#--------------------------------------------------------------------

# Slide 46

r = te.loada("""

$S1 -> S2; k1\*S1

S2 -> ; Vm2\*S2/(Km + S2)

k1 = 0.1

S1 = 10

Vm2 = 0.5; Km = 0.5

""")

#--------------------------------------------------------------------

# Slide 48

r = te.loada("""

$S1 -> S2; k1\*S1

S2 -> S3; k2\*S2

S3 -> S4; k3\*S1

S4 -> ; k4\*S4

k1 = 0.1; k2 = 0.2; k3 = 0.5; k4 = 1.3

S1 = 10

""")

#--------------------------------------------------------------------

# Slide 49

r = te.loada("""

$S1 -> S2; k1\*S1

S2 -> S3; k2\*S2

S3 -> S4; k3\*S3

S4 -> ; k4\*S4

k1 = 0.1; k2 = 0.2; k3 = 0.5; k4 = 1.3

S1 = 10

""")

print ('Distance from steady state = ', r.steadyState())

#--------------------------------------------------------------------

# Slide 53

import tellurium as te

import numpy

r = te.loada ('''

# Model Definition

v1: $Xo -> S1; k1\*Xo;

v2: S1 -> $w; k2\*S1;

# Initialize constants

k1 = 1; k2 = 1; S1 = 15; Xo = 1;

''')

# Time course simulation

m1 = r.simulate (0, 15, 100, ["Time", "S1"]);

r.k1 = r.k1 \* 6;

m2 = r.simulate (15, 40, 100, ["Time", "S1"]);

r.k1 = r.k1 / 6;

m3 = r.simulate (40, 60, 100, ["Time", "S1"]);

m = numpy.vstack ((m1, m2, m3)); # Merge data

r.plotArray (m, labels=["S1"])

#--------------------------------------------------------------------

# Slide 54

import tellurium as te

import numpy

r = te.loada ('''

# Model Definition

v1: $Xo -> S1; k1\*Xo;

v2: S1 -> $w; k2\*S1;

# Initialize constants

k1 = 1; k2 = 1; S1 = 15; Xo = 1;

at time > 15: k1 = k1\*6

at time > 40: k1 = k1/6

''')

# Time course simulation

m = r.simulate (0, 60, 100, ["Time", "S1"]);

r.plot()

#--------------------------------------------------------------------

# Slide 56

import tellurium as te

r = te.loada ('''

$Xo -> S1; k1\*Xo;

S1 -> $X1; k2\*S1;

k1 = 0.2; k2 = 0.4; Xo = 1; S1 = 0;

at (time > 20): S1 = S1 + 0.35

''')

# Simulate the first part up to 20-time units

m = r.simulate (0, 50, 100)

r.plot()

#--------------------------------------------------------------------

# Slide 58

r = te.loada ('''

y' = -k\*y; # Note the apostrophe

y = 1; k = 0.2;

''')

m = r.simulate (0, 50, 100)

r.plot()

#--------------------------------------------------------------------

# Slide 60

import tellurium as te

r = te.loada ('''

x' = sigma\*(y - x);

y' = x\*(rho - z) - y;

z' = x\*y - beta\*z;

x = 0.96259; y = 2.07272; z = 18.65888;

sigma = 10; rho = 28; beta = 2.67;

''')

result = r.simulate (0, 20, 1000, ['time', 'x', 'y', 'z'])

r.plot (result)

#--------------------------------------------------------------------

# Slide 61

import tellurium as te, numpy as np

r = te.loada ('''

J1: $X0 -> S1; k1\*X0;

J2: S1 -> $X1; k2\*S1;

X0 = 1.0; S1 = 0.0; X1 = 0.0;

k1 = 0.4; k2 = 2.3;

''')

m = r.simulate (0, 4, 100, ["Time", "S1"])

label = ['k1=' + str (r.k1)]

for i in range (0,4):

r.k1 = r.k1 + 0.1

label.append ('k1=' + "{:.1f}".format(r.k1))

r.reset()

m = np.hstack ((m, r.simulate (0, 4, 100, ['S1'])))

te.plotArray (m, labels=label)

