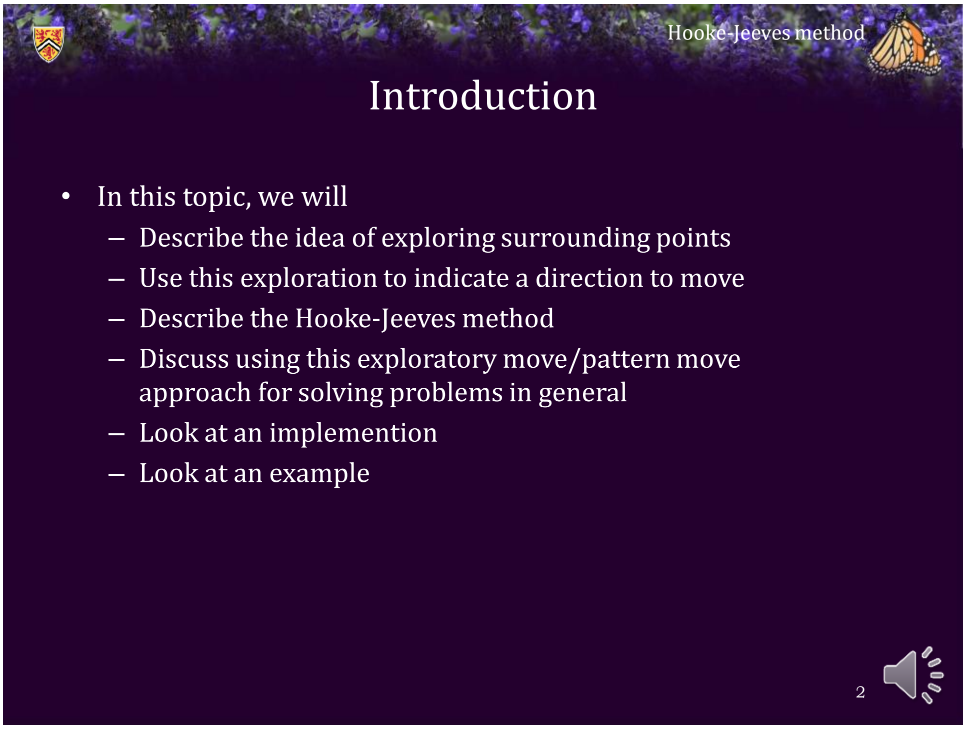
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

ECE 204 *Numerical methods*

Hooke-Jeeves method for finding extrema in *n* dimensions

Douglas Wilhelm Harder, LEL, M.Math.
dwharder@uwaterloo.ca
dwharder@gmail.com

1



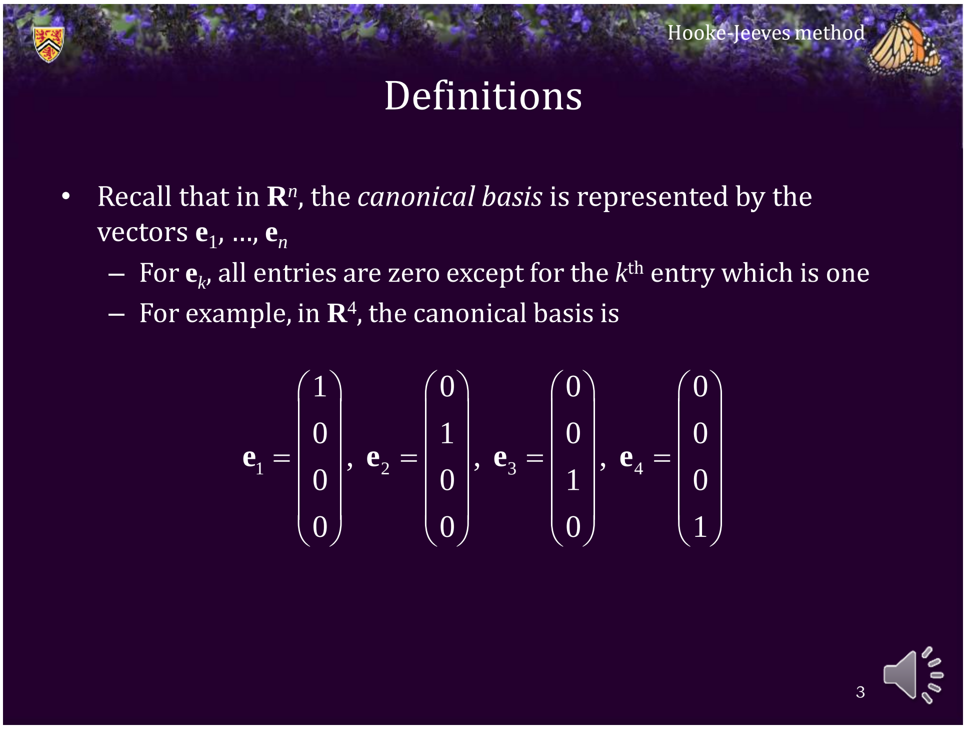
Hooke-Jeeves method

Introduction

- In this topic, we will
  - Describe the idea of exploring surrounding points
  - Use this exploration to indicate a direction to move
  - Describe the Hooke-Jeeves method
  - Discuss using this exploratory move/pattern move approach for solving problems in general
  - Look at an implemention
  - Look at an example

