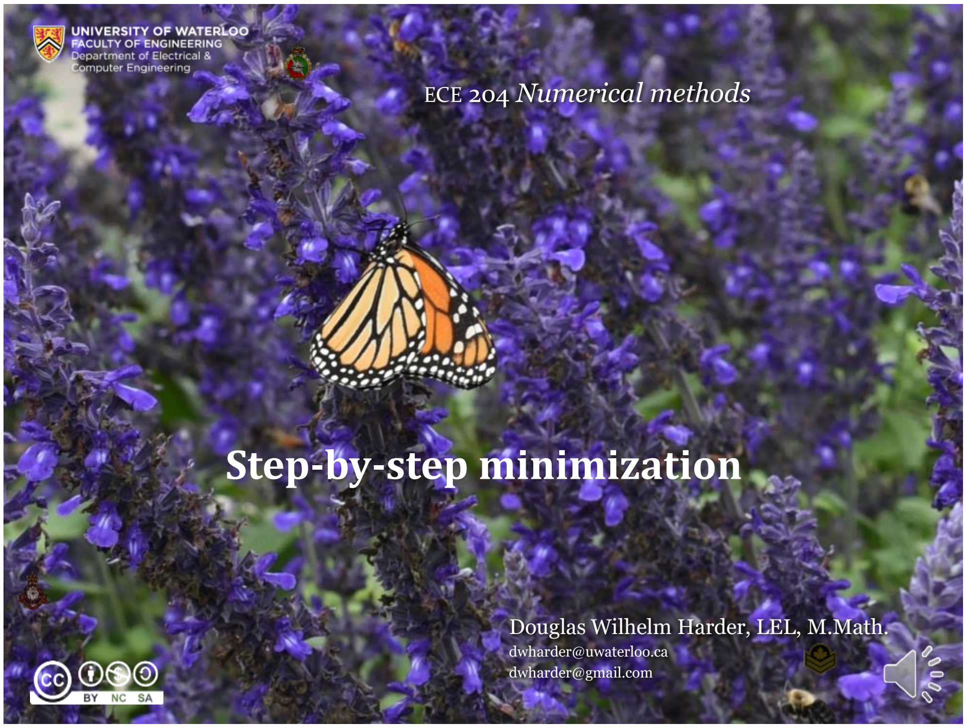



1

Optimization

# Introduction

- In this topic, we will
  - Describe step-by-step minimization
  - Discuss some issues
  - Look at an implementation
  - Look at an example

2

2

Optimization

## Step-by-step minimization

- Here is a straight-forward idea:
  - Keep stepping in the direction that appears to be going towards a minimum until you start going back up again
  - Once at this point, try again with a smaller step size

3

3

Optimization

## Step-by-step minimization

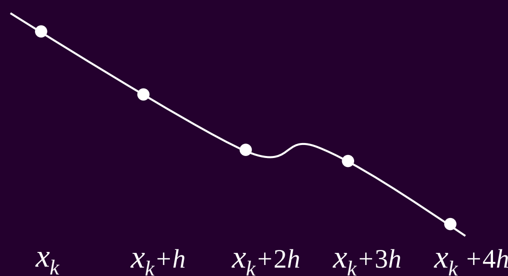
- Given a real-valued function $f(x)$, decide upon some $\varepsilon_{\text{step}}$ that will specify the accuracy required
- Choose an initial point $x_0$ and an initial step size $h$
  1. Given $x_k$, calculate $f(x_k)$, $f(x_k - h)$ and $f(x_k + h)$
     a. If $f(x_k - h) \leq f(x_k)$ and $f(x_k - h) \leq f(x_k + h)$,
        evaluate $f(x_k - 2h)$, $f(x_k - 3h)$, …
        until $f(x_k - (n + 1)h) > f(x_k - nh)$ and set $x_{k+1} \leftarrow x_k - nh$
     b. If $f(x_k + h) \leq f(x_k)$ and $f(x_k + h) \leq f(x_k - h)$,
        evaluate $f(x_k + 2h)$, $f(x_k + 3h)$, …
        until $f(x_k + (n + 1)h) > f(x_k + nh)$ and set $x_{k+1} \leftarrow x_k + nh$
     c. Otherwise, set $x_{k+1} \leftarrow x_k$
  2. If $h < \varepsilon_{\text{step}}$, we are finished;
     otherwise, divide $h$ by two and return to Step 1

