Stochastic Simulation: Lecture 9

Prof. Mike Giles

Oxford University Mathematical Institute

Continuous-time Markov Processes

In these two lectures we will consider the approximation of Continuous-time Markov Processes.

Probably the most important class of applications for these is in the stochastic modelling of chemical reactions in solution, so this is the context we will start with.

Chemical reaction:

$$A + B \longrightarrow C$$

Classical deterministic modelling in a "well-stirred" vessel gives a set of ODEs for the concentrations c_A, c_b, c_C:

$$\begin{aligned} \dot{c}_A &= -\kappa\, c_A\, c_B \\ \dot{c}_B &= -\kappa\, c_A\, c_B \\ \dot{c}_C &= +\kappa\, c_A\, c_B \end{aligned}$$

Continuous-time Markov Processes

This works well when there are lots of molecules of A and B in the solution, but there are applications (particularly in bio-chemistry) when there are very few, and then things become stochastic.

Let X_A, X_B, X_C be number of molecules of A, B, C in some well-mixed container. Reactions require a molecule of A to "bump into" a molecule of B and react, so

$$\mathbb{P}(\text{reaction in time interval } \mathrm{d}t) = \kappa \, X_A \, X_B \, \mathrm{d}t$$

and when a reaction happens

$$\begin{aligned} X_A &\to X_A - 1 \\ X_B &\to X_B - 1 \\ X_C &\to X_C + 1 \end{aligned}$$

Unit rate Poisson Process

A unit rate Poisson process $Y(\tau)$ is a continuous-time random counting process in which

- there is a set of increasing jump times $0 < \tau_1 < \tau_2 < \tau_3 < \ldots$
- setting $\tau_0 = 0$, then $Y(\tau) = j,$ for $\tau \in [\tau_j, \tau_{j+1}),\ j = 0, 1, \ldots$
- the jump intervals $\tau_{j+1} - \tau_j$ are i.i.d. exponential random variables, so for $t > 0$

$$\mathbb{P}(\tau_{j+1} - \tau_j > \tau) = \exp(-\tau)$$

and

$$\mathbb{P}(\tau_{j+1} < \tau + \mathrm{d}t \mid \tau_{j+1} > \tau > \tau_j) = \mathrm{d}t$$

Note: for any time $\tau > 0$, $Y(\tau)$ is a Poisson random variable with mean τ.

Continuous-time Markov Processes

Using a unit rate Poisson process to represent the number of reactions which have taken place we have

$$X(t) = X(0) + R(t) \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix}$$

where

$$R(t) = Y\left(\int_0^t \kappa\, X_A(s)\, X_B(s)\, \mathrm{d}s\right)$$

so the probability of a reaction in time interval $(t, t+\mathrm{d}t)$ is $\kappa\, X_A(t)\, X_B(t)\, \mathrm{d}t$.

Continuous-time Markov Processes

Generalising this, suppose we have d species, and multiple reactions, with the k-th reaction having an intensity function $\lambda_k(t)$ and with each such reaction changing the count of X_i by ζ_{ki}.

Then with independent unit rate processes for each reaction we have

$$X_i(t) = X_i(0) + \sum_k R_k(t)\ \zeta_{ki}$$

where we have the time-change representation

$$R_k(t) = Y_k\left(\int_0^t \lambda_k(s)\,\mathrm{d}s\right)$$

and for the most common law of *mass action kinetics*

$$\lambda_k(t) = \kappa_k \prod_{i=1}^{d} \frac{X_i!}{(X_i - \nu_{ki})!} 1_{\{X_i \geq \nu_{ki}\}}$$

when there are ν_{ki} inputs of species i in reaction k.

Continuous-time Markov Processes

Example from a paper by Anderson and Higham (2012)

$$S_1 \underset{\kappa_2}{\overset{\kappa_1}{\rightleftarrows}} S_2, \quad 2S_2 \xrightarrow{\kappa_3} S_3,$$

then

$$\zeta_1 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \quad \zeta_2 = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad \zeta_3 = \begin{pmatrix} 0 \\ -2 \\ 1 \end{pmatrix},$$

and

$$\lambda_1 = \kappa_1 X_1, \quad \lambda_2 = \kappa_2 X_2, \quad \lambda_3 = \kappa_3 X_2 (X_2 - 1).$$

Stochastic Simulation Algorithm (SSA)

SSA is an exact simulation algorithm originally due to Gillespie (1976, 1977). There have been a number of variants published since – here I first describe his original "Direct Method".

