Principal Component Analysis of Random Walk, Integrated Random Walk

```
(*Determine the order of an integral eigenproblem*)
Clear[IntegralOrder];
IntegralOrder[Integrate[e_, lims__]] := Length[{lims}] + IntegralOrder[e];
IntegralOrder[head_[args___]] := Max @@ (IntegralOrder /@ {args});
IntegralOrder[_Symbol | (_?NumberQ)] = 0;
(*Reduce an integral eigenproblem to a finite-
 dimensional subproblem and a univariate equation for the eigenvalues*)
Clear[ReduceIntegralEquation];
ReduceIntegralEquation[eqn_, fun_, var_] :=
  Module[{m, eqn2, w, vars, A, eigvaleqn, rules, ev},
    (*Find the ODE corresponding to this integral
    equation and obtain its general solution*)
    m = IntegralOrder[eqn];
    eqn2 = D[eqn, {t, m}];
    w = fun /. DSolve[eqn2, fun, var][[1]];
    (*Substitute back into the integral equation and eliminate t by
     equating coefficients to obtain an m-dimensional linear system*)
    eqn2 = (eqn /. {Head[fun][x_] :> (w /. {var -> x})}) // FullSimplify;
    (*Equate coefficients
    (and verify that the expressions are indeed polynomial in var)*)
    vars = Table[C[i], {i, 1, m}];
    eqn2 = FullSimplify[
      (# == 0) & /@ CoefficientList[eqn2[[1]] - eqn2[[2]], var] /. {t :> Assert[False]}];
    A = CoefficientArrays[eqn2, vars][[2]] // Normal;
    (*A must be rank-deficient to have a nonzero solution.*)
    eigvaleqn = (Det[A] == 0) // FullSimplify;
    (*Eliminate oscillatory terms*)
    rules = {
        Sin[ϕ] -> s Sqrt[1 - Cos[ϕ]^2],
        Solve[eigvaleqn, Cos[ϕ]][[1, 1]]
    };
    Assuming[s^2 == 1,
        eqn2 = FullSimplify[eqn2 //. rules];
        (*Find null space*)

    A = CoefficientArrays[eqn2, vars][[2]] // Normal // ExpToTrig // FullSimplify;
        ev = NullSpace[A, ZeroTest -> (PossibleZeroQ[FullSimplify[#]] &)][[1]];
        ev = FullSimplify[ev];
```

```
    ];
    w = w /. Thread[vars → ev];
    (*Return eigenfunction and eigenvalue condition*)
    {w, eigvaleqn}
];
(*Simplify expressions containing a variable s representing ±1*)
ForBothSigns[expr_] := Module[{plus, minus}, Assuming[s^2 == 1,
    plus = FullSimplify[TrigToExp[expr /. {s → +1}]];
    minus = FullSimplify[TrigToExp[expr /. {s → -1}]];
    (plus + minus) / 2 + s (plus - minus) / 2 // FullSimplify]
];
```