2

2



Hooke-Jeeves method

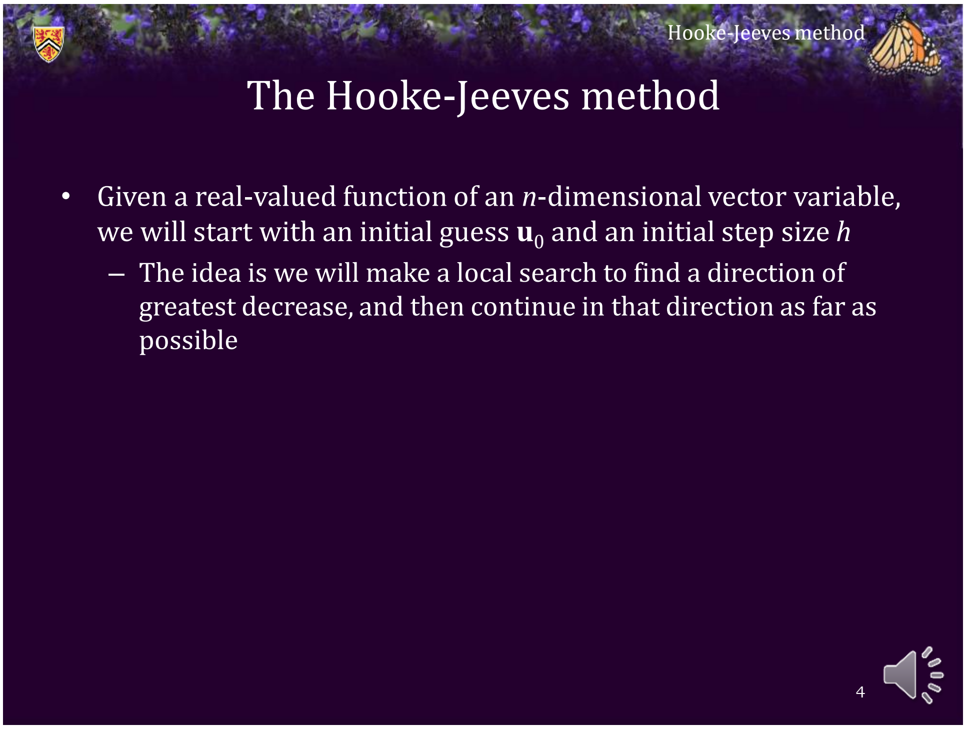
Definitions

- Recall that in $\mathbf{R}^n$, the *canonical basis* is represented by the vectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$
  - For $\mathbf{e}_k$, all entries are zero except for the kth entry which is one
  - For example, in $\mathbf{R}^4$, the canonical basis is

$$\mathbf{e}_1 = \begin{pmatrix}1\\0\\0\\0\end{pmatrix},\ \mathbf{e}_2 = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},\ \mathbf{e}_3 = \begin{pmatrix}0\\0\\1\\0\end{pmatrix},\ \mathbf{e}_4 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

3

3



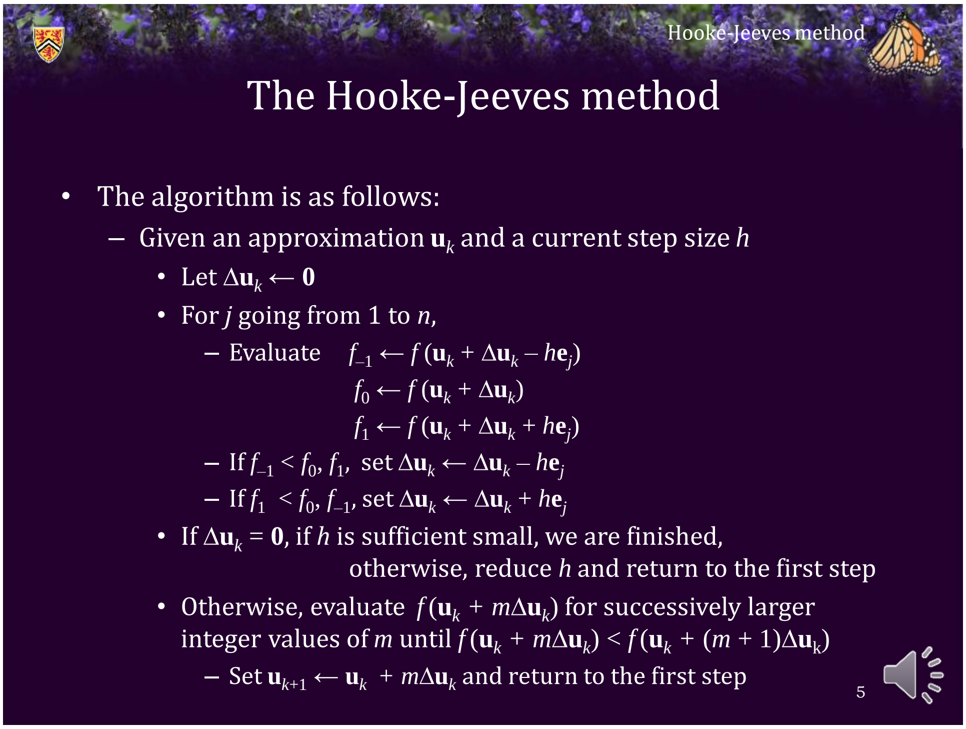
Hooke-Jeeves method

The Hooke-Jeeves method

- Given a real-valued function of an n-dimensional vector variable, we will start with an initial guess $\mathbf{u}_0$ and an initial step size h
  - The idea is we will make a local search to find a direction of greatest decrease, and then continue in that direction as far as possible

