1

Introduction

- In this topic, we will
  - Review initial-value problems (IVPs)
  - Discuss the differences in approaches to finding or approximating solutions to IVPs
  - Introduce cubic splines
  - Describe the upcoming lectures

2

2

Initial-value problems

- An initial-value problem (IVP) is can be:
  - The first derivative described in terms of the independent variable and the function

$$y^{(1)}(t) = f\left(t, y(t)\right) \qquad y^{(1)}(t) = ty(t) + t - 1$$

$$y(t_0) = y_0 \qquad y(0) = 1$$

  - The n^{th} derivative described in terms of the independent variable, lower derivatives and the function

$$y^{(n)}(t) = f\left(t, y(t), y^{(1)}(t), \ldots, y^{(n-1)}(t)\right)$$

$$y(t_0) = y_0 \qquad y^{(3)}(t) = y^{(2)}(t) + 2y^{(1)}(t)\,y(t) + \sin(t)$$

$$y^{(1)}(t_0) = y_0^{(1)} \qquad y(1) = 2$$

$$\vdots \qquad y^{(1)}(1) = 3$$

$$y^{(n-1)}(t_0) = y_0^{(n-1)} \qquad y^{(2)}(1) = 4$$

Approximating solutions to initial-value problems

Initial-value problems

- A system of coupled IVPs, for example

$$y_1^{(1)}(t) = 0.02y_1(t) - 0.1y_1(t)\,y_2(t)$$

$$y_2^{(1)}(t) = -0.04y_2(t) + 0.02y_1(t)\,y_2(t)$$

$$y_1(0) = 5233$$

$$y_2(0) = 323$$

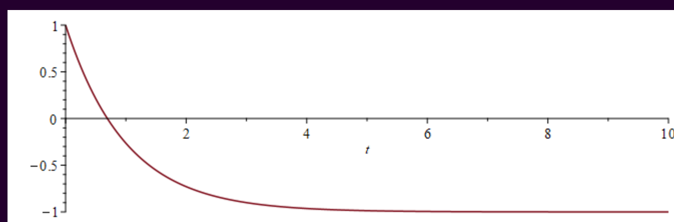
Solutions to IVPs

- Recall your approach in calculus:

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

- In calculus, you find a single exact solution:

$$y(t) = 2e^{-t} - 1$$



  – What if you cannot find an exact solution?

Approximate solutions to IVPs

- What do we have?

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

  – At time $t = 0$, the value is 1
  – The first equation says:
    - If $t = 0$ and $y(0) = 1$, then $y^{(1)}(0) = -0 \cdot 1 - 1 = -1$
  – Taylor series now say that:

$$y(0 + h) \approx y(0) + y^{(1)}(0)h$$
$$= \quad 1 \quad + (-1)h$$

    - Thus, $y(0.1) \approx 0.9$
  - If $t = 0.1$ and $y(0.1) = 0.9$, then $y^{(1)}(0) = -0.1 \cdot 0.9 - 1 = -1.09$
    - Thus $y(0.2) = y(0.1) + y^{(1)}(0.1) = 0.9 + (-1.09)\ 0.1 = 0.791$

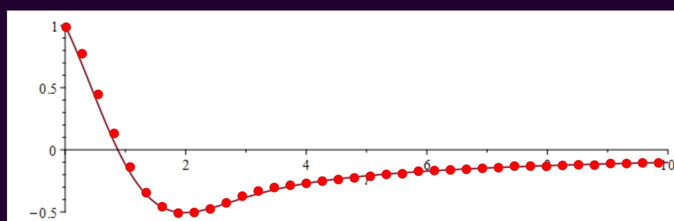
Approximate solutions to IVPs

- In this course,
  we will approximate the solution at specific points:

  $$(t_0, y(t_0)), (t_1, y_1), (t_2, y_2), (t_3, y_3), \ldots$$

  – Thus, $y(t_k) \approx y_k$

This is the initial condition



Approximating at intermediate values of *t*

- Suppose we want to approximate the solution at some point

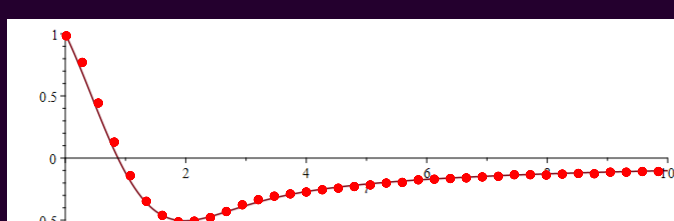
  $$t_{k-1} < t < t_k$$

  – Do we find the interpolating linear polynomial between

  $$(t_{k-1}, y_{k-1}) \text{ and } (t_k, y_k)?$$

  – Do we find the interpolating cubic polynomial between

  $$(t_{k-2}, y_{k-2}), (t_{k-1}, y_{k-1}), (t_k, y_k) \text{ and } (t_{k+1}, y_{k+1})?$$



