Consider the integral

$$\int_a^b \frac{\cos(x^3)\exp(-x)}{\sqrt{x}}dx$$

i) (5 points) For $N = 2$ (only one midpoint), state by hand computations an analytical expression for the approximation of the integral as obtained by the Extended Midpoint method.

Exercise i)

First we compute the f-computation at the midpoint of the function

$$f\left(\frac{a+b}{2}\right) = \frac{\cos\left(\left(\frac{a+b}{2}\right)^3\right) \cdot e^{\frac{a+b}{2}}}{\sqrt{\frac{a+b}{2}}}$$

We then multiply it by the step-size, and since N = 2 the stepsize is:

$$h = b - a$$

$$f\left(\frac{a+b}{2}\right) = \frac{\cos\left(\left(\frac{a+b}{2}\right)^3\right) \cdot e^{\frac{a+b}{2}}}{\sqrt{\frac{a+b}{2}}}(b-a)$$

This is then the analytical expression of N=2 of the extended midpoint method

Exercise ii)

The accuracy was computed using Richardson extrapolation with the following formula

$$\frac{A(h_2) - A(h_1)}{\alpha^k - 1}$$

Where α^k is calculated with the following formula:

$$\alpha^k = \frac{A(h_1) - A(h_2)}{A(h_2) - A(h_3)}$$

And $h_1 > h_2 > h_3$

The code used can be seen here

```
double f_1(double x)
{
    return (cos(pow(x, 3)) * exp(-x)) / sqrt(x);
}

double extended_midpoint(double f(double x), double a, double b, double
threshold = 1e-3)
{
```

```
    double alp_k = 0;
    double error = 100;
    double h_old = 0;
    double integral;
    double integral_m1 = 0;
    double integral_m2 = 0;
    int N = 1;
    int f_eval;
    int i = 1;
    int max_iter = 40;
    std::cout << setw(5) << "i" << setw(15) << "A(hi)" << setw(15) <<
"A(hi-1)-A(hi)" << setw(15) << "rich-alp^k" << setw(15) << "rich-fejl"
<< setw(15) << "f-beregninger" << endl;
    while (abs(error) > threshold && i < max_iter)
    {
        double h = (b - a) / N;
        integral = 0;
        for (double j = a + 0.5 * h; j < b; j += h)
        {
            integral += f(j) * h;
        }
        if (i > 2) // we need at least 3 iterations to calculate alp_k
        {
            alp_k = (integral_m2 - integral_m1) / (integral_m1 -
integral);
            error = (integral - integral_m1) / (alp_k - 1);
        }
        f_eval = pow(2, i - 1);
        std::cout
            << setw(5) << i << setw(15) << integral << setw(15) <<
integral_m1 - integral << setw(15) << alp_k << setw(15) << error <<
setw(15) << f_eval << endl;
        integral_m2 = integral_m1;
        integral_m1 = integral;
        h_old = h;
        N *= 2;
        i++;
    }
    return integral;
}

void title(string title)
{
    cout << endl
         << GREEN << title << RESET << endl;
}
```

```
int main()
{
    double a = 0, b = 4;

    /* ------------------------------ function 1 ---------------------
---------- */

    title("Integrate from a = 0, b = 1: f(x) = cos(x^2)*exp(-x)");

    double integral = extended_midpoint(f_1, a, b, 1e-3);
}
```

The result can be seen in the screenshot below

i	A(hi)	A(hi-1)-A(hi)	alp^k	rich-comp	f-comp
1	-0.0556954	0.0556954	0	100	1
2	0.380737	-0.436433	0	100	2
3	0.629306	-0.248569	1.75578	0.32889	4
4	1.01052	-0.38121	0.652053	-1.0956	8
5	1.13552	-0.125003	3.0496	0.0609888	16
6	1.21473	-0.0792132	1.57806	0.137033	32
7	1.27796	-0.0632325	1.25273	0.2502	64
8	1.32279	-0.0448295	1.41051	0.109204	128
9	1.35432	-0.0315224	1.42215	0.0746713	256
10	1.37654	-0.0222198	1.41866	0.0530738	512
11	1.39222	-0.0156862	1.41652	0.0376599	1024
12	1.40331	-0.0110826	1.41539	0.0266798	2048
13	1.41114	-0.00783326	1.41481	0.018884	4096
14	1.41668	-0.00553777	1.41451	0.0133597	8192
15	1.42059	-0.00391538	1.41436	0.00944911	16384
16	1.42336	-0.00276844	1.41429	0.00668238	32768
17	1.42532	-0.00195753	1.41425	0.00472546	65536
18	1.4267	-0.00138416	1.41423	0.00334152	131072
19	1.42768	-0.000978746	1.41422	0.00236285	262144
20	1.42837	-0.000692075	1.41422	0.0016708	524288
21	1.42886	-0.00048937	1.41422	0.00118144	1048576
22	1.42921	-0.000346037	1.41421	0.000835404	2097152

It is worth noting here that the f_computations are not saved between iterations. If they were, then the f_computations for any iteration should subtract all previous ones.

Exercise iii)

The implementation of DErule from numeric recipes was used, and the implemented code only iterates through iterations of the next() function and prints the relevant information. The code can be seen here:

```
void print_de_rule(DErule<double(Doub, Doub)> de, double threshold =
1e-3)
{
    int its = 60;
    double A, A_m1 = 0, A_m2 = 0, alp_k = 0, error = 0, f_eval = 1;

    std::cout << setw(5) << "i" << setw(15) << "A(hi)" << setw(15) <<
"A(hi-1)-A(hi)" << setw(15) << "alp^k" << setw(15) << "error" <<
setw(15) << "f-comp" << endl;
    for (size_t i = 0; i < its; i++)
    {
        A = de.next();
        if (i > 1)
        {
            alp_k = (A_m2 - A_m1) / (A_m1 - A);
            error = (A - A_m1) / (alp_k - 1);
        }
        f_eval = 1 + 2 * pow(2, i);
        std::cout << setw(5) << i << setw(15) << A << setw(15) << A_m1
- A << setw(15) << alp_k << setw(15) << error << setw(15) << f_eval <<
endl;
        if (abs(A_m1 - A) < threshold)
            break;
        A_m2 = A_m1;
        A_m1 = A;
    }
}
double f_1(double x, double delta)
{
    if (x < 0.1)
        return (cos(pow(delta, 3)) * exp(-delta)) / sqrt(delta);
    else
        return (cos(pow(x, 3)) * exp(-x)) / sqrt(x);
}

int main()
{
    double a = 0, b = 4;

    // use DErule
    title("DErule");
```

```
    DErule<double(Doub, Doub)> de(f_1, a, b);
    print_de_rule(de);

    return 0;
}
```

The results can be seen in the following screenshot.

```
DErule
    i          A(hi)  A(hi-1)-A(hi)          alp^k          error         f-comp
    0      -0.103036       0.103036              0              0              3
    1          1.022       -1.12504              0              0              5
    2          1.545      -0.522994        2.15116       0.454321              9
    3        1.43325       0.111753       -4.67992      0.0196751             17
    4        1.38165      0.0515992        2.16578      -0.0442614             33
    5        1.42954      -0.0478984       -1.07726      -0.0230584             65
    6        1.43004    -0.000498842        96.0191    5.24992e-06            129
```