Accuracy and mean RT in a diffusion superposition model with deadline

Online supplement 1 for "Audiovisual detection at different intensities and delays"

This online supplement provides implementation details on the derivation of the predictions for mean response time and accuracy for the diffusion superposition model (Diederich, 1995; Schwarz, 1994) with a deadline.

Libraries

The code requires the inverse Gaussian distribution package SuppDists (Wheeler, 2016, available from CRAN). In addition, package mvtnorm (Genz et al., 2016) is used for the bivariate Normal distribution.

```
library(SuppDists)
library(mvtnorm)
```

Accuracy

For unimodal/synchronous stimuli, accuracy is given by the inverse Gaussian distribution at the deadline d. For example, Monkey 1's accuracy in Condition v (low intensity visual stimuli) is given by

acc_sync(d=951, c=100, mu=0.35, sigma2=8.55^2).

```
acc_sync = function(d, c, mu, sigma2)
{
        pinvGauss(d, nu=c/mu, lambda=c*c/sigma2)
}
```

For stimuli with onset asynchrony τ, accuracy is given by (12) which is the sum of the inverse Gaussian distribution at time τ and four integrals of the form $q \cdot \int_{-\infty}^{c} \exp(rx) \cdot \phi(x \mid m_1, s_1^2) \cdot \Phi(x \mid m_2, s_2^2)\, dx$:

1. $q = 1, r = 0, m_1 = \mu_V \tau, s_1^2 = \sigma_V^2 \tau, m_2 = c - \mu_{AV} d', s_2^2 = \sigma_{AV}^2 d'$.
2. $q = \exp\left(\frac{2c\mu_{AV}}{\sigma_{AV}^2}\right), r = -\frac{2\mu_{AV}}{\sigma_{AV}^2}, m_1 = \mu_V \tau, s_1^2 = \sigma_V^2 \tau, m_2 = c + \mu_{AV} d', s_2^2 = \sigma_{AV}^2 d'$.
3. $q = -\exp\left(\frac{2c\mu_V}{\sigma_V^2}\right), r = 0, m_1 = 2c + \mu_V \tau, s_1^2 = \sigma_V^2 \tau, m_2 = c - \mu_{AV} d', s_2^2 = \sigma_{AV}^2 d'$.
4. $q = -\exp\left(\frac{2c\mu_V}{\sigma_V^2} + \frac{2c\mu_{AV}}{\sigma_{AV}^2}\right), r = -\frac{2\mu_{AV}}{\sigma_{AV}^2}, m_1 = 2c + \mu_V \tau, s_1^2 = \sigma_V^2 \tau, m_2 = c + \mu_{AV} d', s_2^2 = \sigma_{AV}^2 d'$

with $d' = d - \tau$.

Three convenience functions evaluate these integrals using Eq. 10,010.1 in (Owen, 1980):

1. $\int_{-\infty}^{y} \phi(u) \cdot \Phi(a + bu)\, du = \Phi\Phi\left(\frac{a}{\sqrt{1+b^2}}, y \,\middle|\, \varrho = -\frac{b}{\sqrt{1+b^2}}\right)$, with $m_1' = m_1 + r s_1^2$
2. $\int_{-\infty}^{c} \phi(x \mid m_1', s_1^2) \cdot \Phi(x \mid m_2, s_2^2)\, dx = \int_{-\infty}^{\frac{(c-m_1')}{s_1}} \phi(u) \cdot \Phi\left(\frac{s_1}{s_2} u + \frac{m_1' - m_2}{s_2}\right) du$
3. $\int_{-\infty}^{c} \exp(rx)\, \phi(x \mid m_1, s_1^2) \Phi(x \mid m_2, s_2^2)\, dx = \exp\left(r m_1 + \frac{r^2 s_1^2}{2}\right) \int_{-\infty}^{c} \phi(x \mid m_1', s_1^2) \cdot \Phi(x \mid m_2, s_2^2)\, dx,$

```
# Integrate dnorm(x) * pnorm(a + b*x) from -Inf to y (Owen, 1980, Eq. 10,010.1)
owen10_010.1 = function(y, a, b)
{
	rho = -b/sqrt(1 + b*b)
	pmvnorm(lower=c(-Inf, -Inf), upper=c(a/sqrt(1+b*b), y), corr=matrix(c(1, rho, rho, 1), nrow=2))
}

# Integrate dnorm(x | mu1, sigma1) * pnorm(x | mu2, sigma2) from -Inf to y
owen10_010.1b = function(y, mu1, sigma1, mu2, sigma2)
{
	owen10_010.1(y=(y - mu1)/sigma1, a=(mu1 - mu2)/sigma2, b=sigma1/sigma2)
}

# Integrate exp(rx) * dnorm(x | mu1, sigma1) * pnorm(x | mu2, sigma2) from -Inf to y
owen10_010.1c = function(y, r, mu1, sigma1, mu2, sigma2)
{
	exp(r * mu1 + r^2 * sigma1^2 / 2) * owen10_010.1b(y, mu1 + r * sigma1^2, sigma1, mu2, sigma2)
}
```