Key idea: if we define $\lambda = \sum_k \lambda_k$, then

$$\begin{aligned} \mathbb{P}(\text{reaction } k \text{ occurs in next } \mathrm{d}t) &= \lambda_k \, \mathrm{d}t \\ \mathbb{P}(\text{some reaction occurs in next } \mathrm{d}t) &= \lambda \, \mathrm{d}t \\ \mathbb{P}(\text{next reaction is reaction } k) &= \lambda_k / \lambda \end{aligned}$$

Stochastic Simulation Algorithm (SSA)

Input: initial X, final time T
$t := 0$

while $t < T$ **do**
  compute λ_k and $\lambda := \sum_k \lambda_k$
  generate two uniform r.v.'s U_1, U_2
  next reaction time $t := t - \log(U_1)/\lambda$
  if $t < T$ **then**
    identify reaction k' s.t.

$$\sum_{k<k'} \frac{\lambda_k}{\lambda} < U_2 \leq \sum_{k \leq k'} \frac{\lambda_k}{\lambda}$$

    $X := X + \zeta_{k'}$
  end if
end while

Stochastic Simulation Algorithm (SSA)

Major issue: the cost is proportional to the total number of reactions that take place – could be millions.

This will be addressed by tau-leaping approximation, and MLMC.

Minor issues:

- for each reaction, the Direct Method requires 2 random numbers
- 2 key steps have costs proportional to the number of possible reactions.

The first of these is addressed by Gillespie's Next Reaction Method, and the second was addressed by Gibson & Bruck (2000).

Stochastic Simulation Algorithm (SSA)

Reaction k has the unit rate Poisson process

$$Y_k\left(\int_0^t \lambda_k(s)\,\mathrm{d}s\right)$$

with real jump times $t_1, t_2, \ldots$ and pseudo-times $\tau_1, \tau_2, \ldots$ where

$$\int_0^{t_n} \lambda_k(s)\,\mathrm{d}s = \tau_n \quad \Longrightarrow \quad \int_{t_n}^{t_{n+1}} \lambda_k(s)\,\mathrm{d}s = \tau_{n+1} - \tau_n$$

and $\tau_{n+1} - \tau_n = -\log U$ where U is an $(0, 1)$ uniform r.v.

Putting

$$T_k(t) = \tau_{n+1} - \tau_n - \int_{t_n}^{t} \lambda_k(s)\,\mathrm{d}s$$

means that reaction k occurs when T_k reaches 0.

Stochastic Simulation Algorithm (SSA)

Input: initial X, timers $T_k = -\log(U_k)$, final time T
$t := 0$

loop
  compute λ_k
  set $\Delta t = \min_k(T_k/\lambda_k)$, $k' = \text{argmin}_k\,(T_k/\lambda_k)$,
  $t := t + \Delta t$
  if $t > T$, **stop**
  $X := X + \zeta_{k'}$
  $T_{k'} = -\log U$
  for all $k \neq k'$ **do**
    $T_k := T_k - \lambda_k\,\Delta t$
  end for
end loop

Tau-leaping method

SSA is used extensively but it can be very costly – some simulations may involve millions of individual reactions, and may need to perform up to a million such calculations.

The tau-leaping method is an approximate simulation method.

The Euler-Maruyama SDE approximation treats the drift and diffusion values as constant within a timestep, and only updates them at the end of the timestep.

Tau-leaping adopts the same idea, updating the λ_k only at the beginning/end of each timestep.

Within a timestep of size h, λ_k is fixed so the number of reactions of type k is $P(\lambda_k h)$ where $P(\mu)$ is a Poisson r.v. with mean μ.

