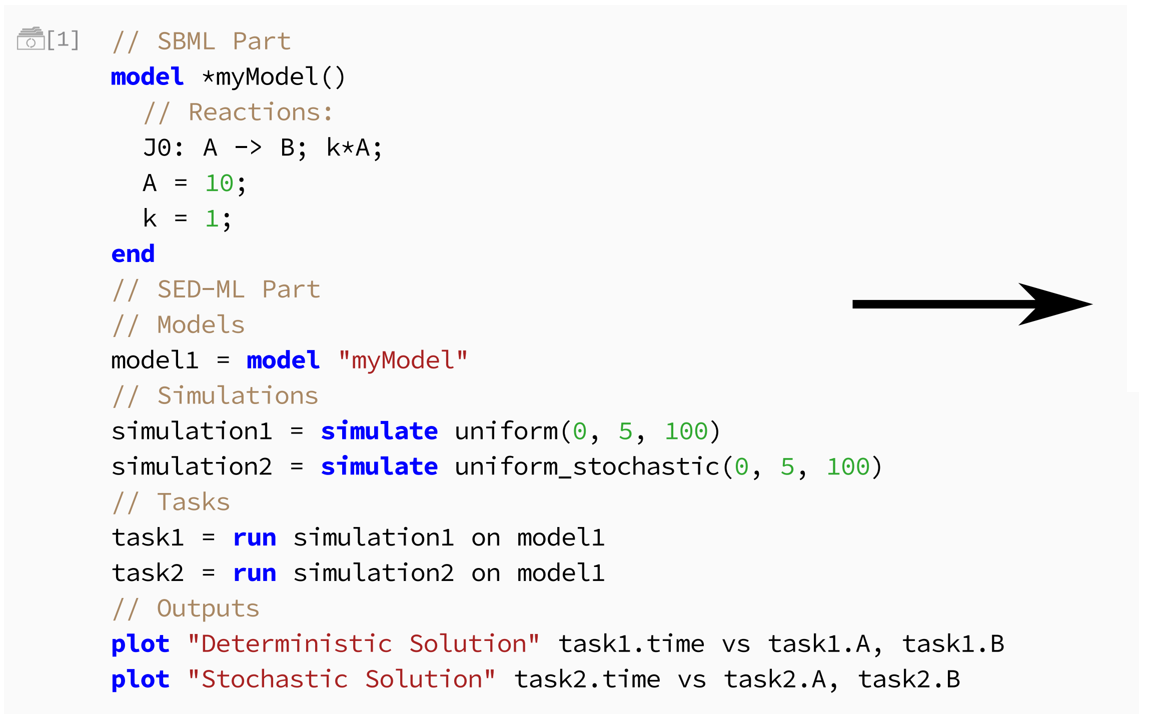
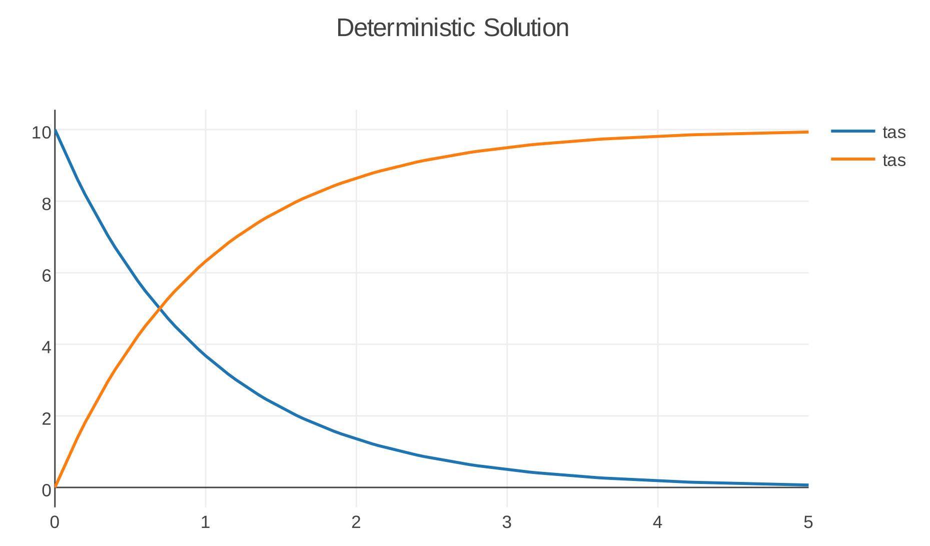
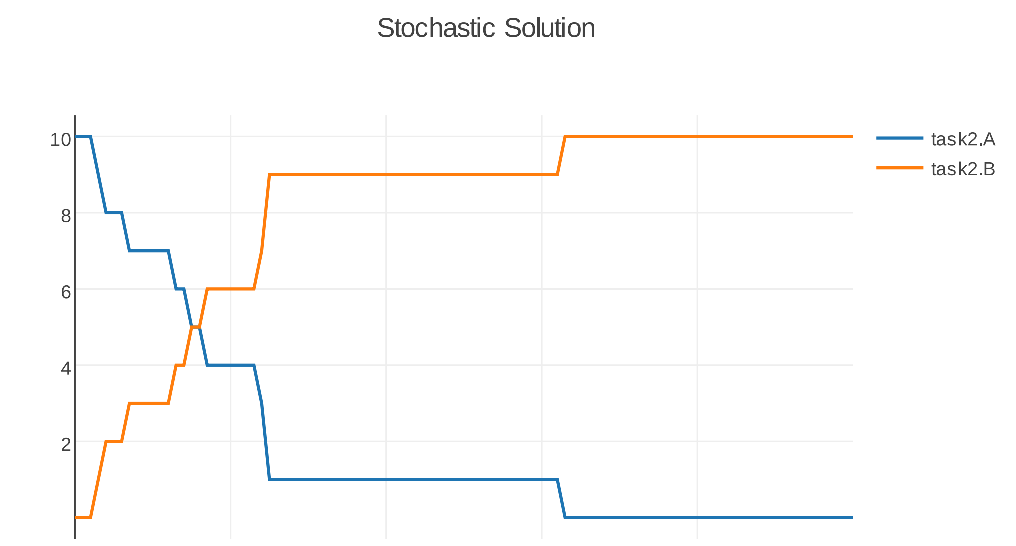
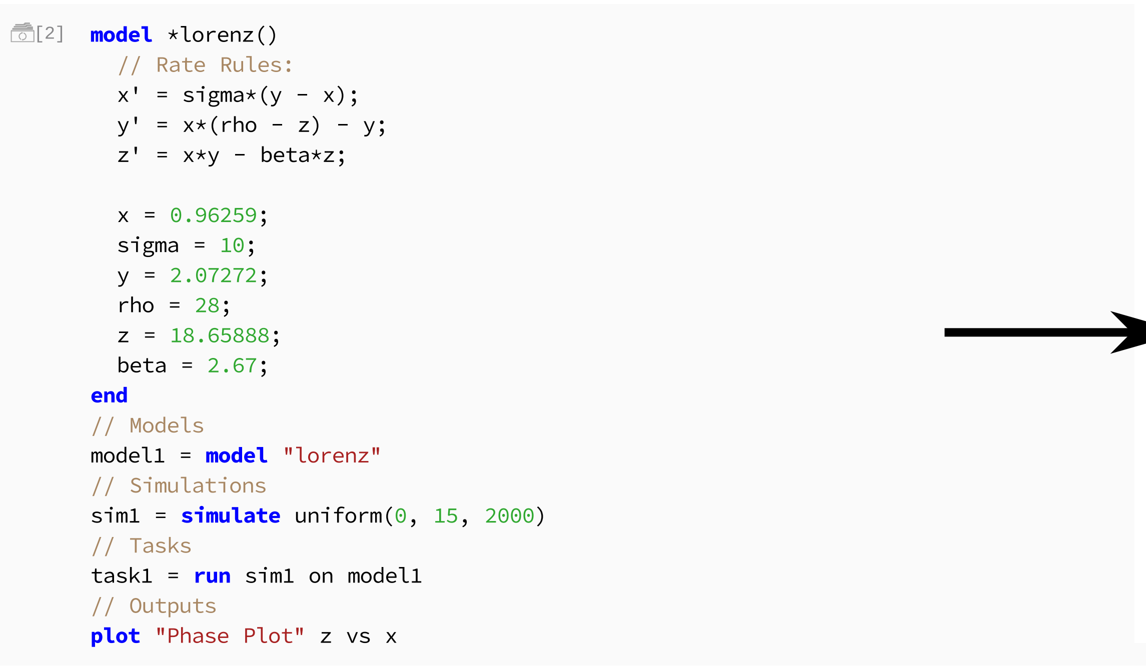


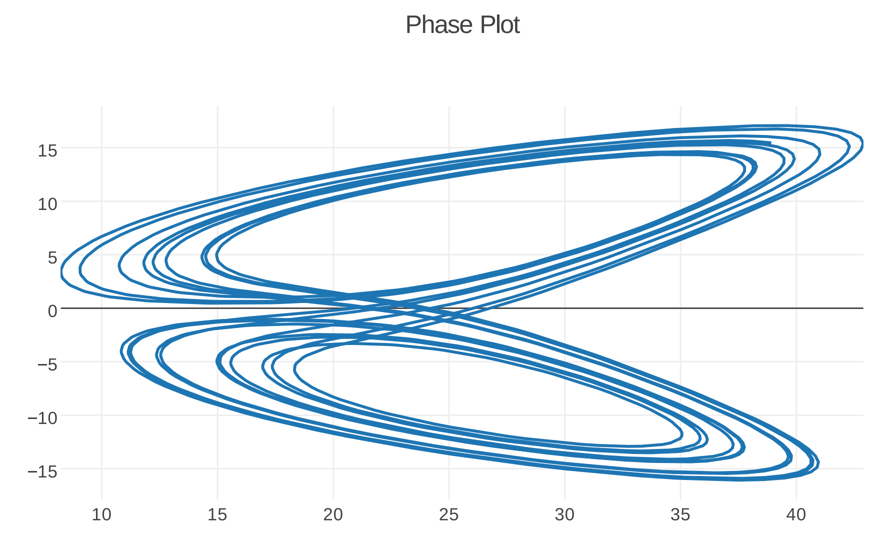
```
[1] // SBML Part
    model *myModel()
      // Reactions:
      J0: A -> B; k*A;
      A = 10;
      k = 1;
    end
    // SED-ML Part
    // Models
    model1 = model "myModel"
    // Simulations
    simulation1 = simulate uniform(0, 5, 100)
    simulation2 = simulate uniform_stochastic(0, 5, 100)
    // Tasks
