1

Euler's method

Introduction

- In this topic, we will
  - Derive Euler's method from Taylor series
  - Look at the technique visually
  - See the error is $O(h^2)$ for a single step
  - Look at two examples of a single step
  - See how to apply Euler's method under multiple steps
    - We will implement this in C++
  - Look at two examples of multiple steps
  - Derive the formula based on integration

2

2

Euler's method

Initial-value problems

- Suppose we have the initial-value problem (IVP):

$$y^{(1)}(t) = f\left(t, y(t)\right)$$
$$y(t_0) = y_0$$

- Recall from calculus that:

$$y(t_0 + h) = y(t_0) + y^{(1)}(t_0)h + \frac{1}{2}y^{(2)}(\tau)h^2$$

- Substitute the derivative and we have:

$$y(t_0 + h) = y(t_0) + hf\left(t_0, y(t_0)\right) + \frac{1}{2}y^{(2)}(\tau)h^2$$

- Substitute the initial condition and we have:

$$y(t_0 + h) = y_0 + hf\left(t_0, y_0\right) + \frac{1}{2}y^{(2)}(\tau)h^2$$

3

3

Euler's method

Euler's method

- Consequently, we may approximate

$$y(t_0 + h) \approx y_0 + hf\left(t_0, y_0\right)$$

  – The error is $O(h^2)$

4

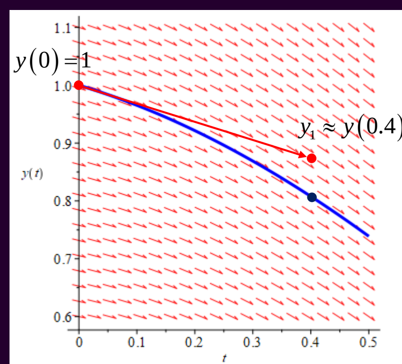
4

Euler's method

- Visually, what are we doing?

$$y^{(1)}(t) = -0.2y(t) - \sin(t) - 0.1$$
$$y(0) = 1$$

  – Estimate $y(0.4) \approx y_1$



Euler's method

- Problem:
  – This only approximates the solution at $t_0 + h$
  – How do we approximate the solution at $t_0 + 2h$ at $t_0 + 3h$?

- Suppose we say that $t_1 = t_0 + h$, so suppose we assign

$$y_1 \leftarrow y_0 + hf(t_0, y_0)$$

  – In this case, $y(t_1) \approx y_1$

- Given $t_k = t_0 + kh$, assume we have an approximation y_k
  – In this case, we can assign

$$y_{k+1} \leftarrow y_k + hf(t_k, y_k)$$

  – Thus, $y(t_{k+1}) \approx y_{k+1}$

Euler's method

One step of Euler's method

- Suppose, for example, we are approximating a solution to:

$$y^{(1)}(t) = -y(t)$$
$$y(0) = 1$$

  – Let $h = 0.1$, in which case, we can calculate

7

7

Euler's method

One step of Euler's method

- Suppose, for example, we are approximating a solution to:

$$y^{(1)}(t) = -y(t) \qquad f(t, y) = -y$$
$$y(0) = 1$$

- Let $h = 0.1$, in which case, we can approximate

$$y(0.1) \approx y_1 \leftarrow y_0 + 0.1 \cdot f(t_0, y_0) = 1 + 0.1 \cdot (-1) = 0.9$$
$$y(0.2) \approx y_2 \leftarrow y_1 + 0.1 \cdot f(t_1, y_1) = 0.9 + 0.1 \cdot (-0.9) = 0.81$$
$$y(0.3) \approx y_3 \leftarrow y_2 + 0.1 \cdot f(t_2, y_2) = 0.81 + 0.1 \cdot (-0.81) = 0.729$$

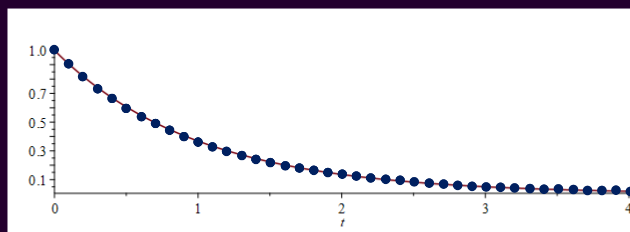
$$y(t) = e^{-t} \qquad \begin{aligned} y(0.1) &= 0.904837418035960 \\ y(0.2) &= 0.818730753077982 \\ y(0.3) &= 0.740818220681718 \end{aligned}$$