Continuous-time random walk

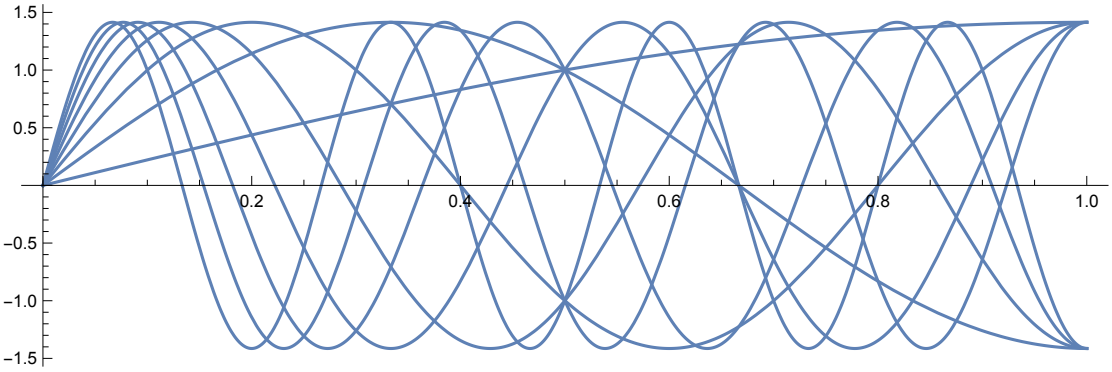
Principal component analysis of the continuous-time random walk.

```
In[•]:= eqn = (ω^-2) w[t] == Integrate[w[t2], {t1, 0, t}, {t2, t1, T}];
(*Eigenproblem for covariance functional of random walk*)
{eigfun, eigvaleqn} =
  Assuming[T > 0 && ϕ > 0, ReduceIntegralEquation[eqn /. {ω → ϕ / T}, w[t], t]];
eigfun = eigfun /. {ϕ → ω T};
<|eigenvalues → (eigvaleqn && λ == ω^-2) /. {ϕ → ω T}, eigenfunctions → eigfun|>
eigfunNorm = Integrate[eigfun^2, {t, 0, T}] // FullSimplify;
neigval = NSolve[eigvaleqn, ϕ ∈ Interval[{0, 25}], Reals];
"First " <> ToString[Length[neigval]] <> " eigenfunctions:"
Plot[(eigfun / Sqrt[eigfunNorm]) /. ({T → 1, ω → ϕ} /. neigval),
 {t, 0, 1}, AspectRatio → 1 / 3, ImageSize → Large]
```

```
Out[•]= ⟨| eigenvalues → Cos[T ω] == 0 && λ == 1/ω^2, eigenfunctions → Sin[t ω] |⟩
```

```
Out[•]= First 8 eigenfunctions:
```