Approximating solutions to initial-value problems

Interpolating cubic polynomials

- Let's implement this function
  - We assume $t_k - t_{k-1} = h$

$$\begin{pmatrix} -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \end{pmatrix} \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_{k-2} \\ y_{k-1} \\ y_k \\ y_{k+1} \end{pmatrix}$$

```
double ivp_interp_4pt( double t,
                       double ts[4],
                       double ys[4] ) {
    double delta{ (t - ts[1])/(ts[2] - ts[1]) };
    assert( (0.0 <= delta) && (delta <= 1.0) );

    return (
        (
            (
                (ys[3] - ys[0])/6.0 + (ys[1] - ys[2])/2.0
            )*delta + ((ys[0] + ys[2])/2.0 - ys[1])
        )*delta + (-ys[3]/6.0 + ys[2] - ys[1]/2.0 - ys[0]/3.0)
    )*delta + ys[1];
}
```

9

9

Approximating solutions to initial-value problems

Splines

- Recall that $y^{(1)}(t) = f(t, y(t))$, so

$$y^{(1)}(t_{k-1}) = f(t_{k-1}, y_{k-1}) \text{ and } y^{(1)}(t_k) = f(t_k, y_k)$$

  - Can we find a cubic polynomial p that satisfies:

$$p(t_{k-1}) = y_{k-1}$$
$$p^{(1)}(t_{k-1}) = f(t_{k-1}, y_{k-1})$$
$$p(t_k) = y_k$$
$$p^{(1)}(t_k) = f(t_k, y_k)$$

$$p(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$p^{(1)}(t) = 3a_3t^2 + 2a_2t + a_1$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ hf(t_{k-1}, y_{k-1}) \\ y_k \\ hf(t_k, y_k) \end{pmatrix}$$

10

10

Splines

- This is now fun:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix}\begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ hf\left(t_{k-1}, y_{k-1}\right) \\ y_k \\ hf\left(t_k, y_k\right) \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 3 & 2 & 0 & 0 \end{pmatrix}\begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ hf\left(t_{k-1}, y_{k-1}\right) \\ y_k - y_{k-1} - hf\left(t_{k-1}, y_{k-1}\right) \\ h\left(f\left(t_k, y_k\right) - f\left(t_{k-1}, y_{k-1}\right)\right) \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ hf\left(t_{k-1}, y_{k-1}\right) \\ y_k - y_{k-1} - hf\left(t_{k-1}, y_{k-1}\right) \\ h\left(f\left(t_k, y_k\right) + f\left(t_{k-1}, y_{k-1}\right)\right) + 2\left(y_{k-1} - y_k\right) \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ hf\left(t_{k-1}, y_{k-1}\right) \\ 3\left(y_k - y_{k-1}\right) - h\left(2f\left(t_{k-1}, y_{k-1}\right) + f\left(t_k, y_k\right)\right) \\ h\left(f\left(t_k, y_k\right) + f\left(t_{k-1}, y_{k-1}\right)\right) + 2\left(y_{k-1} - y_k\right) \end{pmatrix}$$

Splines

- Let's implement this function:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ hf\left(t_{k-1}, y_{k-1}\right) \\ 3\left(y_k - y_{k-1}\right) - h\left(2f\left(t_{k-1}, y_{k-1}\right) + f\left(t_k, y_k\right)\right) \\ h\left(f\left(t_k, y_k\right) + f\left(t_{k-1}, y_{k-1}\right)\right) + 2\left(y_{k-1} - y_k\right) \end{pmatrix}$$

```
double ivp_spline_2pt( double t,     double  ts[2],
                       double ys[2], double dys[2] ) {
    double     h{ ts[1] - ts[0] };
    double delta{ (t - ts[0])/h };
    assert( (0.0 <= delta) && (delta <= 1.0) );

    return (
        (
            (
                h*(dys[0] + dys[1]) + 2.0*(ys[0] - ys[1])
            )*delta - (h*(2.0*dys[0] + dys[1]) + 3.0*(ys[0] - ys[1]))
        )*delta + h*dys[0]
    )*delta + ys[0];
}
```