For example, Monkey 1's accuracy in Condition v100a (low intensity) is given by

acc_async(d=951, c=100, mua=.35, sigma2a=8.55^2, mub=0.01, sigma2b=4.3^2, tau=100).

```
acc_async = function(d, c, mua, sigma2a, mub, sigma2b, tau)
{
	# Probability of absorption within 0…tau
	muab = mua + mub
	sigma2ab = sigma2a + sigma2b
	p0 = acc_sync(d=tau, c=c, mu=mua, sigma2=sigma2a)

	# 1st term of Equation A.3
	q = 1
	r = 0
	mu1 = mua * tau
	sigma1=sqrt(sigma2a * tau)
	mu2 = c - muab * (d - tau)
	sigma2 = sqrt(sigma2ab * (d - tau))
	p1 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

	# 2nd integral
	q = exp(2 * c * muab / sigma2ab)
	r = -2 * muab / sigma2ab
	mu1 = mua * tau
	sigma1 = sqrt(sigma2a * tau)
	mu2 = c + muab * (d - tau)
	sigma2 = sqrt(sigma2ab * (d - tau))
	p2 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)
```

```
        # 3rd integral
        q = -exp(2 * c * mua / sigma2a)
        r = 0
        mu1 = 2 * c + mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c - muab * (d - tau)
        sigma2 = sqrt(sigma2ab * (d - tau))
        p3 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

        # 4th integral
        q = -exp(2 * c * mua / sigma2a + 2 * c * muab / sigma2ab)
        r = -2 * muab / sigma2ab
        mu1 = 2 * c + mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c + muab * (d - tau)
        sigma2 = sqrt(sigma2ab * (d - tau))
        p4 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

        p0 + p1 + p2 + p3 + p4
}
```

Mean response time

For unimodal/synchronous stimuli, the mean RT is given by Equation 4, which integrates the product of the time and the inverse Gaussian density from zero until the deadline d.

```
# Mean RT in unimodal and synchronous stimuli
mrt_sync = function(d, c, mu, sigma2)
{
        # Integral t * density from 0 to d (Schwarz, 1994, Equation A.2)
        m = c / mu * {pnorm(mu*d, c, sqrt(sigma2*d)) -
                exp(2*c*mu / sigma2) * pnorm(pnorm(-mu*d, c, sqrt(sigma2*d))}

        # Normalize with detection accuracy
        m / acc_sync(d, c, mu, sigma2)
}
```

Monkey 1's mean RT in Condition v is given by mrt_sync (d=951, c=100, mu=0.35, sigma2=8.55^2), plus μ_{M}.

For stimuli with onset asynchrony, mean RT is given by (13)–(15), and eight integrals of the forms $q\int_{-\infty}^{c}\exp(rx)\,\phi(x\mid m_1,s_1^2)\Phi(x\mid m_2,s_2^2)\,dx$ and $q\int_{-\infty}^{c}x\,\exp(rx)\,\phi(x\mid m_1,s_1^2)\cdot\Phi(x\mid m_2,s_2^2)\,dx$.

1. $q=\frac{c}{\mu_{\mathrm{AV}}}, r=0, m_1=\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c-\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'.$
2. $q=-\frac{c}{\mu_{\mathrm{AV}}}\cdot\exp\left(-\frac{2c\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}\right), r=-\frac{2\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}, m_1=\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c+\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'.$
3. $q=-\frac{c}{\mu_{\mathrm{AV}}}\cdot\exp\left(\frac{2c\mu_{\mathrm{V}}}{\sigma_{\mathrm{V}}^2}\right), r=0, m_1=2c+\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c-\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'$
4. $q=\frac{c}{\mu_{\mathrm{AV}}}\cdot\exp\left(\frac{2c\mu_{\mathrm{V}}}{\sigma_{\mathrm{V}}^2}+\frac{2c\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}\right), r=-\frac{2\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}, m_1=2c+\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c+\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'$
5. $q=-\frac{1}{\mu_{\mathrm{AV}}}, r=0, m_1=\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c-\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'.$
6. $q=\frac{1}{\mu_{\mathrm{AV}}}\cdot\exp\left(\frac{2c\mu_{\mathrm{V}}}{\sigma_{\mathrm{V}}^2}\right), r=0, m_1=2c+\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c-\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'$
7. $q=\frac{1}{\mu_{\mathrm{AV}}}\cdot\exp\left(-\frac{2c\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}\right), r=-\frac{2\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}, m_1=\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c+\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'.$
8. $q=-\frac{1}{\mu_{\mathrm{AV}}}\cdot\exp\left(\frac{2c\mu_{\mathrm{V}}}{\sigma_{\mathrm{V}}^2}+\frac{2c\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}\right), r=-\frac{2\mu_{\mathrm{AV}}}{\sigma_{\mathrm{AV}}^2}, m_1=2c+\mu_{\mathrm{V}}\tau, s_1^2=\sigma_{\mathrm{V}}^2\tau, m_2=c+\mu_{\mathrm{AV}}d', s_2^2=\sigma_{\mathrm{AV}}^2d'$

