



Mathematical Modelling and Scientific Computing in the Biosciences

| 20 March 2007

Lecture 2: Overview

Topics

- **Mathematica Preliminaries**
- **Mass-Action Kinetics**
- **Michaelis-Menten Kinetics**
- **Perturbation Analysis:**

–regular perturbation
–singular perturbation

2 of 24

Mathematica Preliminaries

List: collection of general expressions [{...}](#)

```
In[8]:= Table[f[i], {i, 1, 3}]
```

```
Out[8]= {f[1], f[2], f[3]}
```

Replacement rule: transforming each subpart of an expression [/.](#)

```
In[1]:= a /. {a → b}
```

```
Out[1]= b
```

```
In[2]:= f[a] /. {a → b}
```

```
Out[2]= f[b]
```

```
In[14]:= a == b /. Equal → Less
```

```
Out[14]= a < b
```

Pattern: for representing forms of Mathematica expressions [x_](#)

```
In[15]:= {ab, cd, ef} /. {x_y_ → yx}
```

```
Out[15]= {ba, dc, fe}
```

```
In[16]:= {a*b == c, c*d == e, e*f == g} /. {x_*y_ → y}
```

```
Out[16]= {b == c, d == e, f == g}
```

3 of 24

Mathematica Preliminaries

Map: apply a function to each element of an expression [Map\[Function, List\]](#)

```
In[12]:= Map[Sin, {a, b, c}]
```

```
Out[12]= {Sin[a], Sin[b], Sin[c]}
```

```
In[7]:= Map[2^# &, {a, b, c}]
```

```
Out[7]= {2^a, 2^b, 2^c}
```

Pick: take elements out of a list, according to some criterion [Pick\[List, Criterion\]](#)

```
In[23]:= NumberList = {2, 3, 4};
Map[# > 2 &, NumberList]
Pick[NumberList, %]
```

```
Out[24]= {False, True, True}
```

```
Out[25]= {3, 4}
```

Solve: solve system of algebraic equations, for selected variables [Solve\[LHS==RHS, SolvedVariables\]](#)

```
In[9]:= Solve[a x^2 + b x + c == 0, x]
```

```
Out[9]= {{x ->  $\frac{-b - \sqrt{b^2 - 4 a c}}{2 a}$ }, {x ->  $\frac{-b + \sqrt{b^2 - 4 a c}}{2 a}$ }}
```



Mass–Action Kinetics

- What is a **flux**?

flux of a chemical species is the rate of change in the **concentration** per unit **time**

Units of **concentration**: mole/liter $\approx 6.022 * 10^{23}$ entities/liter, ...

Units of **time**: second, minute

\therefore Units of **flux**: mole/(liter * second), ...

Mathematical expression for expressing the **flux** of species A: $\frac{d[A]}{dt}$

- Rate laws: how does the **flux** depend on the concentration of the **reactants**?



3 of 4

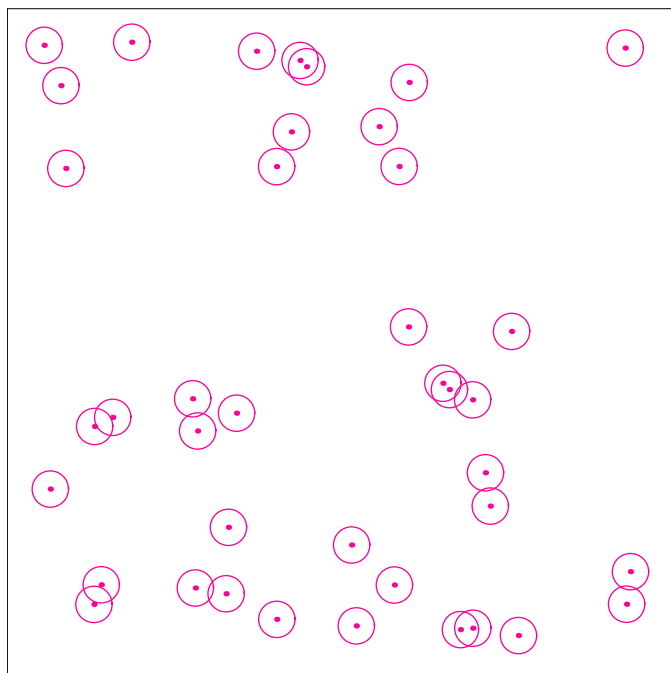
Mass–Action Kinetics

- **Mass–action rate law:** reaction flux is proportional to the probability of finding reacting molecules in a small volume.

```
In[906]:=
NumMolecule1 = 40;
Molecule1Location = Table[{Random[], Random[]}, {i, 1, NumMolecule1}];

PlotCircleRadius = 0.03;
PlotMolecule[Color_, LocationList_] :=
  Show[Graphics[Map[{Color, Circle[#, PlotCircleRadius], Point[#]} &, LocationList]] // Flatten,
  AspectRatio → Automatic, PlotRange → Table[{0 - PlotOffMargin, 1 + PlotOffMargin}, {2}],
  DisplayFunction → Identity];
PlotOffMargin = 0.05;

Plot1 = PlotMolecule[Hue[0.9], Molecule1Location];
Show[Plot1, DisplayFunction → $DisplayFunction, ImageSize → {250, 250}, Frame → True, FrameTicks → None];
```



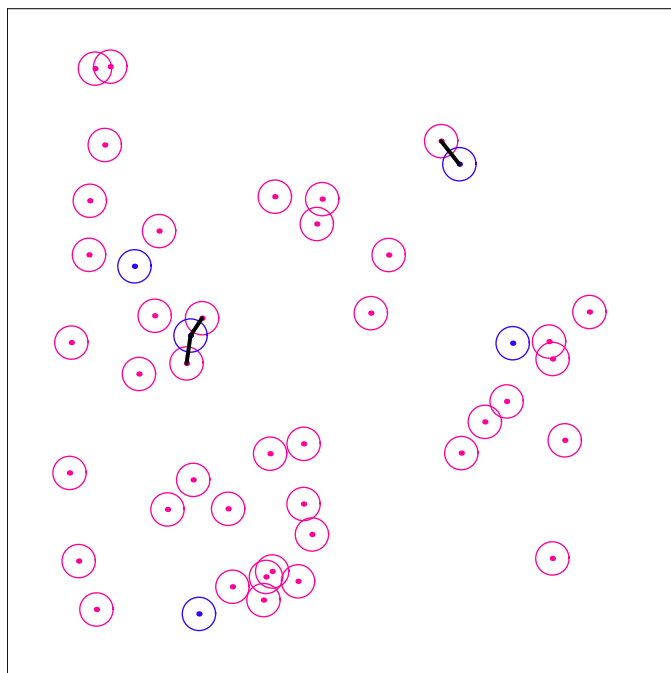
Mass–Action Kinetics

- Mass–action rate law: reaction flux is proportional to the probability of finding reacting molecules in a small volume.

```
In[835]:=
NumMolecule2 = 5;
Molecule2Location = Table[{Random[], Random[]}, {i, 1, NumMolecule2}];

(* Plotting intersection lines *)
DistanceList = Tuples[{Molecule1Location, Molecule2Location}];
PickedElements = Map[Norm[#[[1]] - #[[2]]] < PlotCircleRadius * 2 &, DistanceList];
LineIntersection =
  Show[Graphics[Map[{Thickness[0.006], Line[#]} &, Pick[DistanceList, PickedElements]]], DisplayFunction -> Identity];

Plot2 = PlotMolecule[Hue[0.7], Molecule2Location];
PlotCombinedSpecies_Density5 = Show[{Plot1, Plot2, LineIntersection},
  DisplayFunction -> $DisplayFunction, ImageSize -> {250, 250}, Frame -> True, FrameTicks -> None];
```



