ckpoint: 2 timer siden (autosaved)

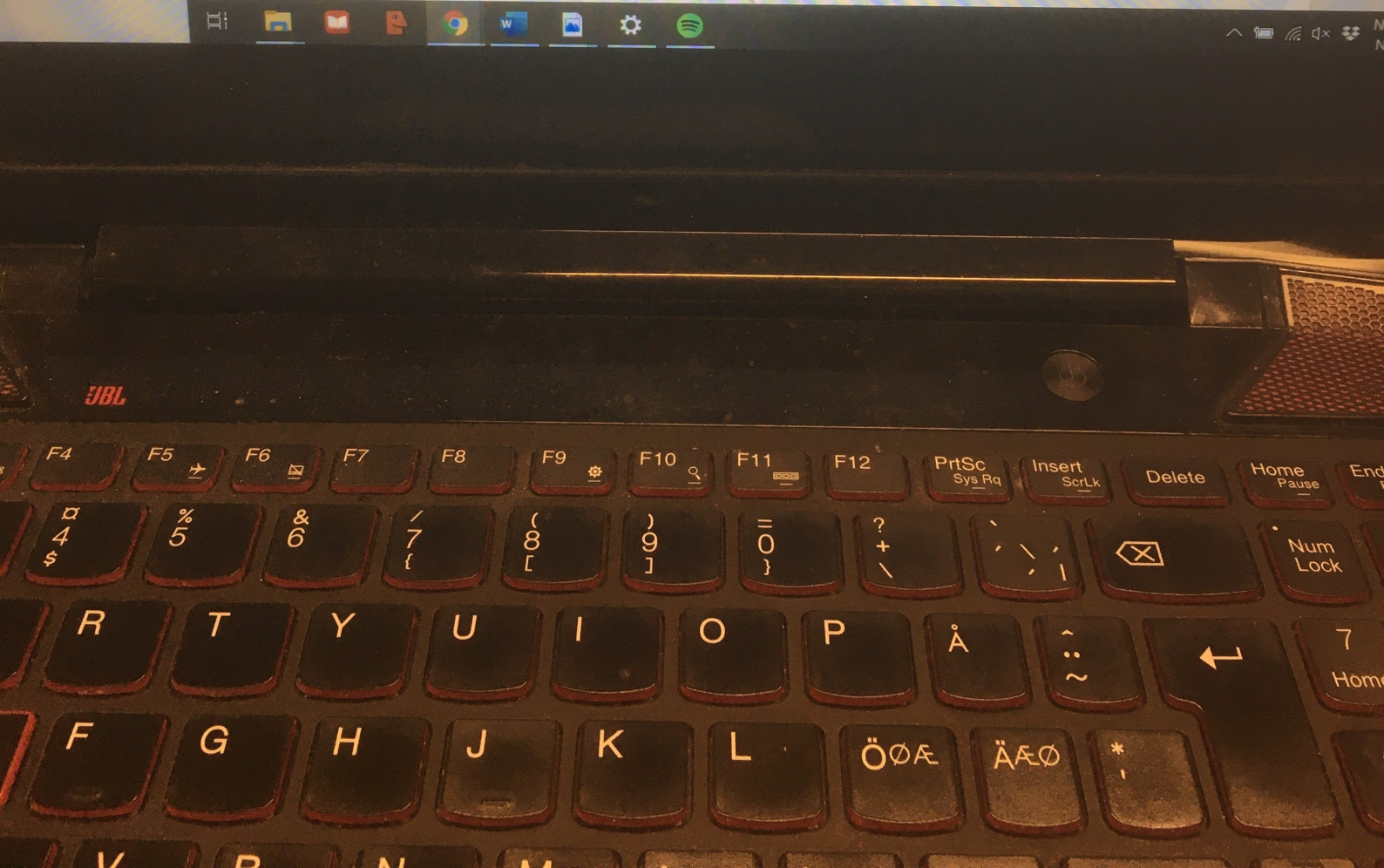
Kernel Navigate Widgets Help

Code nbdiff

1] (J) Implement an adaptive ODE solver by the use of Bogacki–Shampine pair of method, see the list of Runge–Kutta methods on Wikipedia. Use it to solve the Lotka-Volterra equation with tolerance Tol = 10^3. Plot the solutions as well as the stepsizes. Note the number of steps used in this case. Repeat the experiment with Heun–Euler and compare the number of steps used in this case. Repeat the experiment with Tol=10^5 and Tol=10^7. Increase the parameter Max_metodekall if needed.