    task1 = run simulation1 on model1
    task2 = run simulation2 on model1
    // Outputs
    plot "Deterministic Solution" task1.time vs task1.A, task1.B
    plot "Stochastic Solution" task2.time vs task2.A, task2.B
```

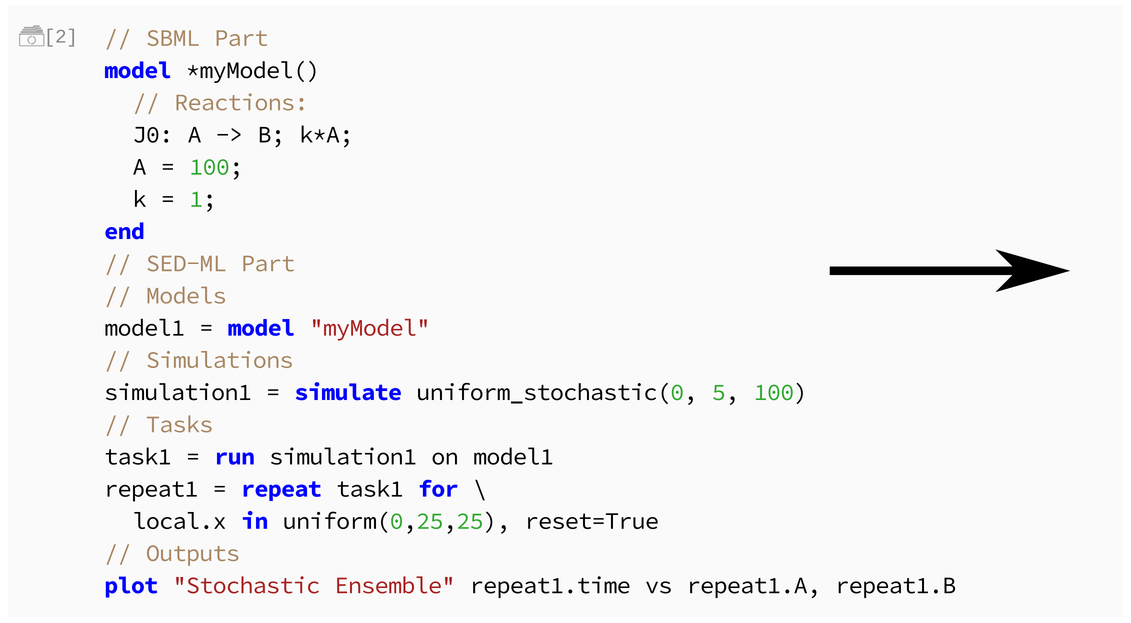




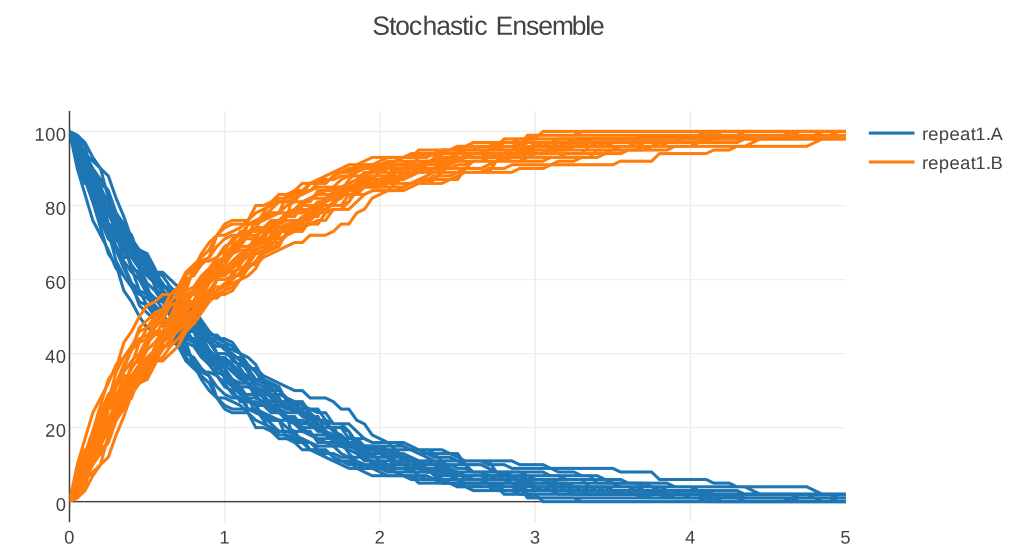


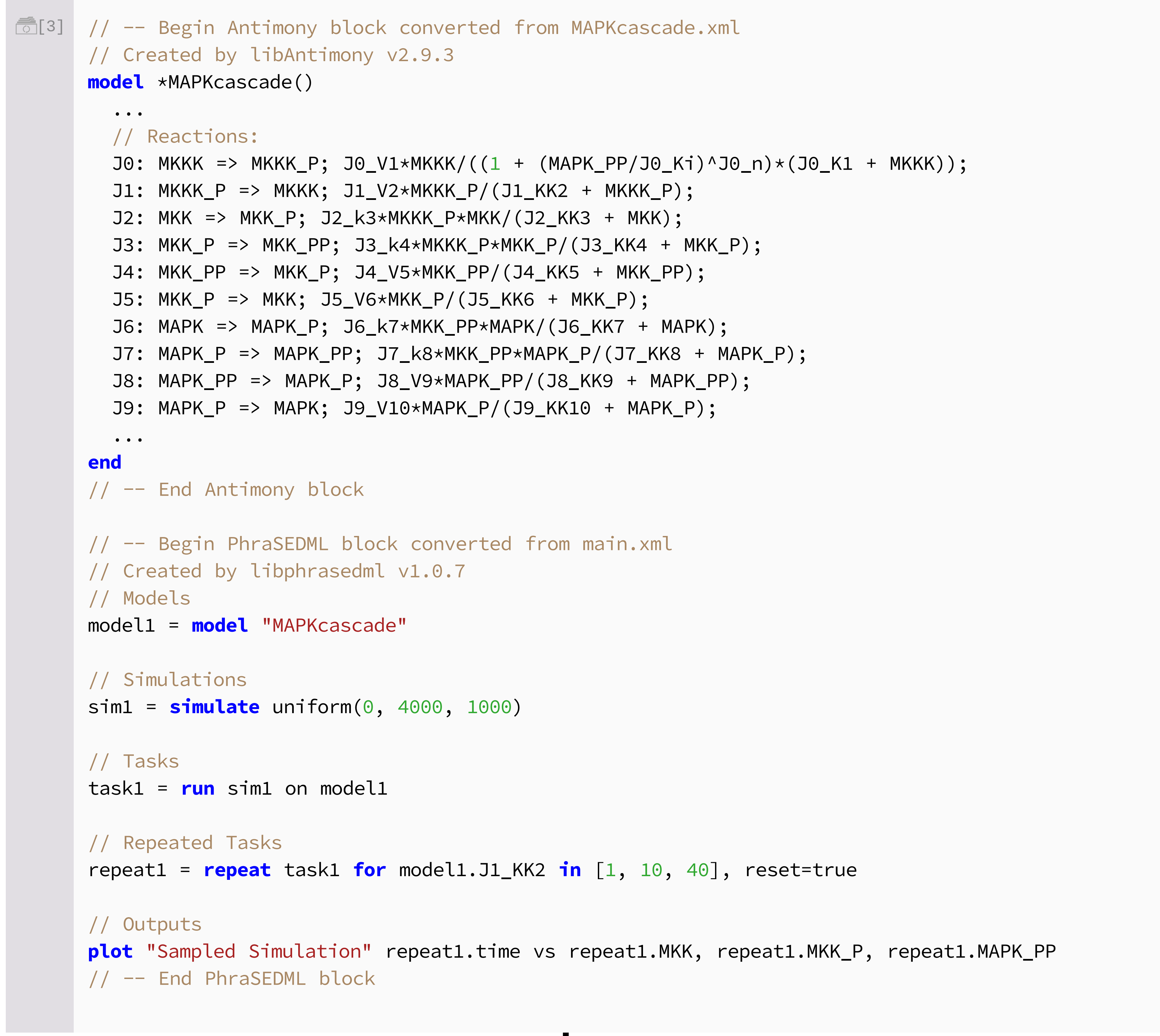
```
[2] model *lorenz()
      // Rate Rules:
      x' = sigma*(y - x);
      y' = x*(rho - z) - y;
      z' = x*y - beta*z;

      x = 0.96259;
      sigma = 10;
      y = 2.07272;
      rho = 28;