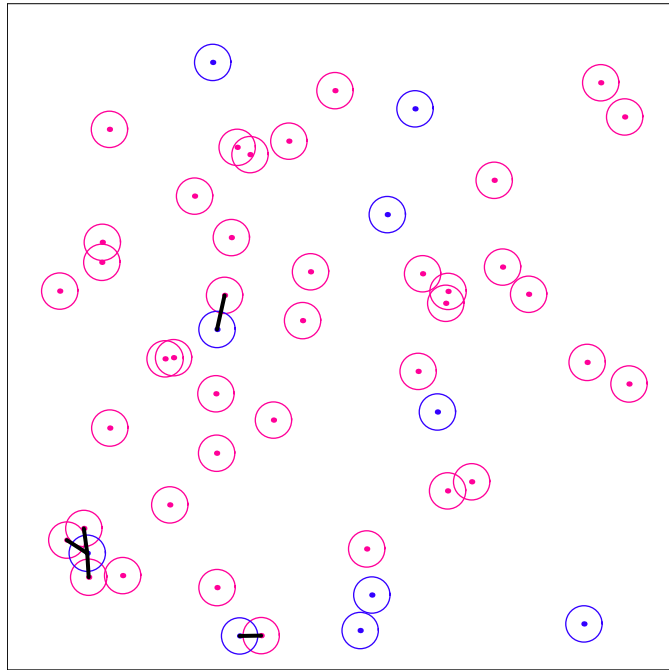
Mass–Action Kinetics

- Mass–action rate law: reaction flux is proportional to the probability of finding reacting molecules in a small volume.

```
In[863]:=
NumMolecule2 = 10;
Molecule2Location = Table[{Random[], Random[]}, {i, 1, NumMolecule2}];

(* Plotting intersection lines *)
DistanceList = Tuples[{Molecule1Location, Molecule2Location}];
PickedElements = Map[Norm#[[1]] - #[[2]] < PlotCircleRadius * 2 &, DistanceList];
LineIntersection =
  Show[Graphics[Map[{Thickness[0.006], Line[#]} &, Pick[DistanceList, PickedElements]]], DisplayFunction -> Identity];

Plot2 = PlotMolecule[Hue[0.7], Molecule2Location];
PlotCombinedSpecies_Density10 = Show[{Plot1, Plot2, LineIntersection},
  DisplayFunction -> $DisplayFunction, ImageSize -> {250, 250}, Frame -> True, FrameTicks -> None];
```



Mass-Action Kinetics

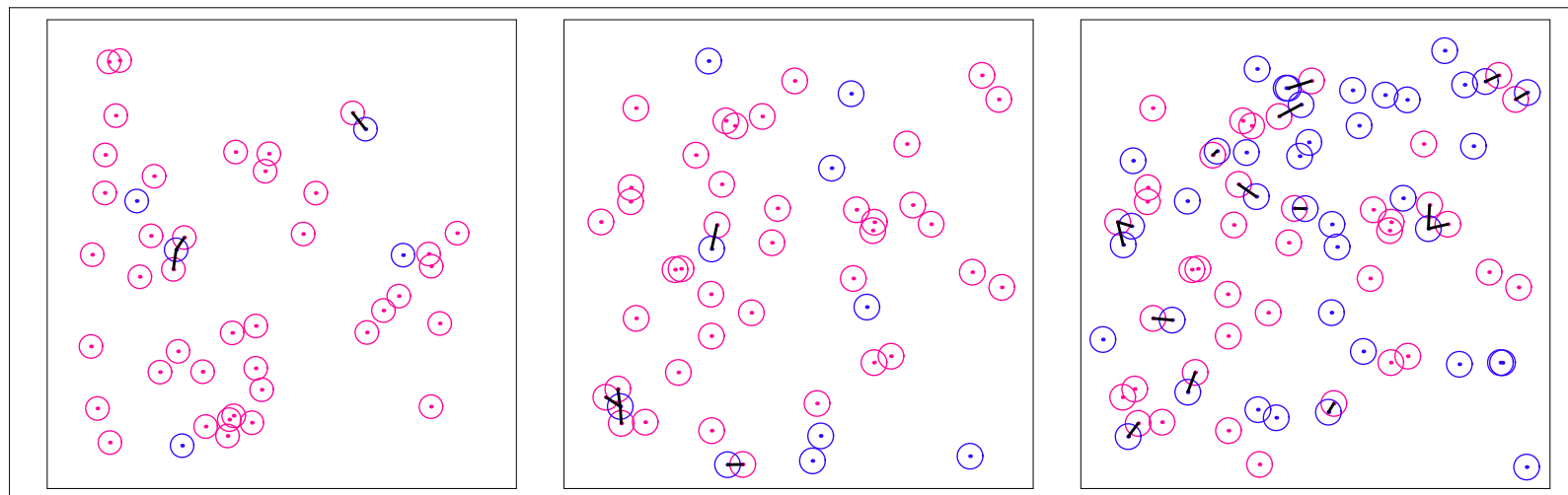
- Mass-action rate law: reaction flux is proportional to the probability of finding reacting molecules in a small volume.

```
In[870]:=
NumMolecule2 = 40;
Molecule2Location = Table[{Random[], Random[]}, {i, 1, NumMolecule2}];

(* Plotting intersection lines *)
DistanceList = Tuples[{Molecule1Location, Molecule2Location}];
PickedElements = Map[Norm#[[1]] - #[[2]] < PlotCircleRadius * 2 &, DistanceList];
LineIntersection =
  Show[Graphics[Map[{Thickness[0.006], Line[#]} &, Pick[DistanceList, PickedElements]], DisplayFunction -> Identity];

Plot2 = PlotMolecule[Hue[0.7], Molecule2Location];
PlotCombinedSpecies_Density40 = Show[{Plot1, Plot2, LineIntersection},
  DisplayFunction -> Identity, ImageSize -> {300, 300}, Frame -> True, FrameTicks -> None];

Show[GraphicsArray[{PlotCombinedSpecies_Density5, PlotCombinedSpecies_Density10, PlotCombinedSpecies_Density40}],
  DisplayFunction -> $DisplayFunction, ImageSize -> {200 * 3, 200}, Frame -> True, FrameTicks -> None];
```



