Buffon's Needle Problem

Manuel Eberl

April 19, 2020

Abstract

In the 18th century, Georges-Louis Leclerc, Comte de Buffon posed and later solved the following problem [1, 2], which is often called the first problem ever solved in geometric probability: Given a floor divided into vertical strips of the same width, what is the probability that a needle thrown onto the floor randomly will cross two strips?

This entry formally defines the problem in the case where the needle's position is chosen uniformly at random in a single strip around the origin (which is equivalent to larger arrangements due to symmetry). It then provides proofs of the simple solution in the case where the needle's length is no greater than the width of the strips and the more complicated solution in the opposite case.

Contents

1 Buffon's Needle Problem

theory *Buffons-Needle*
  imports *HOL−Probability.Probability*
begin

1.1 Auxiliary material

lemma *sin-le-zero'*: $sin\ x \leq 0$ **if** $x \geq -pi\ x \leq 0$ **for** x
  by (*metis minus-le-iff neg-0-le-iff-le sin-ge-zero sin-minus that*(*1*) *that*(*2*))

1.2 Problem definition

Consider a needle of length l whose centre has the x-coordinate x. The following then defines the set of all x-coordinates that the needle covers (i.e. the projection of the needle onto the x-axis.)

definition *needle* :: *real* ⇒ *real* ⇒ *real* ⇒ *real set* **where**
  needle l x φ = closed-segment (x − l / 2 ∗ sin φ) (x + l / 2 ∗ sin φ)

Buffon's Needle problem is then this: Assuming the needle's x position is chosen uniformly at random in a strip of width d centred at the origin, what is the probability that the needle crosses at least one of the left/right boundaries of that strip (located at $x = \pm\frac{1}{2}d$)?

definition *buffon* :: *real* ⇒ *real* ⇒ *bool measure* **where**
  buffon l d =
    do {
      (x, φ) ← uniform-measure lborel ({−d/2..d/2} × {−pi..pi});
      return (count-space UNIV) (needle l x φ ∩ {−d/2, d/2} ≠ {})
    }

1.3 Derivation of the solution

The following form is a bit easier to handle.

lemma *buffon-altdef*:
  buffon l d =
    do {
      (x, φ) ← uniform-measure lborel ({−d/2..d/2} × {−pi..pi});
      return (count-space UNIV)
        (let a = x − l / 2 ∗ sin φ; b = x + l / 2 ∗ sin φ
          in min a b + d/2 ≤ 0 ∧ max a b + d/2 ≥ 0 ∨ min a b − d/2 ≤ 0 ∧
max a b − d/2 ≥ 0)
    }
proof −
  note *buffon-def*[*of l d*]
  also {
    have *(λ(x,φ). needle l x φ ∩ {−d/2, d/2} ≠ {}) =*
        (λ(x,φ). let a = x − l / 2 ∗ sin φ; b = x + l / 2 ∗ sin φ

in $-d/2 \geq$ *min a b* $\wedge$ $-d/2 \leq$ *max a b* $\vee$ *min a b* $\leq d/2 \wedge$ *max a b* $\geq d/2$)
by (*auto simp*: *needle-def Let-def closed-segment-eq-real-ivl min-def max-def*)
also have ... =
(λ(*x*,φ). *let a* = *x* − *l* / *2* ∗ *sin* φ; *b* = *x* + *l* / *2* ∗ *sin* φ
in min a b + *d*/*2* ≤ *0* ∧ *max a b* + *d*/*2* ≥ *0* ∨ *min a b* − *d*/*2* ≤ *0* ∧ *max a b* − *d*/*2* ≥ *0*)
by (*auto simp add*: *algebra-simps Let-def*)
finally have (λ(*x*, φ). *return* (*count-space UNIV*) (*needle l x* φ ∩ {− *d*/*2*, *d*/*2*} ≠ {})) =
(λ(*x*,φ). *return* (*count-space UNIV*)
(*let a* = *x* − *l* / *2* ∗ *sin* φ; *b* = *x* + *l* / *2* ∗ *sin* φ
in min a b + *d*/*2* ≤ *0* ∧ *max a b* + *d*/*2* ≥ *0* ∨ *min a b* − *d*/*2* ≤ *0* ∧ *max a b* − *d*/*2* ≥ *0*))
by (*simp add*: *case-prod-unfold fun-eq-iff*)
}
finally show *?thesis* **.**
qed

