A math equation with text

Description automatically generated with medium confidence

# Exercise i)

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

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

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

Where is calculated with the following formula:

And

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

A screenshot of a computer screen

Description automatically generated

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.

A screenshot of a computer screen

Description automatically generated