Mass–Action Kinetics

Rate law:

"The rate of the **single** chemical reaction ($A+B \rightarrow C$) is directly proportional to the product of the concentrations of the participating species, A and B"

↕

$$\frac{d[A]}{dt} = \frac{d[B]}{dt} = -k_{AB} * [A] * [B]$$

$$\frac{d[C]}{dt} = k_{AB} * [A] * [B]$$

"The rate of the **reversible** chemical reaction ($A+B \rightleftharpoons C+D$) is directly proportional to the product of the concentrations of the participating species, A and B"

↕

$$\frac{d[A]}{dt} = \frac{d[B]}{dt} = -k_{AB} * [A] * [B] + k_{CD} * [C] * [D]$$

$$\frac{d[C]}{dt} = \frac{d[D]}{dt} = k_{AB} * [A] * [B] - k_{CD} * [C] * [D]$$

Chemical Equilibrium:

• Forward rate = backward rate, i.e., $-k_{AB} * [A] * [B] + k_{CD} * [C] * [D] = 0$

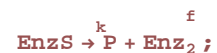
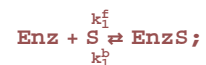
• Characterizing equilibrium solutions: **equilibrium constant** $\equiv \frac{[C][D]}{[A][B]} = k_{AB}/k_{CD}$

```
In[7]:= EquilCondition = (-KAB * Conc[A] * Conc[B] + KCD * Conc[C] * Conc[D] == 0);
ConcSoln = Solve[EquilCondition, Conc[C]] // Flatten;
Conc[C] * Conc[D] / (Conc[A] * Conc[B]) /. ConcSoln
```

```
Out[9]=  $\frac{KAB}{KCD}$ 
```

Michaelis–Menten Kinetics

Conversion of substrate (S) to product (P), via enzyme–substrate (EnzS) formation:



Get the equations using the Cellerator™ package (automatic equation generation for biological modelling; www.cellerator.org)

```
In[370]:=
  << cellerator.m;

  Cellerator™ 1.5.8 (1-July-2005) loaded 20-Mar-2007 11:28 using Mathematica Version 5.2 for Linux (June 20, 2005)

In[371]:=
  reaction1 = {{Enz + S ⇌ EnzS, k1f, k1b}};
  (reaction1ODE = interpret[reaction1] // First) // ColumnForm

Out[372]=
  Enz'[t] == k1b EnzS[t] - k1f Enz[t] S[t]
  EnzS'[t] == -k1b EnzS[t] + k1f Enz[t] S[t]
  S'[t] == k1b EnzS[t] - k1f Enz[t] S[t]

In[373]:=
  reaction2 = {{EnzS → P + Enz, k2f}};
  (reaction2ODE = interpret[reaction2] // First) // ColumnForm

Out[374]=
  Enz'[t] == k2f EnzS[t]
  EnzS'[t] == -k2f EnzS[t]
  P'[t] == k2f EnzS[t]
```



Michaelis–Menten Kinetics

Lets check **conservation relations**:

```
In[524]:=
  EnzTotal[t] = Enz[t] + EnzS[t];
  TimeDerivEnzTotal = D[EnzTotal[t], t];
  reaction2Rule = reaction2ODE /. {Equal → Rule}

Out[526]=
  {Enz'[t] → k2f EnzS[t], EnzS'[t] → -k2f EnzS[t], P'[t] → k2f EnzS[t]}

In[527]:=
  TimeDerivEnzTotal /. reaction2Rule

Out[527]=
  0

In[528]:=
  STotal[t] = S[t] + EnzS[t] + P[t];
  TimeDerivSTotal = D[STotal[t], t];
  reactionsCombined = ({reaction1 // First, reaction2 // First} // interpret // First) /. {Equal → Rule};
  (TimeDerivSTotal /. reactionsCombined) == 0

Out[531]=
  True
```

4 species, 2 conservation relations, therefore obtain 2 independent ODES:

```
In[532]:=
  Clear[EnzTotal];
  EnzTotal[t_] := EnzTotal[0];
  EnzReplaceRule = Solve[EnzTotal[t] == Enz[t] + EnzS[t], Enz[t]] // Flatten;
  SelectedVar = {EnzS'[t], S'[t]};
  TwoDimEnzymeODE = Pick[reactionsCombined, Map[MemberQ[SelectedVar, #] &,
    Map#[[1]] &, reactionsCombined]] /. EnzReplaceRule /. {Rule → Equal};
  % // ColumnForm

Out[537]=
  EnzS'[t] == -k1b EnzS[t] - k2f EnzS[t] + k1f (-EnzS[t] + EnzTotal[0]) S[t]
  S'[t] == k1b EnzS[t] - k1f (-EnzS[t] + EnzTotal[0]) S[t]
```

Michaelis–Menten Kinetics

```
In[538]:=
STotal[t_] := STotal[0];
TransformEqn = {Ŝ[t] == S[t] / STotal[0], EnzŜ[t] == EnzS[t] / EnzTotal[0]};
TransformVariables = Solve[%, SelectedVar /. {Derivative[1][Species_][t_] → Species[t]}] // Flatten;
ODETransformRule = Append[%, Map[D#[[1]], t] → D#[[2]], t] &, %] // Flatten

Out[541]=
{S[t] → STotal[0] Ŝ[t], EnzS[t] → EnzTotal[0] EnzŜ[t], S'[t] → STotal[0] Ŝ'[t], EnzS'[t] → EnzTotal[0] EnzŜ'[t]}
```

```
In[542]:=
ModTwoDimEnzymeODE = Simplify[TwoDimEnzymeODE /. %, Assumptions → {STotal[0] > 0, EnzTotal[0] > 0}];
% // ColumnForm

Out[543]=
EnzŜ[t] (k1b + k2f + k1f STotal[0] Ŝ[t]) + EnzŜ'[t] == k1f STotal[0] Ŝ[t]
EnzTotal[0] EnzŜ[t] (k1b + k1f STotal[0] Ŝ[t]) == STotal[0] (k1f EnzTotal[0] Ŝ[t] + Ŝ'[t])

In[544]:=
TimeChainRule = {t → t̂,
  Derivative[1][Species_][t] → Derivative[1][Species][t̂] * (EnzTotal[0] * k1f)};
ModTwoDimEnzymeODE = Simplify[ModTwoDimEnzymeODE /. TimeChainRule, Assumptions → {STotal[0] > 0, EnzTotal[0] > 0}] // .
{x_[temp_] → x[t̂]}

Out[545]=
{EnzŜ[t̂] (k1b + k2f + k1f STotal[0] Ŝ[t̂]) + k1f EnzTotal[0] EnzŜ'[t̂] == k1f STotal[0] Ŝ[t̂],
  EnzŜ[t̂] (k1b + k1f STotal[0] Ŝ[t̂]) == k1f STotal[0] (Ŝ[t̂] + Ŝ'[t̂])}

In[546]:=
ParamEnzymeODE = Simplify[ModTwoDimEnzymeODE /. {EnzTotal[0] → e STotal[0]}, Assumptions → {e > 0, k1f > 0}];
% // ColumnForm

Out[547]=
EnzŜ[t̂] (k1b + k2f + k1f STotal[0] Ŝ[t̂]) + k1f e STotal[0] EnzŜ'[t̂] == k1f STotal[0] Ŝ[t̂]
EnzŜ[t̂] (k1b + k1f STotal[0] Ŝ[t̂]) == k1f STotal[0] (Ŝ[t̂] + Ŝ'[t̂])
```

