Programming techniques for physical simulations
Exercise 3

September 30, 2009

Simpson library

1. If you have not yet done so, write a C++ function for the Simpson integration using function pointers.
2. What are the arguments of the function? What are the preconditions and the postconditions? Document your function thoroughly and check the conditions using assertions.
3. Take that function and copy it to a different file. Create a header file that declares the function. Compile and link your program.
4. Create a Makefile that compiles the function for you. Make sure you get the dependencies right. Always compile only the files that have changed.
5. Create a library libintegrate.a that contains your Simpson integration function. Rewrite your Makefile to link against it.

Operator overloading and template functions - Implementation of a finite group

Definition We will implement the finite group $\mathbb{Z}_2$ with the following properties:

- $\mathbb{Z}_2 = \{+, -\}$
- The group operation $\cdot$ is defined through

$$+ \cdot + = - \cdot - = + \tag{1}$$
$$- \cdot + = + \cdot - = -. \tag{2}$$

- The representation of a group element $g(\mu), \mu \in \mathbb{Z}_2$ on integer, real or complex numbers is given by

$$g(\mu) = \begin{cases} +1 & : \quad \mu = + \\ -1 & : \quad \mu = - \end{cases}$$

Implementation To represent the group $\mathbb{Z}_2$ in C++, we will use an enumeration type and a function that returns the identity element:

```
enum Z2 { Plus, Minus };

template<class T> T identity_element() { return T(1); }
template<> Z2 identity_element<Z2>() { return Plus; }
```

This is a case of a fully specialized template function, which will be explained in more detail later in the course. To implement the group operation, we will *overload* the * operator, i.e. assign a meaning to expressions such as

```
Z2 p = Plus, m = Minus;
Z2 r = p*m;
```

To this end, we create the following function:

```
Z2 operator*(Z2 a, Z2 b);
```

Furthermore, we want to be able to print our result in a nice form, i.e. using an expression such as `cout << r << endl;`. We will therefore overload

```
ostream& operator<<(ostream& os, Z2 a);
```

in such a way that `Plus` is printed for + and `Minus` for −.

To implement the action of a group element on a number, we will implement the following template function (note that from the point of view of C++, `a*b` is not necessarily the same as `b*a`):

```
template<class T> T operator*(T a, Z2 b);
template<class T> T operator*(Z2 a, T b);
```

Finally, we will implement a templated power function which only relies on the multiplication, so that it can also be used on our $\mathbb{Z}_2$ group:

```
template<class T> T mypow(T a, unsigned int n);
```

Note that for this, you might need to use the `identity_element` function from above.

Exercise

- Implement the functions mentioned above; you can find a C++ file with the necessary structure on the lecture website.
- For each of the templated functions, think about the concepts required and document these!