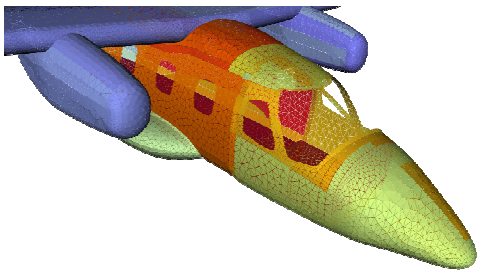
P Sewell Jan. 2014

H63ESD Engineering Software: Design and Implementation

COURSEWORK (100% of the Module Assessment)



Coursework Deadlines And University Regulations **2**
Final Submission deadline: Friday 13th May 2016 ..... *2*
Submission format 2
Required Platform for the Software 2
Course work style 2
Your budget 3
Progress Reporting 3
Friday 26th February: *3*
Friday 18th March: *3*
Friday 22nd April: *3*
Plagiarism 3
Coursework Assignment: part 1, 75%, Overview and Assessment Criteria *4*
Coursework Assignment: part 2, 25%, Overview and Assessment Criteria *4*

Scenario: (Your client's field of business) **6**
Delaunay Triangulations *8*
Numerical issues for Producing Delaunay Triangulations *9*
Speed issues for Producing Delaunay Triangulations - Parallelisation? *9*

Specific Coursework Tasks **10**
Part 1, Task 1: (10%) Code optimisation *10*
Part 1, Task 2: (10%) Serial Characterisation of the Jenna platform *11*
Part 1, Task 3: (10%) An Interval Mathematics Component *11*
Part 1, Task 4: (15%) Manipulation of Triangulated Meshes *12*
Part 1, Task 5: (10%) Basic OpenMP Parallelisation *12*
Part 1, Task 6: (10%) Basic MPI Parallelisation *12*
Part 2: (25%): Delaunay Triangulation *13*
Progress Reporting and Time Management: (10%) *13*

Appendix #1: Linear Interpolation Over Individual Triangles **14**

Appendix #2: The Circumcentre of Individual Triangles **15**

Appendix #3: Use Of The Exact Maths Library, GMP **15**

Appendix #4: File Formats **16**

Appendix #5: Code for task#1 ..... **17**

Coursework Deadlines And University Regulations

Final Submission deadline: Friday 13th May 2016

Submission format

SUBMISSION VIA

1) MOODLE - A zipped file containing all your work, programs AND a PDF of the report as detailed below.

The report is not to contain general background material, just the required responses to the questions and design tasks below.

Usual Faculty submission procedures and University rules regarding late submission apply.

Required Platform for the Software

All software MUST be in C++ and compile with g++ on the Jenna cluster and all *required* timings and memory characterisations must be undertaken upon Jenna : this is part of the Design Specification.

You are permitted to use the STL, but no other 3rd party code or libraries (e.g. Boost) without prior permission of P Sewell.

WARNING: failure to follow these requirements will cost you a notable amount of marks as I (the client) may not be able to assess your work in proper comparison to that of other students and my own codes.

Course work style

Consider the coursework in the following manner. P Sewell wishes to place a contract to undertake a major design and development project with a software developer. He is the *client.* In order to choose to whom to award the work, he has produced a "test" that all hopeful developers must undertake in order to demonstrate their skills. Therefore, as is not uncommon in such circumstances, the test may be:

1) Vague in places
2) Possibly open-ended - where to stop?
3) Lacking in some crucial details

You are expected to interact with your client - ask questions!

You are expected to make your own engineering judgements where the specification and/or the client is unclear on, or else simply doesn't explicitly mention, a particular aspect (e.g. expandability, robustness, the use of a 3rd party library etc).

For this reason there is no overall page limit on the report - again, your engineering judgement on the appropriate style of presentation given the context.

Your budget

A 10 credit module = 100 hours of student time. 22 hours lectures and you will also have to undertake general study of the module material: spend the budget wisely.

You will not be penalised for asking P Sewell questions or seeking any other form of help from him - you are supposed/expected to!

Progress Reporting

As is the case for all real projects, the client (PS) requires to be able to monitor progress. For this reason, you are required to submit a <1 page summary of your activities and/or project progress via Moodle on the following dates. **These are compulsory and are awarded a mark (see below).** The intention of these progress reports is partly to replicate a true industrial atmosphere, but primarily is **to permit PS to *help you***. These are not marked on technical content, but only from the point of view of project management.

Friday 26th February:

PS would expect to hear about study of the lecture material and initial coursework progress

Friday 18th March:

PS would expect to hear about some initial coursework *results* for part 1 and design thoughts for part2

Friday 22nd April:

PS Would expect to hear about the design work for part 2 and any initial implementation/tests.

PS reserves the right to interview any student about their progress reports and to see demonstrations.

