

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

ECE 204 *Numerical methods*

**Adaptive techniques for approximating  
solutions to 1<sup>st</sup>-order initial-value problems**

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

CC BY NC SA

1

Adaptive algorithms for 1st-order initial-value problems

## Introduction

- In this topic, we will
  - Describe adaptive algorithms for initial-value problem (IVP) solvers
  - List the algorithms we will see in this topic
  - Discuss making dynamic changes to the step size
    - This will include limitations
  - Discuss appropriate data structures for such algorithms


2

2

Adaptive algorithms for 1st-order initial-value problems

## Iterative methods

- In order to estimate the error, previously we had to approximate the solution with  $n$  points, and then again with  $2n$  points
  - Our error analysis suggested the error should drop by  $h$ ,  $h^2$  or  $h^4$ , depending on the algorithm
- This potentially requires us to do a significant amount of work!

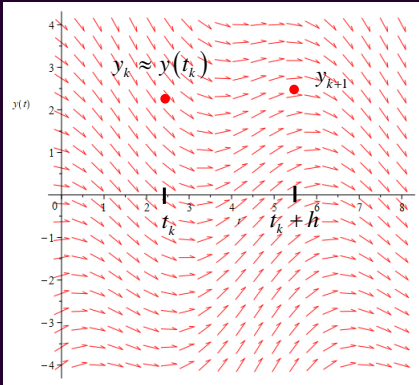
3 


3

Adaptive algorithms for 1st-order initial-value problems

## Adaptive techniques

- Instead, we will use the following approach:
  - At each step, we will try to estimate the error of the next approximation



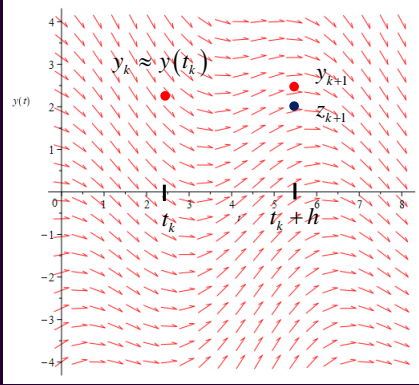
4 

4

Adaptive algorithms for 1st-order initial-value problems

## Adaptive techniques

- Thus, we proceed as follows:
  - Given an approximation  $y_k$  at a point  $t_k$ ,
    - Use two algorithms, one significantly more precise ( $z_{k+1}$ ) than the other ( $y_{k+1}$ )



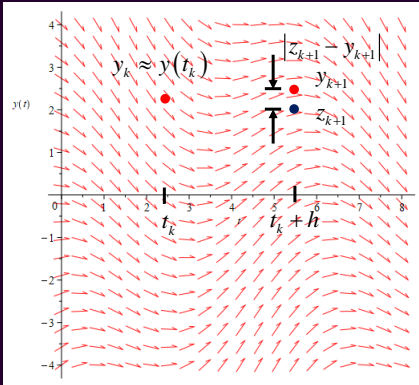
5

5

Adaptive algorithms for 1st-order initial-value problems

## Adaptive techniques

- Neither  $z_{k+1}$  nor  $y_{k+1}$  is exact, but because  $z_{k+1}$  is much more accurate, then  $|z_{k+1} - y_{k+1}|$  is a not-unreasonable approximation of the error of  $y_{k+1}$ 
  - Like before, to be conservative, we'll over-estimate the error:
 
$$2|z_{k+1} - y_{k+1}|$$



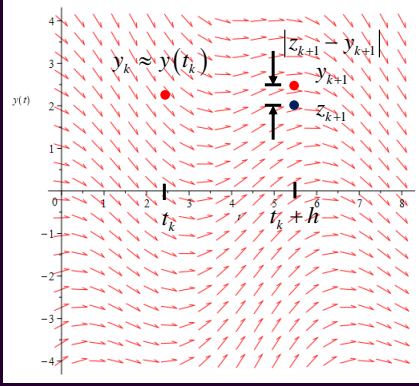
6

6

Adaptive algorithms for 1st-order initial-value problems

## Adaptive techniques

- Because  $\varepsilon_{\text{abs}}$  is the maximum error we are allowed to accept per unit time, the maximum error we are willing to accept is  $\varepsilon_{\text{abs}}h$ 
  - If  $2|z_{k+1} - y_{k+1}| < \varepsilon_{\text{abs}}h$ , we're okay
  - Otherwise, we need to try again with a smaller  $h$ ...




7

7

Adaptive algorithms for 1st-order initial-value problems

## Adaptive techniques

- Thus, we have two possibilities:
  - If  $2|z_{k+1} - y_{k+1}| < \varepsilon_{\text{abs}}h$ , our approximation is too accurate
    - We could use a larger value of  $h$  with the next step
  - If  $2|z_{k+1} - y_{k+1}| \geq \varepsilon_{\text{abs}}h$ , the error is too large
    - We need to try again with a smaller value of  $h$




8

8

Adaptive algorithms for 1st-order initial-value problems

## Adaptive techniques

- Questions:
  - In the first case, how much larger can we make  $h$ ?
  - In the second case, how much smaller must we make  $h$ ?
- Answers:
  - We will calculate a scaling factor  $a$ 
    - If  $a > 1$ , then  $h$  can be made larger, so continue
    - If  $a \leq 1$ , then  $h$  is too large, so try again
- Question:
  - Isn't  $ah$  the "ideal" step size?
- Answer:
  - It approximates the ideal step size for the current point
  - Issue: The chance is 50-50 that the next ideal step size smaller
  - Solution: Multiply  $a$  by 0.9


