Geometric complexity comparison of Rogers' random predator equations
(Holling II and III with no prey replacement).

In the main analysis we assume a Poisson likelihood corresponding to experimental designs where prey are continually replaced. Here we consider a Binomial likelihood corresponding to designs where depletion of prey occurs. We focus on the type II and type III functional forms, hence use Rogers' corresponding random predator equations, which are (among) the most commonly assessed functional-response models in the literature.

Preliminaries

In[•]:=
```
Clear["Global`*"]

AbundSeries = "GoldenRatio";
SetDirectory[
  FileNameJoin[{NotebookDirectory[], "../results/GoldenRatio_binom/"}]];
```

Define functions to compute fisher matrix and expected fisher matrix

Definition using Hessian (applies under mild regularity conditions):

In[•]:=
```
fisher[fun_, pars_] := -D[Apply[fun, pars], {pars, 2}]
```

Expected Fisher Matrix given experimental design (*unused*)

In[•]:=
```
EFisher[FisherMatrix_, func_, PDistE_, NDistE_, subs_] :=
 Expectation[
  Expectation[
   FisherMatrix /. subs, x ≈ BinomialDistribution[N, func /. subs]],
  {P ≈ PDistE, N ≈ NDistE}]
```

Maximum and minimum limits on expected *proportion* of prey eaten

In[•]:=
```
SqrtDetEFlim[func_, Det_, Nvals_, Pvals_, subs_] :=
  Piecewise[
   {{Sqrt[Det],
     Max[func /. subs /. P → Pvals /. N → Nvals] ≤ 1 &&
      Min[func /. subs /. P → Pvals /. N → {Max[Nvals]}] ≥ 1 / Max[Nvals]
   }}];
```

Define functional responses - *proportion* of prey eaten (N_e / N_o)

```
In[•]:= RogersH2 = 1 - 1/(a b N) ProductLog[a b N Exp[-a (P T - b N)]];
        (* Using Citardauq *)
        RogersH3 = 2 a N P T / ((1 + a N P T + a b N^2) + Sqrt[(1 + a N P T + a b N^2)^2 - 4 a^2 b N^3 P T]);
        (* Kratina et al.2009 and Okuyama & Ruyle 2011 *)
        RogersH3a = ((1 + a N P T + a b N^2) - Sqrt[(1 + a N P T + a b N^2)^2 - 4 a^2 b N^3 P T]) / (2 a b N^2);
```

Define Likelihood functions

```
In[•]:= BinomlL[func_] := Log[Factorial[N] / (Factorial[x] Factorial[N - x])] + x Log[func] + (N - x) Log[1 - func]

        lLRogersH2[a_, b_] := BinomlL[RogersH2]
        lLRogersH3[a_, b_] := BinomlL[RogersH3]
```

Define wrapper functions

Geometric Complexity function

```
In[•]:= ClearAll[GeomComplex]
        GeomComplex[
          Nvalues_,
          Pvalues_,
          Model_] :=
         Module[
          {
           Nvals = Nvalues,
           Pvals = Pvalues,
           Tval = 1,
           NIntMethod = {"LocalAdaptive", "SingularityHandler" → Automatic},
           minRec = 150,
           maxRec = 500,
           accgoal = 3,
           precgoal = 3
          },
          Nprobs = ConstantArray[1 / Length[Nvals], Length[Nvals]];
```

```
 Pprobs = ConstantArray[1 / Length[Pvals], Length[Pvals]];
 NDistE = EmpiricalDistribution[Nprobs → Nvals];
 PDistE = EmpiricalDistribution[Pprobs → Pvals];
 subs = {T → Tval};

 ParmRange = {
   {a, 0, Infinity},
   {b, 0, Infinity}
  };

 Which[
  Model == "RogersH2",
  DetRogersH2 =
   Det[EFisher[fisher[lLRogersH2, {a, b}], RogersH2, PDistE, NDistE, subs]];
  NIntRogersH2 =
   Log[
    NIntegrate[
     SqrtDetEFlim[RogersH2, DetRogersH2, Nvals, Pvals, subs],
     ParmRange[[1]],
     ParmRange[[2]],
     AccuracyGoal → accgoal,
     PrecisionGoal → precgoal,
     Method → NIntMethod,
     MinRecursion → minRec,
     MaxRecursion → maxRec]]
  ,
  Model == "RogersH3",
  DetRogersH3 =
   Det[EFisher[fisher[lLRogersH3, {a, b}], RogersH3, PDistE, NDistE, subs]];
  NIntRogersH3 =
   Log[
    NIntegrate[
     SqrtDetEFlim[RogersH3, DetRogersH3, Nvals, Pvals, subs],
     ParmRange[[1]],
     ParmRange[[2]],
     AccuracyGoal → accgoal,
     PrecisionGoal → precgoal,
     Method → NIntMethod,
     MinRecursion → minRec,
     MaxRecursion → maxRec]]
 ]
]
```