Plagiarism

Please refer to the University's rules on plagiarism and take them very seriously - The Faculty will

For this coursework, it is expected that students will talk to each other about the work and even jointly investigate particular avenues of interest and exchange ideas - that's how engineers operate. However, **this is not a group project.** Each student must design and implement their own work separately.

If in doubt please ask PS

There Are Two Parts To This Coursework: All Tasks In Both Parts Are Compulsory.

Both parts of the coursework are focussed upon the same engineering scenario (except for task #1).

Coursework Assignment: part 1, 75%, Overview and Assessment Criteria

This assesses your knowledge and understanding of, and your ability to use, the techniques we have studied on the smaller scale. There are a number of separate tasks each of which considers a particular aspect of the material.

Marks will be allocated on the basis of:

1. Does the code compile properly without ignored warnings?
2. Does the code work properly, i.e. does it produce the correct behaviour?
3. Is the code robust?
4. Is the code efficient - speed/memory?
5. Is the code clearly understandable to our level of software engineer (by the end of this module!): layout, structure, an appropriate level of documentation ?
6. Has the code been designed with a view to future updates/modifications as well as to permit recycling of its parts for other projects?
7. The technical correctness of the required explanations.
8. Demonstration of good engineering practice and judgement.

Coursework Assignment: part 2, 25%, Overview and Assessment Criteria

This is a larger scale design, implement and test study. There is no unique answer! Credit will be given for consideration of the *structure* of the solution, and is *less* concerned (but is not unconcerned) with whether all features properly compile and run: do not be afraid to leave out voluminous *implementation* detail in favour of spending time getting the design and structure correct. However, be careful about the balance - you must convince the *client* of both your design and implementation skills *within a given time frame* - so actually your project management skills are also being assessed.

I expect to see commentary in the code and the report explaining why design choices have been made - I will value seeing the results of any experimentation that support your choices - even where the alternatives proved inferior I want to credit you for considering and assessing them.

The marking criteria encompass the following points:

1. Design: does the overall strategy attempted offer a good design solution?
2. Implementation: is the code well implemented in terms of robustness, maintainability, portability, good engineering practice?
3. Does the code "function" properly?
4. Have you adequately tested your code ? Proof? Are your engineering decisions defensible?
5. Performance: speed and memory.
6. Documentation: suitable for the purpose - adequacy without excess?

7. Credit will be given for correct and appropriate use of material that students discover themselves beyond the lecture material - "impress the client".

If in doubt on any aspect of the criteria used to assess your work for a particular task, please ask PS

The Scenario is described below. No expertise is expected in the area, but as software engineers we will often have to *understand the problem area* before we can start our own work.

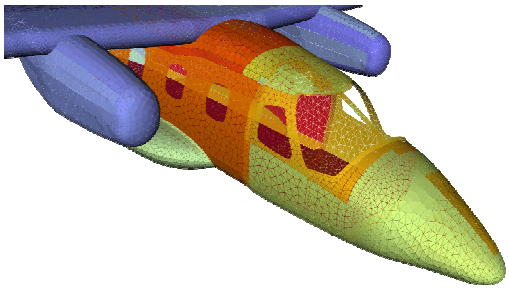
When you turn over the problem is going to look complicated!

That's the whole point! - Divide and conquer!

IF IN DOUBT - ASK THE CLIENT

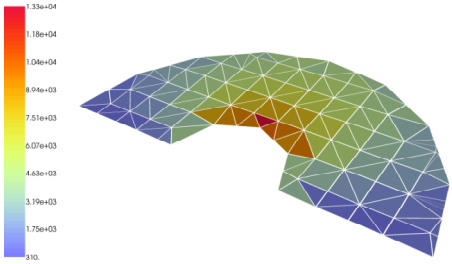
Scenario: (Your client's field of business)

Many engineering disciplines use CAD/CAE packages which require models of geometrical structures.



Often, structures are modelled by a triangulated approximation to their surfaces. Triangles offer the ability to represent small and fine features with different sized triangles.

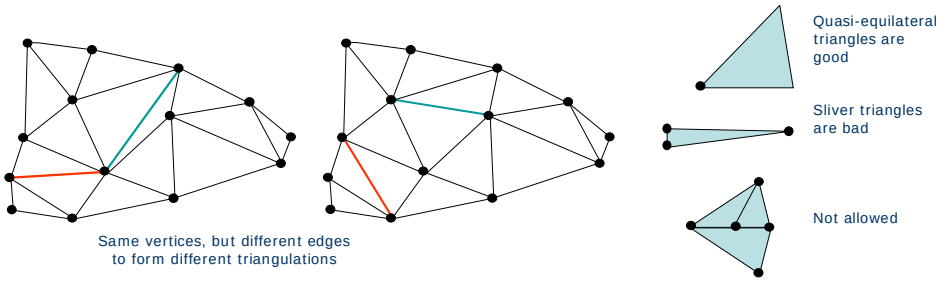
Similarly, we might discretise a flat 2D problem space in order to numerically solve some equations representing a physical process of interest - maybe current flow or temperature. We might visualise this as shown here using a colour map or as shown below using a "height" surface.



