An unbiased ray-marching transmittance estimator - Supplemental

Markus Kettunen, Eugene d'Eon, Jacopo Pantaleoni, Jan Novak
(NVIDIA)
ACM *SIGGRAPH 2021*
doi: https://doi.org/10.1145/3450626.3459937

Summary

In this supplemental material we include code, derivations and plots that expand upon on empirical investigations, validate our derivations and implementations.

Estimators

Code for various transmittance estimation:
inputs: each estimator takes a function f[x], an interval [0,d]
outputs: each estimator returns: {estimate of T, number of density lookups}

helper - Comb filter

In[8]:=
```
equiTupleEval[f_, d_, k_, x_] := Mean[
  Table[f[Mod[x + i d/k, d]], {i, Range[k]}]]
```

In[9]:=
```
equiTupleSamples[f_, d_, k_, x_] :=
 Table[f[Mod[x + i d/k, d]], {i, Range[k]}]
```

helper - figure of merit (FoM) - inverse efficiency

In[10]:=
```
FoM[estimator_, f_, d_, MCn_] := Module[{samples},
  samples = Table[estimator[f, d], {i, Range[MCn]}];
  Mean[Last /@ samples] × Variance[First /@ samples]
 ]
```

In[11]:=
```
checkMean[estimator_, f_, d_, MCn_] := Module[{samples},
  samples = Table[estimator[f, d], {i, Range[MCn]}];
  Mean[First /@ samples]
 ]
```

Biased “exp(mean)” estimators with Comb and end-point matching CV

In[12]:=
```
TBiasedExpMean[f_, x_, n_] := Module[{tau},
  tau = Mean[Table[f[RandomReal[{0, x}]] / (1 / x), {i, Range[n]}]];
  {Exp[-tau], n}
 ]
```

exp(mean) with query size M comb filter

In[13]:=
```
TBiasedExpMeanComb[f_, d_, M_] := Module[{tau},
  tau = d equiTupleEval[f, d, M, RandomReal[{0, d}]];
  {Exp[-tau], M}
 ]
```

exp(mean) with query size M comb filter and end-point matching

In[14]:=
```
TBiasedExpMeanComb[f_, d_, M_] := Module[{tau, fp},
  fp = f[#] + 1/2 (f[0] + f[d]) - #/d f[d] - (d - #)/d f[0] &;
  tau = d equiTupleEval[fp, d, n, RandomReal[{0, d}]];
  {Exp[-tau], n + 2}
 ]
```

Jackknife exp(mean) estimator to reduce bias:

In[15]:=
```
TBiasedExpMeanJackknife[f_, x_, n_] := Module[{tau, samples},
  samples = Table[f[RandomReal[{0, x}]] / (1 / x), {i, Range[n]}];
  tau = Mean[samples];
  {n Exp[-tau] - (n - 1)/n Sum[Exp[-Mean[Delete[samples, i]]], {i, 1, n}], n}
 ]
```

Delta tracking “Track-length” binary transmittance estimator

Σmax = majorant density

```
In[16]:= TDeltaTracking[f_, Σmax_, d_] := Module[{x, w, cost},
   x = -Log[RandomReal[]] / Σmax;
   w = 1;
   cost = 0;
   While[x < d,
    cost += 1;
    If[RandomReal[] < f[x]/Σmax,
     w = 0;
     Break[];
    ];
    x += -Log[RandomReal[]] / Σmax;
   ];
   {w, cost}
  ]
```

Johnson's estimator Eq.(6)

This is a novel variation of the standard track-length / delta-tracking transmittance estimator based on a zero-order Poisson probability estimator by [Johnson 1951]. This can outperform the average of n delta-tracking estimates in some cases (see efficiency comparison below).

```
In[17]:= TJohnson[f_, Σmax_, d_, n_] := Module[{x, w, cost, vsum},
   vsum = 0;
   x = -Log[RandomReal[]] / Σmax;
   cost = 0;
   While[x < n d,
    cost += 1;
    If[RandomReal[] < f[Mod[x, d]]/Σmax,
     vsum += 1;
    ];
    x += -Log[RandomReal[]] / Σmax;
   ];
   {(1 - n^-1)^vsum, cost}
  ]
```

DT Next Flight

In[18]:=
```
TDeltaTrackingNextFlight[f_, Σmax_, d_] := Module[{x, ans, cost},
  ans = Exp[-Σmax d];
  cost = 0;
  x = -Log[RandomReal[]] / Σmax;
  While[x < d,
   cost += 1;
   ans += (1 - f[x]/Σmax) Exp[-Σmax (d - x)];
   If[RandomReal[] < f[x]/Σmax,
    Break[];
   ];
   x += -Log[RandomReal[]] / Σmax;
  ];
  {ans, cost}
 ]
```

Ratio Tracking

basic ratio tracking:

In[19]:=
```
TRatioTracking[f_, Σmax_, d_] := Module[{x, w, cost},
  x = -Log[RandomReal[]] / Σmax;
  cost = 0;
  w = 1;
  While[x < d,
   cost += 1;
   w *= 1 - f[x]/Σmax;
   x += -Log[RandomReal[]] / Σmax;
  ];
  {w, cost}
 ]
```

next-flight ratio tracking:

In[20]:=
```
TNextFlightRatioTracking[f_, Σmax_, d_] := Module[{x, ans, w, cost},
  ans = Exp[-Σmax d];
  x = -Log[RandomReal[]] / Σmax;
  w = 1;
  cost = 0;
  While[x < d,
   cost += 1;
   ans += w (1 - f[x]/Σmax) Exp[-Σmax (d - x)];
   w *= 1 - f[x]/Σmax;
   x += -Log[RandomReal[]] / Σmax;
  ];
  {ans, cost}
 ]
```

Poisson estimator (equivalent to residual ratio tracking):

In[21]:=
```
TPoissonEstimator[f_, controlvariate_, d_, λ_] := Module[{x, w, k, cost},
  x = -Log[RandomReal[]] / λ;
  w = 1;
  cost = 0;
  k = 0;
  While[x < d,
   cost += 1;
   w *= controlvariate - f[x];
   k += 1;
   x += -Log[RandomReal[]] / λ;
  ];
  {w Exp[(λ - controlvariate) d] λ^-k, cost}
 ]
```

Bhanot and Kennedy

Our generalization of the BK estimator with minimum expansion order K and roulette parameter c (BK's original estimator is when K = Floor[c])

In[22]:=
```
TBK[f_, Σmax_, d_, K_, C_] := Module[{xnproduct, result, i, cost},
  xnproduct = 1;
  result = 1;
  cost = 0;
  Do[
   cost += 1;
   xnproduct *= d (Σmax - f[RandomReal[{0, d}]]);
   result += 1/i! xnproduct;
   , {i, Range[K]}
  ];
  i = 1;
  While[RandomReal[] < C/(K + i),
   cost += 1;
   xnproduct *= d (Σmax - f[RandomReal[{0, d}]]);
   result += xnproduct/(C^i K!);
   i += 1;
  ];
  {result Exp[-Σmax d], cost}
 ]
```

Generalized BK symmetrized with U-statistics:

In[23]:=
```
TUBK[f_, Σmax_, d_, K_, c_] := Module[{i, m, positions, x, p, e},
  i = 1;
  While[RandomReal[] < c/(K + i), i += 1;];
  m = K + i - 1;
  (*m total samples*)
  positions = RandomReal[{0, d}, m];
  x = d (Σmax - f[#]) & /@ positions;
  p[0] = 1;
  Do[p[i] = Sum[x[[j]]^i, {j, Range[m]}];, {i, Range[m]}];
  e[0] = 1;
  Do[e[k] = 1/k Sum[(-1)^(i-1) e[k - i] × p[i], {i, 1, k}];, {k, Range[m]}];
  {Exp[-Σmax d] (Sum[1/i! e[i]/Binomial[m, i], {i, 0, K}] +
      Sum[1/(c^(i-K) K!) e[i]/Binomial[m, i], {i, K + 1, m}]), m}
 ]
```

U-BK estimator with query-size M Comb filter (K = Floor[c])

```
In[24]:= TUBKComb[f_, Σmax_, d_, c_, M_] := Module[{K, i, m, positions, x, p, e},
   K = Floor[c];
   i = 1;
   While[RandomReal[] < c/(K + i), i += 1;];
   m = K + i - 1;
   (*m total samples*)
   positions = RandomReal[{0, d}, m];
   x = d (Σmax - equiTupleEval[f, d, M, #]) & /@ positions;
   p[0] = 1;
   Do[p[i] = Sum[x[[j]]^i, {j, Range[m]}];, {i, Range[m]}];
   e[0] = 1;
   Do[e[k] = 1/k Sum[(-1)^(i-1) e[k - i] × p[i], {i, 1, k}];, {k, Range[m]}];
   {Exp[-Σmax d] (Sum[1/i! e[i]/Binomial[m, i], {i, 0, K}] +
       Sum[1/(c^(i-K) K!) e[i]/Binomial[m, i], {i, K + 1, m}]), m M}
  ]
```