Michaelis–Menten Kinetics: Asymptotic Analysis

```
In[548]:=
ZeroOrderApprox = ParamEnzymeODE /. { $\epsilon \rightarrow 0$ }
```

```
Out[548]=
{EnzS[t] (k1b + k2f + k1f STotal[0]  $\hat{S}[t]$ ) == k1f STotal[0]  $\hat{S}[t]$ , EnzS[t] (k1b + k1f STotal[0]  $\hat{S}[t]$ ) == k1f STotal[0] ( $\hat{S}[t] + \hat{S}'[t]$ )}
```

```
In[549]:=
EnzSEquilibrium = Solve[#[[1]], EnzS[t]] // Flatten
```

```
Out[549]=
{EnzS[t]  $\rightarrow \frac{k1f STotal[0] \hat{S}[t]}{k1b + k2f + k1f STotal[0] \hat{S}[t]}$ }
```

```
In[550]:=
ReducedODE = ZeroOrderApprox[[2]] /. EnzSEquilibrium
```

```
Out[550]=
 $\frac{k1f STotal[0] \hat{S}[t] (k1b + k1f STotal[0] \hat{S}[t])}{k1b + k2f + k1f STotal[0] \hat{S}[t]} = k1f STotal[0] (\hat{S}[t] + \hat{S}'[t])$ 
```

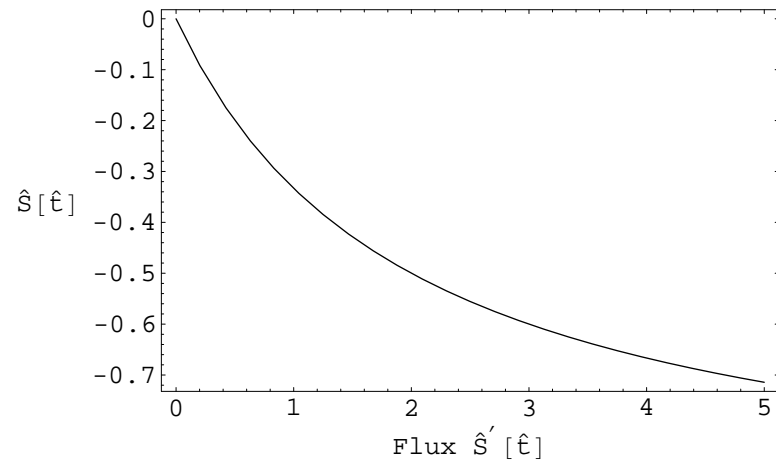
```
In[551]:=
MMKinetic = ReducedODE // Simplify[#, Assumptions  $\rightarrow \{\hat{S}[t] > 0\}$ ] & // solve[#,  $\hat{S}'[t]$ ] & // Flatten
```

```
Out[551]=
{ $\hat{S}'[t] \rightarrow -\frac{k2f \hat{S}[t]}{k1b + k2f + k1f STotal[0] \hat{S}[t]}$ }
```

```
In[552]:=
(#[[1, 2]] // Numerator) / ((#[[1, 2]] // Denominator) / (k1f STotal[0]) // Expand)
```

```
Out[552]=
 $-\frac{k2f \hat{S}[t]}{\frac{k1b}{k1f STotal[0]} + \frac{k2f}{k1f STotal[0]} + \hat{S}[t]}$ 
```

```
In[553]:=
 $\hat{S}'[\hat{t}] /. \text{MMKinetic} /. \{k1b \rightarrow 1, k1f \rightarrow 1, k2f \rightarrow 1, \text{STotal}[0] \rightarrow 1\};$ 
Plot[%, { $\hat{S}[\hat{t}]$ , 0, 5}, PlotRange -> All, Frame -> True, FrameLabel -> {"Flux  $\hat{S}'[\hat{t}]$ ", " $\hat{S}[\hat{t}]$ "}, RotateLabel -> False];
```



14 of 24

Michaelis–Menten Kinetics: Asymptotic Analysis

```
Out[441]=
```

$$-\frac{k2f \hat{S}[\hat{t}]}{\frac{k1b}{k1f \text{STotal}[0]} + \frac{k2f}{k1f \text{STotal}[0]} + \hat{S}[\hat{t}]}$$

Changing the Michaelis constant, $\frac{k_{1b}}{k_{1f} S_{Total}[0]} + \frac{k_{2f}}{k_{1f} S_{Total}[0]}$