In this work we are concerned with the second type of triangulation (also referred to as a *mesh)*

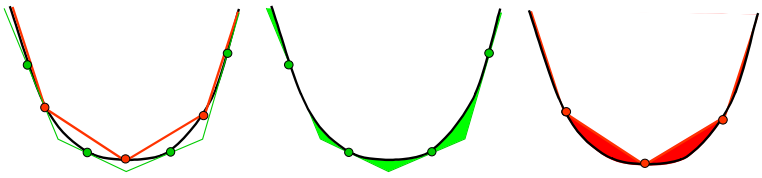
We can "triangulate" a particular problem with <u>given fixed vertices</u> in many different ways. However, we will inevitably ask, or be asked: *What is the "best quality" triangulation?*

Note that a triangulation of given vertices is not unique: "any edge can be flipped".



The definition of a "good" mesh obviously depends upon its intended use. For example, people solving a set of engineering equations are likely to prioritise different criteria than the computer games/animation community.

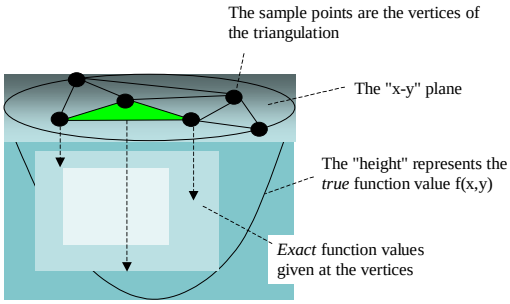
A 1D Interpolation Error Measure: One interesting (meaningful?) measure of *triangulation quality* is to consider the interpolation of a function over the mesh - how close is the interpolated sampled function to the true continuous one?



Above is shown a 1D "mesh" example: How close are the Red and Green approximations to the true black curve? We can consider the "area" of the solid green and red regions as *error measures.*

A 2D Interpolation Error Measure:

This idea also extends to 2D meshes - albeit it is harder to draw!



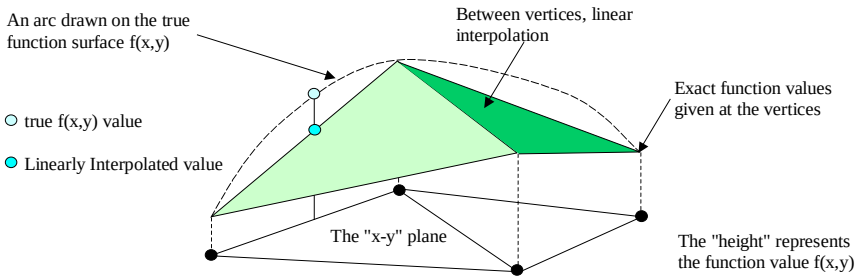
Over a 2D x-y domain we are given a smooth continuous function f(x,y) that we need to sample it in an analogous manner to the 1D case above.

We select some discrete sample points $\{x_i, y_i\}$ for $0<i<N$.

We then join the samples to form a set of triangles t_j for $0<j<T$.

Each triangle defines a locally flat approximation to the function f(x,y).

At any point in each triangle we estimate f(x) by linear interpolation of its known values at the triangle's vertices (our sample points).



Let the linear interpolation approximation to f(x,y) in each triangle t_j be $F_j(x,y)$.

(See appendix #1 for the details of this)

We can define a measure of error, i.e. how good is the interpolation, as follows:

There is a "volume" difference between the true and the interpolated surfaces (i.e. analogous to the red or green regions of the 1D case). This provides a useful measure of error.

It is easiest to evaluate this error, locally, for each triangle and them sum to get the total error.

Local triangle error $E_j = \int_{t_j} dA_j \left| f(x,y) - F_j(x,y) \right|$ where the integral is over the area of t_j.

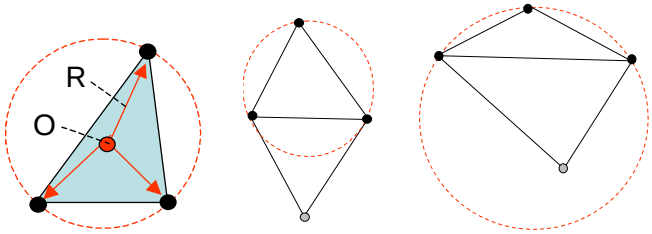
Therefore the global error $E = \sum_j \int_{t_j} dA_j \left| f(x,y) - F_j(x,y) \right|$.

The question is then, which particular triangulation minimises *E* for a given set of vertices (sample points)?

It can be shown (but not here) that there is a particular (unique-ish) choice of triangulation that minimises the interpolation error, the so called *Delaunay Triangulation.* This is what we will generate.