We defined again convenience functions that transform these integrals to two expressions that match Equations 10,010.1 (see accuracy) and 10,011.1 in (Owen, 1980):

1. $\int_{-\infty}^{y}u\cdot\phi(u)\cdot\Phi(a+bu)\,du=\frac{b}{\sqrt{1+b^2}}\cdot\phi\left(\frac{a}{\sqrt{1+b^2}}\right)\cdot\Phi\left(u\sqrt{1+b^2}+\frac{ab}{\sqrt{1+b^2}}\right)-\phi(u)\cdot\Phi(a+bu)$
2. $\int_{-\infty}^{c}x\cdot\phi(x\mid m_1,s_1^2)\cdot\Phi(x\mid m_2,s_2^2)\,dx=\int_{-\infty}^{\frac{c-m_1}{s_1}}(s_1u+m_1)\cdot\phi(u)\cdot\Phi\left(\frac{s_1}{s_2}u+\frac{m_1-m_2}{s_2}\right)du$
   $=m_1\int_{-\infty}^{\frac{c-m_1}{s_1}}\phi(u)\cdot\Phi\left(\frac{s_1}{s_2}u+\frac{m_1-m_2}{s_2}\right)du$ [see accuracy] $+s_1\int_{-\infty}^{\frac{c-m_1}{s_1}}u\cdot\phi(u)\cdot\Phi\left(\frac{s_1}{s_2}u+\frac{m_1-m_2}{s_2}\right)du$
3. $\int_{-\infty}^{c}x\exp(rx)\,\phi(x\mid m_1,s_1^2)\Phi(x\mid m_2,s_2^2)\,dx=\exp\left(rm_1+\frac{r^2s_1^2}{2}\right)\int_{-\infty}^{c}x\cdot\phi(x\mid m_1,s_1^2)\cdot\Phi(x\mid m_2,s_2^2)\,dx$

```
# Integrate dnorm(x) * pnorm(a + b*x) from -Inf to y (Owen, 1980, Eq. 10,011.1)
owen10_011.1 = function(y, a, b)
{
        bb = sqrt(1 + b*b)
        b/bb * dnorm(a/bb) * pnorm(y*bb + a*b/bb) - pnorm(a + b*y)*dnorm(y)
}

# Integrate x * dnorm(x | mu1, sigma1) * pnorm(x | mu2, sigma2) from -Inf to y
owen10_011.1b = function(y, mu1, sigma1, mu2, sigma2)
{
        owen10_011.1(y=(y - mu1)/sigma1, a=(mu1 - mu2)/sigma2, b=sigma1/sigma2) * sigma1 +
          mu1 * owen10_010.1(y=(y - mu1)/sigma1, a=(mu1 - mu2)/sigma2, b=sigma1/sigma2)
}
```

```
# Integrate x * exp(rx) * dnorm(x | mu1, sigma1) * pnorm(x | mu2, sigma2) from -Inf to y
owen10_010.1c = function(y, r, mu1, sigma1, mu2, sigma2)
{
	exp(mu1 * v + sigma1^2 * v^2/2) * owen10_011.1b(y, mu1 + v * sigma1^2, sigma1, mu2, sigma2)
}
```

For example, Monkey 1's mean RT in Condition v100a (low intensity) is predicted to