In[519]:=

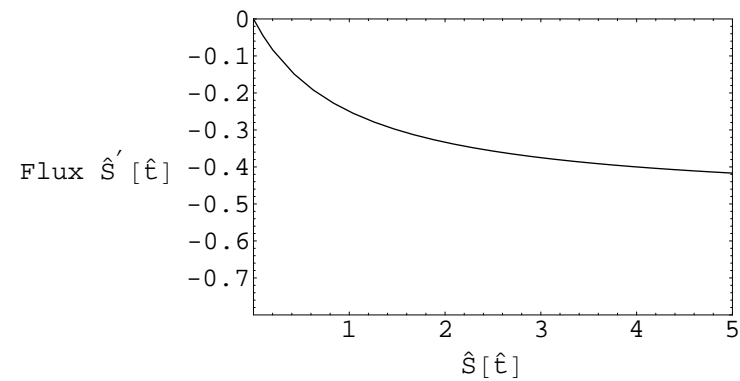
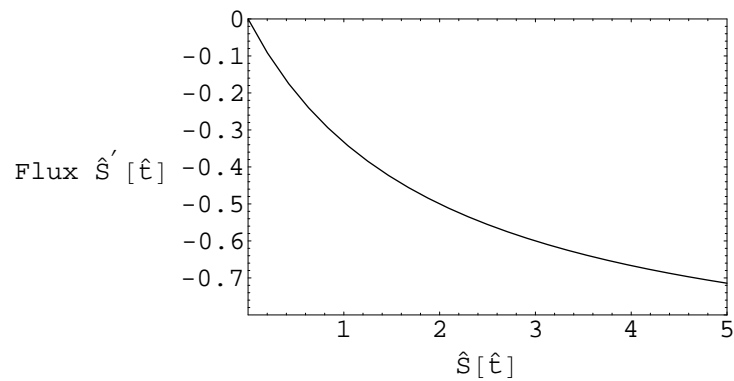
```

 $\hat{S}'[\hat{t}] /. \text{MMKinetic} /. \{k_{1b} \rightarrow 1, k_{1f} \rightarrow 1, k_{2f} \rightarrow 1, S_{Total}[0] \rightarrow 1\};$ 
Plot1 = Plot[%, { $\hat{S}[\hat{t}]$ , 0, 5}, PlotRange -> {{0, 5}, {-0.8, 0}}, Frame -> True,
  FrameLabel -> {" $\hat{S}[\hat{t}]$ ", "Flux  $\hat{S}'[\hat{t}]$ "}, RotateLabel -> False, DisplayFunction -> Identity];

 $\hat{S}'[\hat{t}] /. \text{MMKinetic} /. \{k_{1b} \rightarrow 1, k_{1f} \rightarrow 1, k_{2f} \rightarrow 1, S_{Total}[0] \rightarrow 2\};$ 
Plot2 = Plot[%, { $\hat{S}[\hat{t}]$ , 0, 5}, PlotRange -> {{0, 5}, {-0.8, 0}}, Frame -> True,
  FrameLabel -> {" $\hat{S}[\hat{t}]$ ", "Flux  $\hat{S}'[\hat{t}]$ "}, RotateLabel -> False, DisplayFunction -> Identity];

Show[GraphicsArray[{{Plot1, Plot2}}], ImageSize -> {600, 200}, DisplayFunction -> $DisplayFunction];

```



Michaelis–Menten Kinetics: Asymptotic Analysis

What happens to the solution in the limit $\epsilon \rightarrow 0$?

`In[555]:=`

```

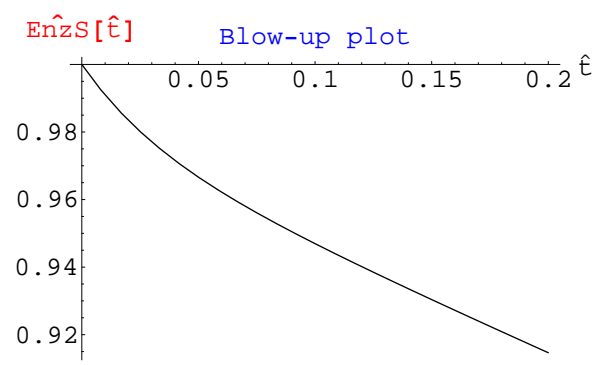
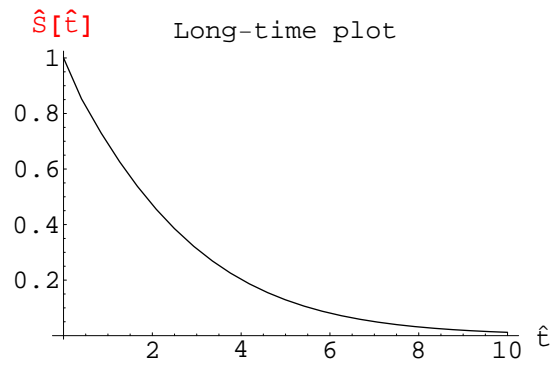
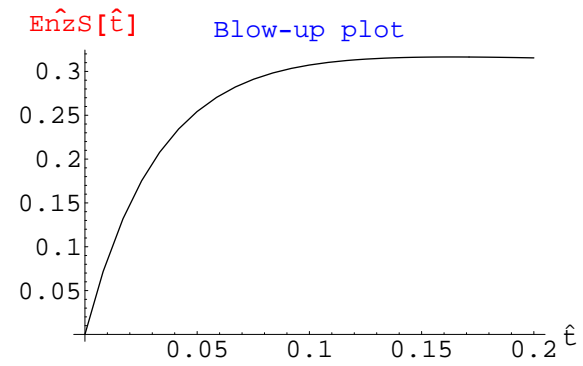
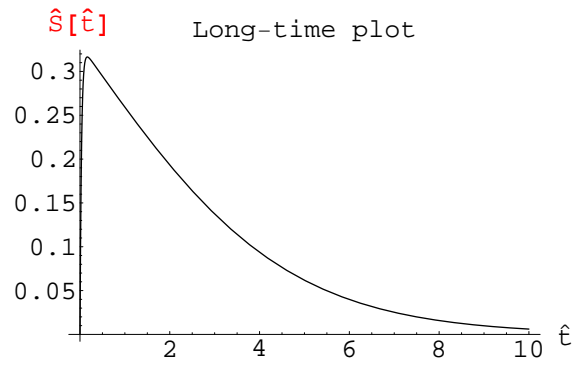
KineticParamReplaceList = {k1b → 1, k1f → 1, k2f → 1, sTotal[0] → 1, ε → 10-1};
Sol = NDSolve[Append[ParamEnzymeODE /. %, {ŝ[0] == 1, EnzS[0] == 0}], {ŝ, EnzS}, {t̂, 0, 10}];

boldBlue[x_] := StyleForm[x, FontColor → RGBColor[0, 0, 1], FontWeight → "Bold", FontSize → 10];
boldRed[x_] := StyleForm[x, FontColor → RGBColor[1, 0, 0], FontWeight → "Bold", FontSize → 10];
EnzymeSubstratePlotLarge = Plot[{EnzS[t̂]} /. Sol, {t̂, 0, 10}, PlotRange → All, ImageSize → {300, 150}, DisplayFunction → Identity, AxesLabel → {"t̂", {"ŝ[t̂]" // boldRed}}, PlotLabel → "Long-time plot"];

EnzymeSubstratePlotExpand =
  Plot[{EnzS[t̂]} /. Sol, {t̂, 0, 0.2}, PlotRange → All, ImageSize → {300, 150}, DisplayFunction → Identity, AxesLabel → {"t̂", {"EnzS[t̂]" // boldRed}}, PlotLabel → ("Blow-up plot" // boldBlue)];
SubstratePlotLarge = Plot[{ŝ[t̂]} /. Sol, {t̂, 0, 10}, PlotRange → All, ImageSize → {300, 150}, DisplayFunction → Identity, AxesLabel → {"t̂", {"ŝ[t̂]" // boldRed}}, PlotLabel → "Long-time plot"];

SubstratePlotExpand =
  Plot[{ŝ[t̂]} /. Sol, {t̂, 0, 0.2}, PlotRange → All, ImageSize → {300, 150}, DisplayFunction → Identity, AxesLabel → {"t̂", {"EnzS[t̂]" // boldRed}}, PlotLabel → ("Blow-up plot" // boldBlue)];
Show[GraphicsArray[{{EnzymeSubstratePlotLarge, EnzymeSubstratePlotExpand}, {SubstratePlotLarge, SubstratePlotExpand}}], ImageSize → {600, 300}];

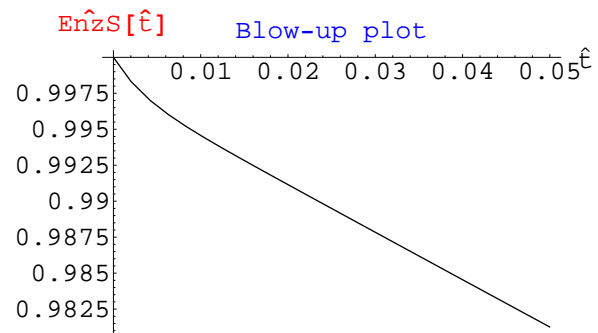
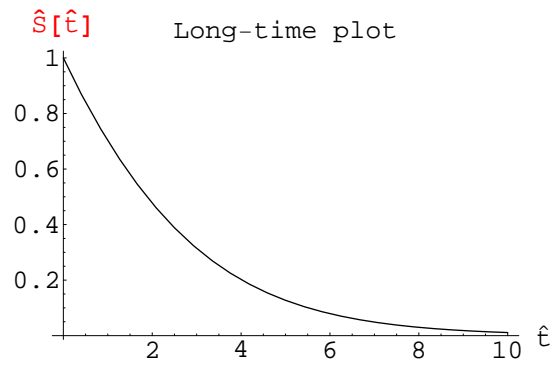
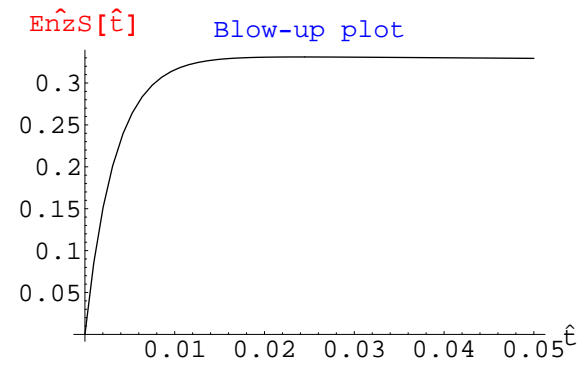
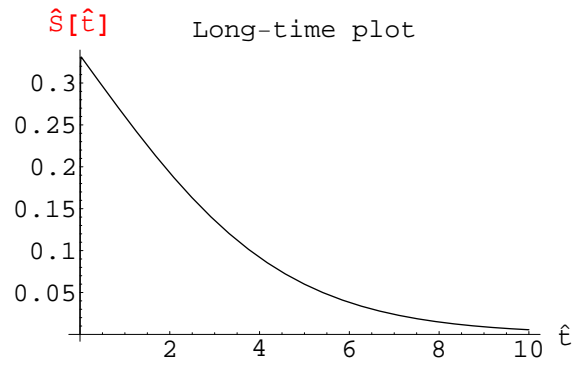
```