Delaunay Triangulations

A Delaunay Triangulation, DT, is defined to be the triangulation where the circumcircle of each triangle is empty, that is it contains no vertices except its own. The circumcircle of a triangle is that circle that passes though its vertices: there are known formulae to get the circumcentre, O, and circumradius, R, from the vertex coordinates. See appendix #2.



Left Delaunay, right non-Delaunay

Generating Delaunay Triangulations

There are a number of algorithms for connecting a given set of vertices to produce a DT. We will use the *incremental Bowyer-Watson* algorithm. We do not need to understand how it works, its just a recipe.

The algorithm inserts the vertices one at a time. We usually start with one "large" triangle big enough to enclose the whole problem and when all our points have been inserted, delete the large triangle's vertices leaving just what we want.

Consider an existing Delaunay mesh and we want to insert a new vertex and re-triangulate: it turns out that this is a local operation.

A number of design issues to consider:

1) In which existing triangle does the new vertex fall: ***fast search needed?***

2) From this seed triangle we need a search over its adjacent triangles and then their neighbours etc until sure *all* triangles whose circumcircles enclose the vertex are found. ***fast search needed?***

3) We need a fast and accurate (see below) implementation of the "in circumcircle" test.

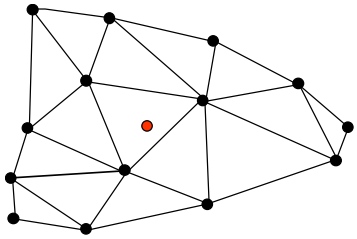
4) A suitable data structure ?

5) Is this parallelisable?

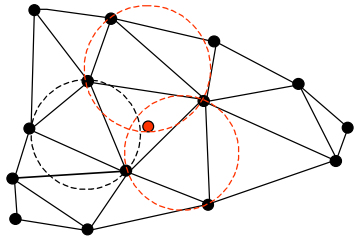
6) What if a new vertex falls *exactly* on the circumcircle of a triangle. The algorithm can choose to either regard this as inside or outside. would be nice to be able to have easy way of switching between either option at the high level of the code without lots of "ifs".

For your <u>interest only</u> I have placed an MSc/PhD level set of slides on meshing on Moodle: <u>you are not expected to study them for this assignment.</u>

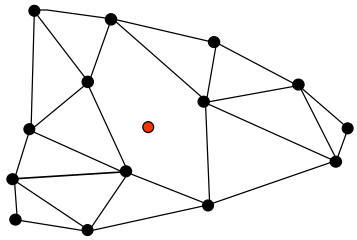
The *Incremental Bowyer-Watson* Algorithm



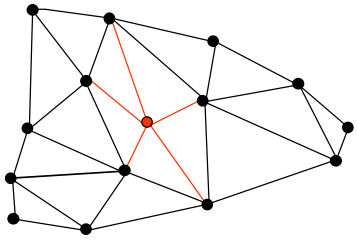
Initial mesh: we want to add the red vertex



Identify those triangles whose circumcircles enclose the new vertex: just the red ones here



Delete them



Each *exposed edge* seeds a new triangle connected to the new vertex

Numerical issues for Producing Delaunay Triangulations

A naive implementation of the Bowyer Watson Algorithm usually fails. This is not the fault of the algorithm, but rather its implementation on a computer with finite numerical precision

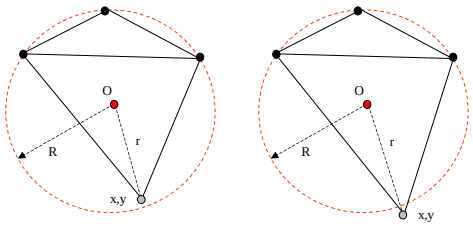
Let us assume that for each existing triangle we know its circumcentre O_x, O_y and its circumradius, R.

Does a new point r=x,y lie inside/outside of the circumcircle?

The test is mathematically easy.

Inside if $|r-O| < R$

or explicitly if $\sqrt{(x-O_x)^2+(y-O_y)^2} < R$



However, numerical rounding error is a big problem. The problem is if rounding errors cause a "mistake" the whole algorithm goes wrong as the triangles are no longer consistent with each other.

A solution is to use "exact maths" calculations. There are libraries available (GMP for us) that store all numbers as integer ratios, e.g. 712/123 and all the maths operations can be done without loss of precision. Example 7/3 + 15/2 = 59/6. However, the size of the integers used grows as necessary and the problem is that the number of digits in each explodes so that the calculations become very very slow.

We need an approach of the form: All "numbers" are stored in two forms, a "rough value" and an "exact value". The above test becomes

if $|r-O|-R$ evaluated using the rough values is <u>definitely</u> >0 or <0 we have a result, else redo the test with the exact values.

For the rough values we will use *interval maths.* Each "number" is given both maximum and minimum values between which we are certain the actual value lies.