It is obvious that the problem boils down to determining the measure of the following set:

definition *buffon-set* :: *real* ⇒ *real* ⇒ (*real* × *real*) *set* **where**
buffon-set l d = {(*x*,φ) ∈ {−*d*/*2*..*d*/*2*} × {−*pi*..*pi*}. *abs x* ≥ *d* / *2* − *abs* (*sin* φ) ∗ *l* / *2*}

By using the symmetry inherent in the problem, we can reduce the problem to the following set, which corresponds to one quadrant of the original set:

definition *buffon-set′* :: *real* ⇒ *real* ⇒ (*real* × *real*) *set* **where**
buffon-set′ l d = {(*x*,φ) ∈ {*0*..*d*/*2*} × {*0*..*pi*}. *x* ≥ *d* / *2* − *sin* φ ∗ *l* / *2*}

lemma *closed-buffon-set* [*simp*, *intro*, *measurable*]: *closed* (*buffon-set l d*)
proof −
have *buffon-set l d* = ({−*d*/*2*..*d*/*2*} × {−*pi*..*pi*}) ∩
(λ*z*. *abs* (*fst z*) + *abs* (*sin* (*snd z*)) ∗ *l* / *2* − *d* / *2*) −‘ {*0*..}
(**is** - = *?A*) **unfolding** *buffon-set-def* **by** *auto*
also have *closed* ...
by (*intro closed-Int closed-vimage closed-Times*) (*auto intro*!: *continuous-intros*)
finally show *?thesis* **by** *simp*
qed

lemma *closed-buffon-set′* [*simp*, *intro*, *measurable*]: *closed* (*buffon-set′ l d*)
proof −
have *buffon-set′ l d* = ({*0*..*d*/*2*} × {*0*..*pi*}) ∩
(λ*z*. *fst z* + *sin* (*snd z*) ∗ *l* / *2* − *d* / *2*) −‘ {*0*..}
(**is** - = *?A*) **unfolding** *buffon-set′-def* **by** *auto*
also have *closed* ...
by (*intro closed-Int closed-vimage closed-Times*) (*auto intro*!: *continuous-intros*)
finally show *?thesis* **by** *simp*
qed

lemma *measurable-buffon-set* [*measurable*]: *buffon-set l d* ∈ *sets borel*
  by *measurable*

lemma *measurable-buffon-set′* [*measurable*]: *buffon-set′ l d* ∈ *sets borel*
  by *measurable*

context
  fixes *d l* :: *real*
  assumes *d*: *d* > *0* **and** *l*: *l* > *0*
begin

lemma *buffon-altdef′*:
  buffon l d = *distr* (*uniform-measure lborel* ({−*d*/*2*..*d*/*2*} × {−*pi*..*pi*}))
    (*count-space UNIV*) (λ*z*. *z* ∈ *buffon-set l d*)
proof −
  let *?P* = λ(*x*,φ). *let a* = *x* − *l* / *2* ∗ *sin* φ; *b* = *x* + *l* / *2* ∗ *sin* φ
    in min a b + *d*/*2* ≤ *0* ∧ *max a b* + *d*/*2* ≥ *0* ∨ *min a b* − *d*/*2*
≤ *0* ∧ *max a b* − *d*/*2* ≥ *0*
  have *buffon l d* =
    uniform-measure lborel ({− *d* / *2*..*d* / *2*} × {−*pi*..*pi*}) ≫=
    (λ*z*. *return* (*count-space UNIV*) (*?P z*))
   unfolding *buffon-altdef case-prod-unfold* **by** *simp*
  also have . . . = *uniform-measure lborel* ({− *d* / *2*..*d* / *2*} × {−*pi*..*pi*}) ≫=
    (λ*z*. *return* (*count-space UNIV*) (*z* ∈ *buffon-set l d*))
  proof (*intro bind-cong-AE AE-uniform-measureI AE-I2 impI refl return-measurable*,
goal-cases)
   show (λ*z*. *return* (*count-space UNIV*) (*?P z*))
     ∈ *uniform-measure lborel* ({− *d* / *2*..*d* / *2*} × {− *pi*..*pi*}) $\rightarrow_M$
      subprob-algebra (*count-space UNIV*)
    unfolding *Let-def case-prod-unfold lborel-prod* [*symmetric*] **by** *measurable*
   show (λ*z*. *return* (*count-space UNIV*) (*z* ∈ *buffon-set l d*))
     ∈ *uniform-measure lborel* ({− *d* / *2*..*d* / *2*} × {− *pi*..*pi*}) $\rightarrow_M$
      subprob-algebra (*count-space UNIV*) **by** *simp*

