In a [previous post](https://stubner.me/2019/07/numerical-integration-in-rcpp/) it has shown that without intervention [RcppNumerical](http://cran.r-project.org/package=RcppNumerical) does not handle integration over infinite ranges. In this post I want to generalize the method to integrals where only one of the limits is infinite. In addition, I want to make it more user friendly, since the below code is written something like

// [[Rcpp::depends(RcppEigen)]]

// [[Rcpp::depends(RcppNumerical)]]

#include

namespace rstub {

// [...]

}

class exp4: public Numer::Func {

private:

double mean;

public:

exp4(double mean\_) : mean(mean\_) {}

double operator()(const double& x) const {

return exp(-pow(x - mean, 4) / 2);

}

};

// [[Rcpp::export]]

Rcpp::NumericVector integrate\_exp4(const double &mean, double lower, double upper) {

exp4 function(mean);

double err\_est;

int err\_code;

double result = rstub::integrate(function, lower, upper, err\_est, err\_code);

return Rcpp::NumericVector::create(Rcpp::Named("result") = result,

Rcpp::Named("error") = err\_est);

}

and have it correctly handle different input:

rbind(

integrate\_exp4(4, 0, 4),

integrate\_exp4(4, -Inf, Inf),

integrate\_exp4(4, 3, Inf),

integrate\_exp4(4, -Inf, 3)

)

## result error

## [1,] 1.0779003 9.252237e-08

## [2,] 2.1558005 1.439771e-06

## [3,] 1.9903282 4.250105e-11

## [4,] 0.1654723 6.251315e-14

The only differences in the above code to the sample code from the [previous post](https://stubner.me/2019/07/numerical-integration-in-rcpp/) is the usage or rstub::integrate instead of Numer:integrate and the as yet unspecified rstub namespace. What is needed in that namespace? First, we will need a template class that does the necessary variable substitutions. In the case where both limits are infinite, we use as before \(x = (1-t)/t\) resulting in

\[  
\int\_{-\infty}^{\infty} \mathrm{d}x f(x) = \int\_0^1 \mathrm{d}t \frac{f((1-t)/t) + f(-(1-t)/t)}{t^2}  
\]

If only one of the limits is infinite, we use the substitutions \(x = a + (1-t)/t\) and \(x = b – (1-t)/t\) resulting in

\[  
\int\_{a}^{\infty} \mathrm{d}x f(x) = \int\_0^1 \mathrm{d}t \frac{f(a+(1-t)/t)}{t^2}  
\]

and

\[  
\int\_{-\infty}^{b} \mathrm{d}x f(x) = \int\_0^1 \mathrm{d}t \frac{f(b-(1-t)/t)}{t^2}  
\]

For the C++ template class aggregation is used instead of inheritance, allowing to easily specify the limits:

template

class transform\_infinite: public Numer::Func {

private:

T func;

double lower;

double upper;

public:

transform\_infinite(T \_func, double \_lower, double \_upper) :

func(\_func), lower(\_lower), upper(\_upper) {}

double operator() (const double& t) const {

double x = (1 - t) / t;

bool upper\_finite = (upper < std::numeric\_limits::infinity());

bool lower\_finite = (lower > -std::numeric\_limits::infinity());

if (upper\_finite && lower\_finite) {

Rcpp::stop("At least on limit must be infinite.");

} else if (lower\_finite) {

return func(lower + x) / pow(t, 2);

} else if (upper\_finite) {

return func(upper - x) / pow(t, 2);

} else {

return (func(x) + func(-x)) / pow(t, 2);

}

}

};

Finally we need a wrapper function for Numer::integrate which checks if both limits are finite or not:

using Numer::Integrator;

template

double integrate(const T& f, double lower, double upper,

double& err\_est, int& err\_code,

const int subdiv = 100, const double& eps\_abs = 1e-8, const double& eps\_rel = 1e-6,

const Integrator::QuadratureRule rule = Integrator::GaussKronrod41) {

if (upper == lower) {

err\_est = 0.0;

err\_code = 0;

return 0.0;

}

if (std::abs(upper) < std::numeric\_limits::infinity() &&

std::abs(lower) < std::numeric\_limits::infinity()) {

return Numer::integrate(f, lower, upper, err\_est, err\_code, subdiv, eps\_abs, eps\_rel, rule);

} else {

double sign = 1.0;

if (upper < lower) {

std::swap(upper, lower);

sign = -1.0;

}

transform\_infinite g(f, lower, upper);

return sign \* Numer::integrate(g, 0.0, 1.0, err\_est, err\_code, subdiv, eps\_abs, eps\_rel, rule);

}

}

If both limits are finite, Numer::integrate is used directly. Otherwise the function is transformed and Numer::integrate is used with adjusted range. In addition, it is first checked that the upper limit is actually larger than the lower limit. If this is not the case, one of the properties of integration is used to swap the limits and change the sign:

\[  
\int\_{a}^{b} \mathrm{d}x f(x) = -\int\_{b}^{a} \mathrm{d}x f(x)  
\]

Thereby we get the correct result even when the limits have been exchanged:

rbind(

integrate\_exp4(4, 3, Inf),

integrate\_exp4(4, Inf, 3)

)

## result error

## [1,] 1.990328 4.250105e-11

## [2,] -1.990328 4.250105e-11

In the end we needed on template class and one template function, which could be put into a separate header file, to generalize Numer::integrate for integration over an infinite interval.