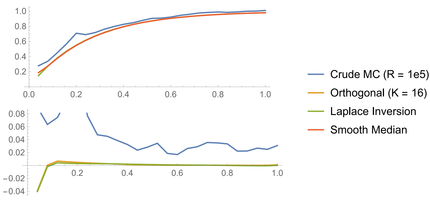
Test 5: Ruin probability: Poisson / Exponential.

```
(* Accuracy parameters *)
K = 16; R = 10^5; numPoints = 25;

(* Problem description *)
c = 1; λ = 4; UDist = ExponentialDistribution[4]; UDist = GammaDistribution[2, 2];

(* Time grid *)
tMax = 1;
dt = tMax / numPoints; ts = Table[i, {i, dt, tMax, dt}];

(* Orthogonal polynomial method *)
Verbosity = 0;

SLPs = Table[
   NDist = PoissonDistribution[λ t];

   r = λ t Mean[UDist]^2 / Moment[UDist, 2];
   m = Moment[UDist, 2] / Mean[UDist];

   SeedRandom[1];
   ps = PseudoGammaCoeffsIndirectly[NDist, UDist, r, m, K, Verbosity];

   OrthogonalSLP = Sum[ps[[i + 1]] (m (r + i) ℐu[r + i + 1, m, c t] - c t ℐu[r + i, m, c t]), {i, 0, K}];

   SeedRandom[1];
   CrudeSLP = First@Last@ReferenceCMC[NDist, UDist, {}, {c t}, R];

   svfStarI = Last@LaplaceInversionMethod[NDist, UDist];
   LapInvSLP = Mean[NDist] Mean[UDist] svfStarI[c t];

   {OrthogonalSLP, LapInvSLP, CrudeSLP}, {t, ts}];


OrthoRuins = 1/(c ts) (λ ts * Mean[UDist] - SLPs[[ ;; , 1]]) // N;
LapInvRuins = 1/(c ts) (λ ts * Mean[UDist] - SLPs[[ ;; , 2]]) // N;
CrudeRuins = 1/(c ts) (λ ts * Mean[UDist] - SLPs[[ ;; , 3]]) // N;


(***** Make the plots *****)
EstNames = {"Crude MC (R = 1e" <> ToString[Log10[R]] <> ")", "Orthogonal (K = " <> ToString[K] <> ")",
   "Laplace Inversion"};
AllRuins = {CrudeRuins, OrthoRuins, LapInvRuins};

RefName = "Smooth Median";
RefRuins = Table[Median[AllRuins[[ ;; , i]]], {i, Length[ts]}] // GaussianFilter[#, 2] &;

WithT[Data_] := Transpose[{ts, Data}];

RuinEst = ListPlot[WithT /@ Append[AllRuins, RefRuins], Joined → True, AspectRatio → 1 / 3];
RuinAbsErr = ListPlot[WithT[AbsErr[#, RefRuins]] & /@ AllRuins, Joined → True, AspectRatio → 1 / 3];
(*RuinRelErr=ListPlot[WithA[RelErr[#,RefRuins]]&/@AllRuins,Joined→True,AspectRatio→1/3];*)
RuinPlot = Legended[GraphicsColumn[{RuinEst, RuinAbsErr(*,RuinRelErr*)}],
  LineLegend[Colours, Append[EstNames, RefName]]]
Export["poisson_exponential_ruin.pdf", RuinPlot];
```



Test 6: Ruin probability: Poisson Pareto

```
(* Accuracy parameters *)
K = 16; R = 10^5; numPoints = 25;

(* Problem description *)
c = 1; λ = 2; a = 5; b = 11; θ = 0;
UDist = ParetoDistribution[a, b, θ];
LTofU[ŝ_] = Expectation[E^-ŝ X, X ≈ UDist];

(* Time grid *)
tMax = 5;
dt = tMax / numPoints; ts = Table[i, {i, 10^-1, tMax, dt}];

(* Orthogonal polynomial method *)
Verbosity = 0;

SLPs = Table[
   NDist = PoissonDistribution[λ t];

   (* Orthogonal polynomial method *)
   Tilt = 1;
   ThetaFracOfM = 1/2;
   mOrig = ThetaFracOfM / Tilt;

   r = Mean[UDist];
   m = mOrig / (1 - mOrig Tilt);

   (*Print["Want to take m > 1/2 Tilt i.e. ", mOrig, " > " ,1/(2 Tilt)];
   Print["Tilting by ", N@Tilt, " which gives a modified m of ", N@m];
   *)
   SeedRandom[1];
   ps = PseudoGammaCoeffsIndirectly[NDist, UDist, r, mOrig, K, Verbosity, LTofU, Tilt];

   OrthogonalSLP = Sum[ps[[i + 1]] (m (r + i) ℐu[r + i + 1, m, c t] - c t ℐu[r + i, m, c t]), {i, 0, K}];

   SeedRandom[1];
   CrudeSLP = First@Last@ReferenceCMC[NDist, UDist, {}, {c t}, R];

   svfStarI = Last@LaplaceInversionMethod[NDist, UDist, LTofU];
   (*LapInvSLP= Mean[NDist]Mean[UDist]svfStarI[c t];*)

   LapInvSLP = Quiet@Catch[Mean[NDist] Mean[UDist] svfStarI[c t], _SystemException];
   If[LapInvSLP === SystemException["MemoryAllocationFailure"], Print["Memory problem..."];
    LapInvSLP = Indeterminate];

   {OrthogonalSLP, LapInvSLP, CrudeSLP}, {t, ts}];


OrthoRuins = 1/(c ts) (λ ts * Mean[UDist] - SLPs[[ ;; , 1]]) // N;
LapInvRuins = 1/(c ts) (λ ts * Mean[UDist] - SLPs[[ ;; , 2]]) // N;
CrudeRuins = 1/(c ts) (λ ts * Mean[UDist] - SLPs[[ ;; , 3]]) // N;

(***** Make the plots *****)
EstNames = {"Crude MC (R = 1e" <> ToString[Log10[R]] <> ")", "Orthogonal (K = " <> ToString[K] <> ")",
   "Laplace Inversion"};
AllRuins = {CrudeRuins, OrthoRuins, LapInvRuins};

RefName = "Smooth Median";
RefRuins = Table[Median[DeleteCases[AllRuins[[ ;; , i]], Indeterminate]], {i, Length[ts]}] //
   GaussianFilter[#, 2] &;

WithT[Data_] := Transpose[{ts, Data}];

RuinEst = ListPlot[WithT /@ Append[AllRuins, RefRuins], Joined → True, AspectRatio → 1 / 3];
RuinAbsErr = ListPlot[WithT[AbsErr[#, RefRuins]] & /@ AllRuins, Joined → True, AspectRatio → 1 / 3];
(*RuinRelErr=ListPlot[WithA[RelErr[#,RefRuins]]&/@AllRuins,Joined→True,AspectRatio→1/3];*)
RuinPlot = Legended[GraphicsColumn[{RuinEst, RuinAbsErr(*,RuinRelErr*)}],
  LineLegend[Colours, Append[EstNames, RefName]]]
Export["poisson_pareto_ruin.pdf", RuinPlot];
```

```
Memory problem...
```