LNA++ quick start

Justin Feigelman

[justin.feigelman@helmholtz-muenchen.de](mailto:justin.feigelman@helmholtz-muenchen.de)

26 January 2015

# Summary

This document provides a very quick guide to LNA++ and demonstrates how to use it for a simple chemical reaction network with a single species stochastic production and degradation.

For more detailed usage information see the **User Guide**.

# Prerequisites

Before using LNA++ a few libraries need to be installed.

The following libraries must be installed on the system:

1. CVODES
2. Blitz++

Source code for both libraries is included in the ‘libraries’ folder and can be installed via the standard GNU Autoconf toolchain (see <https://en.wikipedia.org/wiki/Autoconf>) for further information. Usually it suffices to run the following commands from the source directory after unzipping.

./configure --with-pic

make

sudo make install

This requires administrator privileges. Otherwise, you can run ‘make install --prefix=PATH/TO/INSTALL’

For more information about installation of prerequisites see the **User Guide**.

# MATLAB

To build the example model in MATLAB, cd to the ‘models’ directory within MATLAB. Then, open the ‘BirthDeath.m’ script file. This script contains the necessary code to generate the executable for the model, as well as a number of commands for simulating the model and computing first and second order sensitivities.

The crucial steps for building the model are:

1. **Define the stoichiometry matrix (S) for the model:**

S = [1 -1 0 0; % change to mRNA

0 0 1 -1]; % change to protein

1. **Define the state variables and model parameters as symbols:**

syms k\_m k\_p g\_m g\_p real

syms m p real

phi = [m,p]; % state vector

Theta = [k\_m, k\_p, g\_m, g\_p]; % parameter vector

npar = length(Theta); % 4 parameters

1. **Define the reaction fluxes:**

f = @(phi,t,Theta) ...

[ Theta(1), Theta(3)\*phi(1), Theta(2)\*phi(1), Theta(4)\*phi(2)];

1. **Generate the symbolic C code:**

addpath('../matlab')

model = 'BirthDeath';

generateLNAComponents(model, S, f, phi, Theta, 'BOTH')

1. **Compile the code and generate the Mex file:**

compileLNA(model, S, npar);

Each of the steps is described in more detail in the **User Guide**.

Once the model is built it is run by calling the mex executable with some arguments:

Theta = [20, 25, 10, 1];

tspan = 0:0.1:10;

[MRE, Var] = BirthDeath\_LNA(Theta, tspan);

Make sure the executable is on the MATLAB search path. You can add it by running

addpath('BirthDeath/')

The rest of the BirthDeath.m script shows how to use the executable with various optional inputs and outputs.

# Python

To build the example model in Python, cd to the ‘models’ directory and launch Python. Make sure to use Python 3.0 or greater. Open the ‘BirthDeath.py’ script file. This script contains the necessary code to generate the module for the model, as well as a number of commands for simulating the model and computing first and second order sensitivities.

To build the model, perform the following steps:

1. **Import the LNA++ module:**

from sys import path

path += ['../python']

from LNA import \*

1. **Define the stoichiometric matrix for the model:**

S = Matrix([[1, -1, 0, 0], \

[0, 0, 1, -1]])

1. **Define the symbolic state variables and model parameters:**

phi = symbols('m,p', real=True)

Theta = symbols('k\_m,k\_p,g\_m,g\_p', real=True)

1. **Define the reaction fluxes:**

f = lambda phi,t,Theta: \

[Theta[0], Theta[2]\*phi[0], Theta[1]\*phi[0], Theta[3]\*phi[1]]

1. **Compute the symbolic functions and store them in an object:**

model=’BirthDeath’

tups = generateLNAComponents(model, S, f, phi, Theta, computeSS='BOTH')

1. **Generate the C code and Python module:**

npar = len(Theta) # number of parameters

compileLNA(model, S, tups, npar)

Once the module is compiled, it is placed in the ‘modules’ directory.

To run the Birth Death model, add the ‘modules’ directory to the Python search path:

path += ['modules']

and then import the generated module:

from BirthDeathLNA import LNA

Simulate the model using the generated module’s LNA function, and use numpy to generate a vector of output times for the simulation:

from numpy import arange

# define initial conditions

Y0 = [2, 200]

V0 = [2, 3, 150]

# model parameters

Theta = [20, 25, 10, 1]

# vector of output times

tspan = arange(0,10,0.1).tolist()

# compute the mean and temporal auto-covariance matrix assuming steady-state mean and variance at t=0

obsVar = 2 # second species (Protein) is observed

MRE,Sigma = LNA(Theta,tspan, obsVar=obsVar)

The rest of the Python script shows how to use the generated module with various input and output arguments, including first and second order sensitivities. For more details, see the **User Guide**.