

Implementing data fitting in MATLAB

See codes posted under today's date on website, and Ch. 4 of 301.pdf!!!!

$$u = \operatorname{sech}(x) \quad (1)$$

Derivative:

$$\frac{du}{dx} = -\operatorname{sech}(x)\operatorname{tanh}(x). \quad (2)$$

Plot function and its derivative

```
dx=0.1;
x=-10:dx:10;

y=sech(x);

%exact value of derivative
yd_exact=-sech(x).*tanh(x);

figure(1)
set(gca,'FontSize',18)
plot(x,y,'--','LineWidth',2); hold on
plot(x,yderiv,'LineWidth',2);
legend('y(x)', 'yd_exact(x)')
```

Compute second-order accurate derivative. Use

$$\text{center - difference } O(h^2) : \frac{y_{n+1} - y_{n-1}}{2h} \quad (3)$$

$$\text{forward - difference } O(h^2) : \frac{-3y_n + 4y_{n+1} - y_{n+2}}{2h} \quad (4)$$

$$\text{backward - difference } O(h^2) : \frac{3y_n - 4y_{n-1} + y_{n-2}}{2h}. \quad (5)$$

Here, $y_n = u(x_k)$

$x_k = -10 + dx * (k - 1)$

dx plays role of h .

First, use forward-difference to get derivative $yd(x)$ at left-hand endpoint x_1 .

```
yd(1) = (-3*y(1) + 4*y(2) - y(3)) / (2*dx);
```

Next, use center-difference to get derivative $yd(x)$ at central points $x_2 \dots x_{n-1}$.

```
for j=2:n-1
    yd(j) = (y(j+1) - y(j-1)) / (2*dx);
end
```

Finally, use backward-difference to get derivative $yd(x)$ at right-hand endpoint x_n .

```
yd(n) = (3*y(n) - 4*y(n-1) + y(n-2)) / (2*dx)
```

plot

```
figure(2)
set(gca,'FontSize',18)
plot(x,yd_exact,'o'); hold on
plot(x,yd,'LineWidth',3);
legend('yd exact(x)', 'yd order 2')
```

Repeat – this time using 2nd order forward and backward differences at first TWO and last TWO endpoints and 4TH ORDER center difference at midpoints

```
yd2(1) = (-3*y(1) + 4*y(2) - y(3)) / (2*dx);
yd2(2) = (-3*y(2) + 4*y(3) - y(4)) / (2*dx);
for j=3:n-2
    yd2(j) = (-y(j+2) + 8*y(j+1) - 8*y(j-1) + y(j-2)) / (12*dx);
end
yd2(n-1) = (3*y(n-1) - 4*y(n-2) + y(n-3)) / (2*dx);
yd2(n) = (3*y(n) - 4*y(n-1) + y(n-2)) / (2*dx);
```

Compare plot – see improved accuracy for 4th order method. For 2nd order, error $\approx 10^{-2}$. 4th order, error $\approx 10^{-4}$

FINAL NOTE: if differentiating data, first fit spline \rightarrow smooth differentiated data.

Numerical integration – use built-in MATLAB functions:

Compute $\int_{-10}^{10} \operatorname{sech}(x) dx = \tanh(10) - \tanh(-10) = 2.000000\dots$

Recall $y = \operatorname{sech}(x)$. To compute integral above via composite trapezoidal rule with $h = dx = 0.1$:

```
int_trap=trapz(x,y.^2)
```

```
int_trap = 1.999999991727922e+00
```

Generate cumulative values: $\int_{-10}^x \operatorname{sech}(z) dz$

```
int_trap_cum=cumtrapz(x,y.^2)
```

```
figure(3)
```

```
set(gca,'FontSize',18)
```

```
plot(x,int_trap_cum,'LineWidth',2)
```

Simpson's method with recursive accuracy, default tolerance 10^{-6}

First, define function want to integrate using `inline` syntax:

```
sec_sqd_fun=inline('sech(x).^2');
```

```
int_quad=quad(sec_sqd_fun,-10,10)
```

```
int_quad = 2.000000245590369e+00
```

NOTES ON SYNTAX:

- `inline` returns a function HANDLE – pass directly to `quad`
- could also define function want to integrate in a separate `.m` file.

```
function F=myfun(x) ;  
F=sech(x).^2 ;
```

- Then, pass HANDLE to this function with `quad(@myfun,-10,10)`
- AND ... all of these cases, function must accept VECTOR input and RETURN VECTOR OUTPUT. Use “dot” operations, like `.^` instead of `^`