U-BK estimator with tuple-size M Comb filter and endpoint matching (K = Floor[c])

```
In[25]:= TUBKCombEndpoint[f_, Σmax_, d_, c_, M_] := Module[{K, i, m, positions, x, p, e, fp},
   fp = f[#] + 1/2 (f[0] + f[d]) - #/d f[d] - (d - #)/d f[0] &;
   K = Floor[c];
   i = 1;
   While[RandomReal[] < c/(K + i), i += 1;];
   m = K + i - 1;
   (*m total samples*)
   positions = RandomReal[{0, d}, m];
   x = d (Σmax - equiTupleEval[fp, d, M, #]) & /@ positions;
   p[0] = 1;
   Do[p[i] = Sum[x[[j]]^i, {j, Range[m]}];, {i, Range[m]}];
   e[0] = 1;
   Do[e[k] = 1/k Sum[(-1)^(i-1) e[k - i] × p[i], {i, 1, k}];, {k, Range[m]}];
   {Exp[-Σmax d] (Sum[1/i! e[i]/Binomial[m, i], {i, 0, K}] +
       Sum[1/(c^(i-K) K!) e[i]/Binomial[m, i], {i, K + 1, m}]), m M + 2}
  ]
```

U-BK estimator with tuple-size M Comb filter and pivot estimation (K = Floor[c])

In[26]:=
```
TUBKCombPivotEstimation[f_, d_, c_, combM_, pivotM_] :=
 Module[{K, i, m, positions, x, p, e, fp, piv},
  K = Floor[c];
  fp = f;
  piv = equiTupleEval[fp, d, pivotM, RandomReal[{0, d}]];
  i = 1;
  While[RandomReal[] < c/(K + i), i += 1;];
  m = K + i - 1;
  (*m total samples*)
  positions = RandomReal[{0, d}, m];
  x = d (piv - equiTupleEval[fp, d, combM, #]) & /@ positions;
  p[0] = 1;
  Do[p[i] = Sum[x[[j]]^i, {j, Range[m]}];, {i, Range[m]}];
  e[0] = 1;
  Do[e[k] = 1/k Sum[(-1)^(i-1) e[k - i] × p[i], {i, 1, k}];, {k, Range[m]}];
  {Exp[-piv d] (Sum[1/i! e[i]/Binomial[m, i], {i, 0, K}] +
      Sum[1/(c^(i-K) K!) e[i]/Binomial[m, i], {i, K + 1, m}]), m combM + pivotM}
 ]
```

U-BK estimator with pivot-size M Comb filter, endpoint matching and pivot estimation (K = Floor[c])

In[27]:=
```
TUBKCombEndpointPivotEstimation[f_, d_, c_, combM_, pivotM_] :=
 Module[{K, i, m, positions, x, p, e, fp, piv},
  fp = f[#] + 1/2 (f[0] + f[d]) - #/d f[d] - (d - #)/d f[0] &;
  piv = equiTupleEval[fp, d, pivotM, RandomReal[{0, d}]];
  K = Floor[c];
  i = 1;
  While[RandomReal[] < c/(K + i), i += 1;];
  m = K + i - 1;
  (*m total samples*)
  positions = RandomReal[{0, d}, m];
  x = d (piv - equiTupleEval[fp, d, combM, #]) & /@ positions;
  p[0] = 1;
  Do[p[i] = Sum[x[[j]]^i, {j, Range[m]}];, {i, Range[m]}];
  e[0] = 1;
  Do[e[k] = 1/k Sum[(-1)^(i-1) e[k - i] × p[i], {i, 1, k}];, {k, Range[m]}];
  {Exp[-piv d] (Sum[1/i! e[i]/Binomial[m, i], {i, 0, K}] +
      Sum[1/(c^(i-K) K!) e[i]/Binomial[m, i], {i, K + 1, m}]), m combM + 2 + pivotM}
 ]
```

unbiased ray marching

Algorithm 1:

In[28]:=
```
ElementaryMeans[x_] := Module[{NN, m, i, n},
  NN = Length[x];
  m[0] = 1;
  Do[
   Do[
     m[k] = m[k] + k/n (m[k - 1] × x[[n]] - m[k]);
     , {k, Min[n, NN], 1, -1}
    ];
   , {n, 1, NN}
  ];
  Table[m[i], {i, 0, NN}]
 ]
```

Algorithm 2:

In[29]:=
```
AggressiveBKRoulette[K_, c_, pZ_] := Module[{ws, P, i, u},
  ws = {1};
  P = 1 - pZ;
  u = RandomReal[];
  If[P ≤ u,
   ws,
   Do[AppendTo[ws, 1/P], {i, Range[K]}];
   i = K + 1;
   While[True,
    P = P Min[c/i, 1];
    If[P ≤ u,
     Break[];
    ];
    AppendTo[ws, 1/P];
    i += 1;
   ];
   ws
  ]
 ]
```

Algorithm 5:

Our unbiased ray marching estimator:

In[30]:=
```
TURM[f_, d_, maj_] :=
 Module[{K, i, m, positions, x, p, e, fp, piv, c, taubar, M, Xs, w, NN, cost, ms, T},
  cost = 0;

  (*control optical thickness*)
  taubar = d maj;

  (* Algorithm 4 *)
  M = Max[1, Floor[((0.015 + taubar) (0.65 + taubar) (60.3 + taubar))^(1/3)/(0.31945 + 1) + 0.5]];

  (* endpoint matching *)
  fp = f[#] + 1/2 (f[0] + f[d]) - #/d f[d] - (d - #)/d f[0] &;
  cost += 2;

  (* aggressive roulette *)
  K = c = 2;
  w = AggressiveBKRoulette[K, c, 0.9];
  NN = Length[w] - 1;

  (* combed negative optical depth estimation *)
  Xs = Table[-d equiTupleEval[fp, d, M, RandomReal[{0, d}]], {i, Range[NN + 1]}];
  cost += M (NN + 1);

  (* truncated power series estimation*)
  T = 0;
  Do[
   ms = ElementaryMeans[Delete[Xs, i] - Xs[[i]]];
   T = T + 1/(NN + 1) Exp[Xs[[i]]] Sum[ms[[k + 1]]/(k ! w[[k + 1]]), {k, 0, NN}];
   , {i, Range[NN + 1]}];
  {T, cost}
 ]
```

Algorithm 6:

Biased ray marching

In[31]:=
```
TBRM[f_, d_, maj_] :=
 Module[{K, i, m, positions, x, p, e, fp, piv, c, taubar, M, Xs, w, NN, cost, ms, T},

  (*control optical thickness*)
  taubar = d maj;

  (* Algorithm 4 *)
  M = Max[1, Floor[((0.015 + taubar) (0.65 + taubar) (60.3 + taubar))^(1/3)/(0.31945 + 1) + 0.5]];

  (* endpoint matching *)
  fp = f[#] + 1/2 (f[0] + f[d]) - #/d f[d] - (d - #)/d f[0] &;

  {Exp[-d equiTupleEval[fp, d, M, RandomReal[{0, d}]]], 2 + M}
 ]
```

p-series CMF

[Georgiev et al. 2019]

goal parameter is typically 0.99

In[32]:=
```
TpCMF[f_, Σmax_, d_, goal_] := Module[{x, Tr, w, runningCDF, maxD,
    pdf, exponent, lastPDF, tau, rr, i, tmpT, invI, dense, wi, accept},
  runningCDF = 0;
  i = 1;
  maxD = Σmax;
  pdf = d;
  w = 1;
  tau = pdf * maxD;
  Tr = 0;
  exponent = Exp[-tau];
  lastPDF = exponent;
  rr = RandomReal[];
  While[runningCDF < goal,
   tmpT = RandomReal[] d;
   runningCDF += lastPDF;
   invI = 1.0 / i;
   dense = f[tmpT];
   wi = invI pdf (maxD - dense);
   lastPDF *= tau invI;
   Tr += w;
   w *= wi;
   i += 1;
  ];
  While[True,
   accept = tau / i;
   Tr += w;
   If[accept ≤ rr, Break[]];
   rr /= accept;
   runningCDF += lastPDF;
   tmpT = RandomReal[] d;
   invI = 1 / i;
   dense = f[tmpT];
   wi = invI (maxD - dense) pdf;
   lastPDF *= tau invI;
   w *= (wi / accept);
   i += 1;
  ];
  {Tr exponent, i - 1}
 ]
```

approximate mean number of samples taken by pCMF estimator at 99% mass with majorant optical depth m:

In[33]:=
```
pCMFMeanN[τbar_] := Ceiling[((0.015 + τbar) (0.65 + τbar) (60.3 + τbar))^(1/3)]
```

p-series Cumulative

[Georgiev et al. 2019]