Approximating at intermediate values of *t*

- Which is better?
  - We will take an IVP to which we know the solution and find:
    1. The linear polynomial interpolating
       (0.20, *y*(0.20)), (0.25, *y*(0.25))
    2. The cubic polynomial interpolating
       (0.15, *y*(0.15)), (0.20, *y*(0.20)), (0.25, *y*(0.25)), (0.30, *y*(0.30))
    3. The cubic spline
       (0.20, *y*(0.20)), (0.25, *y*(0.25))
  - We will then evaluate the actual solution and these approximations at the point $t = 0.2353243$

Approximating at intermediate values of *t*

- First, let's start the 1st-order IVP:

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

  - Here, *y*(0.2353243) = 0.7903145090700692

Linear interpolating polynomial:	0.7905207882879153
	0.0002062
Cubic interpolating polynomial:	0.7903144140636057
	0.00000009501
Cubic spline:	0.7903144140636057
	0.000000008924

Approximating at intermediate values of *t*

- Next, let's consider :

$$y^{(1)}(t) = \left(t - y(t) + 1\right)\left(y(t) - t\right) \qquad y(t) = t + \frac{1}{2} + \frac{1}{2}\sqrt{3}\tan\left(\frac{\pi - 3\sqrt{3}t}{6}\right)$$

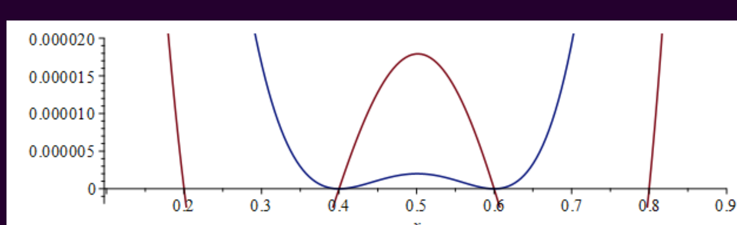
$$y(0) = 1$$

  – Here, $y(0.2353243) = 1.022125607413852$

Linear interpolating polynomial:	1.022252377336976
	0.0001268
Cubic interpolating polynomial:	1.022125194141359
	0.0000004133
Cubic spline:	1.022125568692043
	0.00000003872

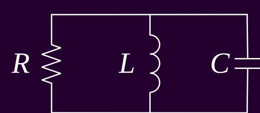
Approximating at intermediate values of *t*

- Also, we can do this with any continuous and differentiable function:
  – Given the sine function, here we see the error of:
    - A cubic polynomial interpolating the values 0.2, 0.4, 0.6, 0.8
    - A cubic spline matching the values and derivatives at 0.4 and 0.6
  – The error of the spline is smaller by a factor of 10



Our approach

- We will begin by approximating the solution to a 1st-order IVP
  - The techniques used here will trivially generalize to allow us to:
    - A system of n coupled 1st-order IVPs

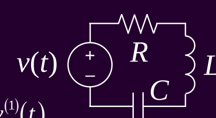


$$v^{(1)}(t) = -\frac{v(t)}{RC} - \frac{i_L(t)}{C}$$

$$i_L^{(1)}(t) = \frac{v(t)}{L}$$

    - An n^{th}-order IVP

$$\theta^{(2)}(t) = -\frac{g}{L}\sin(\theta(t))$$



$$i^{(2)}(t) + \frac{R}{L}i^{(1)}(t) + \frac{1}{CL}i(t) = \frac{1}{L}v^{(1)}(t)$$

    - A system of higher-order coupled IVPs

17

17

Looking ahead

- To approximate a solution to a 1st-order IVP, we will look at:
  - Euler's method
  - Heun's method
  - 4th-order Runge Kutta
  - Adaptive Euler-Heun
  - Dormand-Prince method
  - Stiff ODEs and backward Euler
- We will then generalize these algorithms to approximate the solution to a system of 1st-order coupled IVPs
- We will use such an approach to approximate the solution to an n^{th}-order IVP
- We will then see it is trivial to approximate the solution to a system of higher-order IVPs

18

18

Summary

- Following this topic, you now
  - Understand the various types of initial-value problems
  - Are aware of the approach we will use
  - Know about splines as opposed to interpolating polynomials
  - Are aware that we will approximate solutions to:
    - 1^{st}-order IVPs
    - Systems of 1^{st}-order IVPs
    - Higher-order IVPs
    - Systems of higher-order IVPs

References

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

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.

23