E.g. exact number is pi, as an interval [3,4] or [3.14,3.15] etc.

Now all maths calculations can be overloaded, e.g. [a,b] + [c,d] = [a+c,b+d]

Interval maths ought to be only a few times slower than ordinary maths and coupled with the exact maths provides us with *total numerical robustness!*

Design issues:

> We are provided with an exact maths library (see appendix #3 for instructions) but must generate our own interval maths capability.
>
> We will have to have circumcentre tests for different types of data arguments.

Speed issues for Producing Delaunay Triangulations - Parallelisation?

Real meshes are huge! Operations on them become really time consuming. Can we parallelise?

> Consider the "which triangle does a new vertex fall in" test.
>
> Consider the Bowyer Watson algorithm.

Maybe all parallel processes should be able to simultaneously access any part of the mesh, or else maybe we should *partition the mesh* and each process operates on its own particular part with exchanges of information between processes regarding the interfaces between the partitions?

Specific Coursework Tasks

Part 1, Task 1: (10%) Code optimisation

On Moodle and below in appendix #5, there is a *very poorly* written computational program, *H63ESD_task_1.cpp*. It is a "made up" code that doesn't actually do anything useful, but it has many features common to a range of large-scale numerical simulation codes in engineering.

This is a rather poorly written code for a number of reasons!

I compiled it with *g++ -o task H63ESD_task_1.cpp* and ran it with *time ./task* on Jenna node comp00. The run time data on the screen was,

real 0m14.945s
user 0m14.883s
sys 0m0.022s

Your task is to improve the coding of the algorithm to minimise the run time; you must not change the code's function in any way. Direct use of machine code is not allowed!

Please make sure you get exactly the same results: run the original code as above and make a copy of the file produced, data_out. Then for any new version check for identical results by doing,

diff data_out original_ data_out

You should get no differences at all.

Required results: (1) *A copy of the original listing annotated with brief comments to identify poor programming style and where you think there is scope to improve the code: comments should cover both "coding bad practice" and "speedup" possibilities. (2) your optimised code (3) state in your report your best speed.*

To give ourselves a target, I have put a "forum" on Moodle - post your best result so far - obviously a full 10% will be awarded for the best result.

Part 1, Task 2: (10%) Serial Characterisation of the Jenna platform

Setup a very simple serial C program to add two double vectors.

a[i]=b[i]+c[i+offset] for all 0<i<n-offset.

The arrays are to be dynamically allocated. The arrays b and c are to be initialised so that b[i]=c[i]=i

Time the execution of the loop above (i.e. excluding the array setup and initialisation).

To get reasonable timing accuracy it may be necessary to wrap an outer loop around the above loop so as to repeat the above sum-loop many times.

To do the timings you might use *gettimeofday*, but there are alternatives – if you trust them and they are sufficiently accurate.

```
#include <sys/time.h>

struct timeval start_time,end_time;

gettimeofday(&start_time,NULL);

// Your code to time

gettimeofday(&end_time,NULL);

cout<<"\n\ngettimeofday wall time="<<
        end_time.tv_sec - start_time.tv_sec+(end_time.tv_usec-start_time.tv_usec)/1e6;
```

With offset=0, investigate and plot the timings for different n.

Briefly explain what you see. Does doubling n mean doubling the run time etc?

Based upon the previous results, select a few "interesting" values of n and repeat for different offset values.

Please provide no more than 1 page (excluding plots) of written explanation of what you see

Required results: *Submit both your test codes and the results obtained and the discussions in the report.*

Part 1, Task 3: (10%) An Interval Mathematics Component

Produce an *interval maths* "component" to be used in the above scenario: suitable templating is required. Consider that this component may be recycled for future projects as well.

Provide a suitable "test" main program to demonstrate the functionality of your component.

Required results: *Submit your component code and the test main. No report is required.*

Part 1, Task 4: (15%) Manipulation of Triangulated Meshes

Assume that you are provided with a triangulation stored in files as described in Appendix 4:

Design and implement a data structure and an interface for a C++ class to encapsulate the triangulation. Use of the STL is likely to be appropriate.

The interface must permit the following:

a) File streaming I/O

b) Queries of the form: given a point x,y at run time which triangle contains it

c) Queries of the form: given a function (known at compile time) f(x,y), return the integral of f(x,y) over the domain of the triangulation.

For this we will approximate the integration in two possible ways - see Appendix #1

Required results: *Your codes and a brief section in the report to convince the client that it works correctly (evidence please).*

Part 1, Task 5: (10%) Basic OpenMP Parallelisation

Parallelise task 4 using OpenMP: credit will be given for the degree of speedup achieved.