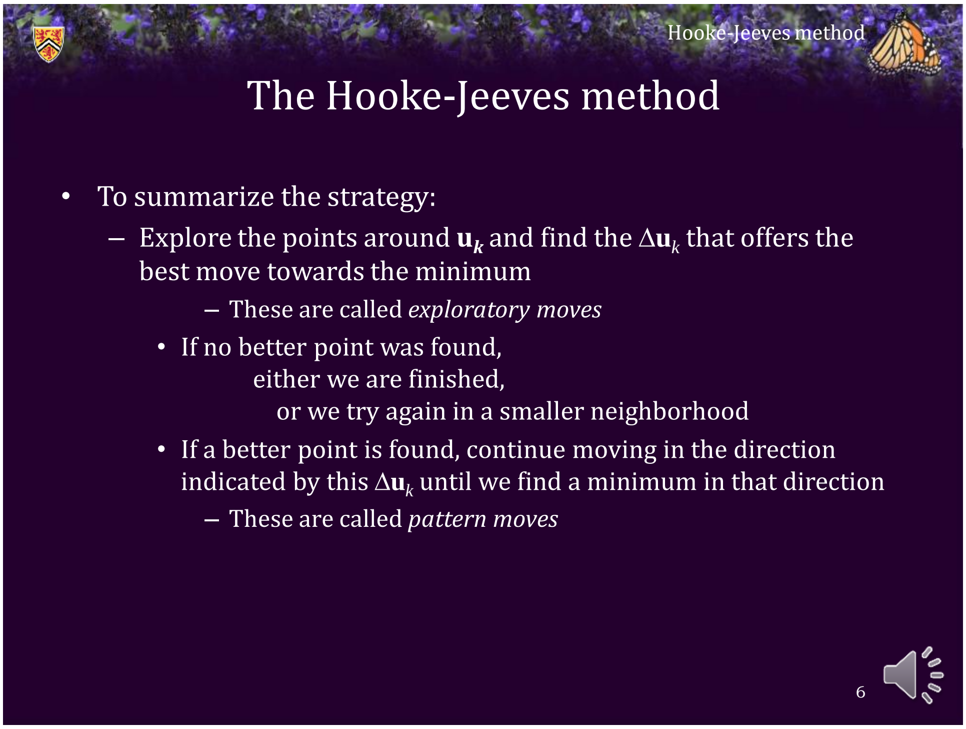
4

4



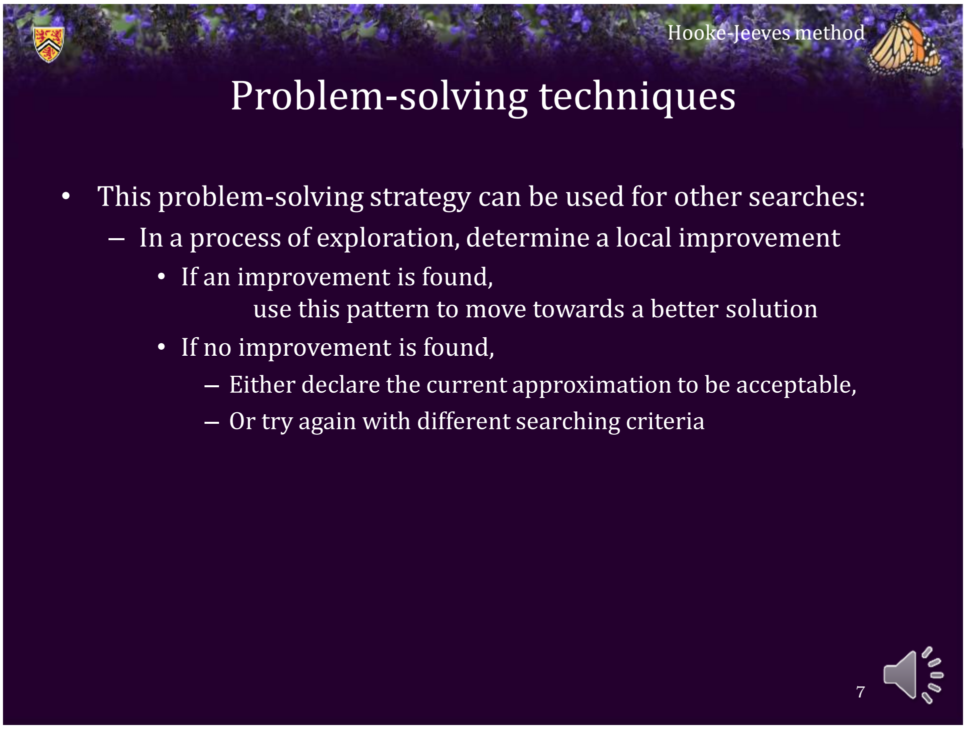
The Hooke-Jeeves method

- The algorithm is as follows:
  - Given an approximation $\mathbf{u}_k$ and a current step size h
    - Let $\Delta\mathbf{u}_k \leftarrow \mathbf{0}$
    - For j going from 1 to n,
      - Evaluate $f_{-1} \leftarrow f(\mathbf{u}_k + \Delta\mathbf{u}_k - h\mathbf{e}_j)$
        $f_0 \leftarrow f(\mathbf{u}_k + \Delta\mathbf{u}_k)$
        $f_1 \leftarrow f(\mathbf{u}_k + \Delta\mathbf{u}_k + h\mathbf{e}_j)$
      - If $f_{-1} < f_0, f_1$, set $\Delta\mathbf{u}_k \leftarrow \Delta\mathbf{u}_k - h\mathbf{e}_j$
      - If $f_1 < f_0, f_{-1}$, set $\Delta\mathbf{u}_k \leftarrow \Delta\mathbf{u}_k + h\mathbf{e}_j$
    - If $\Delta\mathbf{u}_k = \mathbf{0}$, if h is sufficient small, we are finished,
      otherwise, reduce h and return to the first step
    - Otherwise, evaluate $f(\mathbf{u}_k + m\Delta\mathbf{u}_k)$ for successively larger integer values of m until $f(\mathbf{u}_k + m\Delta\mathbf{u}_k) < f(\mathbf{u}_k + (m + 1)\Delta\mathbf{u}_k)$
