HMDGL – Beispiel zur Programmierung in C++

```cpp
// ----------------------------------------------------------------------------
// Simulation einer Pendelschwingung in C++
// ----------------------------------------------------------------------------
#define _USE_MATH_DEFINES
#include <iostream>
#include <vector>
#include <math.h>  // für M_PI
#include "hmdgl.hpp"

// ----------------------------------------------------------------------------
// Die Zustandsvariablen des DGL-Systems werden in Containern des Typs
// <state_type> gespeichert. Als <state_type> sind grundsätzlich alle
// Container geeignet, welche die Methoden begin() und end() besitzen.
// ----------------------------------------------------------------------------
typedef std::vector<double> state_type;
const size_t dim = 2;  // DGL-System aus zwei Gleichungen 1. Ordnung

// ----------------------------------------------------------------------------
// In C++-Programmen können die Funktionen <sys> und <step> direkt
// im Hauptprogramm definiert werden (sog. Lambda-Funktionen).
// ----------------------------------------------------------------------------
int main()
{
    size_t num_steps = 1000;  // Anzahl der Integrationsschritte
    double t_begin   = 0;     // Startzeitpunkt der Simulation
    double t_end     = 10.0;  // Endzeitpunkt der Simulation

    state_type init(dim);
    init[0] = 45.0 * M_PI / 180.0;  // Anfangsauslenkung = 45°
    init[1] = 0;                    // Anfangsgeschwindigkeit = 0

    // DGL-System aus zwei Gleichungen 1. Ordnung:
    // du0/dt = u1  und  du1/dt = -g/L * sin(u0)
    auto sys = [] (double /* t */, state_type u)
    {
        state_type dudt(dim);
        dudt[0] = u[1];
        dudt[1] = -9.81 / 1.0 * sin(u[0]);
        return dudt;
    };

    // Nach jedem Integrationsschritt wird <step> aufgerufen:
    // Aktuelle "Simulationszeit" und Pendelauslenkung ausgeben
    auto step = [] (double t, state_type u)
    {
        std::cout << t << "\t" << 180.0 * u[0] / M_PI << std::endl;
    };

    HMDGL::dgl_rk4(t_begin, t_end, num_steps, sys, step, init);
    // HMDGL::dgl_heun(t_begin, t_end, num_steps, sys, step, init);
    // HMDGL::dgl_euler(t_begin, t_end, num_steps, sys, step, init);
}
```