Out[•]=



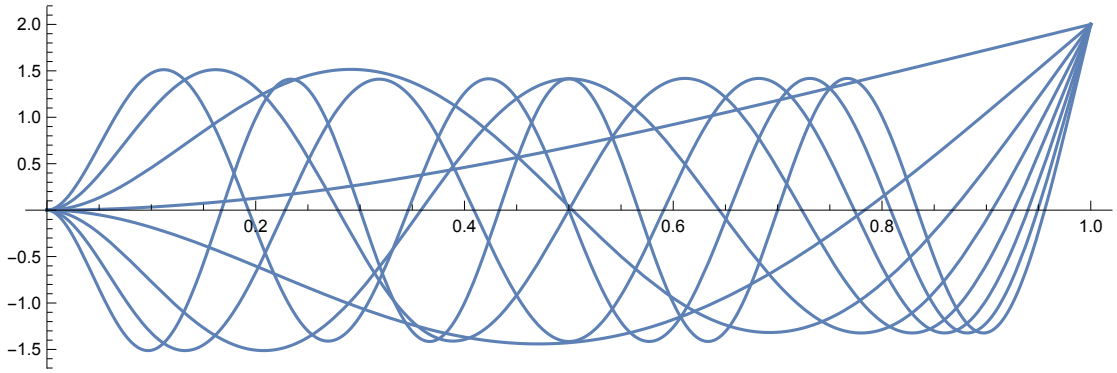
Continuous-time integrated random walk

The integrated random walk is less straightforward, but we can apply the same techniques to obtain the eigenfunctions in closed form, and we can obtain a univariate trigonometric expression whose roots give us the eigenvalues.

```
In[•]:= eqn = (ω^-4) w[t] == Integrate[w[t4], {t1, 0, t}, {t2, 0, t1}, {t3, t2, T}, {t4, t3, T}];
(*Eigenproblem for covariance functional of integrated random walk*)
{eigfun, eigvaleqn} = Assuming[T > 0 && ϕ > 0 && Exp[ϕ] > 0 && Tanh[ϕ] > 0,
    ReduceIntegralEquation[eqn /. {ω → ϕ / T}, w[t], t]];
eigvaleqn = Assuming[Exp[ϕ] > 0, FullSimplify[ExpToTrig[eigvaleqn]]];
eigfun = eigfun /. {ϕ → ω T};
<|eigenvalues → (eigvaleqn && λ == ω^-4) /. {ϕ → ω T}, eigenfunctions → eigfun|>
rules = {Sin[ϕ] → s Sqrt[1 - Cos[ϕ]^2],
    ExpToTrig[Solve[eigvaleqn, Cos[ϕ]][[1, 1]]] // FullSimplify} /. {ϕ → ω T};
eigfunNorm = Assuming[s^2 == 1, Integrate[eigfun^2, {t, 0, T}] // FullSimplify];
eigfunNorm = Assuming[Exp[ω T] > 1 && Tanh[ω T] > 0 && s^2 == 1,
    FullSimplify[TrigExpand[eigfunNorm] //. rules]];
eigfunNorm = ForBothSigns[eigfunNorm];
neigval = NSolve[eigvaleqn, ϕ ∈ Interval[{0, 25}], Reals];
"First " <> ToString[Length[neigval]] <> " eigenfunctions:"
Plot[(eigfun / Sqrt[eigfunNorm]) /. ({T → 1, ω → ϕ, s → Sign[Sin[ϕ]]} /. neigval),
 {t, 0, 1}, AspectRatio → 1 / 3, ImageSize → Large]
```

$$\text{Out[•]=}\ \left\langle\left|\text{eigenvalues} \to 1+\text{Cos}[T\,\omega]\,\text{Cosh}[T\,\omega]==0\ \&\&\ \lambda==\frac{1}{\omega^4},\ \text{eigenfunctions}\to e^{t\,\omega}+\frac{e^{-t\,\omega+T\,\omega}\left(e^{T\,\omega}-s\right)}{-1+e^{T\,\omega}\,s}-\frac{\left(-1+e^{2\,T\,\omega}\right)\text{Cos}[t\,\omega]}{-1+e^{T\,\omega}\,s}+\frac{\left(1+e^{2\,T\,\omega}-2\,e^{T\,\omega}\,s\right)\text{Sin}[t\,\omega]}{-1+e^{T\,\omega}\,s}\right|\right\rangle$$

```
Out[•]= First 8 eigenfunctions:
```

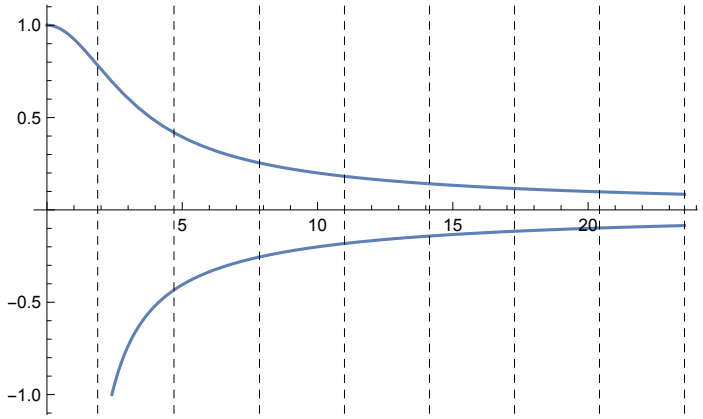


Definite integrals of eigenfunctions

We will need an expression for the integral over [0,T] of each eigenfunction.

In[•]:=
```
eigintNumerator = Assuming[s^2 == 1 && Tanh[ω T] > 0,
    Simplify[Integrate[ExpToTrig[eigfun] // ExpandAll, {t, 0, T}] //. rules]];
eigintNumerator = ForBothSigns[eigintNumerator];
eigintDenominator = √eigfunNorm;
eigint = eigintNumerator / eigintDenominator // FullSimplify;
eigint = Assuming[T > 0 && Exp[ω T] > 1, ForBothSigns[eigint]];
Show[
Plot[(eigint / √T /. {ω → ϕ / T}) /. {s → {-1, +1}}, {ϕ, 0, Last[neigval][[1, 2, 1]]}],
ListLinePlot[Table[Table[{neigval[[i, 1, 2, 1]], 1.6 j}, {j, -1, 1}],
    {i, 1, Length[neigval]}], PlotStyle → Directive[Black, Thin, Dashed]]
]
eigint
```