      - Set $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + m\Delta\mathbf{u}_k$ and return to the first step



The Hooke-Jeeves method

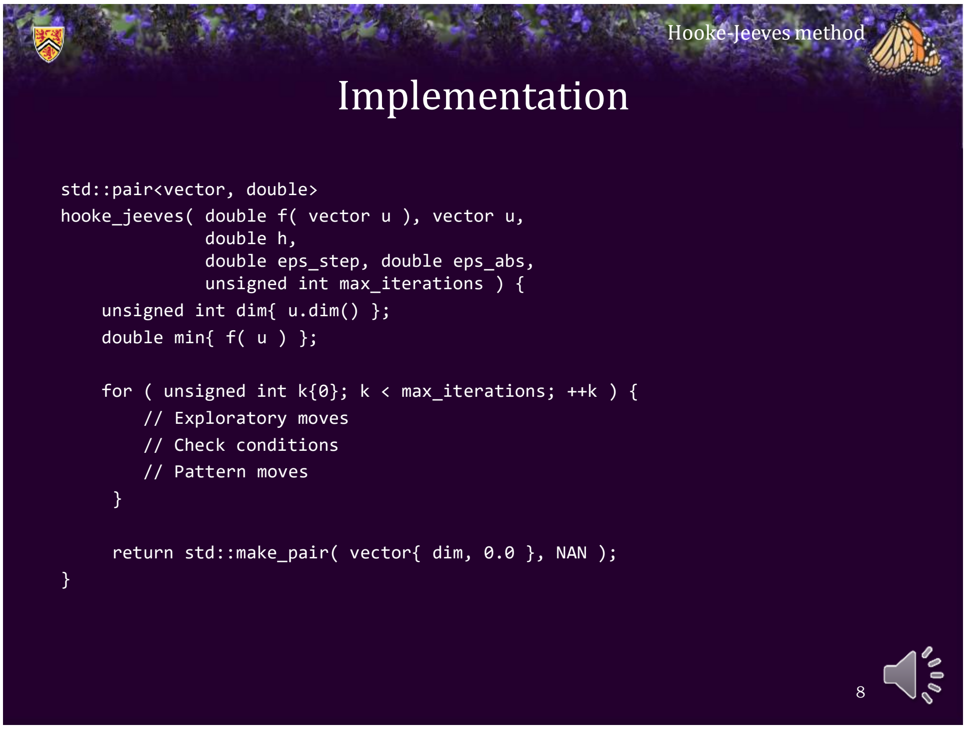
- To summarize the strategy:
  - Explore the points around $\mathbf{u}_k$ and find the $\Delta\mathbf{u}_k$ that offers the best move towards the minimum
      - These are called *exploratory moves*
    - If no better point was found,
      either we are finished,
      or we try again in a smaller neighborhood
    - If a better point is found, continue moving in the direction indicated by this $\Delta\mathbf{u}_k$ until we find a minimum in that direction
      - These are called *pattern moves*



Problem-solving techniques

- This problem-solving strategy can be used for other searches:
  - In a process of exploration, determine a local improvement
    - If an improvement is found,
      use this pattern to move towards a better solution
    - If no improvement is found,
      - Either declare the current approximation to be acceptable,
      - Or try again with different searching criteria