In[34]:=
```
TpCumulative[f_, Σmax_, d_] :=
 Module[{t, W, i, rr, left, x, wi, accept, pdf, tmpT, dense, invI},
  t = 0; W = 1; i = 1;
  pdf = d;
  rr = RandomReal[];
  While[True,
   left = W;
   tmpT = RandomReal[] d;
   dense = f[tmpT];
   invI = 1.0 / i;
   wi = invI pdf (Σmax - dense);
   accept = Min[1, Abs[W wi]];
   If[accept ≤ rr,
    t += left;
    Break[];
   ];
   rr /= accept;
   t += left;
   W *= (wi / accept);
   i += 1.0;
  ];
  {t Exp[-Σmax d], i}
 ]
```

p-series next flight

[Georgiev et al. 2019]

In[35]:=
```
TpNF[f_, pivot_, d_, λ_] := Module[{xnproduct, eN, result, i, cost},
  xnproduct = 1;
  result = 1;
  result = 1;
  eN = RandomVariate[PoissonDistribution[λ]];
  cost = 0;
  Do[
   cost += 1;
   xnproduct *= d (pivot - f[RandomReal[{0, d}]]);
   result += 1 / (i! (1 - CDF[PoissonDistribution[λ]][i - 1])) xnproduct;
   , {i, Range[eN]}
  ];
  {result Exp[-pivot d], cost}
 ]
```

Generalized with U-statistics

```
In[36]:= TUpNF[f_, pivot_, d_, λ_] := Module[{eN, i, m, positions, x, p, e},
   eN = RandomVariate[PoissonDistribution[λ]];
   i = 1;
   m = eN + i - 1; (*m total samples*)
   positions = RandomReal[{0, d}, m];
   x = d (pivot - f[#]) & /@ positions;
   p[0] = 1;
   Do[p[i] = Sum[x[[j]]^i, {j, Range[m]}];, {i, Range[m]}];
   e[0] = 1;
   Do[e[k] = 1/k Sum[(-1)^(i-1) e[k - i] × p[i], {i, 1, k}];, {k, Range[m]}];
   {Exp[-pivot d] (Sum[
        1/(i! (1 - CDF[PoissonDistribution[λ]][i - 1])) e[i]/Binomial[m, i], {i, 0, eN}]), m}
  ]
```

Compare estimators

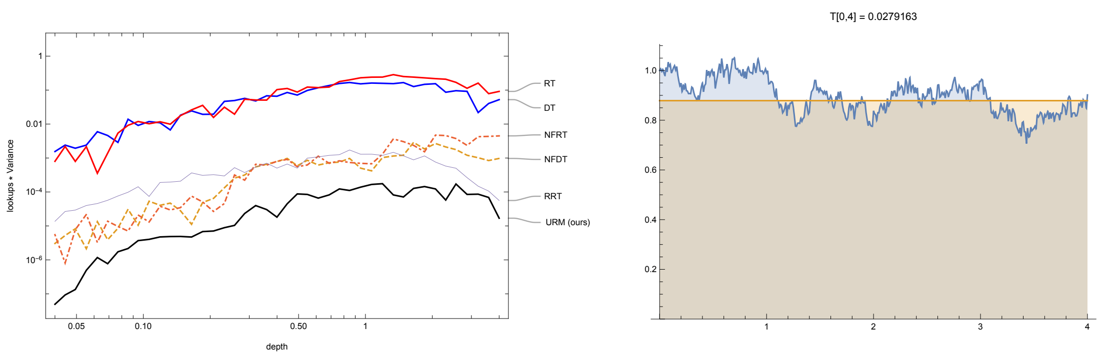
Plot cost * variance for various estimators for a fixed density function f[x] as the interval with [0,d] is varied from d = 0 to d = 4.

Basic estimators

In[•]:=

```
maxpos = 4;
td =
  .1 RandomFunction[FractionalBrownianMotionProcess[.3], {0, maxpos, 0.01}] + 1;
data = TimeSeriesMap[Max[0, #] &, td];
Clear[f];
f = Interpolation[data];
maxT = Exp[-NIntegrate[f[x], {x, 0, maxpos}]];
maj = Max[data];
min = Min[data];
midcontrol = maj / 2 + min / 2;
MCn = 100;
rate = 5;
Magnify[GraphicsRow[{LogLogPlot[
     {
      FoM[TDeltaTracking[#1, maj, #2] &, f, pos, MCn]
      (* delta tracking - tight majorant *),
      FoM[TDeltaTrackingNextFlight[#1, maj, #2] &, f, pos, MCn]
      (* NF delta tracking - tight majorant *),
      FoM[TRatioTracking[#1, maj, #2] &, f, pos, MCn] (* RT - tight majorant *),
      FoM[TNextFlightRatioTracking[#1, maj, #2] &, f, pos, MCn]
      (* NFRT - tight majorant *),
      FoM[TPoissonEstimator[#1, midcontrol + rate, #2, rate] &, f, pos, MCn]
      (* RRT - mid control *),
      FoM[TURM[#1, #2, maj] &, f, pos, MCn] (* unbiased ray marching *)
      ,
     }, {pos, .01 maxpos, maxpos}, MaxRecursion → 1,
     PlotPoints → 22, PlotStyle → {Blue, Dashed, Red, DotDashed, Thin,
       Black, {Red, Dashed}, {Green, Dashed}, {Orange, DotDashed}},
     Frame → True, FrameLabel → {{"lookups * Variance",}, {"depth",}},
     PlotLabels → {"DT", "NFDT", "RT", "NFRT", "RRT", "URM (ours)"}],
    Plot[{f[x], midcontrol}, {x, 0, maxpos}, Filling → Axis, PlotRange → {0, All},
     PlotLabel → "T[0," <> ToString[maxpos] <> "] = " <> ToString[maxT]]
   }, ImageSize → 1200
  ], 0.5]
```

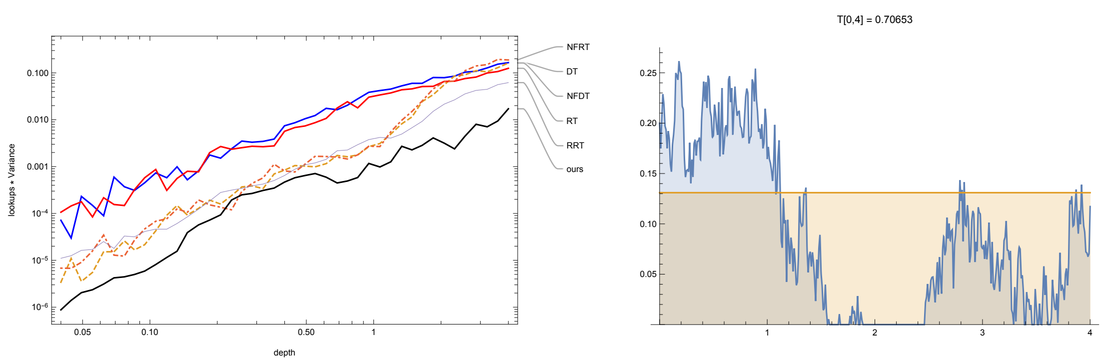
Out[•]=



In[•]:=
```
maxpos = 4;
td =
  .1 RandomFunction[FractionalBrownianMotionProcess[.3], {0, maxpos, 0.01}] + .2;
data = TimeSeriesMap[Max[0, #] &, td];
Clear[f];
f = Interpolation[data];
maxT = Exp[-NIntegrate[f[x], {x, 0, maxpos}]];
maj = Max[data];
min = Min[data];
midcontrol = maj / 2 + min / 2;
MCn = 1000;
rate = 5;
Magnify[GraphicsRow[{LogLogPlot[
     {
      FoM[TDeltaTracking[#1, maj, #2] &, f, pos, MCn]
      (* delta tracking - tight majorant *),
      FoM[TDeltaTrackingNextFlight[#1, maj, #2] &, f, pos, MCn]
      (* NF delta tracking - tight majorant *),
      FoM[TRatioTracking[#1, maj, #2] &, f, pos, MCn] (* RT - tight majorant *),
      FoM[TNextFlightRatioTracking[#1, maj, #2] &, f, pos, MCn]
      (* NFRT - tight majorant *),
      FoM[TPoissonEstimator[#1, midcontrol + rate, #2, rate] &, f, pos, MCn]
      (* RRT - mid control *),
      FoM[TURM[#1, #2, maj] &, f, pos, MCn] (* unbiased ray marching *)

     }, {pos, .01 maxpos, maxpos}, MaxRecursion → 1,
     PlotPoints → 22, PlotStyle → {Blue, Dashed, Red, DotDashed, Thin,
       Black, {Red, Dashed}, {Green, Dashed}, {Orange, DotDashed}},
     Frame → True, FrameLabel → {{"lookups * Variance",}, {"depth",}},
     PlotLabels → {"DT", "NFDT", "RT", "NFRT", "RRT", "ours"}],
    Plot[{f[x], midcontrol}, {x, 0, maxpos}, Filling → Axis, PlotRange → {0, All},
     PlotLabel → "T[0," <> ToString[maxpos] <> "] = " <> ToString[maxT]]
   }, ImageSize → 1200
  ], 0.5]
```