      z = 18.65888;
      beta = 2.67;
    end
    // Models
    model1 = model "lorenz"
    // Simulations
    sim1 = simulate uniform(0, 15, 2000)
    // Tasks
    task1 = run sim1 on model1
    // Outputs
    plot "Phase Plot" z vs x
```



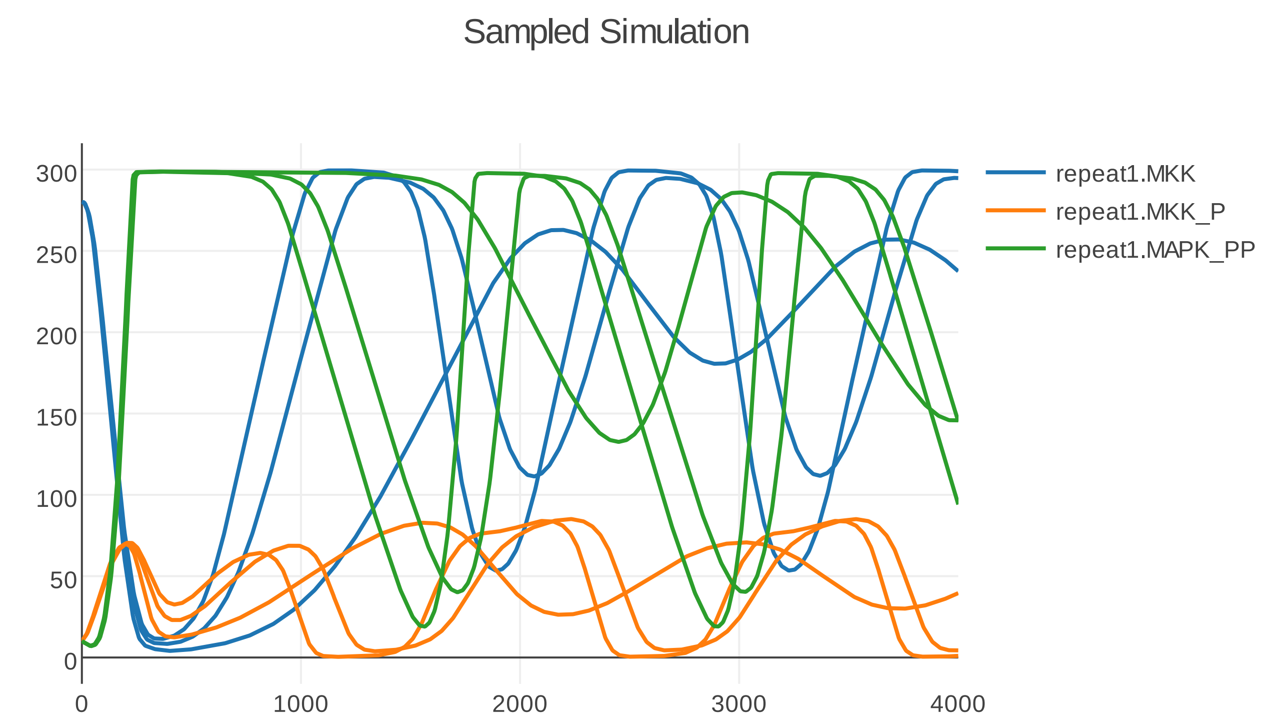


```
[2] // SBML Part
    model *myModel()
      // Reactions:
      J0: A -> B; k*A;
      A = 100;
      k = 1;
    end
    // SED-ML Part
    // Models
    model1 = model "myModel"
    // Simulations
    simulation1 = simulate uniform_stochastic(0, 5, 100)