mrt_async(d=951, c=100, mua=0.35, sigma2a=8.55^2, mub=0.01, sigma2b=4.3^2, tau=100).

```
mrt_async = function(d, c, mua, sigma2a, mub=0.34, sigma2b, tau)
{
	muab = mua + mub
	sigma2ab = sigma2a + sigma2b
	d_ = d - tau

	# Integral t * density from 0 to tau (Schwarz, 1994, Equation A.2)
	m0 = c / mua * {pnorm(mua*tau, c, sqrt(sigma2a*tau)) -
		exp(2*c*mua / sigma2a) * pnorm(pnorm(-mua*tau, c, sqrt(sigma2a*tau))}

	# First term of Eq. 15 (tau * second term of Eq. 12)
	mtau = tau*{acc_async(d, c, mua, sigma2a, mub, sigma2b, tau) - acc_sync(d, c, mua, sigma2a)

	# 1st integral in A.6
	q = c / mua
	r = 0
	mu1 = mua * tau
	sigma1=sqrt(sigma2a * tau)
	mu2 = c - muab * d_
	sigma2 = sqrt(sigma2ab * d_)
	m1 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

	# 2nd integral
	q = -c / muab * exp(2 * c * muab / sigma2ab)
	r = -2 * muab / sigma2ab
	mu1 = mua * tau
	sigma1 = sqrt(sigma2a * tau)
	mu2 = c + muab * d_
	sigma2 = sqrt(sigma2ab * d_)
	m2 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

	# 3rd integral
	q = -c / muab * exp(2 * c * mua / sigma2a)
	r = 0
	mu1 = 2 * c + mua * tau
	sigma1 = sqrt(sigma2a * tau)
	mu2 = c - muab * d_
	sigma2 = sqrt(sigma2ab * d_)
	m3 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

# continued on next page
```

```
        # 4th integral
        q = c / muab * exp(2 * c * mua / sigma2a + 2 * c * muab / sigma2ab)
        r = -2 * muab / sigma2ab
        mu1 = 2 * c + mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c + muab * d_
        sigma2 = sqrt(sigma2ab * d_)
        m4 = q * owen10_010.1c(c, r, mu1, sigma1, mu2, sigma2)

        # 5th integral
        q = -1 / mua
        r = 0
        mu1 = mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c - muab * d_
        sigma2 = sqrt(sigma2ab * d_)
        m5 = q * owen10_011.1c(c, r, mu1, sigma1, mu2, sigma2)

        # 6th integral
        q = 1 / muab * exp(2 * c * muab / sigma2ab)
        r = -2 * muab / sigma2ab
        mu1 = mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c + muab * d_
        sigma2 = sqrt(sigma2ab * d_)
        m6 = q * owen10_011.1c(c, r, mu1, sigma1, mu2, sigma2)

        # 7th integral
        q = 1 / muab * exp(2 * c * mua / sigma2a)
        r = 0
        mu1 = 2 * c + mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c - muab * d_
        sigma2 = sqrt(sigma2ab * d_)
        m7 = q * owen10_011.1c(c, r, mu1, sigma1, mu2, sigma2)

        # 8th integral
        q = -1 / muab * exp(2 * c * mua / sigma2a + 2 * c * muab / sigma2ab)
        r = -2 * muab / sigma2ab
        mu1 = 2 * c + mua * tau
        sigma1 = sqrt(sigma2a * tau)
        mu2 = c + muab * d_
        sigma2 = sqrt(sigma2ab * d_)
        m8 = q * owen10_011.1c(c, r, mu1, sigma1, mu2, sigma2)

        # Return value: normalized integral t * density from 0 to d (Eq. 13)
        p = acc_async(d, c, mua, sigma2a, mub, sigma2b, tau)
        (m0 + mtau + m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8) / p
}
```

Accuracy and mean RT in a diffusion superposition model with two barriers

Online supplement 2 for "Audiovisual detection at different intensities and delays"

This online supplement provides an R implementation of the analytical expressions that we derived for the diffusion superposition model with two absorbing barriers (Blurton, Greenlee, & Gondan, 2014).

Helper functions

Several times in the code below, we use the product of an exponential and the standard Normal distribution function, $\exp(a) \cdot \Phi(x)$. These expressions often raise numerical problems because large exponentials are multiplied with small results of Φ. In these cases, we use an approximation by (Kiani, Panaretos, Psarakis, & Saleem, 2008) that translates the Normal distribution to an exponential, so that the two exponentials can be evaluated together.

```
# exp(a) * pnorm(b) using an approximation by Kiani et al. (2008)
exp_pnorm = function(a, b)
{
	r = exp(a + pnorm(b, log.p=TRUE))
	d = is.nan(r) & b < -5.5
	r[d] = 1/sqrt(2) * exp(a - b[d]*b[d]/2) * (0.5641882/b[d]/b[d]/b[d] - 1/b[d]/sqrt(pi))
	r
}
```

This is the inverse Gaussian distribution, in its pure form and multiplied with an exponential.

```
# Inverse Gaussian distribution
pwald = function(t, c, mu, sigma2)
{
	pnorm((mu*t-c) / sqrt(sigma2*t)) + exp_pnorm(2*c*mu/sigma2, (-mu*t-c) / sqrt(sigma2*t))
}

# exp(e) * Inverse Gaussian distribution
exp_pwald = function(e, t, c, mu, sigma2)
{
	exp_pnorm(e, (mu*t-c) / sqrt(sigma2*t)) + exp_pnorm(e + 2*c*mu/sigma2, (-mu*t-c) / sqrt(sigma2*t))
}
```

We also need the next antiderivative of the inverse Gaussian distribution (multiplied by an exponential).

```
# Integral 0...t x * dwald(x) dx (e.g., Schwarz, 1994, Appendix A.1) (multiplied with an exponential)
exp_ipwald = function(e, t, c, mu, sigma2)
{
	c/mu * {exp_pnorm(e, (mu*t-c)/sqrt(t*sigma2)) - exp_pnorm(e+2*c*mu/sigma2, (-mu*t-c)/sqrt(t*sigma2))}
}
```