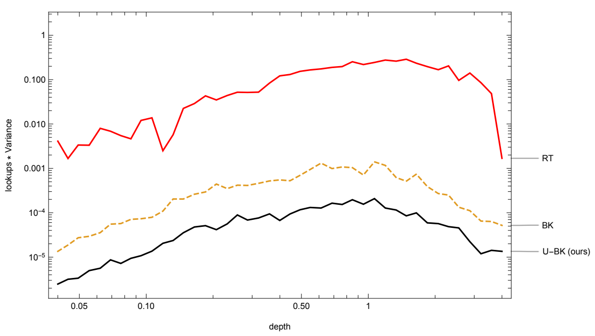
Out[•]=

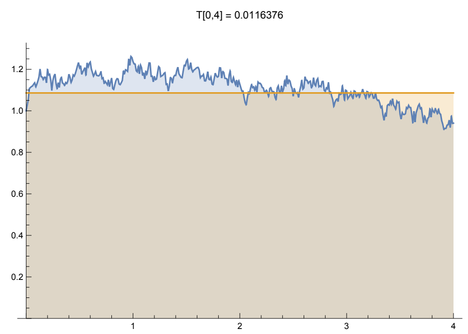


BK vs U-BK

In[∘]:=
```
maxpos = 4;
td =
  .1 RandomFunction[FractionalBrownianMotionProcess[.3], {0, maxpos, 0.01}] + 1;
data = TimeSeriesMap[Max[0, #] &, td];
Clear[f];
f = Interpolation[data];
maxT = Exp[-NIntegrate[f[x], {x, 0, maxpos}]];
maj = Max[data];
min = Min[data];
midcontrol = maj / 2 + min / 2;
MCn = 100;
rate = 5;
K = c = 5;
Magnify[GraphicsRow[{LogLogPlot[
     {
      FoM[TRatioTracking[#1, maj, #2] &, f, pos, MCn] (* RT - tight majorant *),
      FoM[TBK[#1, maj, #2, K, c] &, f, pos, MCn],
      FoM[TUBK[#1, maj, #2, K, c] &, f, pos, MCn]
      (* RRT - mid control *),
     }, {pos, .01 maxpos, maxpos}, MaxRecursion → 1,
     PlotPoints → 22, PlotStyle → {Red, Dashed, Black}, Frame → True,
     FrameLabel → {{"lookups * Variance",}, {"depth",}},
     PlotLabels → {"RT", "BK", "U-BK (ours)"}],
    Plot[{f[x], midcontrol}, {x, 0, maxpos}, Filling → Axis, PlotRange → {0, All},
     PlotLabel → "T[0," <> ToString[maxpos] <> "] = " <> ToString[maxT]]
   }, ImageSize → 1200
  ], 0.5]
```

Out[∘]=





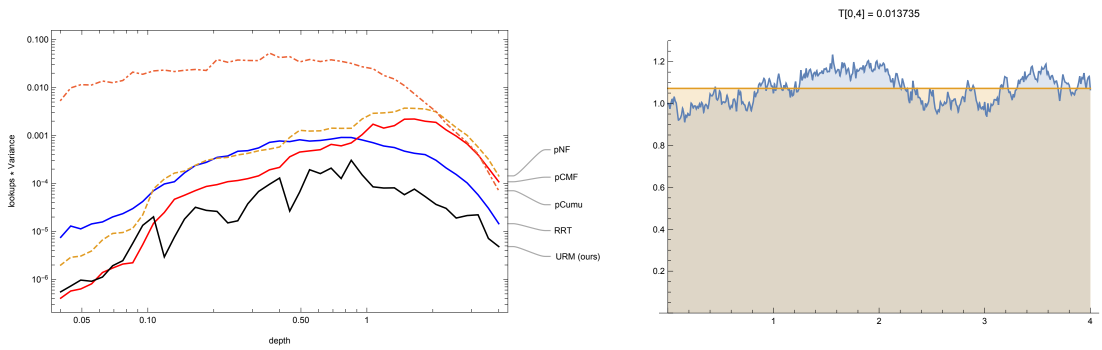
compare p-series NF, pCMF, pCumulative

RRT uses half the tight majorant (for the entire interval) as a control:

p-series Cumulative can perform very poorly for optically thin intervals such as this case:

In[•]:=
```
maxpos = 4;
td =
  .1 RandomFunction[FractionalBrownianMotionProcess[.3], {0, maxpos, 0.01}] + 1;
data = TimeSeriesMap[Max[0, #] &, td];
Clear[f];
f = Interpolation[data];
maxT = Exp[-NIntegrate[f[x], {x, 0, maxpos}]];
maj = Max[data];
min = Min[data];
midcontrol = maj / 2 + min / 2;
MCn = 1000;
rate = 10;
Magnify[GraphicsRow[{LogLogPlot[
     {
      FoM[TPoissonEstimator[#1, midcontrol + rate, #2, rate] &, f, pos, MCn]
      (* RRT - mid control *),
      FoM[TpNF[#1, midcontrol, #2, rate] &, f, pos, MCn],
      FoM[TpCMF[#1, maj, #2, 0.99] &, f, pos, MCn],
      FoM[TpCumulative[#1, maj, #2] &, f, pos, MCn],
      FoM[TURM[#1, #2, maj] &, f, pos, MCn] (* unbiased ray marching *)
     }, {pos, .01 maxpos, maxpos}, MaxRecursion → 1,
     PlotPoints → 22, PlotStyle → {Blue, Dashed, Red, DotDashed,
       Black, {Red, Dashed}, {Green, Dashed}, {Orange, DotDashed}},
     Frame → True, FrameLabel → {{"lookups * Variance",}, {"depth",}},
     PlotLabels → {"RRT", "pNF", "pCMF", "pCumu", "URM (ours)"}],
    Plot[{f[x], midcontrol}, {x, 0, maxpos}, Filling → Axis, PlotRange → {0, All},
     PlotLabel → "T[0," <> ToString[maxpos] <> "] = " <> ToString[maxT]]
   }, ImageSize → 1200
  ], .5]
```

Out[•]=

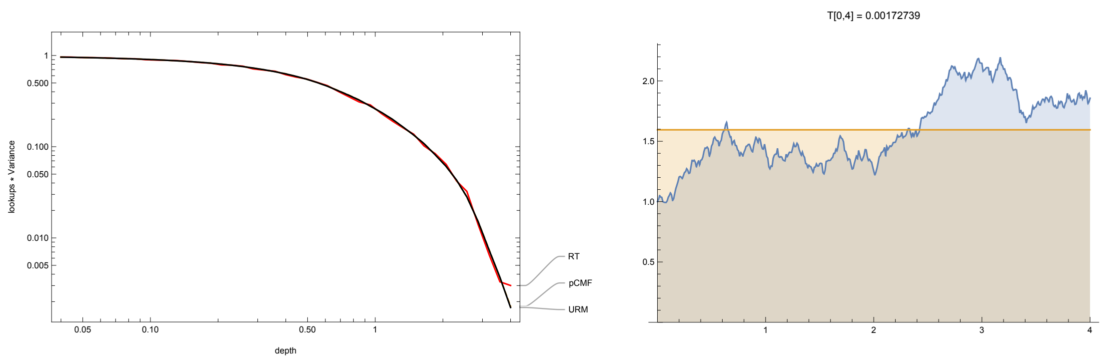


verify correctness/bias

pCMF vs URM (unbiased ray marching) vs RT

In[]:=
```
maxpos = 4;
td =
  .3 RandomFunction[FractionalBrownianMotionProcess[.5], {0, maxpos, 0.01}] + 1;
data = TimeSeriesMap[Max[0, #] &, td];
Clear[f];
f = Interpolation[data];
maxT = Exp[-NIntegrate[f[x], {x, 0, maxpos}]];
maj = Max[data];
min = Min[data];
midcontrol = maj / 2 + min / 2;
MCn = 1000;
rate = 5;
K = c = 2;
M = 5;
pM = 5;
Magnify[GraphicsRow[{LogLogPlot[
     {
      checkMean[TRatioTracking[#1, maj, #2] &, f, pos, MCn]
      (* RT - tight majorant *),
      checkMean[TpCMF[#1, maj, #2, 0.99] &, f, pos, MCn],
      checkMean[TURM[#1, #2, maj] &, f, pos, MCn]
     }, {pos, .01 maxpos, maxpos}, MaxRecursion → 1,
     PlotPoints → 22, PlotStyle → {Red, Dashed, Black}, Frame → True,
     FrameLabel → {{"lookups * Variance",}, {"depth",}},
     PlotLabels → {"RT", "pCMF", "URM"}],
    Plot[{f[x], midcontrol}, {x, 0, maxpos}, Filling → Axis, PlotRange → {0, All},
     PlotLabel → "T[0," <> ToString[maxpos] <> "] = " <> ToString[maxT]]
   }, ImageSize → 1200
  ], 0.5]
```

Out[]=



Minimum number of lookups K of p-series CMF with 99%

mass setting:

Exact result follows from:

In[∘]:=
```
Exp[-τbar] Sum[(τbar)^i / i!, {i, 0, k}]
```

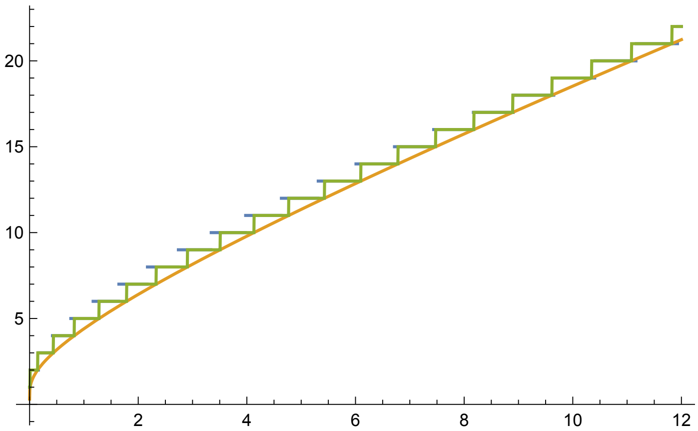
Out[∘]=
```
Gamma[1 + k, τbar] / k!
```

routine that returns K:

In[∘]:=
```
KpCMF[tau_] := Module[{runningCDF, i, exponent, goal, lastPDF, invI},
  runningCDF = 0;
  i = 1;
  exponent = Exp[-tau];
  goal = 0.99;
  lastPDF = exponent;
  While[runningCDF < goal,
   runningCDF += lastPDF;
   invI = 1.0 / i;
   lastPDF *= tau invI;
   i += 1;
  ];
  i - 1
 ]
```

In[∘]:=
```
Plot[
 {
  Ceiling[Sqrt[0.8 + 19 m + 1.5 m^2]],
  FindRoot[Gamma[k, m] / Gamma[k] - 0.99, {k, m}][[-1, -1]],
  KpCMF[m]
 }, {m, 0, 12}]
```

Out[∘]=



The mean number of lookups N of p-series CMF with

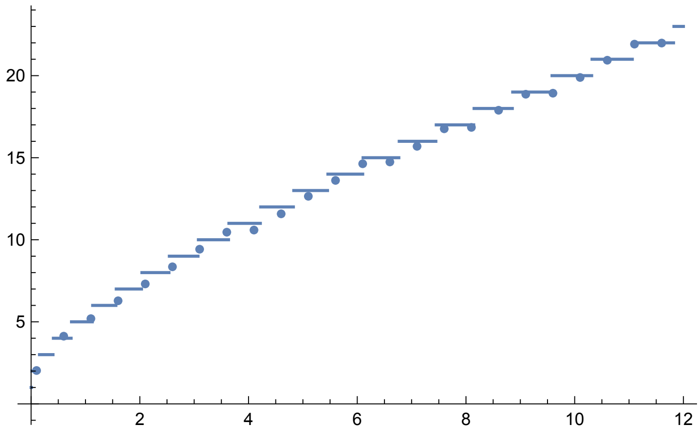
99% mass setting:

Validating our approximation in the paper:

$$\mathbb{E}[N_{\text{CMF}}] \approx \left\lceil \sqrt[3]{(0.015+\bar{\tau})(0.65+\bar{\tau})(60.3+\bar{\tau})} \right\rceil . \qquad (39)$$

In[∘]:=
```
Show[
 Plot[
  {pCMFMeanN[m]
  }, {m, 0, 12}],
 ListPlot[
  Table[{m,
    Mean[Table[TpCMF[1 &, 1, m, 0.99], {i, Range[1000]}]][[2]]
   }, {m, 0.1, 12.0, 0.5}]
 ]
]
```

Out[∘]=



Residual Ratio Tracking and Poisson Estimator relationship

These two equivalent estimators are written in a different and potentially confusing way. This arises because the use of ‘control’ has different meanings.

In graphics, [Novak et al. 2014] presented residual ratio tracking as:

```
TResidualRatioTracking[f_, d_, maj_, control_] := Module[{λ, Yprod, n},
  λ = d (maj - control);
  n = RandomVariate[PoissonDistribution[λ]];
  Yprod = Product[
    (1 - (f[RandomReal[{0, d}]] - control)/(maj - control))
    , {i, Range[n]}];
  Exp[-d control] Yprod
 ]
```

where maj is a majorant function, and control is the control function. The Poisson estimator is written:

```
PoissonEstimator[f_, d_, control_, λ_] := Module[{n, Yprod},
  n = RandomVariate[PoissonDistribution[λ]];
  Yprod = Product[-d (f[RandomReal[{0, d}]] - control), {i, Range[n]}];
  Exp[-d control] 1/(PDF[PoissonDistribution[λ]][n] n!) Yprod
 ]
```

where control is the control variate in the sense of our Eq.(9), μ_c.
τ_c follows from the interval width being d, assuming the control variate is just a constant : $\tau_c = d\,\mu_c$.

The two align when λ = d(maj - RT.control), and Poisson.control = maj.
In this case the terms of the Poisson estimator expand into:

In[•]:=
```
Clear[d, maj];
Table[Exp[-d maj] 1/(PDF[PoissonDistribution[d (maj - RTcontrol)]][n] n!)
   (-d (μ[x] - maj))^n, {n, Range[4] - 1}] // Simplify
```

Out[•]=
$$\left\{e^{-d\,\text{RTcontrol}},\ \frac{e^{-d\,\text{RTcontrol}}\,(\text{maj}-\mu[x])}{\text{maj}-\text{RTcontrol}},\ \frac{e^{-d\,\text{RTcontrol}}\,(\text{maj}-\mu[x])^2}{(\text{maj}-\text{RTcontrol})^2},\ \frac{e^{-d\,\text{RTcontrol}}\,(\text{maj}-\mu[x])^3}{(\text{maj}-\text{RTcontrol})^3}\right\}$$

which is the [Novak et al. 2014] form of RRT.

Generalized Bhanot Kennedy

K = Floor[c]

Verification of the roulette weights:

In[•]:=
```
Clear[K, c, k, X, p, d, x];
Exp[-p d] Sum[(c^k (1 - c/(1+K+k)))/Pochhammer[1 + K, k]
   ((Sum[(d (p - x))^n/n!, {n, 0, K}] + Sum[(d (p - x))^(K+i)/(c^i K!), {i, 1, k}]))
  , {k, 0, Infinity}, Assumptions → c > 0 && K > 0 && x > 0 && p > 0 && d > 0] //
 FullSimplify
```

Out[•]= $e^{-d\,x}$

Derivation of (54) for $E[N]_{BK}$:

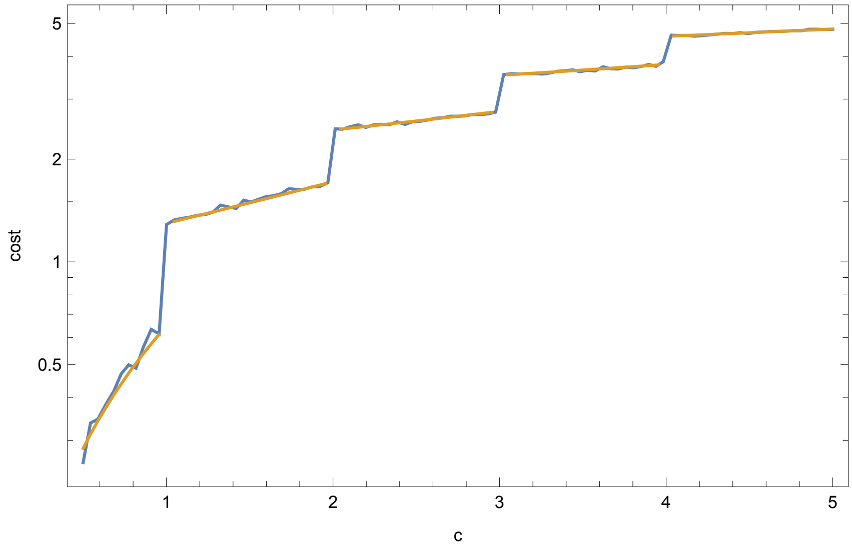
In[∘]:=
```
Clear[c, K];
FullSimplify[Sum[(c^k (1 - c/(1+K+k)))/(Pochhammer[1 + K, k]) (k + K), {k, 0, Infinity}],
 Assumptions → c > 0 && K > 0]
```

Out[∘]= $-1+K+c^{-K}\,e^{c}\,(\text{Gamma}[1+K]-K\,\text{Gamma}[K,c])$

In[∘]:=
```
costGBK[K_, c_] := -1 + K + c^-K e^c (Gamma[1 + K] - K Gamma[K, c])
```

Verify (54) $E[N]_{BK}$:

In[∘]:=
```
f = 1 &;
d = 2;
LogPlot[
 {
  Mean[Last /@ Table[TBK[f, 1.1, d, Floor[c], .5 c], {i, Range[1000]}]],
  costGBK[Floor[c], .5 c]
 }
 , {c, .5, 5}, PlotPoints → 50, MaxRecursion → 1,
 Frame → True, FrameLabel → {{"cost",}, {"c",}}, PlotRange → All
]
```

Out[∘]=


Roulette variance for BK estimator - C.2.1 - derivation and validation

Here we derive and validate the exact variance for the BK and U-BK estimators for the case of a constant-density medium (or any medium where the optical depth is estimated with zero variance). In this case all Y estimates are the same constant, and so U-BK is identical to BK.