Tau-leaping algorithm

Input: timestep h, initial state $\widehat{X}$, final time $T = N\,h$

for $n = 1, N$ **do**

  $\Delta\widehat{X} := 0$

  for each k **do**

    compute $\lambda_k(\widehat{X})$

    generate Poisson r.v.'s $R_k = Poiss(\lambda_k h)$

    $\Delta\widehat{X} := \Delta\widehat{X} + R_k\zeta_k$

  end for

  $\widehat{X} := \widehat{X} + \Delta\widehat{X}$

end for

Output: $f(\widehat{X})$

Tau-leaping method

The cost is $O(T/h)$, but there is now a discretisation error so that for $h \gg 1/\lambda$

$$\mathbb{E}[\, f(X_T) - f(\widehat{X}_T)\,] = O(h).$$

In next lecture will use MLMC to eliminate this error and also reduce the total cost.

Also, Poisson r.v.'s have an unbounded size, so there is a small but finite probability of ending up with negative population counts.

Set reaction rate to zero if negative count of one of the inputs.

Tau-leaping method

For large mean μ, the Poisson distribution is close to the Normal with mean μ and variance μ, rounded to the nearest integer.

This means that we approximately have

$$\widehat{X}_{n+1} = \widehat{X}_n + \sum_k \left(\lambda_k(\widehat{X}_n) h + \sqrt{\lambda_k(\widehat{X}_n)} \sqrt{h}\, Z_{kn} \right) \zeta_k$$

where Z_{kn} are i.i.d. unit Normals random variables.

This corresponds to the Euler-Maruyama discretisation of the *chemical Langevin* SDE approximation

$$\mathrm{d}X = \sum_k \left(\lambda_k(X)\, \mathrm{d}t + \sqrt{\lambda_k(X)}\, \mathrm{d}W_k \right) \zeta_k$$

The fact that MLMC would be very effective for this SDE suggests it might also be useful for tau-leaping.

Approximation hierarchy

Thus chemical kinetics can be modelled at 4 different levels:

- SSA – exact simulation of each and every reaction
- tau-leaping – regular updating of the propensity functions
- Langevin SDE – replacing Poisson distribution by Normal approximation
- ODEs – ignoring stochastic effects entirely

These involve a balance between cost and accuracy, but ideally we would like to achieve both low cost and high accuracy.

Key References

D.T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions", J. Comp. Phys., 22(4):403-434, 1976.

D.T. Gillespie, "Exact stochastic simulation of coupled chemical reactions", J. Phys. Chem. 81(25):2340-2361, 1977.

M.A. Gibson, J. Bruck "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels", J. Phys. Chem., 104(9):1876-1889, 2000.

D.T. Gillespie, A. Ganguly, T.G. Kurtz, "Error analysis of tau-leaping simulation methods", Annals of Applied Probability, 21(6):2226-2262, 2011.

D.T. Gillespie, A. Hellander, L.R. Petzold, "Perspective: Stochastic algorithms for chemical kinetics", J. Chem. Phys., 138(17):170901, 2013.

Poisson CDF and inverse

A discrete Poisson random variable N with rate λ takes integer value n with probability