The following expressions yield the density and the distribution of the not yet absorbed processes at time τ, that is, the onset of the second stimulus (Eqs. B.3a, B.4a, B.3b in Appendix B). We first consider the density (see Cox & Miller, 1965, Eq. 78). The density is used below in the mean detection time. Although there exists two representations of this density (Eqs. 78 and 81 in Cox & Miller), we only implemented the "small-drift" representation (we did not encounter convergence problems or numerical problems).

```
# Density of processes not yet absorbed (Cox & Miller, Eq. 78)
w_2B = function(x, tau, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
        g = dnorm(x, mean=mu*tau, sd=sqrt(sigma2*tau)) -
          exp(2*u*mu/sigma2) * dnorm(x, mean=2*u + mu*tau, sd=sqrt(sigma2*tau))

        k = 1
        repeat
        {
                sk = 2 * k * (u+l)
                s_k = 2 * u - sk
                p1 = exp(sk*mu/sigma2) * dnorm(x, mean=sk + mu*tau, sd=sqrt(sigma2*tau))
                m1 = exp(s_k*mu/sigma2) * dnorm(x, mean=s_k + mu*tau, sd=sqrt(sigma2*tau))

                sk = -2 * k * (u+l)
                s_k = 2 * u - sk
                p2 = exp(sk*mu/sigma2) * dnorm(x, mean=sk + mu*tau, sd=sqrt(sigma2*tau))
                m2 = exp(s_k*mu/sigma2) * dnorm(x, mean=s_k + mu*tau, sd=sqrt(sigma2*tau))

                g = p1 - m2 + p2 - m1 + g
                if(all(p1 <= tol) & all(m1 <= tol) & all(p2 <= tol) & all(m2 <= tol))
                        return(g)
                k = k+1
        }
}
```

The distribution functions follow from integration of Cox and Miller's (1978) Equations 78 and 81. The functions exist in two infinite series representations with different convergence behavior for small and large drifts. We first determine the number K of required terms that guarantee a specific truncation error $\varepsilon > 0$ and then choose the representation that needs fewer terms to converge.

This is the large drift representation (Equation B.3a).

```
# Distribution of the not-yet-absorbed processes at time tau, required terms for the large drift representation
W_2B.large.K = function(tau, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
        ceiling(pmax(1, -sigma2/(u+l)/mu * log(tol)))
}

# continued on next page

# Distribution of the not-yet-absorbed processes at time tau, large drift representation
W_2B.large = function(x, tau, u, l, mu, sigma2, K)
```

```
{
        g = pnorm(x, mean=mu*tau, sd=sqrt(sigma2*tau)) -
          exp(2*u*mu/sigma2 + pnorm(x, mean=2*u + mu*tau, sd=sqrt(sigma2*tau), log=TRUE))

        k = K:1
        sk = 2 * k * (u+l)
        s_k = 2 * u - sk
        p1 = exp(sk*mu/sigma2 + pnorm(x, mean=sk + mu*tau, sd=sqrt(sigma2*tau), log=TRUE))
        m1 = exp(s_k*mu/sigma2 + pnorm(x, mean=s_k + mu*tau, sd=sqrt(sigma2*tau), log=TRUE))

        sk = -2 * k * (u+l)
        s_k = 2 * u - sk
        p2 = exp(sk*mu/sigma2 + pnorm(x, mean=sk + mu*tau, sd=sqrt(sigma2*tau), log=TRUE))
        m2 = exp(s_k*mu/sigma2 + pnorm(x, mean=s_k + mu*tau, sd=sqrt(sigma2*tau), log=TRUE))
        sum(p1 - m2) + sum(p2 - m1) + g
}
```

This is the small-drift representation (Eq. B.3b):

```
# Distribution of the not-yet-absorbed processes at time tau, required terms for the small drift representation
W_2B.small.K = function(x, tau, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
        eps_ = pi * sigma2/2 / ((u+l)*mu/pi + sigma2) / sqrt(2 * (u+l)^2/pi/sigma2/tau) *
          exp(log(tol) + mu^2*tau/2/sigma2 - mu*x/sigma2)
        K = qnorm(pmin(1, pmax(0, eps_)), mean=0, sd=(u+l)/pi/sqrt(sigma2*tau), lower.tail=FALSE)
        pmax(1, ceiling(K))
}

# Distribution of the not-yet-absorbed processes at time tau, small drift representation
W_2B.small = function(x, tau, u, l, mu, sigma2, K)
{
        k = 1:K
        c = mu / sigma2
        ak = 2 / (u + l) * sin(k*pi*l / (u + l))
        bk = k*pi / (u + l)
        lk = 1/2 * (mu^2 / sigma2 + bk^2 * sigma2)
        pk =  1/c * ak / (1 + bk^2/c^2) * {sin(bk * (x + l)) - bk/c * cos(bk * (x + l))} * exp(c*x - lk*tau)
        sum(pk[order(abs(pk))])
}

# continued on next page
```

The wrapper function uses a mirrored solution for the negative drift case (with negative drift, W_2B.large is divergent). It also checks the convergence of the two representations and calls the one with fewer iterations.

```
W_2B = function(x, tau, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
	if(mu < 0)
		return(-W_2B(x=-x, tau=tau, u=l, l=u, mu=-mu, sigma2=sigma2, tol=tol))

	K.large = W_2B.large.K(x=x, tau=tau, u=u, l=l, mu=mu, sigma2=sigma2, tol=tol)
	K.small = W_2B.small.K(x=x, tau=tau, u=u, l=l, mu=mu, sigma2=sigma2, tol=tol)
	if(K.large < K.small)
		return(W_2B.large(x=x, tau=tau, u=u, l=l, mu=mu, sigma2=sigma2, K=K.large))

	# Otherwise
	return(W_2B.small(x=x, tau=tau, u=u, l=l, mu=mu, sigma2=sigma2, K=K.small))
}
```