Bogacki-Shampine with tolerance 10^3

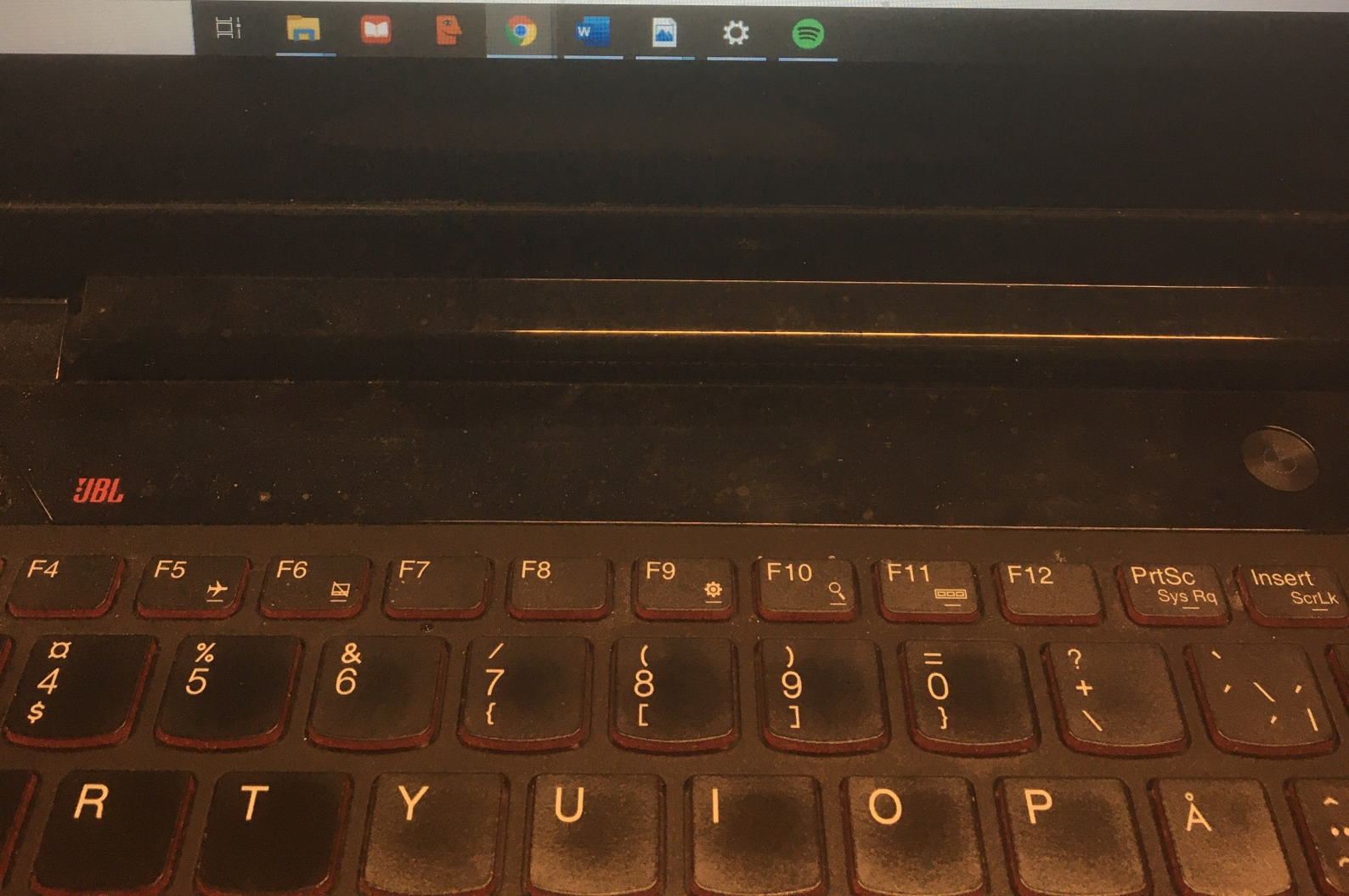
3]:
```
def bogacki_shampine(f, x, y, h):
    k1 = f(x, y)
    k2 = f(x + h, y + (h/2)*k1)
    k3 = f(x + (3 * (h/4)), y + (3*(h/4))*k2)
    y_next = y + (h/9)*(2*k1 + 3*k2 + 4*k3)
    x_next = x + h
    k4 = f(x_next, y_next)
    z_next = y + (h/24)/(7*k1 + 6*k2 + 8*k3 + 3*k4)
    err_est = norm(y_next - z_next)
    p = 2
    return x_next, y_next, err_est, p

def ode_adaptive(f, x0, xend, y0, h0, tol = 1.e-6, method=bogacki_shampine):
    # Adaptive solver for ODEs
    #    y' = f(x,y), y(x0)=y0
    #
    # Input: the function f, x0, xend, and the initial value y0
    #        intial stepsize h, the tolerance tol,
    #         and a function (method) implementing one step of a pair.
    # Ut: Array med x- og y- verdier.

    y_num = array([y0])     # Array for the solutions y
    x_num = array([x0])     # Array for the x-values

    xn = x0                 # Running values for  x, y and the stepsize h
    yn = y0
    h = h0
```



```
Maxcall = 100000          # Maximum allowed calls of method
ncall = 0

# Main loop
while xn < xend - 1.e-10:                    # Buffer for truncation error
    # Adjust the stepsize for the last step
    if xn + h > xend:
        h = xend - xn

    # Gjør et steg med valgt metode
    x_try, y_try, error_estimate, p = method(f, xn, yn, h)
    ncall = ncall + 1

    if error_estimate <= tol:
        # Solution accepted, update x and y
        xn = x_try
        yn = y_try
        # Store the solutions
        y_num = concatenate((y_num, array([yn])))
        x_num = append(x_num, xn)

    # else: The step rejectes and nothing is updated.

    # Adjust the stepsize
    h = 0.8*(tol/error_estimate)**(1/(p+1))*h

    # Stop with a warning in the case of max calls to method
    if ncall > Maxcall:
        print('Maximum number of method calls')
        return x_num, y_num

# Some diagnostic output
print('Number of accepted steps = ', len(x_num)-1)
print('Number of rejected steps = ', ncall - len(x_num)+1)
return x_num, y_num
```

```
    print( Number of rejected steps = ', ncall - len(x_num)+1)
    return x_num, y_num


def lotka_volterra(x, y):
    alpha, beta, delta, gamma = 2, 1, 0.5, 1
    dy = array([alpha*y[0]-beta*y[0]*y[1],
                delta*y[0]*y[1]-gamma*y[1]])

x0, x_end = 0, 10
y0 = array([2, 0.5])
h0 = 10

x_boga, y_boga = ode_adaptive(lotka_volterra, x0, x_end, y0, h0, tol = 1.e-3, method=bogacki_shampine)

plot(x_boga, y_boga, '--')
xlabel('x')
ylabel('y')
grid(True)
title('Lotka-Volterra equation')
legend(['Numerical'])

h_n = diff(x_boga)
x_n = x_boga[0:-1]
semilogy(x_n, h_n, '.-')
xlabel('x')
ylabel('y')
grid(True)
title('Stepsize variations')
```