$$e^{-\lambda} \frac{\lambda^n}{n!}$$

Hence, the cumulative distribution function is

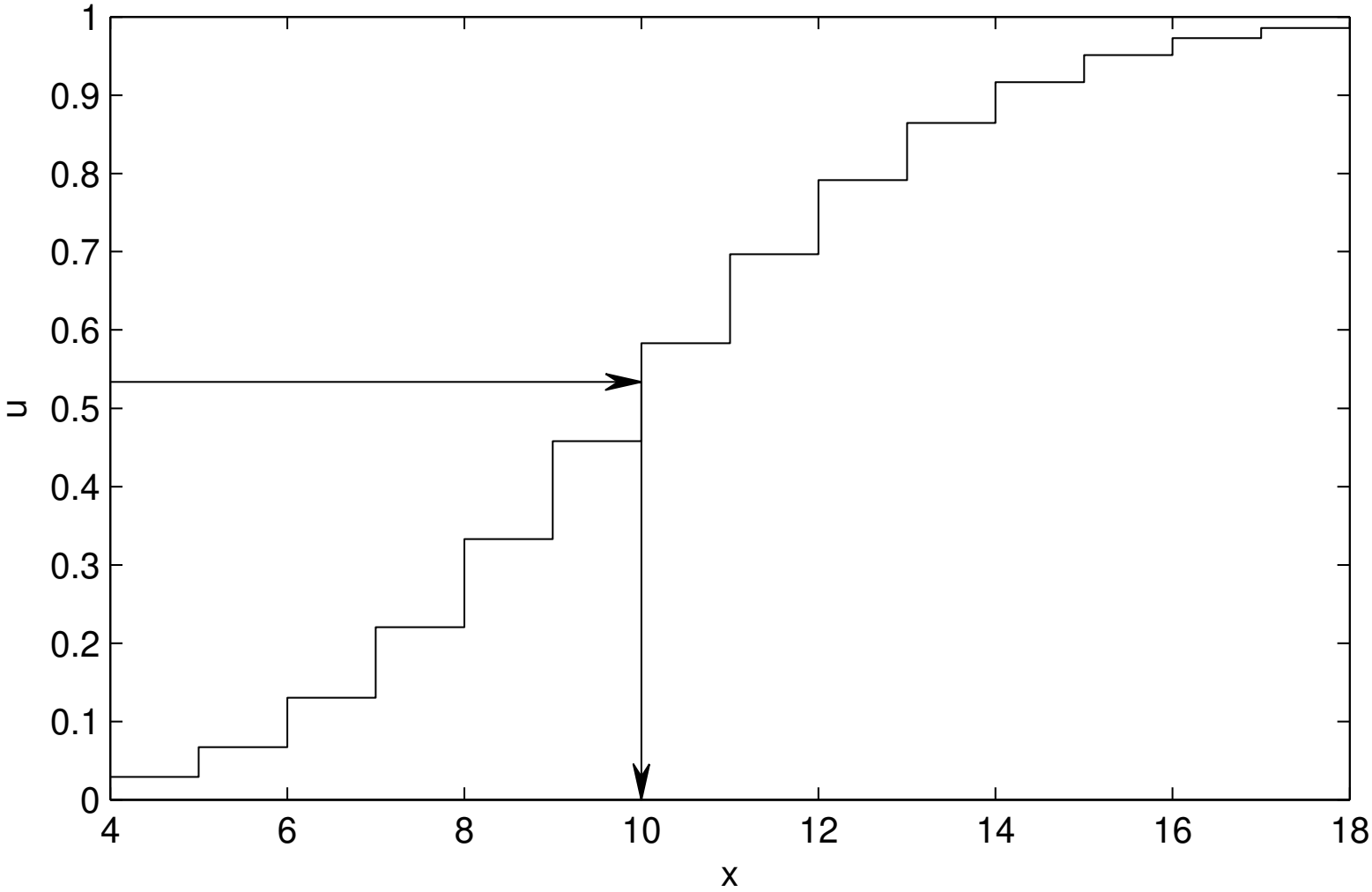
$$\overline{C}(n) \equiv \mathbb{P}(N \leq n) = e^{-\lambda} \sum_{m=0}^{n} \frac{\lambda^m}{m!}.$$

To generate N, can take a uniform $(0,1)$ random variable U and then compute $N = \overline{C}^{-1}(U)$, where N is the smallest integer such that