Out[•]=



Out[•]=
$$\frac{2\left(-1+s\,\text{Cosh}[T\,\omega]\right)\text{Csch}[T\,\omega]}{\sqrt{T}\,\omega}$$

Bounding Probability of Coherence < 1/2

Coherence function

Coherence is a scalar measure of the alignment of a number of phasors for the purpose of coherent detection. In this section, we consider only the discrete-time case.

We suppose that the true phase history of the received signal is a linear combination of eigenfunctions with independent Gaussian coefficients. The receiver's phase model is a truncation of this true phase history to the first m principal components. The phase error for each sample is therefore given by a sum over all the *truncated* principal components, i.e. those indexed by m+1 through n.

In[•]:=
```
f[X_] := (1 / n) Abs[Sum[Exp[ⅈ Sum[X[[j]] Sqrt[λ[[j]]] V[[i, j]], {j, m + 1, n}]], {i, 1, n}]];
```

We seek an upper bound on the probability that f[X] drops below 1/2, signifying loss of coherent integration. We will establish this by showing that f[X] is a sub-Gaussian random variable with parameter

corresponding to the first truncated principal component, and obtaining an approximation for the mean of f[X]. These two pieces of information will allow us to apply a standard sub-Gaussian tail bound.

In[•]:=

```
sum = Inactive[Sum];
reindexRules[i_Symbol] := {i → Unique@i};
reindexRules[{i_Symbol, ___}] := reindexRules[i];
reindexRules[first_, rest__] :=
  {Sequence @@ reindexRules[first], Sequence @@ reindexRules[rest]};
reindex = s : sum[_, lims__] :> (s /. reindexRules[lims]);
inactivate = {sum → sum, Sum → sum};
(*protect Inactive[Sum] from getting double-inactivated*)
unpower = s_sum^p_Integer :> Times @@ Table[Replace[s, reindex, All], {p}];
expand = {
    sum[e1_, lims1__] × sum[e2_, lims2__] :>
    With[{r1 = reindexRules[lims1], r2 = reindexRules[lims2]},
     sum[(e1 /. r1) (e2 /. r2), lims1 /. r1, lims2 /. r2]],
    sum[sum[e_, lims1__], lims2__] :> sum[e, lims1, lims2],
    sum[s_Plus, lims__] :> (sum[#, lims] & /@ s),
    sum[Times[head___, ((n_?NumberQ) | n_Symbol), tail___], lims__] :>
    n sum[Times[head, tail], lims]
};
distribute = Integrate[sum[e_, s__], i__] :> sum[Integrate[e, i], s];
niceindex[expr_] :=
  expr /. (Thread[# → Take[{i, j, k, l, o, p, q, r, s, t, u, v}, Length[#]]] &@
     DeleteDuplicates[
      Cases[expr, s_Symbol /; SymbolName[s] ~StringMatchQ~ "*$*" :> s, -1]]);
doitall[expr_] := Replace[expr /. inactivate, reindex, All] /. unpower //. expand //
   niceindex;
LimVars[i_Symbol] := {i};
LimVars[{i_Symbol, ___}] := {i};
LimVars[first_, rest__] :=
  {Sequence @@ LimVars[first], Sequence @@ LimVars[rest]};
IndependentQ[expr_?NumberQ, _] = True;
IndependentQ[i_Symbol, {}] = True;
IndependentQ[i_Symbol, {{j_Symbol, ___}, tail___}] :=
  (i =!= j) && IndependentQ[i, {tail}];
IndependentQ[head_[args___], lims_] := And @@ ((IndependentQ[#, lims] &) /@ {args});
eval = {
    (* Summation of dimension zero: evaluate. *)
    sum[e_] → e,
    (* Trivial summand: eliminate one dimension. *)
```

```
    sum[x_?NumberQ, {i_, imin_, imax_}, tail___] :> (imax - imin + 1) x sum[1, tail],
     (* Kronecker delta: eliminate one dimension. *)

    sum[Times[head___, (KroneckerDelta[i_, j_] | KroneckerDelta[j_, i_]), tail___],
      lims1___, {i_, imin_, imax_}, lims2___, {j_, jmin_, jmax_}, lims3___] :>
     sum[Times[head, tail] /. {j → i}, lims1, lims2, lims3,
      {i, Max[imin, jmin], Min[imax, jmax]}],
     (* One factor in summand is independent of one
     index: hoist it out of the summation. *)
     sum[Times[head___, e_, tail___], limhead___, {i_, imin_, imax_}, limtail___] /;
      IndependentQ[e, {{i}}] :>
     sum[e sum[Times[head, tail], {i, imin, imax}], limhead, limtail]
};
(*Abs[Sum[Times]≤Sum[Times[Abs]]]*)
absSumUpperBound =
  {Abs[Sum[Times[args_], lims__]] → sum[Times @@ (Abs /@ {args}), lims]};
```

