Appendix S1: Calculation of the IPM integral by numerical quadrature.

We implemented three numerical quadrature algorithms: rectangle (or midpoint), trapezoidal and the alternative extended Simpson's rules (AES) (see Press et al. 1989 for a description of different quadrature rules), in order of decreasing approximating error, respectively. Those three algorithms divide the integration range into smaller subintervals, all of the same length h, and the integrand is then approximated independently by a suitable low-order polynomial within each subinterval. The error committed by that approximation is then proportional to a power of h. A smaller h will in general increase the precision of the results at the cost of having to evaluate the integrand in more subintervals. Therefore, we set out to check the precision and relative CPU time of all three quadrature algorithms with respect to h values (in cm) of 0.5, 0.1, 0.05 and 0.01. We used $h = 0.01$ and the AES algorithm as our benchmark. At every time step, functions $s(x)$, $g(x, y)$ and $F(y)$ in Eq. 4 and 5 were evaluated with pre-calculated parameter values (see below) and the integral in Eq. 5 above was then solved numerically. Values for Δ and total projected time had predetermined fixed values, as described above. The results of the tests showed that, with the AES methodology, there was relatively little gain in precision for $n_{t+\Delta}(y)$ when choosing h values below 0.1 (differences in basal area and total tree number were always $< 10^{-4}$ and $< 10^{-3}$ in units of m^2h^{-1}, for AES and $h \leq 0.1$ cm). When we applied the midpoint or trapezoidal implementations, the differences in $n_{t+\Delta}(y)$ with the benchmark quadrature were always much larger. On the other hand, CPU time increased markedly with smaller h, although for any given h it was always very similar for the three algorithms. Consequently, throughout our study we chose to solve the integral in Eq. 5 numerically with $h = 0.1$ cm and the AES quadrature.