$$U \leq \overline{C}(N)$$

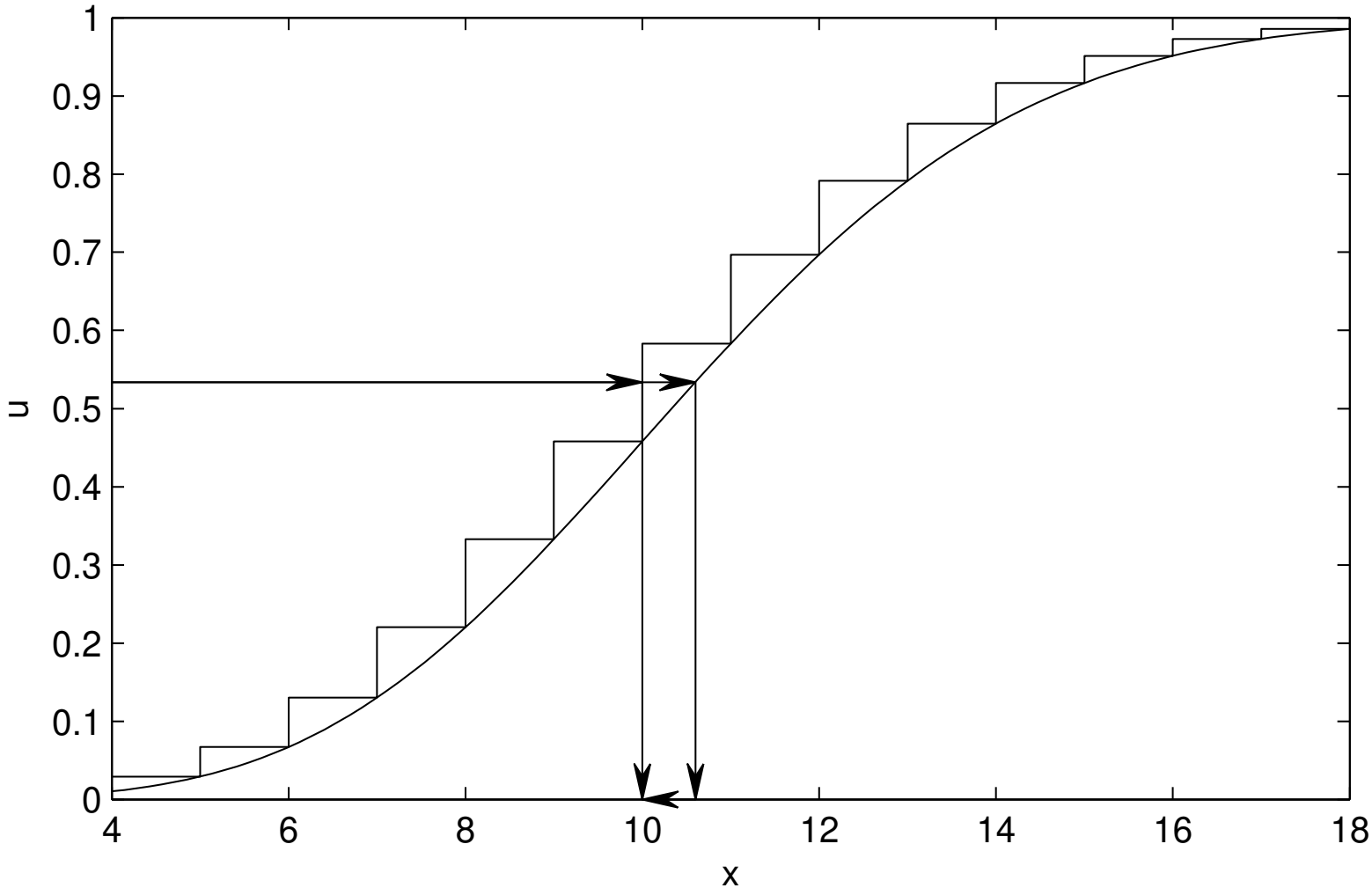
Poisson CDF and inverse

Illustration of the inversion process



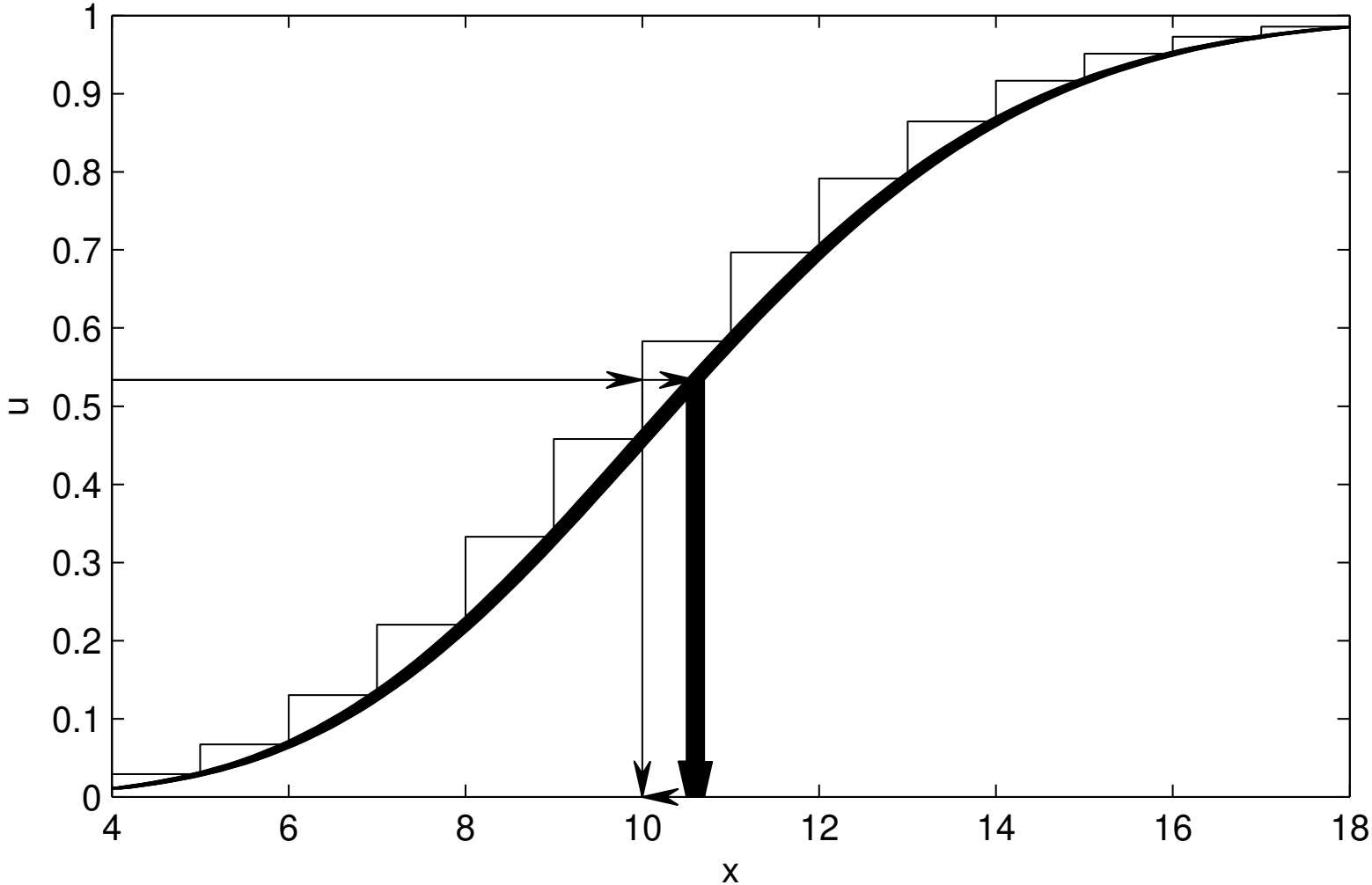
Poisson CDF and inverse

Illustration of the inversion process through rounding down of some $Q(u) \equiv C^{-1}(u)$ to give $\overline{C}^{-1}(u)$



Poisson CDF and inverse

Errors in approximating $Q(u)$ can only lead to errors in rounding down if near an integer



Incomplete Gamma function

If X is a positive random variable with CDF

$$C(x) \equiv \mathbb{P}(X < x) = \frac{1}{\Gamma(x)} \int_\lambda^\infty e^{-t}\, t^{x-1}\, \mathrm{d}t.$$

then integration by parts gives

$$\mathbb{P}(\lfloor X \rfloor \leq n) = \frac{1}{n!} \int_\lambda^\infty e^{-t}\, t^n\, \mathrm{d}t = e^{-\lambda} \sum_{m=0}^{n} \frac{\lambda^m}{m!}$$

$$\Longrightarrow \quad \overline{C}^{-1}(u) = \lfloor C^{-1}(u) \rfloor$$

We will approximate $Q(u) \equiv C^{-1}(u)$ so that
$|\widetilde{Q}(u) - Q(u)| < \delta \ll 1$