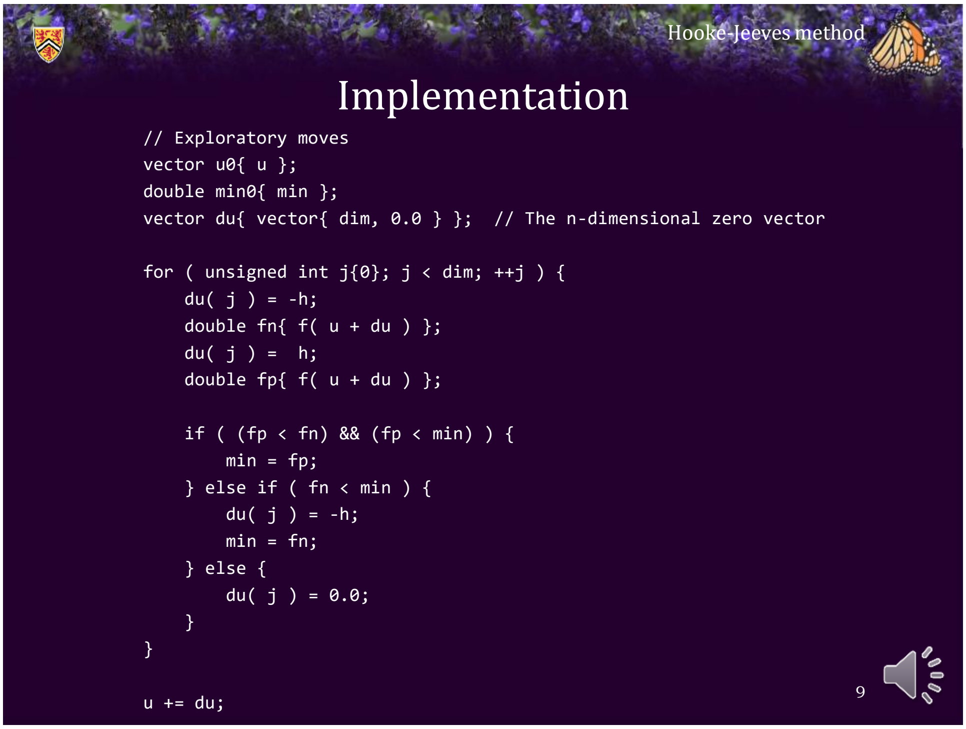
7



Implementation

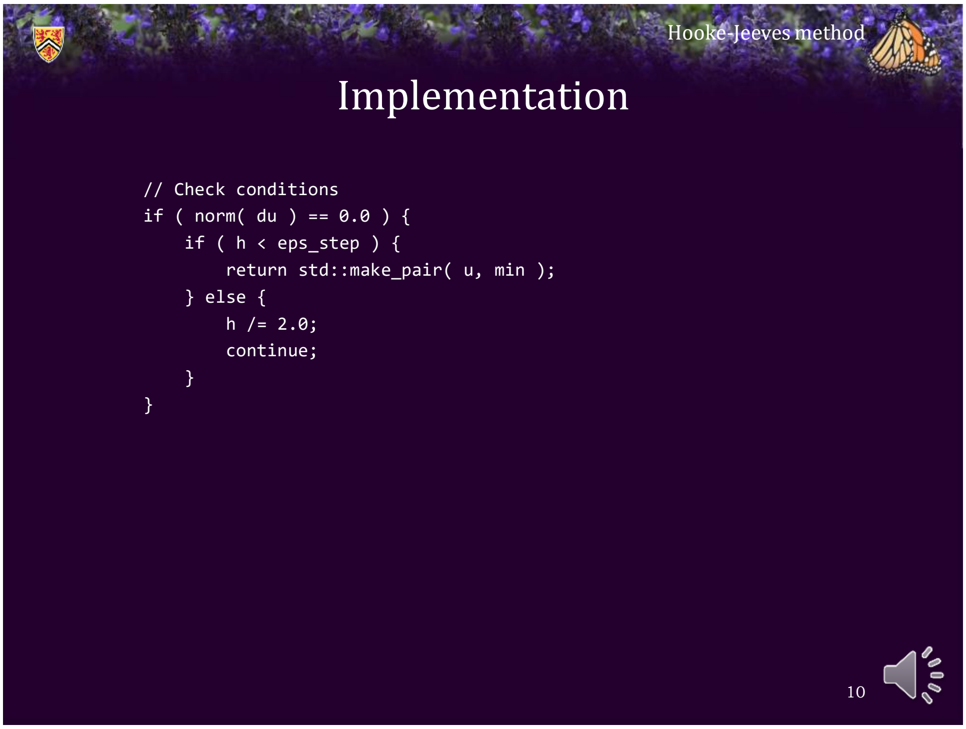
```
std::pair<vector, double>
hooke_jeeves( double f( vector u ), vector u,
              double h,
              double eps_step, double eps_abs,
              unsigned int max_iterations ) {
    unsigned int dim{ u.dim() };
    double min{ f( u ) };

    for ( unsigned int k{0}; k < max_iterations; ++k ) {
        // Exploratory moves
        // Check conditions
        // Pattern moves
     }

     return std::make_pair( vector{ dim, 0.0 }, NAN );
}
```