Upper bound for Lipschitz constant of coherence function

We need to bound the gradient magnitude of the coherence function.

```
Clear[n, m];
vars = Inactive[Table][X[[k]], {k, m + 1, n}];
gradient = Activate[D[f[X], {vars}]];
magnitudeBound = Abs[gradient[[1]] /. inactivate;
magnitudeBound = Activate[magnitudeBound /.
    {sum[KroneckerDelta[j_, k] e_, {j_, 1 + m, n}] :> ReplaceAll[j → k][e]}];
magnitudeBound = magnitudeBound // ComplexExpand // FullSimplify;
magnitudeBound = magnitudeBound /. absSumUpperBound;
eigvecKnowledge = {
    Im[sum[x_, lims__]] → sum[Im[x], lims],
    Im[X[_] Sqrt[λ[_]] V[_, _]] → 0,
    Abs[Sqrt[λ[k_]] e_] :> Sqrt[λ[k]] Abs[e], (*λ≥0*)
    sum[Abs[V[i_, k_]], {i_, 1, n}] → Sqrt[n], (*(1-norm)≤√n×(2-norm)*)
    λ[i_] :> σ^2 (π (i - 1/2)) ^-4
};
magnitudeBound = Assuming[n > 0,
   (doitall[magnitudeBound] //. (eval~Join~eigvecKnowledge)) // FullSimplify];
LipschitzUpperBound = FullSimplify[magnitudeBound /. {k → m + 1}]
```

Out[•]=
$$\frac{4\sqrt{\frac{n\,\sigma^2}{(1+2\,m)^4}}}{n\,\pi^2}$$

Approximation for mean coherence

```
fMeanSeries = f[X];
abs1p[z_] = ((Series[ComplexExpand[Abs[1 + a x + ⅈ a y]], {a, 0, 2}] // Normal) /.
    {a → 1, x → Re[z], y → Im[z]});(*accurate to order O[|z|^3]*)
fMeanSeries = (fMeanSeries /. {Abs[e_] → abs1p[e - 1]}) /. inactivate;
fMeanSeries = (fMeanSeries /.
     {g_[sum[e_, {i_, 1, n}]] → sum[g[ExpToTrig[e]], {i, 1, n}]} // FullSimplify);
fMeanSeries = (fMeanSeries /. {g_[Exp[ⅈ e_]] → g[ExpToTrig[Exp[ⅈ e]]]} //
     FullSimplify) /. {Im[x_] → 0, Re[x_] → x};
fMeanSeries = (fMeanSeries /. {Sin[x_] → x, Cos[x_] → 1 - x^2/2}) // Activate //
   FullSimplify;
fMeanSeries = doitall[fMeanSeries];
fMeanSeries = (fMeanSeries /. {X⟦i_⟧ X⟦j_⟧ :→ KroneckerDelta[i, j]});
eigvecKnowledge = {
    sum[V⟦j_, i_⟧^2, {j_, 1, n}] → 1,
    sum[V⟦k_, i_⟧, {k_, 1, n}] :→ 2 (-1)^i / (π (i - 1/2)),
    λ⟦i_⟧ :→ σ^2 (π (i - 1/2))^-4
};
fMeanSeries = (Replace[#, reindex, All] // niceindex) & /@
   ExpandAll[fMeanSeries //. (eval ~Join~ eigvecKnowledge)];
fMeanSeries = fMeanSeries // FullSimplify // Activate // ExpandAll;
fMeanSeries = Assuming[n ∈ Integers && m ∈ Integers, FullSimplify[fMeanSeries]];
fMeanApprox = 1 + Series[fMeanSeries - 1, {n, ∞, 4}, {m, ∞, 4}] // Normal // FullSimplify
```

