Notes gfp_gaussian_process

Structure

```
CSVconfig config("csv_config.txt");
Parameter_set params("parameter_bounds.txt");

// Read data
std::vector<MOMAdata> cells =  getData(infile,
                                       config.time_col,
                                       config.length_col,
                                       config.fp_col,
                                       config.delm);
// genealogy
build_cell_genealogy(cells);
std::vector<MOMAdata *> root_cells = get_roots(cells);

// minimization
for(int j=0; j<root_cells.size(); ++j){
    numerical_minimization(*root_cells[j], params);
}
```

Likelihhod Calculation

- apply function recursively
- every cell is accessed once and after its parent is calculated

```
void apply_down_tree(MOMAdata &cell,
                     void (*func)(MOMAdata &, Parameter_set &),
                     Parameter_set &params){
    /* applies the function func to the cell cell and the other cells in the genealogy
    * such that the parent cell has already been accessed when the function is applied
    * to the cell.
    *
    * Example (number implies the order in which)
    * ____________________________________________________

               1
             /   \
            2     5
          /   \     \
         3     4     6

    * ____________________________________________________
    */
    apply_down_tree_recr( &cell, func, params);
}

double total_likelihood(MOMAdata &cell, Parameter_set& params){
    /*
    * total_likelihood of cell tree, to be maximized
    */

    apply_down_tree(cell, sc_likelihood, params);

    double total_likelihood=0;
    /*
    * Add likelihoods of all cells
    */
    return total_likelihood;
}

// example function to illustrate how this works
void set_generation(MOMAdata &cell, Parameter_set &params){
    if (cell.parent != nullptr){
        cell.generation = cell.parent->generation + 1;
    } else{
        cell.generation = 0;
    }
}
```

Root library

- choose algorithm at runtime
- allows definition paramter space at runtime

- maybe tricky to install `root`

```
ROOT::Math::Minimizer* min = ROOT::Math::Factory::CreateMinimizer(minName, algoName);

ROOT::Math::Functor f(&total_likelihood,2);
min->SetFunction(f);

if (params.mean_lambda.fixed){
    min->SetFixedVariable(0, "mean_lambda", params.mean_lambda.value);
} else {
    min->SetLimitedVariable(0,"mean_lambda",
                                params.mean_lambda.value,
                                params.mean_lambda.step,
                                params.mean_lambda.lower,
                                params.mean_lambda.upper);
}
min->Minimize();
```

```
parameter_bounds.txt
--------------------

# for free parameter:
# parameter = value, step, lower, upper
mean_lambda = 2, 0.01, -10, 10

# for fixed parameter:
mean_lambda = 4
```