Required results: *Your codes and a brief section in the report to convince the client that it still works correctly and the speedup obtained.*

Part 1, Task 6: (10%) Basic MPI Parallelisation

Design, implement and run test codes to assess the basic performance of the Jenna cluster when running *simple* MPI codes. Explore issues such as scaling with respect to number of processes launched, the load balancing used, how the performance changes if processes are launched on different cores of multi-core CPUS in comparison with each process being on a separate CPU etc. Infiniband versus ethernet, maybe even the MPI configuration settings.

What sort of tests? The scope of MPI coverage in this module is simple point to point data transfer and we have a finite *budget* (i.e. your time!) : keep talking to the client!

This is assessing your ability to "design a problem" rather than just "solving a problem".

Required results: *Up to you - consider the " Course work style" comment above.*

Part 2: (25%): Delaunay Triangulation

Design and implement a serial code to read a set of points from a file *.node in the format of Appendix 4 and to generate a Delaunay Triangulation using the Bowyer Watson algorithm.

Consideration must be given to numerical rounding issues.

Consideration must be given to exception handling.

Validate using test sets of points given on Moodle: don't use random ones as these test sets have been chosen to avoid some awkward issues we are not considering here - more can be provided on request.

Required results: *Your codes and a brief section in the report to convince the client that it works correctly (evidence please). Run times and memory use.*

You are welcome to discuss this with the client as much as you like, because as all real clients do, he is asking for more than he really ought to given the timescales involved. Think careful about how to most effectively spend your valuable time - it is an expensive and finite resource. What would impress the client the most? What are the priorities to be considered if it is not feasible to complete the whole task?

Progress Reporting and Time Management: (10%)

The progress reports described earlier are a compulsory part of the coursework. A mark of 10% will be awarded for these in total and is based upon whether the client felt convinced at the time that the work was proceeding in a suitable manner and is likely to be successfully delivered at the end of the project.

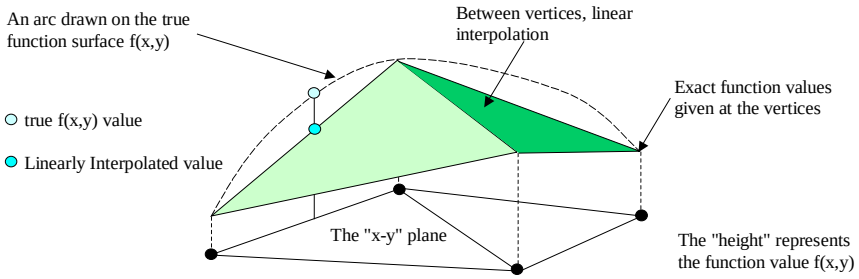
These are not marked on technical content, but only from the point of view of project management.

Indeed the client will be quite happy if the progress reports contain more *questions* than *answers!*

Don't forget that in the real world your client is likely to have a boss to report to as well and he/she will need evidence from you to convince his/her management that they are monitoring the project correctly and thus doing their job properly as well!

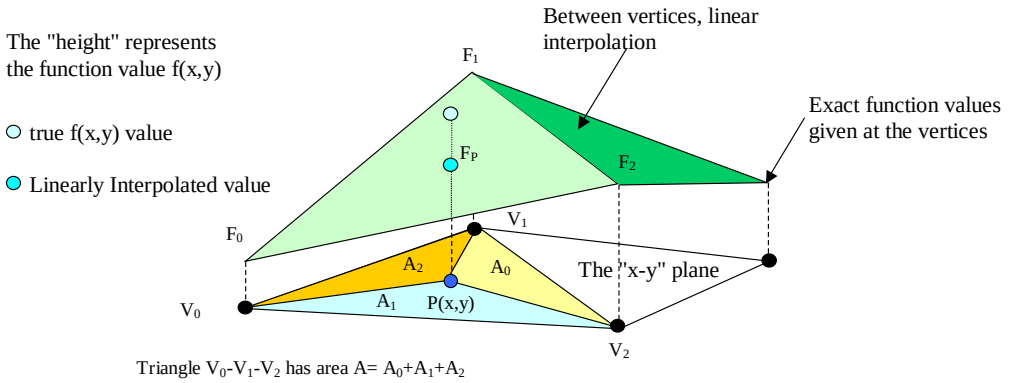
IF IN DOUBT ABOUT ANY OF THE TASKS OR THE ASSEMENT CRITERIA - ASK ME!

Appendix #1: Linear Interpolation Over Individual Triangles



If we have a triangulation and we know the *true* values of the function f(x,y) at its vertices, how do we interpolate to estimate the function at *any* point P in the x, y plane?

Well we consult a friendly mathematician who provides the following:



If the values of the function f(x,y) are known to be F_0, F_1 and F_2 at the vertices V_0 , V_1 and V_2 then the *linearly interpolated value at any point P(x,y) inside the triangle is,*

$$F_P(x,y) = \frac{A_0(x,y)F_0 + A_1(x,y)F_1 + A_2(x,y)F_2}{A}$$

where the A_i are the areas of the sub-triangles shown in the figure - they are functions of where P is - hence x,y. *A* is the total area of the triangle.

