Questions 7

1. How do you allocate GSL vectors and matrices?

   To allocate a GSL vectors, you type:

   `gsl_vector *v1 = gsl_vector_alloc (size_t n);`

   `gsl_vector *v2 = gsl_vector_calloc (size_t n);`

   To allocate a GSL matrix, you type:

   `gsl_matrix *m1 = gsl_vector_alloc (size_t n1, size_t n2);`

   `gsl_matrix *m2 = gsl_vector_calloc (size_t n1, size_t n2);`

   the difference between `alloc` and `calloc`, is that `calloc` initializes all the elements as zeros, where `alloc` doesn't. In the end, you will need to free them, with

   `gsl_vector_free (gsl_vector *v);`

   `gsl_matrix_free (gsl_vector *m);`

Problem 20: Implement the Arctangent function using integral representation.

$$\arctan(x) = \int_0^x \frac{1}{z^2+1}\, dz. \tag{1}$$

To facilitate numerical integration reduce the argument to a reasonable interval (e.g. $[0, 1]$) using the formulae (check them),

$$\arctan(-x) = -\arctan(x) \tag{2}$$

$$\arctan\left(\frac{1}{x}\right) = \frac{\pi}{2} - \arctan(x), \quad \text{if } x > 0. \tag{3}$$

prior to integration. Compare with the corresponding function from `<math.h>` or from `GSL`.

The problem was solved by integrating with `gsl/gsl_integratson.h`. First the integrand from eq. 1 was define:

```
double arctan_integrand (double z, void *params){
        return 1/(pow(z,2) + 1);
}
```

and then the GSL integration rutine was build:

```
double my_arctan (double x){
        if(x<0) return - my_arctan(-x);
        if(x>1) return M_PI/2 - my_arctan(1/x);
        if(x==0) return 0;
```

```
          gsl_function f;
          f.function = arctan_integrand;
          f.params = NULL;

          int limit = 100;
          double a = 0, b = x, epsabs = 1e-9, epsrel = 1e-9, result, error;
          gsl_integration_workspace *w =
                  gsl_integration_workspace_alloc (limit);
          int status = gsl_integration_qags
                  (&f, a, b, epsabs, epsrel, limit, w, &result, &error);
          gsl_integration_workspace_free (w);
          if(status != GSL_SUCCESS) return NAN;
          else return result;
}
```

The three if statements ensures that the function only integrates in the range of $[0, 1]$.

The result of the integration was compared using `atan(x)` from `<math.h>` in the main function, and plottet with points, see fig. 1.

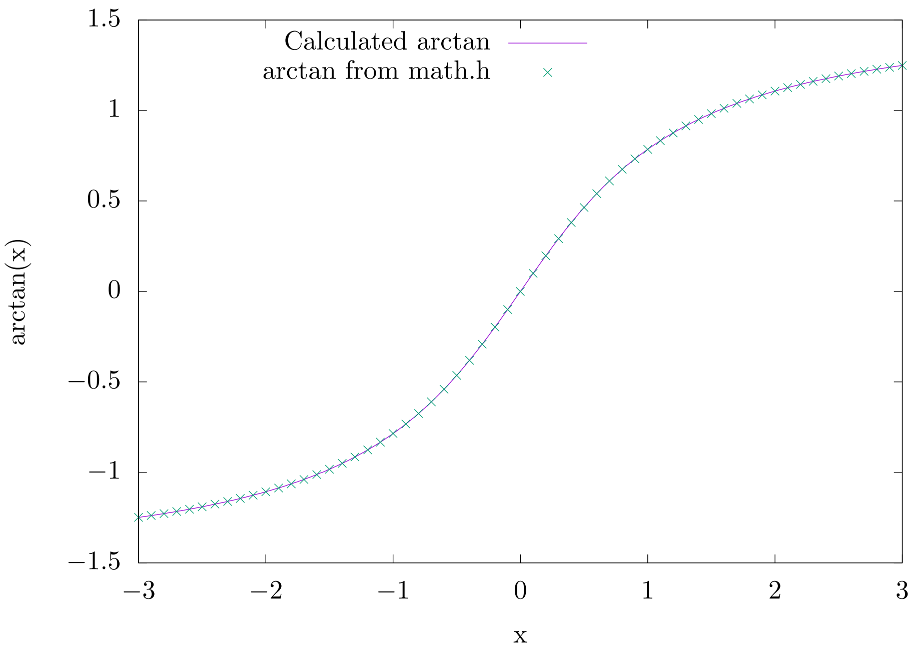

Figure 1: Plot of calculated and exact arctan(x), every 10th result of the arctan from math.h was plottet as points for clarity.