First, compute the expectation of T^2 :

In[∘]:=
```
Clear[c, k, K, Y, τc];
FullSimplify[Sum[(c^k (1 - c/(1+K+k)))/Pochhammer[1 + K, k] (Exp[-τc] (Sum[(Y)^n/n!, {n, 0, K}] +
      Sum[(Y)^(K+i)/(c^i K!), {i, 1, k}]))^2, {k, 0, Infinity}, Assumptions →
   K > 0 && Y > 0 && 0 < c < K + 1], Assumptions → K > 0 && Y > 0 && g > 0 && 0 < c < K + 1]
```

Out[∘]=

$$\frac{1}{(c-Y)\,\text{Gamma}[1+K]^2} e^{-2\,\tau c}\left(-2\,c\,e^{Y}\,K\,Y^{K}\,\text{Gamma}[K,Y]+e^{2\,Y}\,K^2\,(-c+Y)\,\text{Gamma}[K,Y]^2+\text{Gamma}[1+K]\left(2\,c\,e^{Y}\,Y^{K}-c^{K}\,e^{\frac{Y^2}{c}}\,(c+Y)+2\,e^{2\,Y}\,K\,(c-Y)\,\text{Gamma}[K,Y]\right)+c^{K}\,e^{\frac{Y^2}{c}}\,K\,(c+Y)\,\text{Gamma}\left[K,\frac{Y^2}{c}\right]\right)$$

then subtract $E[T]^2$

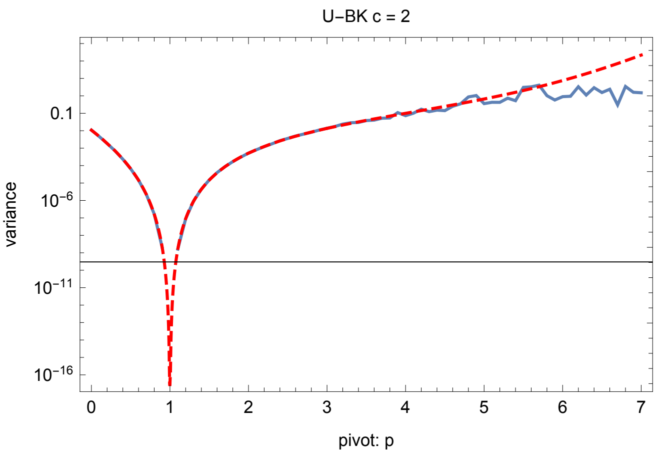
In[47]:=
```
varGBK[K_, c_, Y_, τ_, τc_] :=
  1/((c - Y) Gamma[1 + K]^2) e^(-2 τc) (-2 c e^Y K Y^K Gamma[K, Y] + e^(2 Y) K^2 (-c + Y) Gamma[K, Y]^2 +
     Gamma[1 + K] (2 c e^Y Y^K - c^K e^(Y^2/c) (c + Y) + 2 e^(2 Y) K (c - Y) Gamma[K, Y]) +
     c^K e^(Y^2/c) K (c + Y) Gamma[K, Y^2/c]) - Exp[-τ]^2
```

But the expectation of Y (a constant) is τc - τ:

In[57]:=
```
varGBK[K_, c_, τ_, τc_] := varGBK[K, c, τc - τ, τ, τc]
```

Monte Carlo validation of roulette variance formula for generalized BK:

In[130]:=
```
x = 1;
f = x &;
d = 1;
c = 2; K = Floor[c];
MCn = 1000;
maxplot = 7 x;
ppointsdx = 2 x / 40;
pts1000 = Table[{p, Variance[Table[TBK[f, p, d, K, c][[1]], {i, Range[MCn]}]]},
    {p, 0, maxplot, 0.1}];
Show[
 ListLogPlot[
  pts1000, PlotRange → All, Joined → True
 ],
 LogPlot[varGBK[K, c, x d, d p], {p, 0, maxplot},
  PlotRange → All, PlotStyle → {Dashed, Red}],
 PlotRange → All, Frame → True, FrameLabel →
  {{"variance",}, {"pivot: p", "U-BK c = 2"}}
]
```

Out[138]=


Derivation: variance of BK at optimal pivot

Here we consider the BK and U-BK estimators with optimal pivot p = $-\tau_c$, where truncation is fixed to deterministic order N. Despite the truncation, the estimators still have the correct expectation, because the pivot is optimal (therefore the expectation of Y^k is 0).
This comparison clearly and quantitatively shows the benefit of using $U-$statistics. We will show that the two estimators BK and U-BK have variances:

$$\mathrm{Var}[\widehat{T}_{BK}] = e^{-2\tau} \sum_{k=1}^{N} \frac{\mathbb{E}[Y^2]^k}{(k!)^2}$$

and

$$\mathrm{Var}[\widehat{T}_{UBK}] = e^{-2\tau} \sum_{k=1}^{N} \frac{\mathbb{E}[Y^2]^k}{\binom{N}{k}(k!)^2}.$$

The binomial denominators reduce the variance relative to the non-symmetrized estimator. For small $\mathbb{E}[Y^2]$ (low Y-variance), the linear term sees a variance reduction of $1/N$ relative to the non-symmetrized version, with diminishing gains for the higher order terms. At the optimal pivot, the variance reduction between U-BK and BK approaches $1/N$ as Y-variance goes to 0.

fixed-order truncated (biased) estimator:

pivot p

Let $V = \frac{1}{d}\int_0^d (d(p - f[x]))^2\, dx$

Compute the expectation $Z = E[T^2] / T^2$: look at the pattern for various orders:

In[775]:=
```
Clear[K, p, d, X, X2, V];
Table[{K,
    D[Expand[((Sum[Product[d (p - X[i]), {i, Range[n]}] / n!, {n, 0, K}]))^2
      ] /. X[_]^2 → X2 /. X[_] → X, {p, 0}]}, {K, Range[8]}
  ] /. X2 → (V - d^2 p^2 + 2 d^2 p X) / d^2 /. p → X // Expand // TableForm
```

Out[776]//TableForm=

1	$1 + V$
2	$1 + V + \frac{V^2}{4}$
3	$1 + V + \frac{V^2}{4} + \frac{V^3}{36}$
4	$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576}$
5	$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576} + \frac{V^5}{14400}$
6	$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576} + \frac{V^5}{14400} + \frac{V^6}{518400}$
7	$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576} + \frac{V^5}{14400} + \frac{V^6}{518400} + \frac{V^7}{25401600}$
8	$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576} + \frac{V^5}{14400} + \frac{V^6}{518400} + \frac{V^7}{25401600} + \frac{V^8}{1625702400}$

Note that this is generated simply by:

In[∘]:=
```
Table[Sum[V^k/(k!)^2, {k, 0, i}], {i, Range[6]}] // TableForm
```

Out[∘]//TableForm=

$$1 + V$$
$$1 + V + \frac{V^2}{4}$$
$$1 + V + \frac{V^2}{4} + \frac{V^3}{36}$$
$$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576}$$
$$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576} + \frac{V^5}{14\,400}$$
$$1 + V + \frac{V^2}{4} + \frac{V^3}{36} + \frac{V^4}{576} + \frac{V^5}{14\,400} + \frac{V^6}{518\,400}$$

Which generates the function

In[784]:=
```
ZFixedBKOptPivot[n_, V_] :=
  (BesselI[0, 2 Sqrt[V]] - V^(1+n) HypergeometricPFQ[{1}, {2 + n, 2 + n}, V] / ((1 + n)!)^2)
```

fixed order UBK estimator:

In[780]:=
```
Clear[K, p, X, V, d, X2, Y, w];
list =
  Table[
   Expand[
      ((Sum[1/n! Mean[Table[Product[Y[i], {i, s}], {s, Subsets[Range[K], {n}]}]],
           {n, 0, K}]))^2
     ] /. Mean[{}] → 1 /. Y[_]^2 → V /. Y[_] → 0
   , {K, Range[6]}
  ];
list // Expand // TableForm
```

Out[782]//TableForm=

$$1 + V$$
$$1 + \frac{V}{2} + \frac{V^2}{4}$$
$$1 + \frac{V}{3} + \frac{V^2}{12} + \frac{V^3}{36}$$
$$1 + \frac{V}{4} + \frac{V^2}{24} + \frac{V^3}{144} + \frac{V^4}{576}$$
$$1 + \frac{V}{5} + \frac{V^2}{40} + \frac{V^3}{360} + \frac{V^4}{2880} + \frac{V^5}{14\,400}$$
$$1 + \frac{V}{6} + \frac{V^2}{60} + \frac{V^3}{720} + \frac{V^4}{8640} + \frac{V^5}{86\,400} + \frac{V^6}{518\,400}$$