Accuracy

For unimodal/synchronous stimuli, accuracy is given by the proportion of absorptions of a single stage process at the upper barrier (e.g., Horrocks & Thompson, 2004). The limit for zero drift needs handled specially, so we define again a wrapper function that checks the size of the drift.

```
# Probability for absorption at upper barrier, nonzero drift (Horrocks & Thompson, 2004, Eq. 5)
Pu_nonzero = function(u, l, mu, sigma2)
{
	expm1(-2*l*mu/sigma2) / expm1(-2*(u+l)*mu/sigma2)
}

# Zero drift case
Pu_zero = function(u, l)
{
	l / (u+l)
}

# Wrapper for both cases
Pu_single = function(u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
	# Zero drift
	if(abs(mu / sqrt(sigma2)) < tol)
	  return(Pu_zero(u, l))

	# Standard case
	Pu_nonzero(u, l, mu, sigma2)
}
```

In the two-stage process (asynchronous stimuli), the upper absorptions occur either before or after the onset of the second stimulus. The following two functions yield the first-passage time distributions in the two-barrier diffusion model. The upper barrier absorptions are obtained by a series of weighted inverse Gaussian distributions (Gondan, Blurton, & Kesselmeier, 2014). This is the so-called "small time" representation (Navarro & Fuss, 2009). The series is alternating and absolutely decreasing; for this reason, we do not need to determine the number of terms in advance but just stop evaluation if the last term is below the truncation tolerance. The absorptions at the lower barrier are obtained by a shift of the function arguments.

```
# First passage time distribution at upper barrier, small time formula
F_upper = function(t, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
	G = numeric(length(t))
	k = 0
	repeat
	{
		s2k = 2*(u+l)*k
		P = exp_pwald(-s2k*mu / sigma2, t, s2k + u, mu, sigma2)

		k = k + 1
		s_2k = 2*(u+l)*k
		M = exp_pwald((-s_2k + 2*u)*mu / sigma2, t, s_2k - u, mu, sigma2)

		G = P - M + G
		if(abs(P[length(P)]) <= tol & abs(M[length(M)]) <= tol)
			return(G)
	}
}

# Distribution of absorptions at lower barrier
F_lower = function(t, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
	F_upper(t, l, u, -mu, sigma2, tol)
}
```

As mentioned in Appendix B, there are two representations for the absorptions in the interval $[\tau, \infty]$. The first one is based on Cox and Miller (1965, Eq. 78) and is is better suited for large drifts (Expression B.7a).

```
# Integral w_2B(x, tau) * Pu(u-x, l+x): Determine number of terms for Expression B.7a
Pu_mix.large.K = function(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, tol=sqrt(.Machine$double.eps))
{
	sigma2_ab = sigma2_a + sigma2_b
	mu_ab = mu_a + mu_b
	mu_ = mu_a / sigma2_a - 2 * mu_ab / sigma2_ab
	K = -(log(tol) - 2*u*mu_) / 2 / (u+l) / mu_
	pmax(1, ceiling(K))
}
```

```
# Integral w_2B(x, tau) * Pu(u-x, l+x): Expression B.7a
Pu_mix.large = function(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, K)
{
	sigma2_ab = sigma2_a + sigma2_b
	mu_ab = mu_a + mu_b
	conv = mu_a / sigma2_a - 2 * mu_ab / sigma2_ab

	# Antiderivative of w_2B(x, tau) * Pu_single
	F = function(x)
	{
	  mn = mu_a*tau - 2*mu_ab/sigma2_ab*sigma2_a*tau
	  g = exp(pnorm(x, mean=mn, sd=sqrt(sigma2_a*tau), log=TRUE)) -
	  	exp(2*u*conv + pnorm(x, mean=mn + 2*u, sd=sqrt(sigma2_a*tau), log=TRUE))
	  k = K:1
	  sk = 2*k*(u+l)
	  s_k = 2*u - sk
	  p = exp(sk*conv + pnorm(x, mean=mn + sk, sd=sqrt(sigma2_a*tau), log=TRUE)) -
	  	exp(s_k*conv + pnorm(x, mean=mn + s_k, sd=sqrt(sigma2_a*tau), log=TRUE))
	  sk = -sk
	  s_k = 2*u - sk
	  m = exp(sk*conv + pnorm(x, mean=mn + sk, sd=sqrt(sigma2_a*tau), log=TRUE)) -
	  	exp(s_k*conv + pnorm(x, mean=mn + s_k, sd=sqrt(sigma2_a*tau), log=TRUE))

	  g = sum(p + m) + g
	  c = exp(2*(mu_ab/sigma2_ab)^2*sigma2_a*tau - 2*(l+mu_a*tau)*mu_ab/sigma2_ab)
	  return((c*g - W_2B(x, tau=tau, u=u, l=l, mu=mu_a, sigma2=sigma2_a, tol=tol)) /
	  	expm1(-2*(u+l) * mu_ab/sigma2_ab))
	}

	# Integrate
	return(F(u) - F(-l))
}
```

