

```

//Denker's g-drag_coefficient*v^2
// numerical integration of its equation of motion

#include <iostream>
#include <fstream>           /* for output file */
#include <cmath>
#include <iomanip>           /* for setw() */
#include <stdlib.h>         /* for exit() */
using namespace std;

double const pi = 4*atan(1);

//////////
// printout routine
//
void p(ofstream& ouch,
      double const t,
      double const distance,
      double const speed

){
    ouch << setw(15) << t           << ", "
        << setw(18) << distance    << ", "
        << setw(18) << speed       << ", "
        << endl;
}

// The expression "\" is just too ugly.
// Define a nicer-looking synonym:
string const Q = "\"";

// C++ modulo operator "%" only works for integers
// Generalize to doubles
double mod(double const num, double const denom){
    return num - denom * floor(num / denom);
}

//////////
// Main program
//

int main() {
    /* cout << "boiler plate for lin. (OR non) Damped Simple Pendulum"
    << endl;
    cout << "w/ deadbeat type drive and g change" << endl; */

    cout << "//////////" << endl;
    cout << "This version is Cromer's, fall with quadratic damping,
```

```

only" << endl;

    double t=0;
    double distance = 0;
    double speed = 0;
    double g = 9.8;
    double deltat;
    cout << "Enter step interval in seconds, i.e. delta t; e.g. 1e-3 "
<< endl;

    cin >> deltat;

    cout << "Enter time of fall in seconds: " << endl;
    double stop = 0;
    cin >> stop;
    stop = stop / deltat;    //the total number of iterations

    cout << "Enter printouts per second." << endl;

    double po_per_stop = 0;
    cin >> po_per_stop;
    po_per_stop = po_per_stop;

    double mod_number = stop / po_per_stop; //

    //double mod_number = steps_per_cycle / po_per_cycle;
    //double period;
    cout << "Data output filename hint! string.txt: ";
    string out_filename;
    cin >> out_filename;

    ofstream ouch;
    ouch.open(out_filename.c_str());
    if (!ouch.good()) {
        cerr << "Failed to open output file '" << out_filename << "'" <<
endl;
        exit(1);
    }

    // column headers:

    ouch << setw(15) << setprecision (12) << Q+"t/s"+Q
<< ", "

    // ouch << setw(10) << Q+"t/s"+Q                << ", "
    << setw(18) << Q+"distance (m)"<< Q            << ", "

```

```

        << setw(18) << Q+"speed (m/s)+"Q          << ", "
        << endl;

// print the initial state:
p(ouch, t, distance, speed);

////////////////////////////////////////
// main loop
for (int N = 1; N <= stop; N++){

    /*////////////////////////////////
    // Cromer:

    double A = g - 0.003*speed * speed;
    speed = speed + A * deltat;
    distance = distance + speed * deltat;

    t = t + deltat;          //step t for next iteration.

    //////////////////////////////////*/

// acceleration at beginning of interval:

    double A = g - 0.003*speed * speed;

    speed = speed + A * deltat /2 ;

    distance = distance + speed * deltat;

////////// since A is A(speed) not position, however, recalc is with new
speed was: recalculate acceleration and end of interval, using new
theta:

    A = g - 0.003*speed * speed;

    speed = speed + A*deltat/2;

    t = t + deltat;          // step time for next iteration
// Print out something every mod_number steps,
// (or as close thereto as we can, if mod_number is not an integer).
// Also print out the very last result
// (which will not be evenly spaced if total duration
// is not an integer multiple of the spacing between
printouts).  //////////////////////////////////*/

    if (N == stop || (mod(N + 0.5, mod_number)) < 1) {

```

```
        p(ouch, t, distance, speed);
    }
} // end main loop
}
```