Discrete Functions, Euler's Method, 1st Order DE

1 DISCRETE FUNCTIONS

Discretize the following functions $f(t)$ over the interval I with step size h. Answers must be vectors of numbers.

(**a**) $f(t) = 2t+1$, $I = [-1,3]$, $h = 1$

(**b**) $f(t) = t^2$, $I = [-1,0]$, $h = 1/4$

(**c**) $f(t) = \sin(t)$, $I = [0,\pi]$, $h = \pi/4$

(**d**) $f(t) = \cos(t)$, $I = [0,\pi]$, $h = \pi/3$

(**e**) $f(t) = \delta(t)$, $I = [-1,1]$, $h = 1/2$

(**f**) $f(t) = \delta(t - 1/2)$, $I = [-1,1]$, $h = 1/2$

2 EULER'S METHOD

I. Use Euler's Method to solve the following initial value problems over the interval $I = [-2,2]$ with
(i) step size $h = 2$ **(ii)** step size $h = 1$.

(a) $y' = 2t$ with $y(-2) = 0$.

(b) $y' = 2t+1$ with $y(-2) = 2$.

(b') $y' = 2t+1$ with $y(2) = 6$. (Backwards Euler)

(c) $y' = \dfrac{2y}{t+3}$ with $y(-2) = 1$.

(d) $y' = 3(t+3) + \dfrac{y}{t+3}$ with $y(-2) = -6$.

II. Compare your answers above to the discretization of their continuous solutions on $I = [-2,2]$.

(a) $y = t^2 - 4$

(b) $y = t^2 + t$

(c) $y = (t+3)^2$

(d) $y = 3t(t+3)$

3 DISCRETE DIFFERENTIAL EQUATIONS (1ST ORDER)

I. Convert the following differential equations to point-wise discrete formulas (using y_n and t_n, but no y'_n).
II. Convert your point-wise formulas to matrix equations of the form $A\mathbf{y} = \mathbf{f}$ solving over the indicated interval with the indicated initial values and step-size.

(**a**) $y' = 4t+1$, $y(1) = 0$, $I = [1,2]$, $h = 1/2$

(**b**) $y' = 4t+1$, $y(2) = 0$, $I = [1,2]$, $h = 1/2$

(**c**) $y' + y = 2$, $y(0) = 1$, $I = [0,1]$, $h = 1/3$

(**d**) $y' + ty = 4t$, $y(0) = 0$, $I = [0,1]$, $h = 1/3$

(**e**) $y' + y = \delta(t)$, $y(-2) = 2$, $I = [-2,2]$, $h = 1$

(**f**) $y' + \delta(t)y = 0$, $y(-1) = 1$, $I = [-1,1]$, $h = 1/2$

4 THE DISCRETE IMPULSE BASIS

Write the following discrete functions as sums of impulses.

(**a**) $\mathbf{f} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$ with $h = \frac{1}{2}$.

(**b**) $\mathbf{f} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$ with $h = \frac{1}{3}$.

(**c**) $\mathbf{f} = \begin{bmatrix} -3 \\ 2 \\ 0 \\ 0 \\ 2 \end{bmatrix}$ with $h = \frac{1}{4}$.

Notation: Use $\boldsymbol{\delta}^{(k)}$ for the impulse at position k with step-size h. For example $\boldsymbol{\delta}^{(2)} = \begin{bmatrix} 0 \\ 1/h \\ 0 \\ \vdots \end{bmatrix}$

5 MatLab

- In MatLab, for loops repeat a group of commands a fixed number of times, each time using the next value for the *index variable*. The format for this command is

  for (<index var> = <vector>) .... end

  For example the code

```
>> y = 0
>> for( i = [1 3 5 7 9] )
     y = y + i
   end
```

  will repeat the MatLab command “y = y + i” five times, first with i=1, then with i=3, etc. After the final repetition, with i=9, it will stop. (This code computes the sum of the odd integers from 1 to 9.) Usually this is combined with the <start>:<step>:<end> command which creates vectors. For example

```
>> 1 : 2 : 9
```

  makes the vector [1 3 5 7 9]. If the <step> part is not included, then MatLab assumes you want to use step-size 1. For example

```
>> 1 : 9
```

  makes the vector [1 2 3 4 5 6 7 8 9].

- We can use a for loop to apply Euler's method solving $y' = t^2 y + t$, $y(0) = 2$, on $I = [0, 10]$ with $h = 0.1$.

```
>> y = 2;                                   % set the first value, y(0)=2
>> for ( t = 0 : 0.1 : 10 )                 % from t=0 to t=10 with step-size h=0.1
     dy = t^2 * y(end) + t;                 % |   slope at current t value
     y  = [ y, (y(end) + 0.1 * dy) ];       % |   compute next value and append to y
   end                                      % |_______
```

  The code above results in a vector y of values [y(0), y(0.1), ..., y(10), y(10.1)]. The command y(end) in the code above is used to get the last value in the vector y. The command y = [y ...] is used to add a new element after the end of y.

  Actually, this is **bad** MatLab code. The <vector> = [<itself> ...] command is slow for big vectors (i.e. size > 1 million) so you should not use it in loops where it is run over and over. It is usually faster to instead begin by creating a vector of the correct size before running the loop.

```
>> t = 0 : 0.1 : 10;                          % vector of sample points
>> y = zeros(1, length(t));                   % zero vector of correct size
>> y(1) = 2;                                  % set the first value
>> for ( i = 1 : length(y)-1 )                % loop through index of y
     dy     = t(i)^2 * y(i) + t(i);           % |   slope at current t value
     y(i+1) = y(i) + 0.1 * dy;                % |   compute next value of y
   end                                        % |_______
```

- We can also apply Euler's method using a while loop. While loops are more powerful than for loops, but also more dangerous, because you can create infinite loops using while. Consider the following code.

```
>> y = 2;   t = 0;
>> while (t < 10 )
     y = y + 0.1 * t^2
   end
```

  Since t will always be < 10, MatLab will keep computing in this loop forever, or until you stop it.

  <u>IMPORTANT:</u> Press <Ctrl>+C to break MatLab out of infinite loops or long computations.

  We can use a while loop to find $y(10)$ for $y' = t^2 y + t$, $y(0) = 2$ with $h = 0.1$.

```
>> t = 0;     y = 2;                        % starting value:  y(0)=2
>> while ( t < 10 )                         % loop until  y(10)
     dy = t^2 * y + t;                      % |   slope at current t value
     y = y + 0.1 * dy;                      % |   compute next value of y
     t = t + 0.1;                           % |   compute next value of t
   end                                      % |_______
```