9 

9

Adaptive algorithms for 1st-order initial-value problems

## Looking ahead

- We will look at two adaptive algorithms:
  - In order to introduce this concept, we will look at using both Euler and Heun
  - The second algorithm is the Dormand-Prince algorithm


10 

10

Adaptive algorithms for 1st-order initial-value problems

## Warnings

- We will determine better values of  $h$ , however, these algorithms will require:
  - A minimum value of  $h_{\min}$  to use
  - A maximum value of  $h_{\max}$  based on the problem at hand
    - If  $h$  is too large, we may lose detail in the differential equation
  - An initial value of  $h$  to use
    - We will use  $\sqrt{h_{\min} h_{\max}}$ , the geometric mean
- Also, while we'd like to get  $h$  to the ideal width, we will be using approximations to change the size of  $h$ 
  - Again, being conservative:
    - We will never let  $h$  be more than doubled with any step
    - We will never let  $h$  be less than halved in any step


11 

11

Adaptive algorithms for 1st-order initial-value problems

## Data structures?

- What data structures will we use for these algorithms?
  - Problem: We don't know how many steps we will require
  - If we allocate an array, it may occur that:
    - The array is far too large, so we have wasted memory
    - The array is too small, and we must allocate a larger array and copy everything over...
      - Doubling the array capacity may result in  $O(n)$  wasted memory
  - Instead, the algorithm will use a data structure that:
    - Allows  $O(1)$  insertions with all operations
    - Has  $O(1)$  wasted memory
    - That is, a queue!
  - At the end, we will convert it back to an array

12 

12

Adaptive algorithms for 1st-order initial-value problems

## Implementation


```

std::tuple<unsigned int, double *, double *, double *> adaptive_algorithm(
    double f( double t, double y ), std::pair<double, double> t_rng, double y0,
    std::pair<double, double> h_rng, double eps_abs
) {
    assert( h_rng.first > 0.0 );
    assert( h_rng.second > h_rng.first );
    double h{ std::sqrt( h_rng.first *h_rng.second ) };

    std::queue<double> qt{};
    std::queue<double> qy{};
    std::queue<double> qdy{};

    qt.push( t_rng.first );
    qy.push( y0 );
    qdy.push( f( t_rng.first, y0 ) );

```

13 

13

Adaptive algorithms for 1st-order initial-value problems

## Implementation

```

while ( qt.back() < t_rng.second ) {
    bool found{ false };


    do {
        double y{ /* First approximation */ };
        double z{ /* Second (better) approximation */ };

        double a{ /* Calculate scaling factor for 'h' */ };

        if ( (a > 1.0) || (h == h_rng.first) ) {
            qt.push( qt.back() + h );
            qy.push( z );
            qdy.push( f( qt.back(), z ) );
            found = true;
        }

    } while ( !found );

```

14 

14

Adaptive algorithms for 1st-order initial-value problems

## Implementation


```

a *= 0.9;

if ( a >= 2.0 ) {
    h *= 2.0;
} else if ( a <= 0.5 ) {
    h *= 0.5;
} else {
    h *= a;
}

if ( h < h_rng.first ) {
    h = h_rng.first;
} else if ( h > h_rng.second ) {
    h = h_rng.second;
}
} while ( !found );
}

```

15 

15

Adaptive algorithms for 1st-order initial-value problems

## Implementation

```


unsigned int n{ static_cast<unsigned int>( qt.size() ) };

double *ts = new double[n];
double *ys = new double[n];
double *dys = new double[n];

for ( unsigned int k{0}; k < n; ++k ) {
    ts[k] = qt.front();
    qt.pop();
    ys[k] = qy.front();
    qy.pop();
    dys[k] = qdy.front();
    qdy.pop();
}

return std::make_tuple( n - 1, ts, ys, dys );
}

```

16 


16



Adaptive algorithms for 1st-order initial-value problems

## Summary

- Following this topic, you now
  - Understand what an adaptive IVP solver is
  - Know we will use two different approximations
    - The better approximation allows us to estimate the error of the worse approximation
  - Are aware we will introduce an Euler-Heun adaptive algorithm, and then we will describe the Dormand-Prince method
  - Understand that this will allow us to dynamically change the value of  $h$
  - Understand there are restrictions to both the magnitude of  $h$  and restrictions to any changes to  $h$
  - Are aware of data structure issues in implementations


17 

17

Adaptive algorithms for 1st-order initial-value problems



## References

- [1] [https://en.wikipedia.org/wiki/Adaptive\\_algorithm](https://en.wikipedia.org/wiki/Adaptive_algorithm)
- [2] [https://en.wikipedia.org/wiki/Adaptive\\_step\\_size](https://en.wikipedia.org/wiki/Adaptive_step_size)

18 


18

Adaptive algorithms for 1st-order initial-value problems

# Acknowledgments

None so far.

19 

19

Adaptive algorithms for 1st-order initial-value problems




# 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.






20 

20



Adaptive algorithms for 1st-order initial-value problems 

## 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.

21 