The other representation shows better convergence for small drifts (Appendix B, Expression B.7b).

```
# Integral w_2B(x, tau) * Pu(u-x, l+x): Determine number of terms for Expression B.7b
Pu_mix.small.K = function(x, u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, tol=sqrt(.Machine$double.eps))
{
	sigma2_ab = sigma2_a + sigma2_b
	mu_ab = mu_a + mu_b
	mu_ = mu_a / sigma2_a - 2 * mu_ab / sigma2_ab
	eps_ = exp(log(tol) + 2*l*mu_ab/sigma2_ab) * abs(expm1(-2*(u+l)*mu_ab/sigma2_ab))
	eps__ = eps_ / (2/pi * sqrt(2*(u+l)^2/pi/sigma2_a/tau) * ((u+l)*mu_/pi + 1) *
	  exp(mu_*x-mu_a^2*tau/2/sigma2_a))
	K = qnorm(pmax(0, pmin(1, eps__)), mean=0, sd=(u+l)/pi/sqrt(sigma2_a*tau), lower.tail=FALSE)
	pmax(1, ceiling(K))
}
```

```
# Integral w_2B(x, tau) * Pu(u-x, l+x): Expression B.7b
Pu_mix.small = function(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, K)
{
	sigma2_ab = sigma2_a + sigma2_b
	mu_ab = mu_a + mu_b
	mu_ = mu_a / sigma2_a - 2 * mu_ab / sigma2_ab

	# Antiderivative
	F = function(x)
	{
		k = 1:K
		bk = k*pi / (u + l)
		ak = 2 / (u + l) * sin(bk * l)
		lk = 1/2 * (mu_a^2 / sigma2_a + k^2 * pi^2 * sigma2_a / (u + l)^2)
		pk = ak / (1 + bk^2/mu_^2) *
			{1/mu_ * sin(bk * (x + l)) - bk/mu_^2 * cos(bk * (x + l))} *
			exp(mu_*x - 2*l*mu_ab/sigma2_ab - lk * tau)
		g = sum(pk[order(abs(pk))])

		f = expm1(-2*(u+l)*mu_ab/sigma2_ab)
		1 / f * (g - W_2B(x, tau=tau, u=u, l=l, mu=mu_a, sigma2=sigma2_a, tol=tol*abs(f)))
	}

	# Integrate
	return(F(u) - F(-l))
}
```

In order to determine the upper absorptions, we check which of the two Expressions B.7a and B.7b shows better convergence behavior (i.e., needs less terms to converge). We then return the sum of the upper absorptions in the first stage (F_upper) and the mixture. A few tests are needed to handle special cases, for example, whether all processes have been absorbed in the first stage.

```
# Upper absorptions for asynchronous stimuli
Pu_async = function(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, tol=sqrt(.Machine$double.eps))
{
	# First interval
	P1  = F_upper(tvec=tau, u=u, l=l, mu=mu_a, sigma2=sigma2_a, tol=tol)
	P1l = F_lower(tvec=tau, u=u, l=l, mu=mu_a, sigma2=sigma2_a, tol=tol)

	# If at time tau, everything has already been absorbed, return
	if(P1 + P1l >= 1-tol)
		return(P1)

	sigma2_ab = sigma2_a + sigma2_b
	mu_ab = mu_a + mu_b

	# continued on next page
```

```
        # Check if the problem converges; otherwise, solve mirrored problem
        conv = mu_a / sigma2_a - 2 * mu_ab / sigma2_ab
        if(conv < 0)
                return(1 - Pu_async(l, u, -mu_a, sigma2_a, tau, -mu_b, sigma2_b, tol))

        # Infinite series
        K.large = Pu_mix.large.K(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, tol)
          K.small = max(Pu_mix.small.K(x=c(u, -l), u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, tol))
          if(K.small < K.large)
                return(P1 + Pu_mix.small(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, K=K.small))

        return(P1 + Pu_mix.large(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, K=K.large))
}
```

For example, Monkey 1's detection probability for the weak v100a stimulus is predicted to be

Pu_async(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b)