Michaelis–Menten Kinetics: Asymptotic Analysis

What happens to the solution in the limit $\epsilon \rightarrow 0$?

```
In[564]:=
KineticParamReplaceList = {k1b → 1, k1f → 1, k2f → 1, sTotal[0] → 1,  $\epsilon \rightarrow 10^{-2}$ };
Sol = NDSolve[Append[ParamEnzymeODE /. %, {ŝ[0] == 1, EnzS[0] == 0}], {ŝ, EnzS}, {t̂, 0, 10}];
EnzymeSubstratePlotLarge = Plot[{EnzS[t̂]} /. Sol, {t̂, 0, 10}, PlotRange → All,
  ImageSize → {300, 150}, DisplayFunction → Identity, AxesLabel → {"t̂", ("ŝ[t̂]" // boldRed)}, PlotLabel → "Long-time plot"];
EnzymeSubstratePlotExpand = Plot[{EnzS[t̂]} /. Sol, {t̂, 0, 0.05}, PlotRange → All, ImageSize → {300, 150},
  DisplayFunction → Identity, AxesLabel → {"t̂", ("EnzS[t̂]" // boldRed)}, PlotLabel → ("Blow-up plot" // boldBlue)];
SubstratePlotLarge = Plot[{ŝ[t̂]} /. Sol, {t̂, 0, 10}, PlotRange → All, ImageSize → {300, 150},
  DisplayFunction → Identity, AxesLabel → {"t̂", ("ŝ[t̂]" // boldRed)}, PlotLabel → "Long-time plot"];
SubstratePlotExpand = Plot[{ŝ[t̂]} /. Sol, {t̂, 0, 0.05}, PlotRange → All, ImageSize → {300, 150},
  DisplayFunction → Identity, AxesLabel → {"t̂", ("EnzS[t̂]" // boldRed)}, PlotLabel → ("Blow-up plot" // boldBlue)];
Show[GraphicsArray[{{EnzymeSubstratePlotLarge, EnzymeSubstratePlotExpand}, {SubstratePlotLarge, SubstratePlotExpand}}, ImageSize → {600, 300}];
```



Michaelis–Menten Kinetics: Asymptotic Analysis

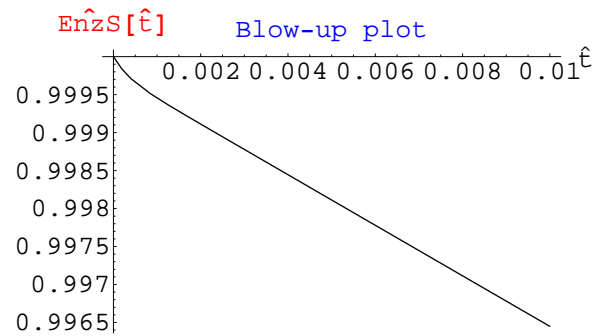
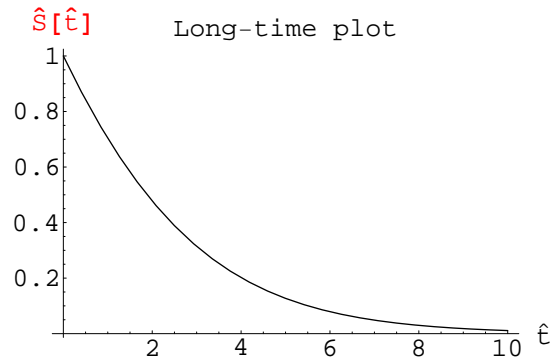
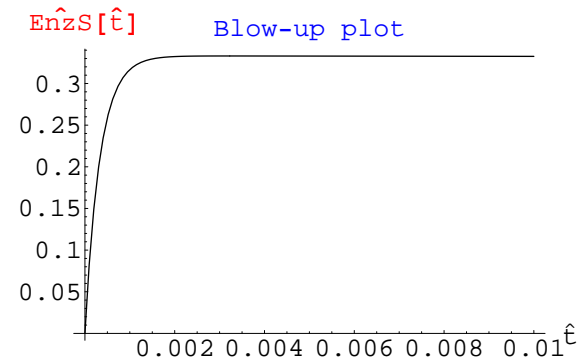
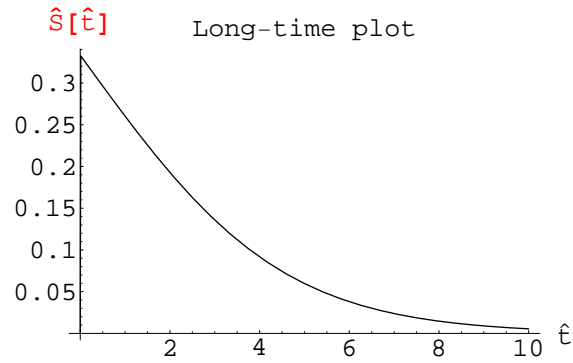
What happens to the solution in the limit $\epsilon \rightarrow 0$? Need to use **singular perturbation analysis**, to match outer and **boundary layer** solution