For interest the quantities $\left\{\frac{A_0(x,y)}{A}, \frac{A_1(x,y)}{A}, \frac{A_2(x,y)}{A}\right\}$ *are called the area, or barycentric coordinates of the point P with respect to the triangle. These have many convenient properties.*

How to numerical approximate the integral of the function with respect to x and y ? Again the same friendly mathematician provides:

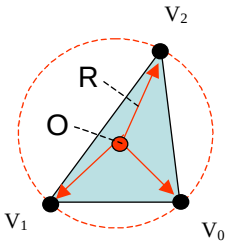
There are two ways we could evaluate an approximation to $\iint_T dx\,dy\ f(x,y)$ where T is a triangle:

1) ***Constant value approximation*** $\iint_T dx\,dy\ \ f(x,y) \approx A_T\ \ f(O_{xT}, O_{yT})$ i.e. evaluate the function at the circumcentre, O, of each triangle and weight by the triangle's area.

2) ***Linear interpolation approximation*** using the above and the useful, but non-obvious, fact that:

$$\iint_{\substack{Triangle\\ area}} dx\ dy\ A_i(x,y) = A/3$$ *Not sure about this - consult your client!*

Appendix #2: The Circumcentre of Individual Triangles



It can be shown that if vertex V_i has coordinates X_i , Y_i that the coordinates of the circumcentre O_x , O_y and the circumradius R are found from solutions of,

$$\begin{pmatrix} X_0^2+Y_0^2 \\ X_1^2+Y_1^2 \\ X_2^2+Y_2^2 \end{pmatrix} = \begin{pmatrix} X_0 & Y_0 & 1 \\ X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \end{pmatrix} \begin{pmatrix} 2O_x \\ 2O_y \\ R^2-O_x^2-O_y^2 \end{pmatrix}$$

(See if you can prove this - just for fun, no marks.)

This order 3 matrix equation can be solved quite effectively using basic algebraic techniques.

You are welcome to discuss and seek further information/detail/clarification regarding any aspect of the "geometry" or "algebra" from P Sewell

Appendix #3: Use Of The Exact Maths Library, GMP

See http://gmplib.org/ and the manual is on http://gmplib.org/#DOWNLOAD.

We will use the integer quotient form. All numbers are expressed as P/Q where P and Q are integers whose length, i.e. number of digits, grows/contracts as necessary. P ad Q can easily have 10000's of digits so speed and memory is an issue.

Basic use

```
#include "gmpxx.h"

mpq_class x(3,1);          // a=3/1

mpq_class y(5,2);          // b=5/2

mpq_class a(x+y*y);        // All the expected maths operators overloaded but don't expect sqrt() - as ?

cout<<"\n"<<a.get_d();     // get.d() returns the double precision approximation to the exact value.
```

To link your programs you need to either link with -lgmp -lgmpxx flags if the libraries are installed on the platform or if not directly with the object libraries

g++ code.cpp libgmpxx.a libgmp.a

I have various builds of these for cygwin and MinGW as well as Jenna - just request them from me.

You are welcome to discuss and seek further information/detail/clarification regarding any aspect of the GMP library from P Sewell

Appendix #4: File Formats

Ascii file format for reading/writing 2D meshes. This is your client's legacy format - can't be changed. Triangulation files should have the extension .tri and simple vertex files, .node.

"vertices.node"

no_points dimensions (2 or 3) no_attributes_per_point

i $\mathbf{x_i}$ $\mathbf{y_i}$ $\mathbf{\{z_i\}}$ **attribute** $_1$. . . . . // no_points off

"triangulation.tri"