Mean response time

For unimodal and synchronous presentations, an analytical solution for the expected first passage time at the upper barrier is given by Equation 13 in Grasman, Wagenmakers, and van der Maas (2009). Grasman et al.'s formula refers to the lower barrier, but is easily mirrored to yield the expected absorption time at the upper barrier. The implementation of this function also covers the special cases of 0%/100% absorptions at the upper barrier as well as the zero drift case. The prediction for the observable response time is obtained by adding the residual μ_{M} to the result.

```
# Expected first passage at upper barrier
Eu_nonzero = function(u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
        # Nothing absorbed at upper barrier. In this case, the result can be any number
        if(Pu_nonzero(u, l, mu, sigma2) <= tol)
          return(Eu_nonzero(u, l, -mu, sigma2))

        # 100% absorbed at upper barrier: then we return the one-barrier result
        if(Pu_nonzero(l, u, -mu, sigma2) <= tol)
                return(u/mu)

        # Otherwise, Grasman formula
        v = mu/sqrt(sigma2)
        z = l/sqrt(sigma2)
        a = (u+l)/sqrt(sigma2)
        return(((a+z) * expm1(2*v*(z-a)) - (a-z) * {exp(-2*v*a) - exp(2*v*z)}) /
                v / expm1(2*v*z) * exp(2*v*a) / expm1(2*a*v))
}



# continued on next page

# Zero drift case
Eu_zero = function(u, l, sigma2)
```

```
{
	u * (u+2*l) / sigma2 / 3
}

# Wrapper for all cases
Eu_single = function(u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
	E = numeric(length(z))

	if(abs(mu) <= tol)
		return(Eu_zero(u, l, sigma2))

	Eu_nonzero(u, l, mu, sigma2, tol)
}
```

For asynchronous stimuli, the expected detection time is a weighted sum of the expected absorption before and after the onset of the second stimulus (i.e., the SOA τ). The first part is obtained by the integration of the distribution of the upper absorptions (function IF_upper below). This integral can be obtained term by term from the series of the distribution function, and is again an absolutely decreasing alternating series, each term being an itself integral of the inverse Gaussian distribution function. We only use the small-time representation (because τ is small and finite in our experiment).

```
# Integral 0...t x * F_upper(x) dx
IF_upper = function(t, u, l, mu, sigma2, tol=sqrt(.Machine$double.eps))
{
	S = numeric(length(t))
	k = 0
	repeat
	{
		s2k = 2*(u+l)*k + u
		P = exp_ipwald(s2k*mu/sigma2, t, s2k, -mu, sigma2)
	  s2k = 2*(u+l)*(k+1) - u
		M = exp_ipwald(s2k*mu/sigma2, t, s2k, -mu, sigma2)

		S = S + (P - M)
		if(all(abs(P) <= tol) & all(abs(M) <= tol))
			return(exp(mu*u/sigma2) * S)
		k = k + 1
	}
}
```

The second part results from a compound of the density of not-yet absorbed processes between the two absorbing barriers, multiplied by the expected absorption time.

```
# Two-stage process
Eu_async = function(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b, tol=sqrt(.Machine$double.eps))
{
        # Absorptions at first stage
        P1 = F_upper(tau, u, l, mu_a, sigma2_a, tol)
        E1 = IF_upper(tau, u, l, mu_a, sigma2_a, tol)

        # T > tau
        P2 = Pu_async(u, l, mu_a, sigma2_a, tau, mu_b, sigma2_b)
        E2 = tau * (P2 - P1)

        f = function(x)
          w_2B(x, tau, u, l, mu_a, sigma2_a) *
          Pu_single(u-x, l+x, mu_a + mu_b, sigma2_a + sigma2_b) *
          Eu_single(u-x, l+x, mu_a + mu_b, sigma2_a + sigma2_b)^
        E3 = integrate(Vectorize(f), lower=-l, upper=u)$value # built-in function

        return((E1 + E2 + E3)/P2)
}
```

The prediction for the observable response time is obtained by adding the residual μ_{M} to the result.

References

Blurton, S. P., Greenlee, M., & Gondan, M. (2014). Multisensory processing of redundant information in go/no-go and choice responses. *Atten Percept Psychophys,* 76(4), 1212-1233.

Diederich, A. (1995). Intersensory Facilitation of Reaction Time: Evaluation of Counter and Diffusion Coactivation Models. *Journal of Mathematical Psychology,* 39(2), 197-215.

Gondan, M., Blurton, S. P., & Kesselmeier, M. (2014). Even faster and even more accurate first-passage time densities and distributions for the Wiener diffusion model. *Journal of Mathematical Psychology,* 60, 20-22.

Grasman, R. P. P. P., Wagenmakers, E.-J., & van der Maas, H. L. J. (2009). On the mean and variance of response times under the diffusion model with an application to parameter estimation. *Journal of Mathematical Psychology,* 53(2), 55-68.

Kiani, M., Panaretos, J., Psarakis, S., & Saleem, M. (2008). Approximations to the normal distribution function and an extended table for the mean range of the normal variables.

Navarro, D. J., & Fuss, I. G. (2009). Fast and accurate calculations for first-passage times in Wiener diffusion models. *Journal of Mathematical Psychology,* 53(4), 222-230.

Owen, D. B. (1980). A table of Normal integrals. *Communications in Statistics – Simulation and Computation,* 9, 389–419.

Schwarz, W. (1994). Diffusion, Superposition and the Redundant-Targets Effect. *Journal of Mathematical Psychology,* 38(4), 504-520.