8

8

One step of Euler's method

- Thus, we can approximate the solution at 0.1, 0.2, 0.3, 0.4, …, 4.0



One step of Euler's method

- Let us observe that the error is O(h^2)
  - We will look at two initial-value problems and approximate $y(t_0 + h)$ for successively smaller values of h

One step of Euler's method

- Let's approximate the solution at $y(0 + h)$ to

$$y^{(1)}(t) = -y(t)$$
$$y(0) = 1$$

n	$h = 2^{-n}$	Exact	Approximation	Error	$\frac{1}{2}y^{(2)}(0)h^2$	Ratio
1	0.5	0.606530659712633	0.5	0.1065	0.1250	
2	0.25	0.7788007830714049	0.75	0.02880	0.03125	0.2704
3	0.125	0.8824969025845955	0.875	0.007497	0.007812	0.2603
4	0.0625	0.9394130628134758	0.9375	0.001913	0.001953	0.2552
5	0.03125	0.9692332344763441	0.96875	0.0004832	0.0004883	0.2526
6	0.015625	0.9844964370054085	0.984375	0.0001214	0.0001221	0.2513
7	0.0078125	0.9922179382602435	0.9921875	0.00003044	0.00003052	0.2507
8	0.00390625	0.9961013694701175	0.99609375	0.000007619	0.000007629	0.2503
9	0.001953125	0.9980487811074755	0.998046875	0.000001906	0.000001907	0.2502
10	0.0009765625	0.9990239141819757	0.9990234375	0.0000004767	0.0000004768	0.2501

One step of Euler's method

- Let's approximate the solution at $y(0 + h)$ to

$$y^{(1)}(t) = -0.2y(t) - \sin(t) - 0.1$$
$$y(0) = 1$$
$$y(t) = \frac{-13 + 25\cos(t) - 5\sin(t) + 14e^{-\frac{t}{5}}}{26}$$

n	$h = 2^{-n}$	Exact	Approximation	Error	$\frac{1}{2}y^{(2)}(0)h^2$	Ratio
1	0.5	0.738852315643913	0.85	–0.1111	–0.1175	
2	0.25	0.8962693342116731	0.925	–0.02873	–0.02938	0.2585
3	0.125	0.9552271839309132	0.9625	–0.007273	–0.007344	0.2531
4	0.0625	0.9794223225078814	0.98125	–0.001828	–0.001836	0.2513
5	0.03125	0.9901670100357426	0.990625	–0.0004580	–0.0004590	0.2506
6	0.015625	0.9951978758220640	0.9953125	–0.0001146	–0.0001147	0.2503
7	0.0078125	0.9976275785667972	0.99765625	–0.00002867	–0.00002869	0.2501
8	0.00390625	0.9988209552460877	0.998828125	–0.000007170	–0.000007172	0.2501
9	0.001953125	0.9994122698263201	0.9994140625	–0.000001793	–0.000001793	0.2500
10	0.0009765625	0.9997065830522891	0.99970703125	–0.0000004482	–0.0000004482	0.2500

Multiple steps of Euler's method

- Issue: As with integration, we need to approximate the solution on an interval
  - First, we will implement a function to find the *n* approximations by dividing a range $[t_0, t_f]$ into *n* sub-intervals
  - For two IVPs with the initial condition $y(0) = 1$,
    we will approximate $y(5)$ by using 2^n intervals

Implementation

- The implementation is straight-forward:

```
std::tuple<double *, double *, double *> euler(
    double f( double t, double y ), std::pair<double, double> t_rng, double y0,
    unsigned int n
) {
    double h{ (t_rng.second - t_rng.first)/n };

    double  *ts{ new double[n + 1] };
    double  *ys{ new double[n + 1] };
    double *dys{ new double[n + 1] };

    ts[0] = t_rng.first;
    ys[0] = y0;
    dys[0] = f( ts[0], ys[0] );

    for ( unsigned int k{0}; k < n; ++k ) {
         ts[k + 1] = t_rng.first + h*(k + 1);     // ts[k + 1] = ts[k] + h;
         ys[k + 1] = ys[k]      + h*dys[k];
        dys[k + 1] = f( ts[k + 1], ys[k + 1] );
    }

    return std::make_tuple( ts, ys, dys );
}
```