Out[•]= $1 + \frac{\left(m^3 - n^3\right) \sigma^2}{6\, m^3\, n^4\, \pi^4}$

Tail Bound

We now have the pieces to evaluate a concentration bound for the sub-Gaussian random variable f[X], where X is a vector of i.i.d. standard Gaussian variables.

In the case where fMeanApprox > 1/2, the probability that the coherence function is less than 1/2 is at most

In[•]:=
```
incoherenceProbabilityBound =
  (Exp[-(fMeanApprox - 1/2)^2/2/LipschitzUpperBound^2] /. {n → Fs T}) //
   FullSimplify
```

Out[•]= $e^{-\frac{Fs\,(\pi + 2\,m\,\pi)^4\,T\left(3 + \frac{\left(m^3 - Fs^3\,T^3\right)\sigma^2}{Fs^4\,m^3\,\pi^4\,T^4}\right)^2}{1152\,\sigma^2}}$

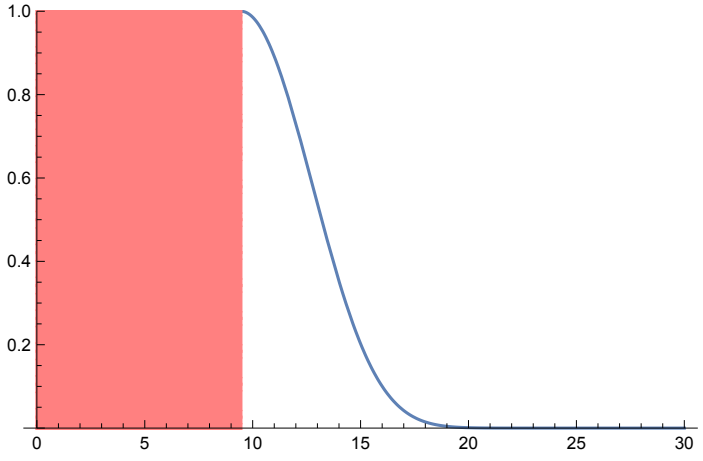
Physical units

The integrated random walk process evaluated at time t may be scaled to units of radians by incorporating a factor σ.

In[•]:=
```
ConcreteProblem[] := (
    λ = UnitConvert[Quantity["SpeedOfLight"] / Quantity[75., "GigaHertz"]];
    vRMS[Quantity[1, "Seconds"]] = Quantity[1, "Meters" / "Seconds"];
    T = Quantity[10, "Seconds"]; (*experiment duration*)
    Fs = UnitConvert[Quantity[1000, "Hertz"]];
   (*sampling frequency (i.e. chirp rate)*)
    n = Fs T; (*number of chirps*)
    σ0 = 2 π / λ Sqrt[1 / Quantity[1, "Seconds"]] vRMS[Quantity[1, "Seconds"]];
    σ = σ0 T^(3 / 2);
    );
AbstractProblem[] := (
    Clear[λ, vRMS, T, Fs, σ, σ0, n]
);
```