8



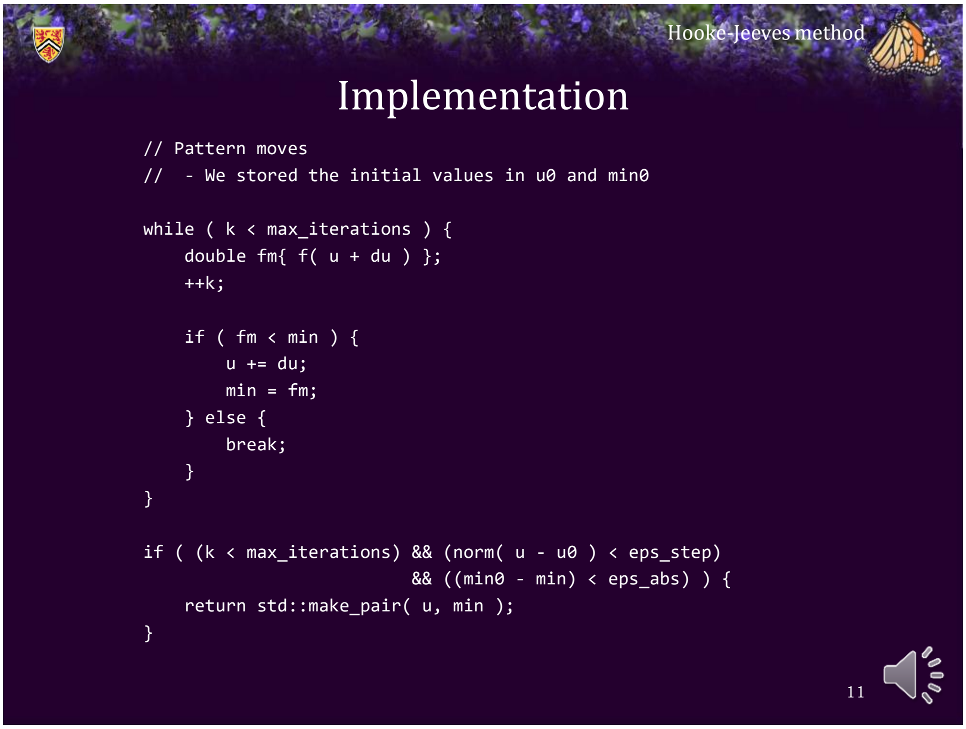
Implementation

```
// Exploratory moves
vector u0{ u };
double min0{ min };
vector du{ vector{ dim, 0.0 } };  // The n-dimensional zero vector

for ( unsigned int j{0}; j < dim; ++j ) {
    du( j ) = -h;
    double fn{ f( u + du ) };
    du( j ) =  h;
    double fp{ f( u + du ) };

    if ( (fp < fn) && (fp < min) ) {
        min = fp;
    } else if ( fn < min ) {
        du( j ) = -h;
        min = fn;
    } else {
        du( j ) = 0.0;
    }
}

u += du;
```



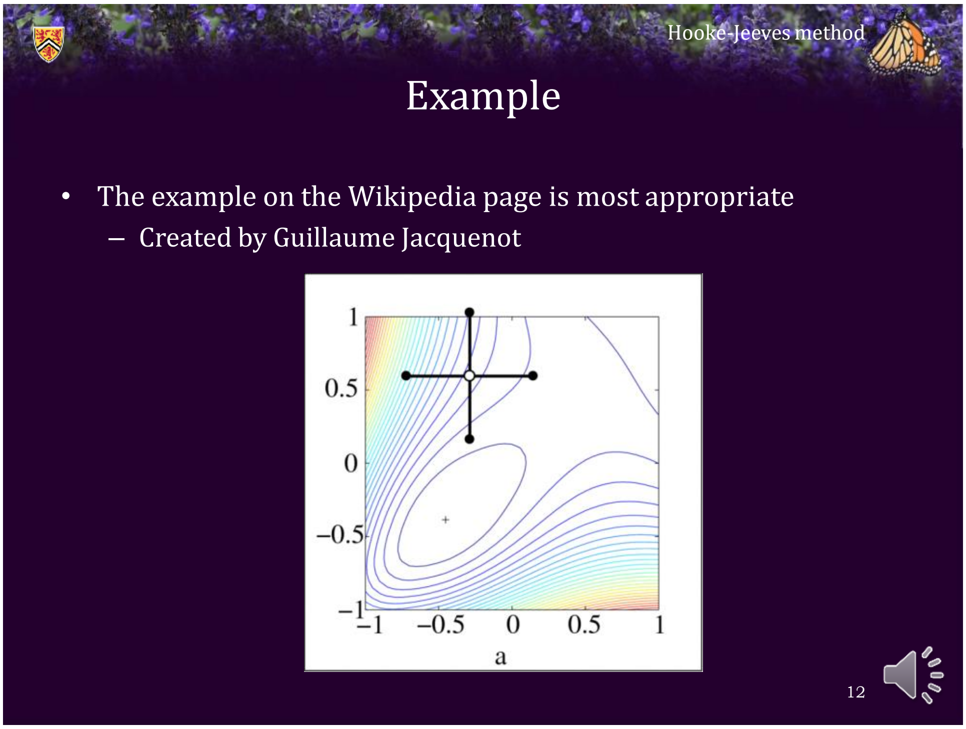
Implementation

```
// Check conditions
if ( norm( du ) == 0.0 ) {
    if ( h < eps_step ) {
        return std::make_pair( u, min );
    } else {
        h /= 2.0;
        continue;
    }
}
```