This will round down correctly except when $Q(u)$ is within δ of an integer – then we need to check some $\overline{C}(m)$

Temme expansion

Temme (1979) derived a uniformly convergent asymptotic expansion
for $C(x)$ of the form

$$C(x) = \Phi\left(\lambda^{\frac{1}{2}} f(r)\right) + \lambda^{-\frac{1}{2}} \phi\left(\lambda^{\frac{1}{2}} f(r)\right) \sum_{n=0}^{\infty} \lambda^{-n} a_n(r)$$

where $r = x/\lambda$ and

$$f(r) \equiv \sqrt{2\,(1 - r + r \log r)},$$

with the sign of the square root matching the sign of $r-1$.

Temme expansion

Based on this, can prove that the quantile function is

$$Q(u) \approx \lambda\, r + c_0(r)$$

where

$$r = f^{-1}(w/\sqrt{\lambda}), \qquad w = \Phi^{-1}(u)$$

and

$$c_0(r) = \frac{\log\left(\, f(r)\,\sqrt{r}/(r-1)\,\right)}{\log r}$$

Both $f^{-1}(s)$ and $c_0(r)$ can be approximated very accurately over a central range by polynomials, and an additional *ad hoc* correction gives

$$\widetilde{Q}_T(u) = \lambda\ r + p_2(r) + p_3(r)/\lambda$$

$C(m)$ evaluation

In double precision, when $\widetilde{Q}(u)$ is too close to an integer $m+1$, we need to evaluate $C(m)$ to choose between m and $m+1$.

When $\frac{1}{2}\,\lambda \leq m \leq 2\,\lambda$, this can be done very accurately using another approximation due to Temme (1987).

Outside this range, a modified version of bottom-up / top-down summation can be used, because successive terms decrease by factor 2 or more.

In single precision this "correction" procedure does not improve the accuracy.

Single precision algorithm

given inputs: λ, u

if $\lambda > 4$

$w := \Phi^{-1}(u)$

$s := w/\sqrt{\lambda}$

if $s_{min} < s < s_{max}$ main branch

$r := p_1(s)$

$x := \lambda r + p_2(r) + p_3(r)/\lambda$

else

$r := f^{-1}(w/\sqrt{\lambda})$ Newton iteration

$x := \lambda r + c_0(r)$

$x := x - 0.0218/(x+0.065\lambda)$

end

$n := \lfloor x \rfloor$

Single precision algorithm

```
if x > 10
    return n
end
end

use bottom-up summation to determine n

if u > 0.5 and not accurate enough
    use top-down summation to determine n
end
```

Top-down summation finds smallest n such that

$$1-u \;\geq\; e^{-\lambda} \sum_{m=n+1}^{\infty} \frac{\lambda^m}{m!}$$

Double precision algorithm

given inputs: λ, u

if $\lambda > 4$

$w := \Phi^{-1}(u)$

$s := w/\sqrt{\lambda}$

if $s_{min} < s < s_{max}$

$r := p_1(s)$

$x := \lambda r + p_2(r) + p_3(r)/\lambda$

$\delta := 2 \times 10^{-5}$

else

$r := f^{-1}(w/\sqrt{\lambda})$

$x := \lambda r + c_0(r)$

$x := x - 0.0218/(x + 0.065\lambda)$

$\delta := 0.01/\lambda$

end

$n := \lfloor x + \delta \rfloor$

Double precision algorithm

```
if x > 10
    if x − n > δ
        return n
    else if C(n) < u          "correction" test
        return n
    else
        return n − 1
    end
end

use bottom-up summation to determine n

if u > 0.5 and not accurate enough
    use top-down summation to determine n
end
```

Conclusions

- By approximating the inverse incomplete Gamma function, developed an approach for inverting the Poisson CDF for $\lambda > 4$
- Computational cost is roughly cost of inverse Normal CDF function plus three polynomials of degree 8–12
- Paper: "Approximation of the inverse Poisson cumulative distribution function", *ACM Transactions on Mathematical Software*, 42(1), 2015
- Paper and open source software are available: `http://people.maths.ox.ac.uk/gilesm/codes/poissinv/`