   case (*4 z*)
   hence *?P z* ⟷ *z* ∈ *buffon-set l d*
   proof (*cases snd z* ≥ *0*)
    case *True*
    with *4* **have** *fst z* − *l* / *2* ∗ *sin* (*snd z*) ≤ *fst z* + *l* / *2* ∗ *sin* (*snd z*) **using** *l*
     by (*auto simp*: *sin-ge-zero*)
    moreover from *True* **and** *4* **have** *sin* (*snd z*) ≥ *0* **by** (*auto simp*: *sin-ge-zero*)
    ultimately show *?thesis* **using** *4 True* **unfolding** *buffon-set-def*
     by (*force simp*: *field-simps Let-def min-def max-def case-prod-unfold abs-if*)
   next
    case *False*
    with *4* **have** *fst z* − *l* / *2* ∗ *sin* (*snd z*) ≥ *fst z* + *l* / *2* ∗ *sin* (*snd z*) **using** *l*
     by (*auto simp*: *sin-le-zero′ mult-nonneg-nonpos*)

moreover from *False* **and** *4* **have** $sin\ (snd\ z) \leq 0$ **by** (*auto simp*: *sin-le-zero'*)
ultimately show *?thesis* **using** *4* **and** *False*
unfolding *buffon-set-def* **using** *l d*
by (*force simp*: *field-simps Let-def min-def max-def case-prod-unfold abs-if*)
qed
thus *?case* **by** (*simp only*:)
qed (*simp-all add*: *borel-prod* [*symmetric*])
also have ... $=$ *distr* (*uniform-measure lborel* $(\{-d/2..d/2\} \times \{-pi..pi\})$)
(*count-space UNIV*) ($\lambda z.\ z \in$ *buffon-set l d*)
by (*rule bind-return-distr'*) *simp-all*
finally show *?thesis* **.**
qed

lemma *buffon-prob-aux*:
emeasure (*buffon l d*) {*True*} $=$ *emeasure lborel* (*buffon-set l d*) / *ennreal* $(2 * d * pi)$
proof $-$
have [*measurable*]: $A \times B \in$ *sets borel* **if** $A \in$ *sets borel* $B \in$ *sets borel*
for *A B* :: *real set* **using** *that* **unfolding** *borel-prod* [*symmetric*] **by** *simp*

have *emeasure* (*buffon l d*) {*True*} $=$
emeasure (*uniform-measure lborel* $(\{-\ (d\ /\ 2)..d\ /\ 2\} \times \{-pi..pi\})$)
(($\lambda z.\ z \in$ *buffon-set l d*) $-$‘ {*True*}) (**is** - $=$ *emeasure ?M* -)
by (*simp add*: *buffon-altdef' emeasure-distr*)
also have ($\lambda z.\ z \in$ *buffon-set l d*) $-$‘ {*True*} $=$ *buffon-set l d* **by** *auto*
also have *buffon-set l d* $\subseteq \{-d/2..d/2\} \times \{-pi..pi\}$
using *l d* **by** (*auto simp*: *buffon-set-def*)
hence *emeasure ?M* (*buffon-set l d*) $=$
emeasure lborel (*buffon-set l d*) / *emeasure lborel* $(\{-\ (d\ /\ 2)..d\ /\ 2\} \times \{-pi..pi\})$
by (*subst emeasure-uniform-measure*) (*simp-all add*: *Int-absorb1*)
also have *emeasure lborel* $(\{-\ (d\ /\ 2)..d\ /\ 2\} \times \{-pi..pi\}) =$ *ennreal* $(2 * pi * d)$
using *d* **by** (*simp add*: *lborel-prod* [*symmetric*] *lborel.emeasure-pair-measure-Times*
ennreal-mult algebra-simps)
finally show *?thesis* **by** (*simp add*: *mult-ac*)
qed