In[783]:=
```
Table[Sum[V^k/(Binomial[i, k] (k!)^2), {k, 0, i}], {i, Range[6]}] // Expand // TableForm
```

Out[783]//TableForm=

$$1 + V$$
$$1 + \frac{V}{2} + \frac{V^2}{4}$$
$$1 + \frac{V}{3} + \frac{V^2}{12} + \frac{V^3}{36}$$
$$1 + \frac{V}{4} + \frac{V^2}{24} + \frac{V^3}{144} + \frac{V^4}{576}$$
$$1 + \frac{V}{5} + \frac{V^2}{40} + \frac{V^3}{360} + \frac{V^4}{2880} + \frac{V^5}{14400}$$
$$1 + \frac{V}{6} + \frac{V^2}{60} + \frac{V^3}{720} + \frac{V^4}{8640} + \frac{V^5}{86400} + \frac{V^6}{518400}$$

In[1177]:=
```
ZFixedUBKOptPivot[n_, V_] := Sum[V^k/(Binomial[n, k] (k!)^2), {k, 0, n}]
```

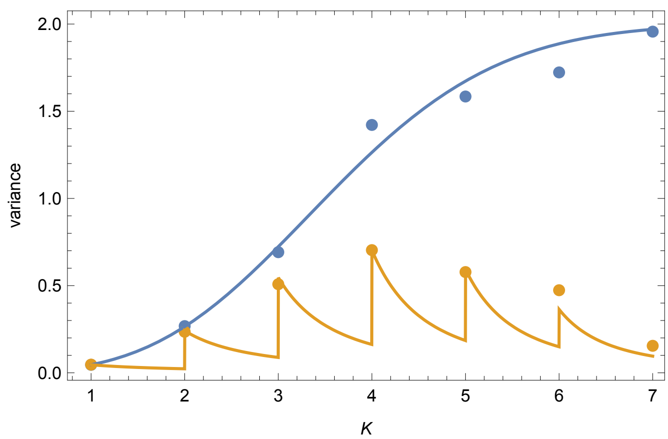
numerical validation

In[933]:=
```
varBKOptPivot[X_, V_, K_, d_] := Module[{},
  Exp[-2 d X] (ZFixedBKOptPivot[K, V] - 1)
 ]
```

In[1178]:=
```
varUBKOptPivot[X_, V_, K_, d_] := Module[{},
  Exp[-2 d X] (ZFixedUBKOptPivot[K, V] - 1)
 ]
```

In[1266]:=
```
maxpos = 4;
td =
  3 RandomFunction[FractionalBrownianMotionProcess[.3], {0, maxpos, 0.01}] + 2;
data = TimeSeriesMap[Max[0, #] &, td];
Clear[f];
f = Interpolation[data];
d = 3;
X = NIntegrate[f[x], {x, 0, d}] / d;
X2 = NIntegrate[(d (X - f[x]))^2, {x, 0, d}] / d;
MCn = 10 000;
Show[
 ListPlot[
  {
   Table[{K, Variance[Table[TBK[f, X, d, K, 0][[1]], {i, Range[MCn]}]]},
    {K, Range[7]}],
   Table[{K, Variance[Table[TUBK[f, X, d, K, 0][[1]], {i, Range[MCn]}]]},
    {K, Range[7]}]
  }
  , PlotRange → All, PlotStyle → PointSize[.02]
 ], Plot[{varBKOptPivot[X, X2, K, d], varUBKOptPivot[X, X2, K, d]},
  {K, 1, 7}, PlotRange → All], PlotRange → All,
 Frame → True, FrameLabel → {{"variance",}, {K,}}
]
```

Out[1275]=



Variance at mean pivot - general BK:

In future work, to rigorously analyze the gain in comb filtering the density/extinction field along a ray by increasing query size M, we need to know the change in variance at the mean/optimal pivot p, as a function of expansion parameter c. This can be used to tell if the net gain in lowering c is made up for by the variance reduction Var[Y] that increasing M gives. To aid this future work, we noted the follow expression for the variance of the UBK estimator at the optimal pivot (E[Y] = 0). This was achieved by expanding the variance calculations up to a fixed order and noting patterns in the powers of $E[Y^2] = V$ at various orders using FindSequenceFunction[].

special case K = c = 1

We first test a very simple result for K = c = 1:

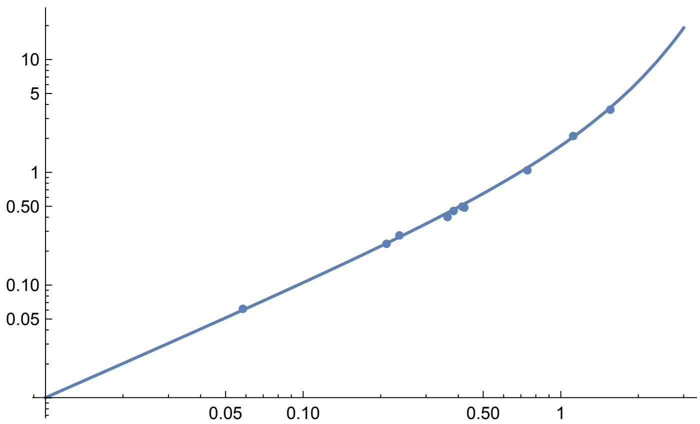
For K = c = 1 - the variance of BK simplifies to a remarkably simple result:
Var[T] = $e^{-2\tau}(e^{V}$-1)

MC check

In[∘]:=
```
MCn = 1000;
points = Table[
   (d = 2;
    td =
     .6 RandomFunction[FractionalBrownianMotionProcess[.6], {0, d, 0.01}] + 0.2;
    data = TimeSeriesMap[Max[0, #] &, td];
    Clear[f];
    f = Interpolation[data];
    X = NIntegrate[f[x], {x, 0, d}] / d;
    V = NIntegrate[(d (X - f[x]))^2, {x, 0, d}] / d;
    {X, V, Variance[Table[TBK[f, X, d, 1, 1][[1]], {i, Range[MCn]}]]}
   ), {i, Range[10]}
  ];
```

In[∘]:=
```
Show[
 LogLogPlot[E^V - 1, {V, 0.01, 3}, PlotRange → All],
 ListLogLogPlot[{#[[2]], #[[3]]/Exp[-2 d #[[1]]]} & /@ points, PlotRange → All],
 PlotRange → All
]
```

Out[∘]=



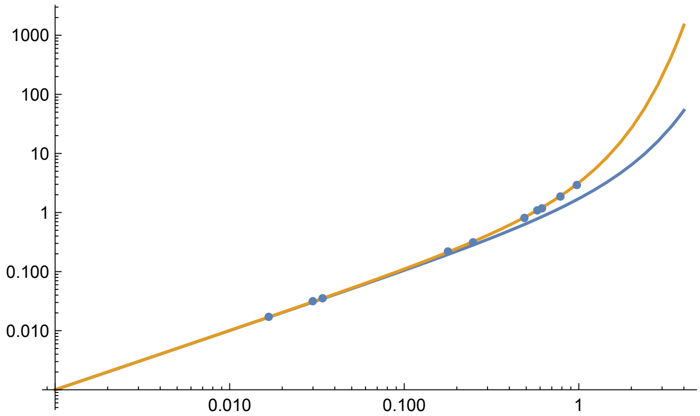
In[•]:=
```
MCn = 30000;
points = Table[
   (d = 2;
    td =
     .8 RandomFunction[FractionalBrownianMotionProcess[.5], {0, d, 0.01}] + 0.2;
    data = TimeSeriesMap[Max[0, #] &, td];
    Clear[f];
    c = 0.5;
    f = Interpolation[data];
    X = NIntegrate[f[x], {x, 0, d}] / d;
    V = NIntegrate[(d (X - f[x]))^2, {x, 0, d}] / d;
    {X, V, Variance[Table[TBK[f, X, d, 1, c][[1]], {i, Range[MCn]}]]}
   ), {i, Range[10]}
  ];
```

In[•]:= Clear[V, c]; $\left(1+\left(-1+e^{\frac{V}{c}}\right)c\right)-1$ /. c → 1/2

Out[•]= $\frac{1}{2}\left(-1+e^{2V}\right)$

In[•]:=
```
Show[
 LogLogPlot[{E^V - 1, 1/2 (-1 + e^(2 V))}, {V, 0.001, 4}, PlotRange → All],
 ListLogLogPlot[{#[[2]], #[[3]] / Exp[-2 d #[[1]]]} & /@ points, PlotRange → All],
 PlotRange → All
]
```

Out[•]=



general K, c

We check the pattern in the expansion up to various orders:

This gives a list of probabilities for terms of size N and their estimates:

In[1]:=
```
Clear[K, c, p, X, V, d, X2];
list = With[{K = 3, c = 3},
   Table[
    {(c^k (1 - c/(1+K+k)))/Pochhammer[1 + K, k], Expand[((Sum[1/n! Mean[Table[Product[Y[i], {i, s}], {s,
            Subsets[Range[k + K], {n}]}]], {n, 0, Floor[c]}] +
          Sum[1/(c^i K!) Mean[Table[Product[Y[i], {i, s}], {s, Subsets[
                Range[k + K], {Floor[c] + i}]}]], {i, 1, k}]))^2
      ] /. Mean[{}] → 1 /. Y[_]^2 → V /. Y[_] → 0}, {k, Range[9] - 1}]
  ];
list // Expand // TableForm
```

Out[3]//TableForm=

$\frac{1}{4}$	$1 + \frac{V}{3} + \frac{V^2}{12} + \frac{V^3}{36}$
$\frac{3}{10}$	$1 + \frac{V}{4} + \frac{V^2}{24} + \frac{V^3}{144} + \frac{V^4}{324}$
$\frac{9}{40}$	$1 + \frac{V}{5} + \frac{V^2}{40} + \frac{V^3}{360} + \frac{V^4}{1620} + \frac{V^5}{2916}$
$\frac{9}{70}$	$1 + \frac{V}{6} + \frac{V^2}{60} + \frac{V^3}{720} + \frac{V^4}{4860} + \frac{V^5}{17\,496} + \frac{V^6}{26\,244}$
$\frac{27}{448}$	$1 + \frac{V}{7} + \frac{V^2}{84} + \frac{V^3}{1260} + \frac{V^4}{11\,340} + \frac{V^5}{61\,236} + \frac{V^6}{183\,708} + \frac{V^7}{236\,196}$
$\frac{27}{1120}$	$1 + \frac{V}{8} + \frac{V^2}{112} + \frac{V^3}{2016} + \frac{V^4}{22\,680} + \frac{V^5}{163\,296} + \frac{V^6}{734\,832} + \frac{V^7}{1\,889\,568} + \frac{V^8}{2\,125\,764}$
$\frac{27}{3200}$	$1 + \frac{V}{9} + \frac{V^2}{144} + \frac{V^3}{3024} + \frac{V^4}{40\,824} + \frac{V^5}{367\,416} + \frac{V^6}{2\,204\,496} + \frac{V^7}{8\,503\,056} + \frac{V^8}{19\,131\,876} + \frac{V^9}{19\,131\,876}$
$\frac{81}{30\,800}$	$1 + \frac{V}{10} + \frac{V^2}{180} + \frac{V^3}{4320} + \frac{V^4}{68\,040} + \frac{V^5}{734\,832} + \frac{V^6}{5\,511\,240} + \frac{V^7}{28\,343\,520} + \frac{V^8}{95\,659\,380} + \frac{V^9}{191\,318\,760} + \frac{V^{10}}{172\,186\,884}$
$\frac{729}{985\,600}$	$1 + \frac{V}{11} + \frac{V^2}{220} + \frac{V^3}{5940} + \frac{V^4}{106\,920} + \frac{V^5}{1\,347\,192} + \frac{V^6}{12\,124\,728} + \frac{V^7}{77\,944\,680} + \frac{V^8}{350\,751\,060} + \frac{V^9}{1\,052\,253\,180} + \frac{V^{10}}{1\,894\,05}$

We notice that these match the following relations:

$$\mathbb{E}[T^2] = e^{-2\tau} \sum_{k=0}^{\infty} \frac{\left(c^k \left(1 - \frac{c}{1+K+k}\right)\right)}{(1+K)_k} \left(\sum_{n=0}^{K} \frac{\mathbb{E}[Y^2]^n}{(n!)^2 \binom{k+K}{n}} + \sum_{i=1}^{k} \frac{c^{-2i}\, \mathbb{E}[Y^2]^{K+i}}{(K!)^2 \binom{K+k}{K+i}} \right)$$

the variance follows ($\mathrm{Var}[T] = \mathbb{E}[T^2] - \mathbb{E}[T]^2$).

In[4]:=
```
maxk = 8;
listtest = With[{K = 3, c = 3},
   Table[
    {(c^k (1 - c/(1+K+k)))/Pochhammer[1 + K, k],
     Sum[V^n/(Binomial[k + K, n] (n!)^2), {n, 0, K}] +
      Sum[(c^(-2 i) V^(K+i))/((K!)^2 Binomial[k + K, K + i]), {i, 1, k}]
     }
    , {k, 0, maxk}]
  ];
listtest // TableForm
```

Out[6]//TableForm=

$\frac{1}{4}$	$1+\frac{V}{3}+\frac{V^2}{12}+\frac{V^3}{36}$
$\frac{3}{10}$	$1+\frac{V}{4}+\frac{V^2}{24}+\frac{V^3}{144}+\frac{V^4}{324}$
$\frac{9}{40}$	$1+\frac{V}{5}+\frac{V^2}{40}+\frac{V^3}{360}+\frac{V^4}{1620}+\frac{V^5}{2916}$
$\frac{9}{70}$	$1+\frac{V}{6}+\frac{V^2}{60}+\frac{V^3}{720}+\frac{V^4}{4860}+\frac{V^5}{17\,496}+\frac{V^6}{26\,244}$
$\frac{27}{448}$	$1+\frac{V}{7}+\frac{V^2}{84}+\frac{V^3}{1260}+\frac{V^4}{11\,340}+\frac{V^5}{61\,236}+\frac{V^6}{183\,708}+\frac{V^7}{236\,196}$
$\frac{27}{1120}$	$1+\frac{V}{8}+\frac{V^2}{112}+\frac{V^3}{2016}+\frac{V^4}{22\,680}+\frac{V^5}{163\,296}+\frac{V^6}{734\,832}+\frac{V^7}{1\,889\,568}+\frac{V^8}{2\,125\,764}$
$\frac{27}{3200}$	$1+\frac{V}{9}+\frac{V^2}{144}+\frac{V^3}{3024}+\frac{V^4}{40\,824}+\frac{V^5}{367\,416}+\frac{V^6}{2\,204\,496}+\frac{V^7}{8\,503\,056}+\frac{V^8}{19\,131\,876}+\frac{V^9}{19\,131\,876}$
$\frac{81}{30\,800}$	$1+\frac{V}{10}+\frac{V^2}{180}+\frac{V^3}{4320}+\frac{V^4}{68\,040}+\frac{V^5}{734\,832}+\frac{V^6}{5\,511\,240}+\frac{V^7}{28\,343\,520}+\frac{V^8}{95\,659\,380}+\frac{V^9}{191\,318\,760}+\frac{V^{10}}{172\,186\,884}$
$\frac{729}{985\,600}$	$1+\frac{V}{11}+\frac{V^2}{220}+\frac{V^3}{5940}+\frac{V^4}{106\,920}+\frac{V^5}{1\,347\,192}+\frac{V^6}{12\,124\,728}+\frac{V^7}{77\,944\,680}+\frac{V^8}{350\,751\,060}+\frac{V^9}{1\,052\,253\,180}+\frac{V^{10}}{1\,894\,05}$

We verify that they cancel:

In[7]:=
```
list - listtest // Expand // TableForm
```

Out[7]//TableForm=

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Johnson estimator: efficiency

Here we compare the efficiency of standard delta-tracking to the generalization that we propose, which follows from [Johnson 1951] where the Poisson process N(t) is sampled n times over the interval and those values are combined to produce a non-binary estimate for n > 1. This answers the question: *if you have n estimates of N(t), can you do better than the mean of n binary estimates*? The answer turns out to be: yes (sometimes).

The efficiency comparison requires the derivation by [Georgiev et al. 2019] for the mean number of lookups for the binary estimator that can terminate early at the first real collision. The mean cost of Johnson's $n > 1$ estimator is simply $M\, n\, \tau$. Here we will assume $M = 1$. The required variances are given in Appendix C, Eqs. (54) and (55).

To understand the relative efficiency, below we plot the ratio of the two efficiencies as a function of τ. This is completely general, as the variances do not depend on the specific variation in $\mu(x)$. We notice that Johnson's estimator is always more efficient than the simple binary estimator as n increases *provided τ is sufficiently large*. For small τ and small n, the early termination of the $n = 1$ estimator wins. The large increase in efficiency at large τ makes sense given that a binary estimator has a hard time estimating a very low transmittance accurately.

In[◦]:=
```
Show[
 LogPlot[(e^-τ (-1 + e^τ)^2)/((-1 + e^(τ/n)) n τ) /. n → {2, 4, 8, 16} // Evaluate,
  {τ, 0, 7}, Frame → True, PlotLabels → {"n=2", "n=4", "n=8", "n=16"},
  FrameLabel → {{"Eff[T_J]/Eff[T_dt]",}, {τ,}}],
 LogPlot[1, {x, 0, 7}, PlotStyle → Dashed]
]
```

Out[◦]=