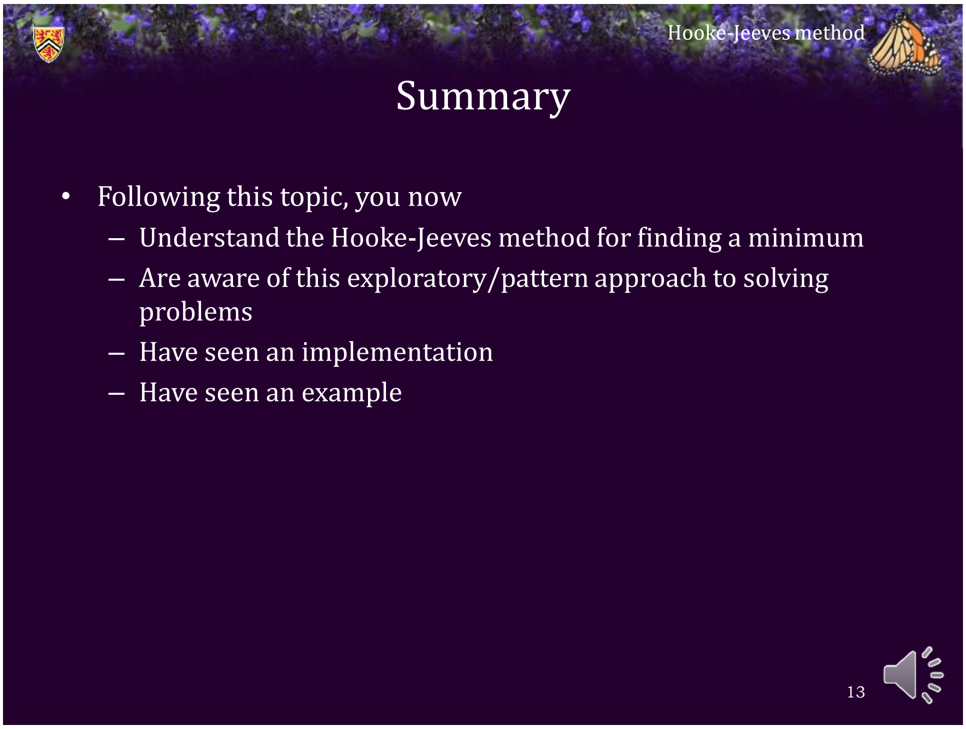
Implementation

```
// Pattern moves
//  - We stored the initial values in u0 and min0

while ( k < max_iterations ) {
    double fm{ f( u + du ) };
    ++k;

    if ( fm < min ) {
        u += du;
        min = fm;
    } else {
        break;
    }
}

if ( (k < max_iterations) && (norm( u - u0 ) < eps_step)
                          && ((min0 - min) < eps_abs) ) {
    return std::make_pair( u, min );
}
```



Example

- The example on the Wikipedia page is most appropriate
  - Created by Guillaume Jacquenot

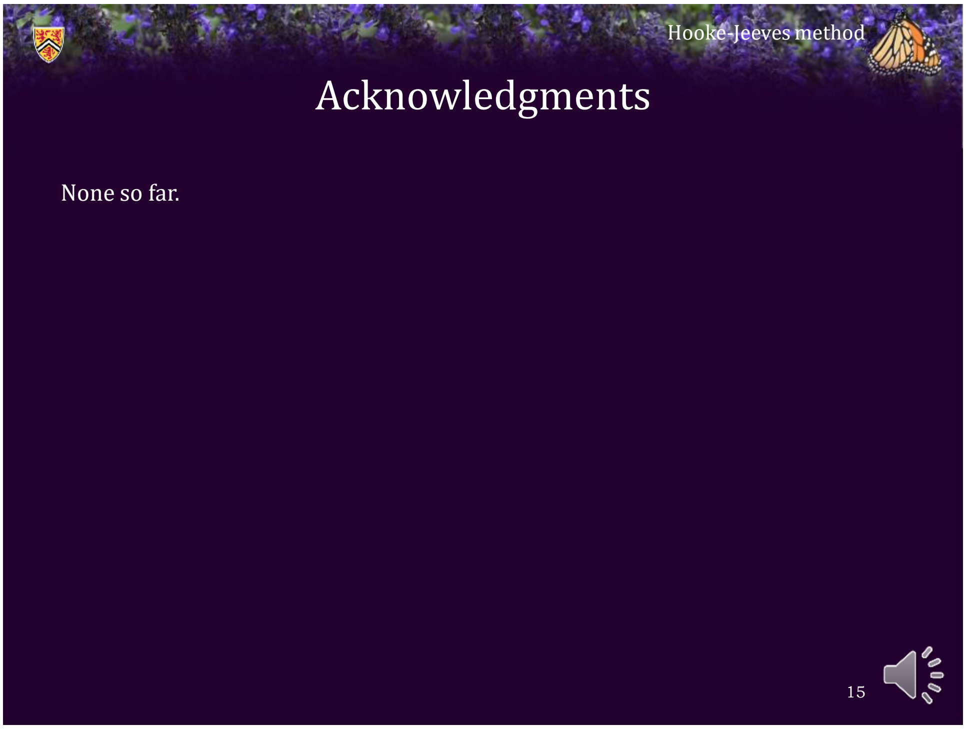


Summary

- Following this topic, you now
  - Understand the Hooke-Jeeves method for finding a minimum
  - Are aware of this exploratory/pattern approach to solving problems
  - Have seen an implementation
  - Have seen an example