In[571]:=

```

KineticParamReplaceList = {k1b → 1, k1f → 1, k2f → 1, sTotal[0] → 1, ε → 10-3};
sol = NDSolve[Append[ParamEnzymeODE /. %, {ŝ[0] = 1, EnzS[0] = 0}], {ŝ, EnzS}, {t̂, 0, 10}];
EnzymeSubstratePlotLarge = Plot[{EnzS[t̂]} /. sol, {t̂, 0, 10}, PlotRange → All,
    ImageSize → {300, 150}, DisplayFunction → Identity, AxesLabel → {"t̂", ("ŝ[t̂]" // boldRed)}, PlotLabel → "Long-time plot"];
EnzymeSubstratePlotExpand = Plot[{EnzS[t̂]} /. sol, {t̂, 0, 0.01}, PlotRange → All, ImageSize → {300, 150},
    DisplayFunction → Identity, AxesLabel → {"t̂", ("EnzS[t̂]" // boldRed)}, PlotLabel → ("Blow-up plot" // boldBlue)];
SubstratePlotLarge = Plot[{ŝ[t̂]} /. sol, {t̂, 0, 10}, PlotRange → All, ImageSize → {300, 150},
    DisplayFunction → Identity, AxesLabel → {"t̂", ("ŝ[t̂]" // boldRed)}, PlotLabel → "Long-time plot"];
SubstratePlotExpand = Plot[{ŝ[t̂]} /. sol, {t̂, 0, 0.01}, PlotRange → All, ImageSize → {300, 150},
    DisplayFunction → Identity, AxesLabel → {"t̂", ("EnzS[t̂]" // boldRed)}, PlotLabel → ("Blow-up plot" // boldBlue)];
Show[GraphicsArray[{{EnzymeSubstratePlotLarge, EnzymeSubstratePlotExpand}, {SubstratePlotLarge, SubstratePlotExpand}], ImageSize → {600, 300}];

```



Regular Perturbation Analysis

Singular perturbation involves more complex ideas; start off with **regular** perturbation analysis. Suppose we want to solve the ODE:

```
In[260]:=
ODESys = {x''[t] + ε x'[t]^2 + x[t] == 0, x[0] == 1, x'[0] == 0};
```

where we know that ϵ is very small ($0 < \epsilon \ll 1$). Can we use the smallness of ϵ to obtain approximate solutions? Since ϵ enters the equation as a small parameter, consider:

In[261]:=

```
MaxOrder = 2;  xtest[t_] :=  $\sum_{i=0}^{\text{MaxOrder}} \epsilon^i x[i][t];$ 
```

In[263]:=

```
ExpansionODE = ODESys /. {x[t_] -> xtest[t], Derivative[n_][x][t_] -> Derivative[n][xtest][t]}; % // ColumnForm
```

Out[264]=

```
x[0][t] +  $\epsilon$  x[1][t] +  $\epsilon^2$  x[2][t] +  $\epsilon$  (x[0]'[t] +  $\epsilon$  x[1]'[t] +  $\epsilon^2$  x[2]'[t])2 + x[0]''[t] +  $\epsilon$  x[1]''[t] +  $\epsilon^2$  x[2]''[t] == 0
x[0][0] +  $\epsilon$  x[1][0] +  $\epsilon^2$  x[2][0] == 1
x[0]'[0] +  $\epsilon$  x[1]'[0] +  $\epsilon^2$  x[2]'[0] == 0
```

Now lets look at equation for each order of ϵ :

In[265]:=

```
eOrder = 0;
Order0ODE = Thread[ Equal[ Coefficient[ Map[# // First &, ExpansionODE],  $\epsilon$ , eOrder], Map[# // Last &, ExpansionODE] ] ]
```

Out[266]=

```
{x[0][t] + x[0]''[t] == 0, x[0][0] == 1, x[0]'[0] == 0}
```

In[267]:=

```
eOrder = 1;
Order1ODE = Thread[ Equal[ Coefficient[ Map[# // First &, ExpansionODE],  $\epsilon$ , eOrder], Map[# // Last &, ExpansionODE] ] ]
```

Out[268]=

```
{x[1][t] + x[0]'[t]2 + x[1]''[t] == 0, x[1][0] == 1, x[1]'[0] == 0}
```

In[269]:=

```
eOrder = 2;
Order2ODE = Thread[ Equal[ Coefficient[ Map[# // First &, ExpansionODE],  $\epsilon$ , eOrder], Map[# // Last &, ExpansionODE] ] ]
```

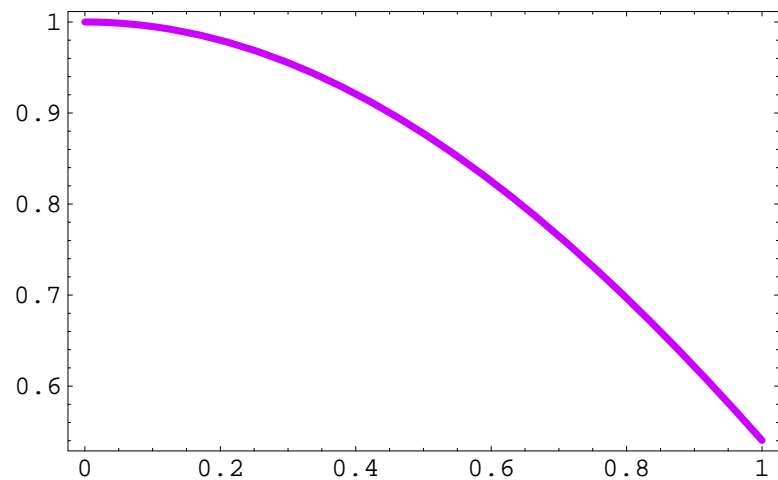
Out[270]=

```
{x[2][t] + 2 x[0]'[t] x[1]'[t] + x[2]''[t] == 0, x[2][0] == 1, x[2]'[0] == 0}
```