lemma *emeasure-buffon-set-conv-buffon-set'*:
emeasure lborel (*buffon-set l d*) $= 4 *$ *emeasure lborel* (*buffon-set' l d*)
proof $-$
have *distr-lborel* [*simp*]: *distr M lborel f* $=$ *distr M borel f* **for** *M* **and** *f* :: *real* $\Rightarrow$ *real*
by (*rule distr-cong*) *simp-all*

define *A* **where** $A =$ *buffon-set' l d*
define *B C D* **where** $B = (\lambda x.\ (-fst\ x,\ snd\ x))\ -‘\ A$ **and** $C = (\lambda x.\ (fst\ x,\ -snd\ x))\ -‘\ A$ **and**
$D = (\lambda x.\ (-fst\ x,\ -snd\ x))\ -‘\ A$

have *meas* [*measurable*]:
$(\lambda x{::}real \times real.\ (-fst\ x,\ snd\ x)) \in borel\text{-}measurable\ borel$
$(\lambda x{::}real \times real.\ (fst\ x,\ -snd\ x)) \in borel\text{-}measurable\ borel$
$(\lambda x{::}real \times real.\ (-fst\ x,\ -snd\ x)) \in borel\text{-}measurable\ borel$
unfolding *borel-prod* [*symmetric*] **by** *measurable*
have *meas'* [*measurable*]: $A \in sets\ borel\ B \in sets\ borel\ C \in sets\ borel\ D \in sets\ borel$
unfolding *A-def B-def C-def D-def* **by** (*rule measurable-buffon-set' measurable-sets-borel meas*)+

have ∗: $buffon\text{-}set\ l\ d = A \cup B \cup C \cup D$
proof (*intro equalityI subsetI*, *goal-cases*)
case (*1 z*)
show *?case*
proof (*cases* $fst\ z \geq 0$; *cases* $snd\ z \geq 0$)
assume $fst\ z \geq 0\ snd\ z \geq 0$
with *1* **have** $z \in A$
by (*auto split*: *prod.splits simp*: *buffon-set-def buffon-set'-def sin-ge-zero A-def*)
thus *?thesis* **by** *blast*
next
assume $\neg(fst\ z \geq 0)\ snd\ z \geq 0$
with *1* **have** $z \in B$
by (*auto split*: *prod.splits simp*: *buffon-set-def buffon-set'-def sin-ge-zero A-def B-def*)
thus *?thesis* **by** *blast*
next
assume $fst\ z \geq 0\ \neg(snd\ z \geq 0)$
with *1* **have** $z \in C$
by (*auto split*: *prod.splits simp*: *buffon-set-def buffon-set'-def sin-le-zero' A-def C-def*)
thus *?thesis* **by** *blast*
next
assume $\neg(fst\ z \geq 0)\ \neg(snd\ z \geq 0)$
with *1* **have** $z \in D$
by (*auto split*: *prod.splits simp*: *buffon-set-def buffon-set'-def sin-le-zero' A-def D-def*)
thus *?thesis* **by** *blast*
qed
qed (*auto simp*: *buffon-set-def buffon-set'-def sin-ge-zero sin-le-zero' A-def B-def C-def D-def*)

have $A \cap B = \{0\} \times (\{0..pi\} \cap \{\varphi.\ sin\ \varphi * l - d \geq 0\})$
using *d l* **by** (*auto simp*: *buffon-set'-def A-def B-def C-def D-def*)
moreover have *emeasure lborel* ... = *0*
unfolding *lborel-prod* [*symmetric*] **by** (*subst lborel.emeasure-pair-measure-Times*) *simp-all*
ultimately have *AB*: $(A \cap B) \in null\text{-}sets\ lborel$
unfolding *lborel-prod* [*symmetric*] **by** (*simp add*: *null-sets-def*)

have $C \cap D = \{0\} \times (\{-pi..0\} \cap \{\varphi.\ -sin\ \varphi * l - d \geq 0\})$
using *d l* **by** (*auto simp*: *buffon-set′-def A-def B-def C-def D-def*)
moreover have *emeasure lborel* ... = *0*
unfolding *lborel-prod* [*symmetric*] **by** (*subst lborel.emeasure-pair-measure-Times*) *simp-all*
ultimately have *CD*: $(C \cap D) \in$ *null-sets lborel*
unfolding *lborel-prod* [*symmetric*] **by** (*simp add*: *null-sets-def*)