References

[1] https://en.wikipedia.org/wiki/Pattern_search_(optimization)



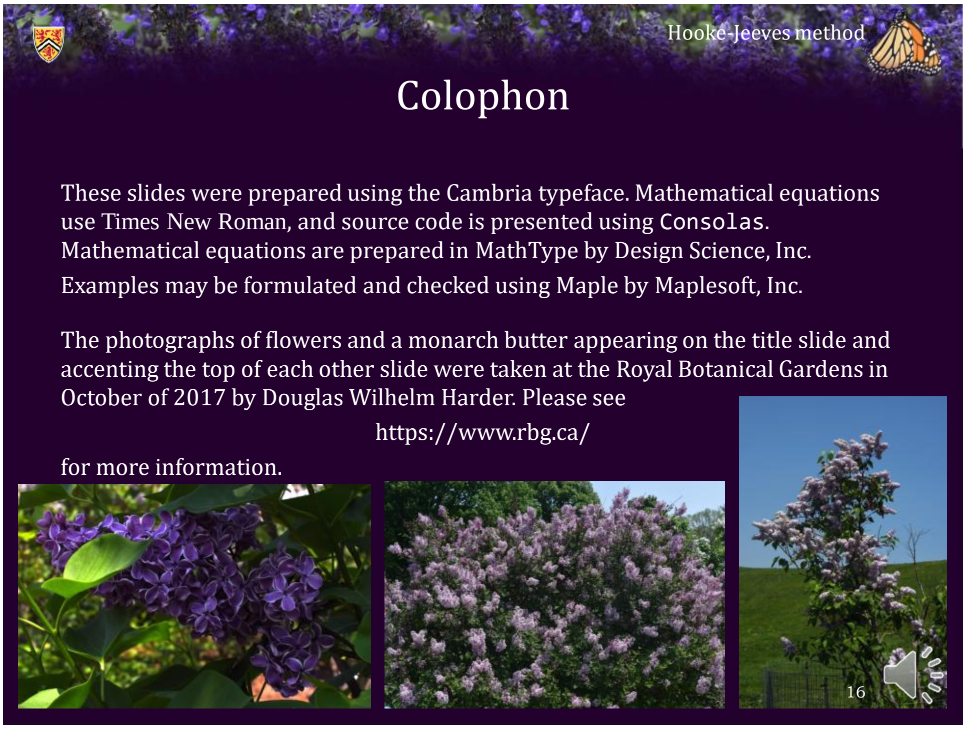
Hooke-Jeeves method

Acknowledgments

None so far.

15

15



Hooke-Jeeves method

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.

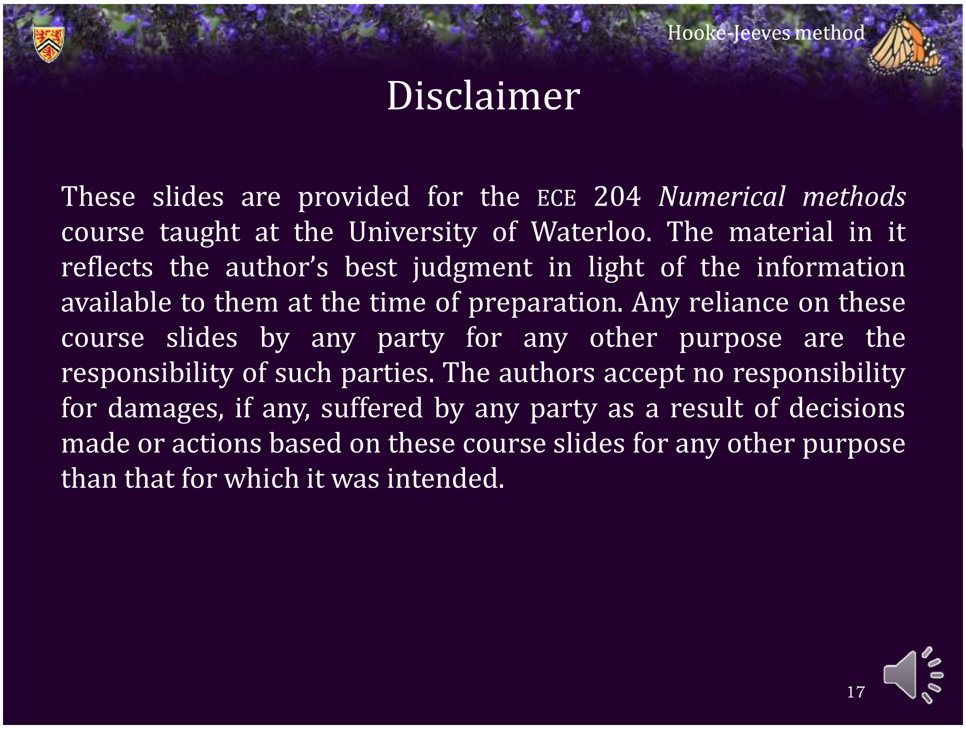
The photographs of flowers and a monarch butter appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens in October of 2017 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

16

16



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.

17