executed in 134ms, finished 20:27:07 2019-11-19



WORK SHEET 11 ANF

2]

$$y'(x) = f(y(x)), \quad y(x_0) = y_0$$

$$k_1 = f(y_n), \quad k_2 = f(y_n + h a_{21} k_1)$$

$$y_{n+1} = y_n + h(b_1 k_1 + b_2 k_2)$$

a) $d_1 = y(x_0 - h) - y_1$, express as a power series of h:

Taylor expansion of $y(x+h)$:

$$y(x+h) = y(x) + h y'(x) + \frac{h^2}{2!} y''(x) + \frac{h^3}{3!} y'''(x)$$

The derivative:

$$\underline{y'(x) = f}, \quad y''(x) = f_x + f_y y' = f_x f_y f \Rightarrow \underline{y'' = f_y f}$$

$$y'''(x) = f_{xx} + f_{xy} y' + f_{yx} f + f_{yy} y' f + f_y f_x + f_y f_y y'$$

$$= f_{xx} + 2 f_{xy} f + f_{yy} f^2 + f_y f_x + (f_y)^2 f$$

$$\Rightarrow \underline{y''' = f_{yy} f^2 + (f_y)^2 f}$$

which gives the exact solution

$$\underline{y(x_0 + h) \; y_0 + hf + \frac{h^2}{2!}(f_x + f_y f) + \frac{h^3}{3!}(f_{xx} + 2 f_{xy} + \dots)}$$

WORK SHEET 11 ANF

2] a) cont. Which conditions on the coefficients a_{21}, b_1 and b_2 have to be satisfied for the method to be of order 1? order 2? order 3 possible?