have $A \cap D = \{\}$ $B \cap C = \{\}$ **using** *d l*
by (*auto simp*: *buffon-set′-def A-def D-def B-def C-def*)
moreover have $A \cap C = \{(d/2,\ 0)\}$ $B \cap D = \{(-d/2,\ 0)\}$
using *d l* **by** (*auto simp*: *case-prod-unfold buffon-set′-def A-def B-def C-def D-def*)
ultimately have *AD*: $A \cap D \in$ *null-sets lborel* **and** *BC*: $B \cap C \in$ *null-sets lborel* **and**
AC: $A \cap C \in$ *null-sets lborel* **and** *BD*: $B \cap D \in$ *null-sets lborel* **by** *auto*

note ∗
also have *emeasure lborel* $(A \cup B \cup C \cup D)$ = *emeasure lborel* $(A \cup B \cup C)$ + *emeasure lborel D*
using *AB AC AD BC BD CD* **by** (*intro emeasure-Un′*) (*auto simp*: *Int-Un-distrib2*)
also have *emeasure lborel* $(A \cup B \cup C)$ = *emeasure lborel* $(A \cup B)$ + *emeasure lborel C*
using *AB AC BC* **using** *AB AC AD BC BD CD* **by** (*intro emeasure-Un′*) (*auto simp*: *Int-Un-distrib2*)
also have *emeasure lborel* $(A \cup B)$ = *emeasure lborel A* + *emeasure lborel B*
using *AB* **using** *AB AC AD BC BD CD* **by** (*intro emeasure-Un′*) (*auto simp*: *Int-Un-distrib2*)
also have *emeasure lborel B* = *emeasure* (*distr lborel lborel* $(\lambda(x,y).\ (-x,\ y))$) *A*
(**is** - = *emeasure ?M* -) **unfolding** *B-def*
by (*subst emeasure-distr*) (*simp-all add*: *case-prod-unfold*)
also have *?M* = *lborel* **unfolding** *lborel-prod* [*symmetric*]
by (*subst pair-measure-distr* [*symmetric*]) (*simp-all add*: *sigma-finite-lborel lborel-distr-uminus*)
also have *emeasure lborel C* = *emeasure* (*distr lborel lborel* $(\lambda(x,y).\ (x,\ -y))$) *A*
(**is** - = *emeasure ?M* -) **unfolding** *C-def*
by (*subst emeasure-distr*) (*simp-all add*: *case-prod-unfold*)
also have *?M* = *lborel* **unfolding** *lborel-prod* [*symmetric*]
by (*subst pair-measure-distr* [*symmetric*]) (*simp-all add*: *sigma-finite-lborel lborel-distr-uminus*)
also have *emeasure lborel D* = *emeasure* (*distr lborel lborel* $(\lambda(x,y).\ (-x,\ -y))$) *A*
(**is** - = *emeasure ?M* -) **unfolding** *D-def*
by (*subst emeasure-distr*) (*simp-all add*: *case-prod-unfold*)
also have *?M* = *lborel* **unfolding** *lborel-prod* [*symmetric*]
by (*subst pair-measure-distr* [*symmetric*]) (*simp-all add*: *sigma-finite-lborel lborel-distr-uminus*)
finally have *emeasure lborel* (*buffon-set l d*) =

of-nat (*Suc* (*Suc* (*Suc* (*Suc* *0*)))) ∗ *emeasure lborel A*
unfolding *of-nat-Suc ring-distribs* **by** *simp*
also have *of-nat* (*Suc* (*Suc* (*Suc* (*Suc* *0*)))) = (*4* :: *ennreal*) **by** *simp*
finally show *?thesis* **unfolding** *A-def* **.**
qed

It only remains now to compute the measure of *buffon-set′*. We first reduce this problem to a relatively simple integral:

lemma *emeasure-buffon-set′*:
emeasure lborel (*buffon-set′ l d*) =
ennreal (*integral* {*0*..*pi*} (λ*x*. *min* (*d* / *2*) (*sin x* ∗ *l* / *2*)))
(**is** *emeasure lborel ?A* = -)
proof −
have *emeasure lborel ?A* = *nn-integral lborel* (λ*x*. *indicator ?A x*)
by (*intro nn-integral-indicator* [*symmetric*]) *simp-all*
also have (*lborel* :: (*real* × *real*) *measure*) = *lborel* $\bigotimes_M$ *lborel*
by (*simp only*: *lborel-prod*)
also have *nn-integral* . . . (*indicator ?A*) = (∫⁺φ. ∫⁺*x*. *indicator ?A* (*x*, φ) ∂*lborel* ∂*lborel*)
by (*subst lborel-pair.nn-integral-snd* [*symmetric*]) (*simp-all add*: *lborel-prod borel-prod*)
also have . . . = (∫⁺φ. ∫⁺*x*. *indicator* {*0*..*pi*} φ ∗ *indicator* {*max 0* (*d*/*2* − *sin* φ ∗ *l* / *2*) .. *d*/*2*} *x* ∂*lborel* ∂*lborel*)
using *d l* **by** (*intro nn-integral-cong*) (*auto simp*: *indicator-def field-simps buffon-set′-def*)
also have . . . = ∫⁺ φ. *indicator* {*0*..*pi*} φ ∗ *emeasure lborel* {*max 0* (*d* / *2* − *sin* φ ∗ *l* / *2*)..*d* / *2*} ∂*lborel*
by (*subst nn-integral-cmult*) *simp-all*
also have . . . = ∫⁺ φ. *ennreal* (*indicator* {*0*..*pi*} φ ∗ *min* (*d* / *2*) (*sin* φ ∗ *l* / *2*)) ∂*lborel*
(**is** - = *?I*) **using** *d l* **by** (*intro nn-integral-cong*) (*auto simp*: *indicator-def sin-ge-zero max-def min-def*)
also have *integrable lborel* (λφ. (*d* / *2*) ∗ *indicator* {*0*..*pi*} φ) **by** *simp*
hence *int*: *integrable lborel* (λφ. *indicator* {*0*..*pi*} φ ∗ *min* (*d* / *2*) (*sin* φ ∗ *l* / *2*))
by (*rule Bochner-Integration.integrable-bound*)
(*insert l d*, *auto intro*!: *AE-I2 simp*: *indicator-def min-def sin-ge-zero*)
hence *?I* = *set-lebesgue-integral lborel* {*0*..*pi*} (λφ. *min* (*d* / *2*) (*sin* φ ∗ *l* / *2*))
by (*subst nn-integral-eq-integral*, *assumption*)
(*insert d l*, *auto intro*!: *AE-I2 simp*: *sin-ge-zero min-def indicator-def set-lebesgue-integral-def*)
also have . . . = *ennreal* (*integral* {*0*..*pi*} (λ*x*. *min* (*d* / *2*) (*sin x* ∗ *l* / *2*)))
(**is** - = *ennreal ?I*) **using** *int* **by** (*subst set-borel-integral-eq-integral*) (*simp-all add*: *set-integrable-def*)
finally show *?thesis* **by** (*simp add*: *lborel-prod*)
qed

We now have to distinguish two cases: The first and easier one is that where the length of the needle, *l*, is less than or equal to the strip width, *d*:

context

assumes *l-le-d*: $l \leq d$
begin

lemma *emeasure-buffon-set′-short*: *emeasure lborel* (*buffon-set′ l d*) = *ennreal l*
proof −
  have *emeasure lborel* (*buffon-set′ l d*) =
        ennreal (*integral* {*0..pi*} (λ*x*. *min* (*d* / *2*) (*sin x* ∗ *l* / *2*))) (**is** - = *ennreal ?I*)
    by (*rule emeasure-buffon-set′*)
  also have ∗: *sin* φ ∗ *l* ≤ *d* **if** φ ≥ *0* φ ≤ *pi* **for** φ
     using *mult-mono*[*OF l-le-d sin-le-one - sin-ge-zero*] *that d* **by** (*simp add*: *algebra-simps*)
  have *?I* = *integral* {*0..pi*} (λ*x*. (*l* / *2*) ∗ *sin x*)
    using *l d l-le-d*
    by (*intro integral-cong*) (*auto dest*: ∗ *simp*: *min-def sin-ge-zero*)
  also have . . . = *l* / *2* ∗ *integral* {*0..pi*} *sin* **by** *simp*
  also have (*sin has-integral* (−*cos pi* − (− *cos 0*))) {*0..pi*}
    by (*intro fundamental-theorem-of-calculus*)
     (*auto intro*!: *derivative-eq-intros simp*: *has-field-derivative-iff-has-vector-derivative* [*symmetric*])
  hence *integral* {*0..pi*} *sin* = −*cos pi* − (−*cos 0*)
    by (*simp add*: *has-integral-iff*)
  finally show *?thesis* **by** (*simp add*: *lborel-prod*)
qed