In[•]:=
```
ConcreteProblem[];
Show[
 Plot[incoherenceProbabilityBound, {m, 1, 30}, PlotRange → {0, 1}],
 RegionPlot[Evaluate[FullSimplify[fMeanApprox < 1 / 2]],
  {m, -1, 30}, {y, -1, 2}, PlotStyle
   → Directive[Opacity[1.0], RGBColor[1.0, 0.5, 0.5]], BoundaryStyle → None]
]
AbstractProblem[];
```

Out[•]=



In[•]:=
```
Clear[λ, vRMS, T, Fs, σ, σ0, n]
```

In[•]:=

In[•]:=
```
degreeRelation = Factor /@ (Times[#, 1152 σ^2 T^7 Fs^7 π^4] & /@ FullSimplify[
      (Log /@ (incoherenceProbabilityBound == ϵ)) /. {Log[Exp[x_]] → x}]);
degreeRelation = FullSimplify[degreeRelation /. {T → n / Fs}];
asymptoticDegree = Assuming[T > 0 && σ0 > 0 && Log[ϵ] < 0, AsymptoticSolve[
     degreeRelation /. {σ → σ0 T^(3/2)}, m, n → ∞, Reals] // FullSimplify];
asymptoticDegree = asymptoticDegree /. {n → Fs T}
```

Out[•]=
$$\left\{\left\{m \to -\frac{1}{2} - \frac{2^{3/4}\, T^{3/4}\, \sqrt{\sigma 0}\, \left(-\frac{\text{Log}[\epsilon]}{\text{Fs}\, T}\right)^{1/4}}{\pi}\right\}, \left\{m \to -\frac{1}{2} + \frac{2^{3/4}\, T^{3/4}\, \sqrt{\sigma 0}\, \left(-\frac{\text{Log}[\epsilon]}{\text{Fs}\, T}\right)^{1/4}}{\pi}\right\},$$
$$\left\{m \to \frac{\left(\frac{1}{\text{Fs}\, T}\right)^{1/3} T\, \sigma 0^{2/3} \left(3\, \pi^2 - 8\, \sqrt{2}\, T^{3/2}\, \sigma 0\, \sqrt{-\frac{\text{Log}[\epsilon]}{\text{Fs}\, T}}\right)}{3 \times 3^{1/3}\, \pi^{10/3}}\right\},$$
$$\left\{m \to \frac{\left(\frac{1}{\text{Fs}\, T}\right)^{1/3} T\, \sigma 0^{2/3} \left(3\, \pi^2 + 8\, \sqrt{2}\, T^{3/2}\, \sigma 0\, \sqrt{-\frac{\text{Log}[\epsilon]}{\text{Fs}\, T}}\right)}{3 \times 3^{1/3}\, \pi^{10/3}}\right\}\right\}$$

We want the solution in which m is an increasing function of $\sigma 0$ and T, ruling out the first and third options:

In[•]:=
```
Assuming[Log[ϵ] < 0 && n > 0 && Fs > 0 && T > 0,
 Limit[asymptoticDegree⟦;; , 1, 2⟧, σ0 → ∞]]
Assuming[Log[ϵ] < 0 && n > 0 && Fs > 0 && σ0 > 0, Limit[asymptoticDegree⟦;; , 1, 2⟧, T → ∞]]
```

Out[•]= $\{-\infty, \infty, -\infty, \infty\}$

Out[•]= $\{-\infty, \infty, -\infty, \infty\}$

Examining a concrete example corresponding to the plot above suggests that the second option is the right one.