Define functions to create experimental designs

The “GoldenRatio” series of abundance levels (starting at 3 for prey and 1 for predators) enabling the generation of arbitrary numbers of equidistant points between min and (specified) max abundances.
The series will be approximately Fibonacci when PreyMax is set to *Fibonacci[n]* for a desired series length *n*.
The “Arithmetic series” spaces out levels equidistantly in arithmetic space to the same maximum as used for the “GoldenRatio” series.

In[•]:=
```
logGRSpace[a_, b_, n_] := Round[GoldenRatio^Range[a, b, (b - a) / (n - 1)]]

PreyVals[n_, PreyMax_, AbundSeries_] :=
 Which[
  AbundSeries == "GoldenRatio",
  logGRSpace[2, Log[GoldenRatio, PreyMax] + 3, n],
  AbundSeries == "Arithmetic",
  Round[Range[3, Max[logGRSpace[2, Log[GoldenRatio, PreyMax] + 3, n]],
    (Max[logGRSpace[2, Log[GoldenRatio, PreyMax] + 3, n]] - 3) / (n - 1)]]
 ]
PredVals[n_, PredMax_, AbundSeries_] :=
 If[n == 1,
  {1}, (* If only a single level is requested,
  specify a single predator individual *)
  Which[ (* Otherwise, determine predator levels according to GoldenRatio
    or Arithmetic series beginning with 1 predator individual *)
   AbundSeries == "GoldenRatio",
   logGRSpace[0, Log[GoldenRatio, PredMax] + 1, n],
   AbundSeries == "Arithmetic",
   Round[Range[1, Max[logGRSpace[0, Log[GoldenRatio, PredMax] + 1, n]],
     (Max[logGRSpace[0, Log[GoldenRatio, PredMax] + 1, n]] - 1) / (n - 1)]]
  ]]
```

Specify AbundanceSeries and export designs

Be sure to copy numbers to GeomComp[] function below.

In[•]:=
```
PreyMinLevels = 5;
PreyMaxLevelsVar = 10;
PreyMaxLevelsFix = 10;
PredMinLevels = 1;
PredMaxLevelsVar = 5;
PredMaxLevelsFix = 4;
```

Export "Var" designs - increasing length and increasing maximum value

In[•]:=
```
VarDesigns = Table[
    {PreyVals[i, Fibonacci[i], AbundSeries],
     PredVals[j, Fibonacci[j], AbundSeries]},
    {i, PreyMinLevels, PreyMaxLevelsVar},
    {j, PredMinLevels, PredMaxLevelsVar}
   ];
TableForm[VarDesigns]
Dimensions[VarDesigns]
```

Out[•]//TableForm=

3 4 7 13 21 1	3 4 7 13 21 1 2	3 4 7 13 21 1 2 3
3 4 7 12 20 34 1	3 4 7 12 20 34 1 2	3 4 7 12 20 34 1 2 3
3 4 7 12 20 33 55 1	3 4 7 12 20 33 55 1 2	3 4 7 12 20 33 1 2 3
3 4 7 12 20 32 54 89 1	3 4 7 12 20 32 54 89 1 2	3 4 7 12 20 32 1 2 3
3 4 7 12 19 32 53 87 144 1	3 4 7 12 19 32 53 87 144 1 2	3 4 7 12 19 32 1 2 3
3 4 7 12 19 32 52 86 141 233 1	3 4 7 12 19 32 52 86 141 233 1 2	3 4 7 12 19 32 1 2 3

Out[•]= {6, 5, 2}

Export "Fix" designs - varying length and constant maximum value

In[•]:=
```
FixDesigns = Table[
   {PreyVals[i, Fibonacci[PreyMaxLevelsFix], AbundSeries],
    PredVals[j, Fibonacci[PredMaxLevelsFix], AbundSeries]},
   {i, PreyMinLevels, PreyMaxLevelsFix},
   {j, PredMinLevels, PredMaxLevelsFix}
  ];
TableForm[FixDesigns]
Dimensions[FixDesigns]
```