lemma *emeasure-buffon-set-short*: *emeasure lborel* (*buffon-set l d*) = *4* ∗ *ennreal l*
   by (*simp add*: *emeasure-buffon-set-conv-buffon-set′ emeasure-buffon-set′-short l-le-d*)

theorem *buffon-short*: *emeasure* (*buffon l d*) {*True*} = *ennreal* (*2* ∗ *l* / (*d* ∗ *pi*))
proof −
  have *emeasure* (*buffon l d*) {*True*} = *ennreal* (*4* ∗ *l*) / *ennreal* (*2* ∗ *d* ∗ *pi*)
     using *d l* **by** (*subst buffon-prob-aux*) (*simp add*: *emeasure-buffon-set-short ennreal-mult*)
  also have . . . = *ennreal* (*4* ∗ *l* / (*2* ∗ *d* ∗ *pi*))
    using *d l* **by** (*subst divide-ennreal*) *simp-all*
  also have *4* ∗ *l* / (*2* ∗ *d* ∗ *pi*) = *2* ∗ *l* / (*d* ∗ *pi*) **by** *simp*
  finally show *?thesis* **.**
qed

end

The other case where the needle is at least as long as the strip width is more complicated:

context
  assumes *l-ge-d*: $l \geq d$
begin

lemma *emeasure-buffon-set′-long*:

```
  emeasure lborel (buffon-set′ l d) =
     ennreal (l ∗ (1 − sqrt (1 − (d / l)^2)) + arccos (d / l) ∗ d)
proof −
  define φ′ where φ′ = arcsin (d / l)
  have φ′-nonneg: φ′ ≥ 0 unfolding φ′-def using d l l-ge-d arcsin-le-mono[of 0
d/l]
    by (simp add: φ′-def)
  have φ′-le: φ′ ≤ pi / 2 unfolding φ′-def using arcsin-bounded[of d/l] d l l-ge-d
    by (simp add: field-simps)
  have ge-phi′: sin φ ≥ d / l if φ ≥ φ′ φ ≤ pi / 2 for φ
     using arcsin-le-iff[of d / l φ] d l-ge-d that φ′-nonneg by (auto simp: φ′-def
field-simps)
  have le-phi′: sin φ ≤ d / l if φ ≤ φ′ φ ≥ 0 for φ
   using le-arcsin-iff[of d / l φ] d l-ge-d that φ′-le by (auto simp: φ′-def field-simps)

  let ?f = (λx. min (d / 2) (sin x ∗ l / 2))
  have emeasure lborel (buffon-set′ l d) = ennreal (integral {0..pi} ?f) (is - =
ennreal ?I)
    by (rule emeasure-buffon-set′)
  also have ?I = integral {0..pi/2} ?f + integral {pi/2..pi} ?f
    by (rule Henstock-Kurzweil-Integration.integral-combine [symmetric]) (auto in-
tro!: integrable-continuous-real continuous-intros)
  also have integral {pi/2..pi} ?f = integral {−pi/2..0} (?f ∘ (λφ. φ + pi))
    by (subst integral-shift) (auto intro!: continuous-intros)
  also have . . . = integral {−(pi/2)..−0} (λx. min (d / 2) (sin (−x) ∗ l / 2))
by (simp add: o-def)
  also have . . . = integral {0..pi/2} ?f (is - = ?I) by (subst Henstock-Kurzweil-Integration.integral-reflect-real
simp-all
  also have . . . + . . . = 2 ∗ . . . by simp
  also have ?I = integral {0..φ′} ?f + integral {φ′..pi/2} ?f
    using l d l-ge-d φ′-nonneg φ′-le
     by (intro Henstock-Kurzweil-Integration.integral-combine [symmetric]) (auto
intro!: integrable-continuous-real continuous-intros)
  also have integral {0..φ′} ?f = integral {0..φ′} (λx. l / 2 ∗ sin x)
    using l by (intro integral-cong) (auto simp: min-def field-simps dest: le-phi′)
  also have ((λx. l / 2 ∗ sin x) has-integral (− (l / 2 ∗ cos φ′) − (− (l / 2 ∗ cos
0)))) {0..φ′}
    using φ′-nonneg
    by (intro fundamental-theorem-of-calculus)
       (auto simp: has-field-derivative-iff-has-vector-derivative [symmetric] intro!:
derivative-eq-intros)
  hence integral {0..φ′} (λx. l / 2 ∗ sin x) = (1 − cos φ′) ∗ l / 2
    by (simp add: has-integral-iff algebra-simps)
  also have integral {φ′..pi/2} ?f = integral {φ′..pi/2} (λ-. d / 2)
    using l by (intro integral-cong) (auto simp: min-def field-simps dest: ge-phi′)
  also have . . . = arccos (d / l) ∗ d / 2 using φ′-le d l l-ge-d
    by (subst arccos-arcsin-eq) (auto simp: field-simps φ′-def)
  also have cos φ′ = sqrt (1 − (d / l)ˆ2)
   unfolding φ′-def by (rule cos-arcsin) (insert d l l-ge-d, auto simp: field-simps)
```