#--------------------------------------------------------------------

# Slide 62

import tellurium as te

import teUtils

r = te.loada ('''

J1: $X0 -> S1; k1\*X0;

J2: S1 -> $X1; k2\*S1;

X0 = 1.0; S1 = 0.0; X1 = 0.0;

k1 = 0.4; k2 = 2.3;

''')

teUtils.parameterScanning.simpleTimeCourseScan (r, 'k1', 'S1', 0.4, 0.8, 5, timeEnd=4, formatStr='{:.1f}

#--------------------------------------------------------------------

# Slide 63

import tellurium as te, numpy as np, teUtils

r = te.loada ('''

// Reactions:

J0: $X0 => S1; (VM1\*(X0 - S1/Keq1))/(1 + X0 + S1 + S4^h);

J1: S1 => S2; (10\*S1 - 2\*S2)/(1 + S1 + S2);

J2: S2 => S3; (10\*S2 - 2\*S3)/(1 + S2 + S3);

J3: S3 => S4; (10\*S3 - 2\*S4)/(1 + S3 + S4);

J4: S4 => $X1; (V4\*S4)/(KS4 + S4);

// Species initializations:

S1 = 0; S2 = 0;

S3 = 0; S4 = 0;

X0 = 10; X1 = 0;

// Variable initializations:

VM1 = 10; Keq1 = 10;

h = 1; V4 = 2.5;

KS4 = 0.5;

''')

teUtils.scanning.simpleTimeCourseScan (r, 'h', 'S1', 1, 8, 8, timeEnd=20, formatStr='{:.1f}')

#-------------------------------------------------------------------

# Exercise 1:

import tellurium as te

import numpy

r = te.loada ('''

E + S -> ES; k1\*E\*S

ES -> E + S; k\_1\*ES

ES -> E + P; k2\*ES

k1 = 20; k\_1 = 1; k2 = 10

E = 2; S = 20

''')

m = r.simulate (0, 1.5, 100)

r.plot()

#-------------------------------------------------------------------

# Exercise 2:

Page 5

r = te.loada("""

$S1 -> S2; k1\*S1

S2 -> ; Vm2\*S2/(Km + S2)

k1 = 0.1

S1 = 10

Vm2 = 0.5; Km = 0.5

""")

m = r.simulate (0, 20, 100)

r.plot()

#--------------------------------------------------------------------

# Exercise 3:

Page 7

r = te.loada("""

$S1 -> S2; k1\*S1

S2 -> S3; k2

S3 -> S4; k3\*S3

S4 -> ; k4\*S4

k1 = 0.1; k2 = 0.2; k3 = 0.5; k4 = 1.3

S1 = 10

""")

m = r.simulate (0, 20, 100)

r.plot()

#--------------------------------------------------------------------

# Exercise 4:

Page 8

r = te.loada("""

$S1 -> S2; k1\*S1

S2 -> S3; k2\*S2

S3 -> S4; k3\*S1

S4 -> ; k4\*S4

k1 = 0.1; k2 = 0.2; k3 = 0.5; k4 = 1.3

S1 = 10

""")

m = r.simulate (0, 20, 100)

r.plot()

#--------------------------------------------------------------------

# Exercise 5

import tellurium as te, roadrunner, teUtils

import matplotlib.pyplot as plt

r = te.loada("""

$Xo -> S1; k1\*Xo

S1 -> S2; k2\*S1

S2 -> S3; k3\*S2

S3 -> $S4; k4\*S3

k1 = 0.1; k2 = 0.2; k3 = 0.15; k4 = 0.4;

Xo = 10

""")

for k in r.getGlobalParameterIds():

for S in r.getFloatingSpeciesIds():

teUtils.scanning.simpleTimeCourseScan (r, k, S, 1, 8, 8, timeEnd=20, formatStr='{:.1f}')

plt.show()