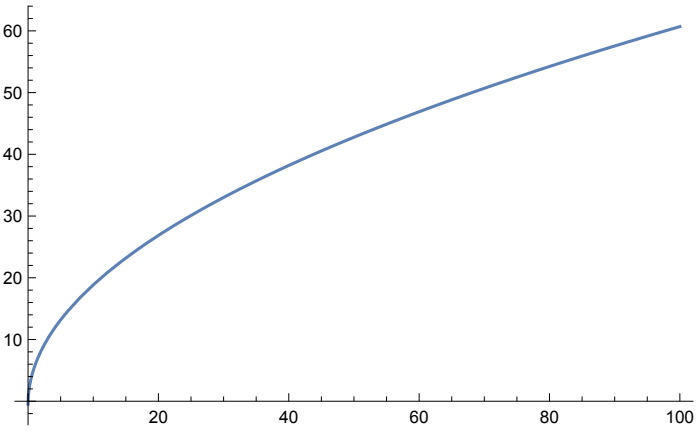
In[•]:=
```
ConcreteProblem[]
N[asymptoticDegree⟦;; , 1⟧ /. {ϵ → 10^-3}]
AbstractProblem[]
```

Out[•]= $\{m \to -19.8494, m \to 18.8494, m \to -4710.95, m \to 4729.86\}$

In[•]:=
```
asymptoticDegree[[2, 1]]
ConcreteProblem[];
Clear[T];
curve = Assuming[T > 0, FullSimplify[asymptoticDegree[[2, 1]] /. {ϵ → .001}]];
Plot[m /. curve, {T, Quantity[0, "Seconds"], Quantity[100, "Seconds"]}]
AbstractProblem[];
```

Out[•]=
$$m \to -\frac{1}{2} + \frac{2^{3/4}\, T^{3/4}\, \sqrt{\sigma 0}\, \left(-\frac{\text{Log}[\epsilon]}{\text{Fs}\, T}\right)^{1/4}}{\pi}$$

Out[•]=



Asymptotic form for the roots of eigvaleqn

As a curio, while the eigenvalue relation for the integrated random walk lacks an obvious closed-form solution, we can express its solutions as a power series in Exp[-π(n-1/2)].

```
In[•]:= ϕ0 = π (n - 1 / 2);
Assuming[s^2 == 1 && 1 / s == s && n ∈ Integers,
 residual = Assuming[n ∈ Integers,
   FullSimplify[(((Equal @@ Solve[eigvaleqn, Cos[ϕ]][[1, 1]]) /. {ϕ → ϕ0 + e}) //
        ExpandAll // TrigExpand) /. {Sin[n π] → 0}]];
 residual = FullSimplify[residual /. {(-1)^n → s, n → -(1 / π) Log[u] + 1 / 2}];
 mRoot = 8;
 residualSeries = ExpToTrig[
   Series[TrigToExp[residual[[1]] - residual[[2]]], {e, 0, mRoot - 1}] // FullSimplify];
 root = (InverseSeries[residualSeries, r] // Normal) /. {r → 0};
 rootSeries = FullSimplify[Expand[root] + O[u]^mRoot /. {s → (-1)^n}];
 rootSeries = ϕ0 + rootSeries;
 approxRoot[n_] = Normal[rootSeries] /. {u → Exp[-ϕ0]}
]
```

$$\text{Out[•]=}\ -\frac{124\,834}{63}(-1)^n e^{-7\left(-\frac{1}{2}+n\right)\pi} - \frac{1516}{3} e^{-6\left(-\frac{1}{2}+n\right)\pi} - \frac{2006}{15}(-1)^n e^{-5\left(-\frac{1}{2}+n\right)\pi} - \frac{112}{3} e^{-4\left(-\frac{1}{2}+n\right)\pi} - \frac{34}{3}(-1)^n e^{-3\left(-\frac{1}{2}+n\right)\pi} - 4 e^{-2\left(-\frac{1}{2}+n\right)\pi} - 2(-1)^n e^{-\left(-\frac{1}{2}+n\right)\pi} + \left(-\frac{1}{2}+n\right)\pi$$

Numerical test:

```
In[•]:= neigval = NSolve[eigvaleqn && ϕ > 0,
     ϕ ∈ Interval[{0, 200}], Reals, WorkingPrecision → 200][[ ;; , 1, 2, 1]];
Log10[Abs[approxRoot[{1, 2, 3, 4, 5}] - neigval[[1 ;; 5]]]] // N
```

```
Out[•]= {-1.82203, -12.4539, -23.386, -34.3003, -45.2153}
```