Implementation

- We can now use this as follows:

```
int main() {
    std::cout.precision( 16 );

    // Initial condition y(0.0) = 1.0
    auto soln{ euler( f0, std::make_pair( 0.0, 5.0 ), 1.0, 100 ) };

    std::cout << "y(4.0) = " << std::get<1>( soln )[100]
              << std::endl;
    delete[] std::get<0>( soln );
    delete[] std::get<1>( soln );
    delete[] std::get<2>( soln );

    return 0;
}

// The differential equation
//                 2
//     y'(t) = -t y(t) - (t + 1) y(t) - t + 1
double f0( double t, double y ) {
    return -t*t*y - (t + 1)*y - t + 1;
}
```


Multiple steps of Euler's method

- Let's approximate the solution at $y(5)$ to

$$y^{(1)}(t) = -y(t)$$

$$y(0) = 1$$

n	Approximation	Error	Ratio
2	2.25	–2.243	
4	0.00390625	0.002832	– 0.00126
8	0.0003910660743713379	0.006347	2.2414
16	0.00249093984716035	0.004247	0.6691
32	0.004353526043772298	0.002384	0.5614
64	0.005482901073904813	0.001255	0.5264
128	0.006095062303453198	0.0006429	0.5122
256	0.006412709688676666	0.0003252	0.5059
512	0.00657438539362522	0.0001636	0.5029
1024	0.006655931188587414	0.00008202	0.5014
	0.006737946999085467		

Multiple steps of Euler's method

- Let's approximate the solution at $y(5)$ to

$$y^{(1)}(t) = -0.2y(t) - \sin(t) - 0.1$$
$$y(0) = 1$$

$$y(t) = \frac{-13 + 25\cos(t) - 5\sin(t) + 14e^{-\frac{t}{5}}}{26}$$

n	Approximation	Error	Ratio
2	–1.621180360259891	1.776	
4	–0.539261422153926	0.6945	0.3910
8	–0.1586451131346017	0.3139	0.4520
16	0.005169109098356661	0.1501	0.4781
32	0.08176730598490181	0.07348	0.4896
64	0.118879447723685	0.03637	0.4950
128	0.1371550215458734	0.018095	0.4975
256	0.1462246406606689	0.009025	0.4988
512	0.1507426484301704	0.004507	0.4994
1024	0.152997481619969	0.002252	0.4997

0.1552495456267901

17

17

Error analysis

- For the same reason integration error is reduced, so is the error for reduced
  - Given n intervals, at each step we have an error, and that error accumulates
  - We have $\sum_{k=1}^{n}\left(\frac{1}{2}y^{(2)}(\tau_k)h^2\right)$ where $t_{k-1} < \tau_k < t_k$
  - Thus, $\sum_{k=1}^{n}\left(\frac{1}{2}y^{(2)}(\tau_k)h^2\right) = \frac{h^2}{2}\sum_{k=1}^{n} y^{(2)}(\tau_k)$

$$= \frac{nh^2}{2}\sum_{k=1}^{n}\frac{1}{n}y^{(2)}(\tau_k)$$

$$= (t_n - t_0)\frac{h}{2}y^{(2)}(\tau) \qquad t_0 < \tau < t_n$$

$$h = \frac{t_n - t_0}{n}$$

$$nh = t_n - t_0$$

18

18

Integration?

- Recall I mentioned that finding a solution to an initial-value problem is equivalent to performing an integration

$$y^{(1)}(t) = f(t, y(t))$$
$$y(t_0) = y_0$$
$$y(t_0 + h) = y(t_0) + \int_{t_0}^{t_0+h} f(\tau, y(\tau))\,d\tau$$
$$\approx y(t_0) + \int_{t_0}^{t_0+h} f(t_0, y(t_0))\,d\tau$$
$$= y_0 + \int_{t_0}^{t_0+h} f(t_0, y_0)\,d\tau$$
$$= y_0 + hf(t_0, y_0)$$

Summary

- Following this topic, you now
  - Understand Euler's method for approximating a solution to a 1st-order initial-value problem
  - Are aware of a visual interpretation with respect to slopes
  - Understand the error is $O(h^2)$ for a single step
  - Are aware that we must apply this technique multiple times to estimate the solution on a larger interval
  - Know that the error drops in this case to $O(h)$
  - Have seen a number of examples and an implementation
  - Understand the derivation and equivalence to Riemann sums

References

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

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.

23

23

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.

24

24