Finding y_1

$$k_1 = f \;,\; k_2 = f + h\,a_{21}\,f_y f + \frac{(h a_{21})^2}{2!} f_{yy} f^2 + \dots$$

$$y_1 = y_0 + h b_1 f + h b_2 \left(f + h a_{21} f_y f + \frac{(h a_{21})^2}{2!} f_{yy} f^2 + \dots\right)$$

$$\Rightarrow d_1 = h(1 - b_1 - b_2) + h^2 f_y f\left(\tfrac{1}{2!} - b_2 a_{21}\right) + h^3\left(\tfrac{1}{3!}\left(f_{yy} f^2 + (f_y)^2 f\right) - \tfrac{1}{2!} b_2 a_{21}^2 f_{yy} f^2\right)$$

check for order $p=1$, $p=2$, $p=3$, runge-kutta conditions:

p	condition
1	$\sum b_i = 1$
2	$\sum b_i c_i = 1/2$
3	$\sum b_i c_i^2 = 1/3$
	$\sum b_i a_{ij} c_j = 1/6$

First order:

$1 - b_1 - b_2 = 0 \Rightarrow b_1 + b_2 = 1$ ✓

Second order:

$\frac{1}{2} - b_2 a_{21} = 0 \Rightarrow \frac{1}{2} a_{21} = \frac{1}{2} \Rightarrow a_{21} = 1$ ✓

Third order: we cannot say anything about third order

WORK SHEET 11 ANF

2] b) Find the optimal choice of parameters

we want the highest order possible, 2.
and the optimal parameters can be derived from 2] a)

3] Given the 3th order Runge-kutta method

$$k_1 = f(x_n, y_n), \quad k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right)$$

$$k_3 = f\left(x_n + \frac{3h}{4}, y_n + \frac{3h}{4} k_2\right)$$

$$y_{n+1} = y_n + \frac{h}{9}(2k_1 + 3k_2 + 4k_3)$$

a) Find the stability function $R(z)$ for this method

$$y_{n+1} = R(z)\, y_n \quad \text{where} \quad z = \lambda h$$

$$\Rightarrow R(z) = \frac{y_{n+1}}{y_n}$$

$$k_1 = f(x_n, y_n) = \lambda y_n$$

$$k_2 = f(y_n, h/2 \lambda y_n) = \lambda y_n (1 + h/2 \lambda)$$

$$k_3 = f\left(x_n + \frac{3h}{4}, y_n + \frac{3h}{4} y_n (1 + h/2 \lambda)\right)$$

$$= \lambda y_n + \frac{3h}{4} y_n \left(1 + \frac{h}{2} \lambda\right)$$

WORK SHEET 11 ANF

3] a) cont.

$$y_{n+1} = y_n + h/6\left(2\lambda y_n + 3\left(y_n\left(1 + \frac{h}{2}\lambda\right)\right)\right.$$
$$\left. + 4y_n\left(1 + \frac{3h}{4}\lambda + 3/8\, h^2\lambda^2\right)\right)$$
$$= y_n + h/6\left(2\lambda y_n + 3\lambda y_n + 3/2\, h\, y_n\lambda^2 + 4y_n\lambda\right.$$
$$\left. + 3y_n\lambda^2 h + 3/2\, y_n h^2\lambda^2\right)$$

$R(z) = y_{n+1} / y_n$

$$\Rightarrow \frac{y_n + y_n\lambda h + \frac{1}{6}h^2 y_n\lambda^2 + \frac{1}{3}y_n\lambda h^2 + \frac{1}{6}y_n h^3\lambda^3}{y_n}$$

$$= 1 + \lambda h + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3$$

with $z = \lambda h$ we get

$$R(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3$$

left side:

$$-2 \leq z + \frac{1}{2}z^2 + \frac{1}{6}z^3$$

$\Rightarrow z = -2.51275$

stability interval S:

$$\left|1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3\right| \leq 1$$

$$-1 \leq 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 \leq 1$$

Right side:

$$z + \frac{1}{2}z^2 + \frac{1}{6}z^3 \leq 0 \Rightarrow 1 + \frac{1}{2}z^2 + \frac{1}{6}z^3 \leq 0$$