Out[•]//TableForm=

3 8 25 76 233 1	3 8 25 76 233 1 5	3 8 25 76 233 1 2 5
3 6 16 39 95 233 1	3 6 16 39 95 233 1 5	3 6 16 39 95 2 1 2 5
3 6 12 25 52 110 233 1	3 6 12 25 52 110 233 1 5	3 6 12 25 52 1 1 2 5
3 5 9 18 34 65 123 233 1	3 5 9 18 34 65 123 233 1 5	3 5 9 18 34 65 1 2 5
3 5 8 14 25 43 76 133 233 1	3 5 8 14 25 43 76 133 233 1 5	3 5 8 14 25 43 1 2 5
3 4 7 12 19 32 52 86 141 233 1	3 4 7 12 19 32 52 86 141 233 1 5	3 4 7 12 19 32 1 2 5

Out[•]= {6, 4, 2}

Define GeomComp[] to apply GeomComplex[] across experimental designs

In[•]:=
```
ClearAll[GeomComp];
GeomComp[
  ModelAbb_,
  Type_,
  AbundSeries_] :=
 Module[
  {
   (******* Be sure to match the following with exported designs above *******)
   PreyMinLevels = 5,
   PreyMaxLevelsVar = 10,
   PreyMaxLevelsFix = 10,
   PredMinLevels = 1,
   PredMaxLevelsVar = 5,
   PredMaxLevelsFix = 4
  }
  ,
  Which[
   Type == "Var",
   Flatten[
```

```
   ParallelTable[
    Flatten[{
      Max[PreyVals[i, Fibonacci[i], AbundSeries]], (* Maximum prey level *)
      Max[PredVals[j, Fibonacci[j], AbundSeries]], (* Maximum pred level *)
      Length[PreyVals[i, Fibonacci[i], AbundSeries]],
      (* Number of prey levels *)
      Length[PredVals[j, Fibonacci[j], AbundSeries]],
      (* Number of pred levels *)
      GeomComplex[
       PreyVals[i, Fibonacci[i], AbundSeries],
       PredVals[j, Fibonacci[j], AbundSeries],
       Model = ModelAbb]
     }],
    {i, PreyMinLevels, PreyMaxLevelsVar},
    {j, PredMinLevels, PredMaxLevelsVar}
   ],
   1]
  ,
  Type == "Fix",
  Flatten[
   ParallelTable[
    Flatten[{
      Max[PreyVals[i, Fibonacci[PreyMaxLevelsFix], AbundSeries]],
      (* Maximum prey level *)
      Max[PredVals[j, Fibonacci[PredMaxLevelsFix], AbundSeries]],
      (* Maximum pred level *)
      Length[PreyVals[i, Fibonacci[PreyMaxLevelsFix], AbundSeries]],
      (* Number of prey levels *)
      Length[PredVals[j, Fibonacci[PredMaxLevelsFix], AbundSeries]],
      (* Number of pred levels *)
      GeomComplex[
       PreyVals[i, Fibonacci[PreyMaxLevelsFix], AbundSeries],
       PredVals[j, Fibonacci[PredMaxLevelsFix], AbundSeries],
       Model = ModelAbb]
     }],
    {i, PreyMinLevels, PreyMaxLevelsFix},
    {j, PredMinLevels, PredMaxLevelsFix}
   ],
   1]
 ]
]
```

Test

```
In[•]:= (* GeomComp[Model="RogersH3",Type="Var",AbundSeries=AbundSeries] *)
```

Apply across designs

```
In[•]:= varRogersH2 = GeomComp[Model = "RogersH2", Type = "Var", AbundSeries = AbundSeries];
        DeleteFile["k2varRogersH2.txt"]; Save["k2varRogersH2.txt", varRogersH2]

In[•]:= fixRogersH2 = GeomComp[Model = "RogersH2", Type = "Fix", AbundSeries = AbundSeries];
        DeleteFile["k2fixRogersH2.txt"]; Save["k2fixRogersH2.txt", fixRogersH2]

In[•]:=


In[•]:= varRogersH3 = GeomComp[Model = "RogersH3", Type = "Var", AbundSeries = AbundSeries];
        DeleteFile["k2varRogersH3.txt"]; Save["k2varRogersH3.txt", varRogersH3]

In[•]:= fixRogersH3 = GeomComp[Model = "RogersH3", Type = "Fix", AbundSeries = AbundSeries];
        DeleteFile["k2fixRogersH3.txt"]; Save["k2fixRogersH3.txt", fixRogersH3]
```