    // Tasks
    task1 = run simulation1 on model1
    repeat1 = repeat task1 for \
      local.x in uniform(0,25,25), reset=True
    // Outputs
    plot "Stochastic Ensemble" repeat1.time vs repeat1.A, repeat1.B
```





```
[3] // -- Begin Antimony block converted from MAPKcascade.xml
    // Created by libAntimony v2.9.3
    model *MAPKcascade()
      ...
      // Reactions:
      J0: MKKK => MKKK_P; J0_V1*MKKK/((1 + (MAPK_PP/J0_Ki)^J0_n)*(J0_K1 + MKKK));
      J1: MKKK_P => MKKK; J1_V2*MKKK_P/(J1_KK2 + MKKK_P);
      J2: MKK => MKK_P; J2_k3*MKKK_P*MKK/(J2_KK3 + MKK);
      J3: MKK_P => MKK_PP; J3_k4*MKKK_P*MKK_P/(J3_KK4 + MKK_P);
      J4: MKK_PP => MKK_P; J4_V5*MKK_PP/(J4_KK5 + MKK_PP);
      J5: MKK_P => MKK; J5_V6*MKK_P/(J5_KK6 + MKK_P);
      J6: MAPK => MAPK_P; J6_k7*MKK_PP*MAPK/(J6_KK7 + MAPK);
      J7: MAPK_P => MAPK_PP; J7_k8*MKK_PP*MAPK_P/(J7_KK8 + MAPK_P);
      J8: MAPK_PP => MAPK_P; J8_V9*MAPK_PP/(J8_KK9 + MAPK_PP);
      J9: MAPK_P => MAPK; J9_V10*MAPK_P/(J9_KK10 + MAPK_P);
      ...
    end
    // -- End Antimony block

    // -- Begin PhraSEDML block converted from main.xml
    // Created by libphrasedml v1.0.7
    // Models
    model1 = model "MAPKcascade"

    // Simulations
    sim1 = simulate uniform(0, 4000, 1000)

    // Tasks
    task1 = run sim1 on model1

    // Repeated Tasks
    repeat1 = repeat task1 for model1.J1_KK2 in [1, 10, 40], reset=true

    // Outputs
    plot "Sampled Simulation" repeat1.time vs repeat1.MKK, repeat1.MKK_P, repeat1.MAPK_PP
    // -- End PhraSEDML block
```


Sampled Simulation
repeat1.MKK
repeat1.MKK_P
repeat1.MAPK_PP