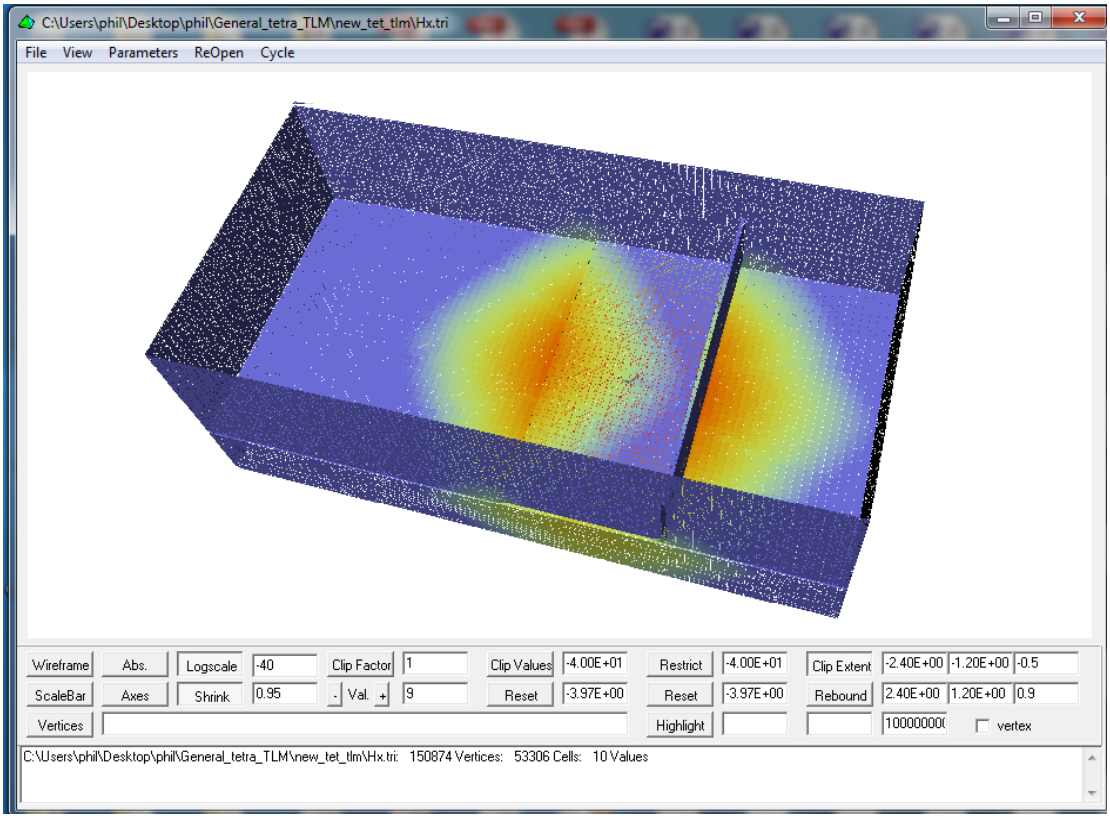
no_points dimensions (2 or 3) no_attributes_per_point

i $\mathbf{x_i}$ $\mathbf{y_i}$ $\mathbf{\{z_i\}}$ **attribute** $_1$. . . . . // no_points off

no_cells no_vertices_per_cell no_attributes_per_cell

i v_1. . . . . $v_{\text{no\_vertices\_per\_cell}}$ **attribute**$_1$. . . . . // no_cells off

But at least that means you can use his "visualisation" software, (available on Moodle): its use will be explained in the lectures as well.



Appendix #5: Code for task#1

```
#include <iostream>
#include <fstream>
#include <math.h>

using namespace std;

int main(int argc, char* argv[]) {

	int nx(10000);

	int ny(200);

	int nt(200);

	double** vi=new double*[nx];

	double** vr=new double*[nx];

	double pi=(4.*atan(1.));

	for(int i=0;i<nx;i++) {

		vi[i]=new double[ny];

		vr[i]=new double[ny];

	}

	for(int i=0;i<nx;i++) {

		for(int j=0;j<ny;j++) {

			vi[i][j]=double(i*i)*double(j)*sin(pi/double(nx)*double(i));

			vr[i][j]=0.;
		}
	}


	ofstream fout("data_out");

	for(int t=0;t<nt;t++) {

		cout<<"\n"<<t;cout.flush();

		for(int i=0;i<nx;i++) {

			for(int j=0;j<ny;j++) {

				if(i>0&&i<nx-1&&j>0&&j<ny-1) {

					vr[i][j]=(vi[i+1][j]+vi[i-1][j]+vi[i][j-1]+vi[i][j+1])/4.;

				} else if(i==0&&i<nx-1&&j>0&&j<ny-1) {

					vr[i][j]=(vi[i+1][j]+10.+vi[i][j-1]+vi[i][j+1])/4.;

				} else if(i>0&&i==nx-1&&j>0&&j<ny-1) {
```

```
                    vr[i][j]=(5.+vi[i-1][j]+vi[i][j-1]+vi[i][j+1])/4.;

                } else if(i>0&&i<nx-1&&j==0&&j<ny-1) {

                    vr[i][j]=(vi[i+1][j]+vi[i-1][j]+15.45+vi[i][j+1])/4.;

                } else if(i>0&&i<nx-1&&j>0&&j==ny-1) {

                    vr[i][j]=(vi[i+1][j]+vi[i-1][j]+vi[i][j-1]-6.7)/4.;

                }
            }
        }

        for(int i=0;i<nx;i++) {

            for(int j=0;j<ny;j++) {

                if(fabs(fabs(vr[i][j])-fabs(vi[i][j]))<1e-2) fout<<"\n"<<t<<" "<<i<<" "<<j<<"
                                                                "<<fabs(vi[i][j])<<" "<<fabs(vr[i][j]);
            }
        }

        for(int i=0;i<nx;i++) {

            for(int j=0;j<ny;j++) vi[i][j]=vi[i][j]/2.+vr[i][j]/2.;

        }
    }
}
```