh_density, EnvMat &envmat);
void temp_on_mesh(InfCylinder &icyl, Secs secs, int mesh_density, EnvMat
&envmat);
void temp_on_mesh(Sphere &s, Secs secs, int mesh_density, EnvMat &envmat);
void temp_on_mesh(InfRectBar &irb, Secs secs, int mesh_density, EnvMat
&envmat);
void temp_on_mesh(Cylinder &cyl, Secs secs, int mesh_density, EnvMat &envmat);
void temp_on_mesh(RectBar &rb, Secs secs, int mesh_density, EnvMat &envmat);
```

USE CASE:

```
RectBar rb = RectBar(.09, .1, .11, “st”, 500);
Envmat envmat = EnvMat("water", 300);
temp_on_mesh(rb, 100, 20, envmat);
cout << rb.temp_dist()[5].temp() << endl;
```

Calculate Avg Temperature

To calculate the average temperature of the geometry at a specific time, use the following functions:

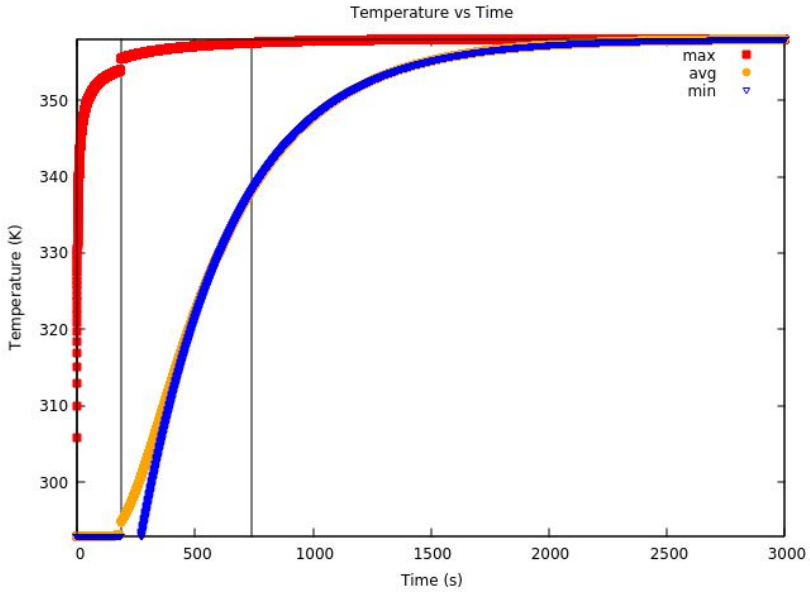
```
float avg_temp_at_time(Sphere &s, Secs time, EnvMat &envmat);
float avg_temp_at_time(PlaneWall &s, Secs time, EnvMat &envmat);
float avg_temp_at_time(InfCylinder &s, Secs time, EnvMat &envmat);
float avg_temp_at_time(Cylinder &s, Secs time, EnvMat &envmat);
float avg_temp_at_time(RectBar &s, Secs time, EnvMat &envmat);
float avg_temp_at_time(InfRectBar &s, Secs time, EnvMat &envmat);
```

The return value is the average temperature at the specified time. `Secs` is a type alias to `float`.

USE CASE:

```
InfCylinder ic = InfCylinder(0.1, “st”, 500);
Envmat envmat = EnvMat("water", 300);
cout << “avg temperature is ” << avg_temp_at_time(ic, 10, envmat) << endl;
```

Temperature vs. Time Plot



To plot geometries at a specific time, you must provide a Geometry, a start time, an end time, an interval to sample times at, and an environment. Plots can be generated both for an entire object as well as for specific points. Plots of an entire object display the object's minimum, maximum, and average temperature at all points and at the specified rate. In addition to the data, vertical lines are drawn indicating the transition between different methods of temperature approximation. Use the following functions:

```
void plot(Sphere &s, Secs start, Secs end, Secs intrv, EnvMat &envmat);
void plot(PlaneWall &s, Secs start, Secs end, Secs intrv, EnvMat &envmat);
void plot(InfCylinder &s, Secs start, Secs end, Secs intrv, EnvMat &envmat);
void plot(Cylinder &s, Secs start, Secs end, Secs intrv, EnvMat &envmat);
void plot(RectBar &s, Secs start, Secs end, Secs intrv, EnvMat &envmat);
void plot(InfRectBar &s, Secs start, Secs end, Secs intrv, EnvMat &envmat);

void plot(Sphere &s, vector<SpherePoint> &p, Secs start, Secs end, Secs intrv,
EnvMat &envmat);
void plot(PlaneWall &s, vector<PlaneWallPoint> &p, Secs start, Secs end, Secs
intrv, EnvMat &envmat);
void plot(InfCylinder &s, vector<InfCylinderPoint> &p, Secs start, Secs end,
Secs intrv, EnvMat &envmat);
void plot(Cylinder &s, vector<CylinderPoint> &p, Secs start, Secs end, Secs
intrv, EnvMat &envmat);
void plot(RectBar &s, vector<RectBarPoint> &p, Secs start, Secs end, Secs
intrv, EnvMat &envmat);
```

```
void plot(InfRectBar &s, vector<InfRectBarPoint> &p, Secs start, Secs end, Secs
intrv, EnvMat &envmat);
```

Temperature Conversion

We also provide four utility functions to convert Kelvin temperature to Fahrenheit and Celsius, and the other way around:

```
float kelvin_to_fahrenheit(Kelvin k);
float kelvin_to_celcius(Kelvin k);
Kelvin  fahrenheit_to_kelvin(float f);
Kelvin celcius_to_kelvin(float c);
```

USE CASE:

```
     PlaneWall w = PlaneWall(0.1, “st”, 500);
     PlaneWallPoint p = PlaneWallPoint(0.05, 100);
     Envmat envmat = EnvMat("water", 300);
     temp_at_point(w, p, envmat);
     cout << kelvin_to_celcius(p.temp()) << “ degrees in celcius” <<
endl;
```