Minimization

David Eberly
Magic Software, Inc.
http://www.magic-software.com

Created: March 2, 1999

I used the ideas from Numerical Recipes in C to find local minima of functions of one variable. The method uses inverse parabolic interpolation to fit three points of the function with a parabola and to use the vertex as an update function value to include on the next step. The idea is that the vertices should converge to a local minima. The class declaration is

```
#include <iostream.h>

class mgcMin1D
{
public:
    typedef float (*RealFunction)(float);

    mgcMin1D (float _tmin, float _tmax, RealFunction _F);
    int Minimum (float t, float step, float tolerance, float& tmin,
        float &fmin);

private:
    RealFunction F;    // function to minimize
    float tmin, tmax;  // function domain

    float t0, t1, t2, f0, f1, f2;
    int Bracket (float t, float step);

// error handling
public:
    static int verbose;
    static unsigned error;
    static void Report (ostream& ostr);
private:
    static const unsigned at_endpoint;
    static const unsigned bracket_exceeded;
    static const unsigned minimum_exceeded;
    static const char* message[];
    static int Number (unsigned single_error);
    static void Report (unsigned single_error);
};
```

I'm not going to describe the code here. The source file `minimize.c` has comments which indicate abstractly what the algorithm is. It works okay on simple examples, but I had trouble with it in more complicated situations (failure to converge is the most notable problem). Here is an example anyway

```
#include <iostream.h>

float F (float x)
{
    // F has local minima at |x| = sqrt(0.5)
    float x2 = x*x;
    return x2*x2-1;
}

int main ()
{
    float xmin = 0.1, xmax = 1.0;
    mgcMin1D mini(xmin,xmax,F);

    float xguess = 0.5;
    float step = 0.01;
    float tolerance = 1e-06;
    float xtrue, ftrue;
    if ( mini.Minimum(xguess,step,tolerance,xtrue,ftrue) )
        cout << "minimum at x = " << xtrue << ", f = " << ftrue << endl;
    else
        cout << "minimum not found" << endl;

    return 0;
}
```

The higher dimensional minimizers (min1, min2, min3, minn) use Powell's direction set method to avoid taking derivatives. I'll describe these at a later date. The file `minimize.cpp` uses the conjugate gradient method to determine the search direction, but then resorts to inverse parabolic interpolation along the 1–dimensional subspace. I'll describe this later also.