also have *2 ∗ ((1 − sqrt (1 − (d / l)*^{2}*)) ∗ l / 2 + arccos (d / l) ∗ d / 2) =*
l ∗ (1 − sqrt (1 − (d / l)^2*)) + arccos (d / l) ∗ d*
using *d l* **by** (*simp add*: *field-simps*)
finally show *?thesis* **.**
qed

lemma *emeasure-buffon-set-long*: *emeasure lborel* (*buffon-set l d*) =
4 ∗ ennreal (*l ∗* (*1 − sqrt* (*1 −* (*d / l*)^2)) + *arccos* (*d / l*) *∗ d*)
by (*simp add*: *emeasure-buffon-set-conv-buffon-set′ emeasure-buffon-set′-long l-ge-d*)

theorem *buffon-long*:
emeasure (*buffon l d*) {*True*} =
ennreal (*2 / pi ∗* ((*l / d*) *− sqrt* ((*l / d*)^2 *− 1*) + *arccos* (*d / l*)))
proof −
have ∗: *l ∗ sqrt* ((*l*^{2} *− d*^{2}) / *l*^{2}) + *0 ≤ l + d ∗ arccos* (*d / l*)
using *d l-ge-d* **by** (*intro add-mono mult-nonneg-nonneg arccos-lbound*) (*auto simp*: *field-simps*)
have *emeasure* (*buffon l d*) {*True*} =
ennreal (*4 ∗* (*l − l ∗ sqrt* (*1 −* (*d / l*)^2) + *arccos* (*d / l*) *∗ d*)) / *ennreal* (*2 ∗ d ∗ pi*)
using *d l l-ge-d ∗* **unfolding** *buffon-prob-aux emeasure-buffon-set-long ennreal-numeral* [*symmetric*]
by (*subst ennreal-mult* [*symmetric*])
(*auto intro*!: *add-nonneg-nonneg mult-nonneg-nonneg simp*: *field-simps*)
also have ... = *ennreal* ((*4 ∗* (*l − l ∗ sqrt* (*1 −* (*d / l*)^2) + *arccos* (*d / l*) *∗ d*)) / (*2 ∗ d ∗ pi*))
using *d l ∗* **by** (*subst divide-ennreal*) (*auto simp*: *field-simps*)
also have (*4 ∗* (*l − l ∗ sqrt* (*1 −* (*d / l*)^2) + *arccos* (*d / l*) *∗ d*)) / (*2 ∗ d ∗ pi*) =
2 / pi ∗ (*l / d − l / d ∗ sqrt* ((*d / l*)ˆ*2 ∗* ((*l / d*)ˆ*2 − 1*)) + *arccos* (*d / l*))
using *d l* **by** (*simp add*: *field-simps*)
also have *l / d ∗ sqrt* ((*d / l*)ˆ*2 ∗* ((*l / d*)ˆ*2 − 1*)) = *sqrt* ((*l / d*) ˆ *2 − 1*)
using *d l l-ge-d* **unfolding** *real-sqrt-mult real-sqrt-abs* **by** *simp*
finally show *?thesis* **.**
qed

end
end

end

References

[1] J. F. Ramaley. Buffon's Noodle Problem. *The American Mathematical Monthly*, 76(8):916–918, 1969.

[2] E. W. Weisstein. MathWorld – Buffon's Needle Problem.

http://mathworld.wolfram.com/BuffonsNeedleProblem.html.