$\Rightarrow z = 0$

this gives us the stability interval

$$S = [-2.51275, 0]$$

WORK SHEET 11 ANF

3] b) $y' = Ay$, $A = \begin{bmatrix} -41 & 38 \\ 19 & -22 \end{bmatrix}$

Find the eigenvalues of A:

$$|A - \lambda I| = \begin{bmatrix} -41-\lambda & 38 \\ 19 & -22-\lambda \end{bmatrix}$$

$$= (-41-\lambda)(-22-\lambda) - 38 \cdot 19 = \lambda^2 + 63\lambda + 180$$

$$= (\lambda+3)(\lambda+60)$$

$\Rightarrow \underline{\lambda_1 = -3}$, $\underline{\lambda_2 = -60}$, Find max stepsize h:

$z_1 \geq \lambda_1 h$, $z_2 \leq \lambda_2 h$

$h \geq 0 \Rightarrow h \leq -25/-60 = 0.4167$

$\Rightarrow \underline{\text{Max } h = 0.4167}$

WORK SHEET 11 ANF

4] a) Prove that the implicit midpoint rule

$$y_{n+1} = y_n + h f\left(x_n + \frac{h}{2}, \frac{1}{2}(y_n + y_{n+1})\right) \text{ is } A(0)\text{-stable}$$

it is stable if $|R(z)| \leq 1$ for all $z \leq 0$

$$f(x,y) = \lambda y$$

$$y_{n+1} = y_n + h\lambda(y_n + y_{n+1})/2 = y_n\left(1 + \frac{h\lambda}{2}\right) + \frac{h\lambda}{2} y_{n+1}$$

$$\Rightarrow y_{n+1} = y_n\left(\frac{1 + h\lambda/2}{1 - h\lambda/2}\right) \quad \text{and} \quad R(z) = \frac{y_{n+1}}{y_n}$$

$$\Rightarrow R(z) = \frac{y_n\left(\frac{1 + h\lambda/2}{1 - h\lambda/2}\right)}{y_n} = \frac{1 + h\lambda/2}{1 - h\lambda/2} = \frac{1 + z/2}{1 - z/2}$$

when $z \leq 0 \Rightarrow 1 + z/2 < 1 - z/2$ which means

$R(z) \leq 1$ for all $z \leq 0$

$\Rightarrow$ The implicit midpoint rule is $A(0)$ stable

WORK SHEET 11 ANF

5] a) Find the polynomial

$$p(\theta) = a\theta^3 + b\theta^2 + c\theta + d$$

satisfying

$$p(0) = y_n \qquad p(1) = y_{n+1}$$

$$\frac{dp}{d\theta}(0) = h y'_n \qquad \frac{dp}{d\theta}(1) = h y'_{n+1}$$

this means

$$a + b + h y'_n + y_n = y_{n+1}$$

$$\Rightarrow 3a + 2b + h y'_n = h y_{n+1}$$

$$a = y_{n+1} - y_n - h y'_n - b$$

$$3(y_{n+1} - y_n - h y'_n - 3b + 2b + h y'_n = h y_{n+1}$$

$$\underline{b = 3y_{n+1} - 3y_n - h y'_{n+1} - 2h y'_n}$$

$$a = y_{n+1} - y_n - h y'_n - 3y_{n+1} + 3y_n + h y'_{n+1} + 2h y'_n$$

$$a = 2y_n - 2y_{n+1} + h y'_n + h y'_{n+1}$$

which gives the polynomial

$$p(\theta) = (2y_n - 2y_{n+1} + h y'_n + h y'_{n+1})\theta^3 + (3y_{n+1} - 3y_n - h y'_{n+1} - 2h y'_n)\theta^2 + h y'_n \theta + y_n$$

WORK SHEET 11 ANF

5] b) From the result in a), find $p(\theta)$ for

$h = 0.5$, $y_n = 6.03$, $y_{n+1} = 5.91$, $y'_n = 5.41$, $y'_{n+1} = -8.29$

$$p(\theta) = (2 \cdot 6.03 - 2 \cdot 5.91 + 0.5 \cdot 5.41 - 0.5 \cdot 8.29)\theta^3$$
$$+ (3 \cdot 5.91 - 3 \cdot 6.03 + 0.5 \cdot 8.29 - 2 \cdot 0.5 \cdot 5.41)\theta^2$$
$$+ (0.5 \cdot 5.41)\theta + 6.03$$

$$\underline{\underline{p(\theta) = -(1.2)\theta^3 - (1.625)\theta^2 + (2.705)\theta + 6.03}}$$