4

4

# Benefits

- A nice property of this algorithm is that once it is within the vicinity of a minima, it becomes a binary search
- If, however, the minima is very far away from $x_0$ and $h$ is small, it may require a significant number of steps to find it
- Also, if $h$ is too large, we may completely miss a given minima










# Implementation

```
std::pair<double, double> step(
    double f( double x ), double x0, double h,
    double eps_step, double eps_abs,
    unsigned int max_iterations
) {
    double f0{ f( x0 ) };

    for ( unsigned int k{0}; k < max_iterations; ++k ) {
        double fn{ f( x0 - h ) };
        double fp{ f( x0 + h ) };
        double x1, f1;
```

Optimization

# Implementation

```
if ( (f0 <= fn) && (f0 <= fp) ) {
    if ( (h < eps_step) && ( (std::min(fp, fn) - f0) < eps_abs) ) {
        return std::make_pair( x0, f0 );
    }

    h /= 2.0;
    continue;
} else if ( fn <= fp ) {
    x1 = x0 - h;
    f1 = fn;
    fn = f( x1 - h );

    while ( fn < f1 ) {
        ++k;
        x1 -= h;
        f1 = fn;
        fn = f( x1 - h );
    }
} else {
```

7

7

Optimization

# Implementation

```
} else {
    x1 = x0 + h;
    f1 = fp;
    fp = f( x1 + h );

    while ( fp < f1 ) {
        ++k;
        x1 += h;
        f1 = fp;
        fp = f( x1 + h );
    }
}

if ( (std::abs( x0 - x1 ) < eps_step) && ((f0 - f1) < eps_abs) ) {
    return std::make_pair( x1, f1 );
} else {
    x0 = x1;
    f0 = f1;
    h /= 2.0;
}
    }

    return std::make_pair( NAN, NAN );
}
```

8

8

# Example

- Minimize sin($x$) with $x_0$ = 0 and $h$ = 1

| $k$ | $h$ | $x_k$ | $f(x_k)$ |
|---|---|---|---|
| 1 | 1 | –1 | –0.8414709848078965 |
| 2 | 1 | –2 | –0.9092974268256817 |
| 3 | 0.5 | –1.5 | –0.9974949866040544 |
| 4 | 0.25 | –1.5 | |
| 5 | 0.125 | –1.625 | –0.9985313405398316 |
| 6 | 0.0625 | –1.5625 | –0.9999655856782489 |
| 7 | 0.03125 | –1.5625 | |
| 8 | 0.015625 | –1.578125 | –0.9999731453947223 |
| 9 | 0.0078125 | –1.5703125 | –0.9999998829558185 |
| 10 | 0.00390625 | –1.5703125 | |
| 11 | 0.001953125 | –1.5703125 | |
| 12 | 0.0009765625 | –1.5703125 | |
| 13 | 0.00048828125 | –1.57080078125 | –0.9999999999900789 |







# Summary

- Following this topic, you now
  - Understand the step-by-step minimization technique
  - Are aware of the benefits and issues
  - Have seen an implementation
  - Have seen an example

# References

[1] https://en.wikipedia.org/wiki/Mathematical_optimization







# Acknowledgments

None so far.

# Colophon

These slides were prepared using the Cambria typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.
Mathematical equations are prepared in MathType by Design Science, Inc.
Examples may be formulated and checked using Maple by Maplesoft, Inc.

The photographs of flowers and a monarch butter appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens in October of 2017 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.







# Disclaimer

These slides are provided for the ECE 204 *Numerical methods* course taught at the University of Waterloo. The material in it reflects the author’s best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.