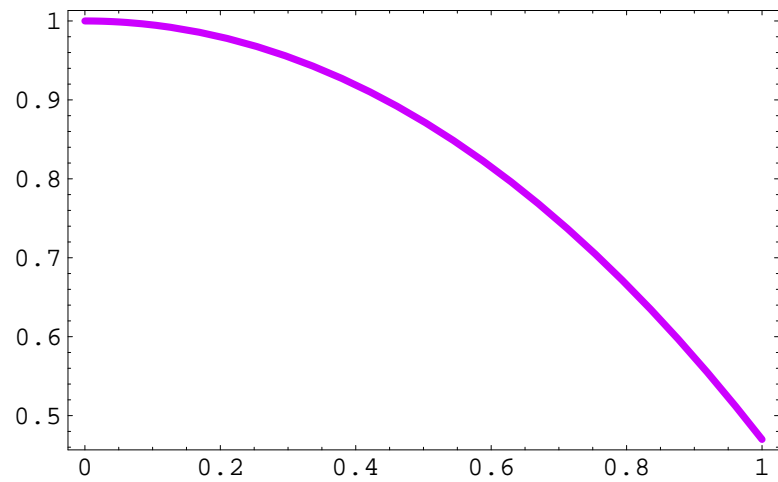
Regular Perturbation Analysis

Now we can solve the expansion equations at each order:

```
In[333]:=
Order0Soln = NDSolve[Order0ODE, x[0], {t, 0, 1}];
Plot[x[0][t] /. Order0Soln, {t, 0, 1},
  PlotStyle -> {Hue[0.8], Thickness[0.01]}, Frame -> True, PlotRange -> All, Axes -> None];
```



```
In[335]:=
Order1Soln = NDSolve[Order1ODE /. Order0Soln, x[1], {t, 0, 1}];
Plot[x[1][t] /. Order1Soln, {t, 0, 1},
  PlotStyle -> {Hue[0.8], Thickness[0.01]}, Frame -> True, PlotRange -> All, Axes -> None];
```



```
In[337]:=
Order2Soln = NDSolve[Order2ODE /. Order0Soln /. Order1Soln, x[2], {t, 0, 1}];
```



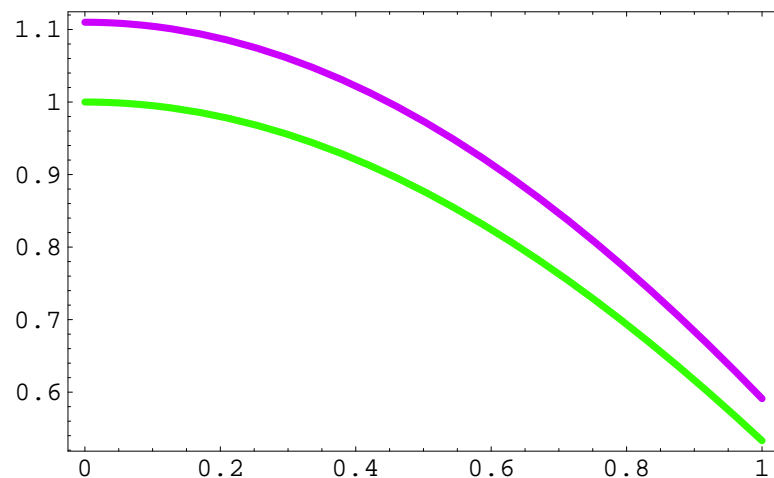
Regular Perturbation Analysis

Lets compare the expansion solution with the solution for the original ODE system:

```
In[362]:=
ODESoln = NDSolve[ODESys /. { $\epsilon \rightarrow 0.1$ }, x, {t, 0, 1}];
ODESolnPlot = Plot[x[t] /. ODESoln, {t, 0, 1}, PlotStyle -> {Hue[0.3], Thickness[0.01]},
  Frame -> True, PlotRange -> All, DisplayFunction -> Identity, Axes -> None];

ExpansionODESolnPlot = Plot[xtest[t] /. (Join[{ $\epsilon \rightarrow 0.1$ }, Order0Soln, Order1Soln, Order2Soln] // Flatten), {t, 0, 1},
  PlotStyle -> {Hue[0.8], Thickness[0.01]}, Frame -> True, PlotRange -> All, DisplayFunction -> Identity, Axes -> None];

Show[ODESolnPlot, ExpansionODESolnPlot, DisplayFunction -> $DisplayFunction];
```



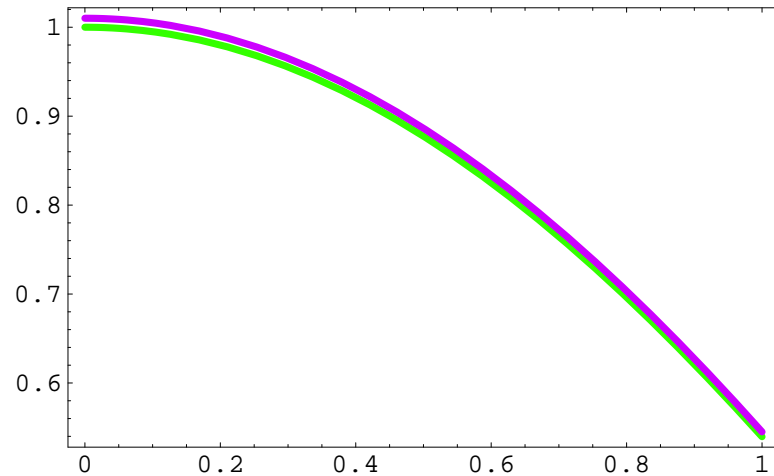
Regular Perturbation Analysis

Lets compare the expansion solution with the solution for the original ODE system:

```
In[366]:=
ODESoln = NDSolve[ODESys /. { $\epsilon \rightarrow 0.01$ }, x, {t, 0, 1}];
ODESolnPlot = Plot[x[t] /. ODESoln, {t, 0, 1}, PlotStyle -> {Hue[0.3], Thickness[0.01]},
  Frame -> True, PlotRange -> All, DisplayFunction -> Identity, Axes -> None];

ExpansionODESolnPlot = Plot[xtest[t] /. (Join[{ $\epsilon \rightarrow 0.01$ }, Order0Soln, Order1Soln, Order2Soln] // Flatten), {t, 0, 1},
  PlotStyle -> {Hue[0.8], Thickness[0.01]}, Frame -> True, PlotRange -> All, DisplayFunction -> Identity, Axes -> None];

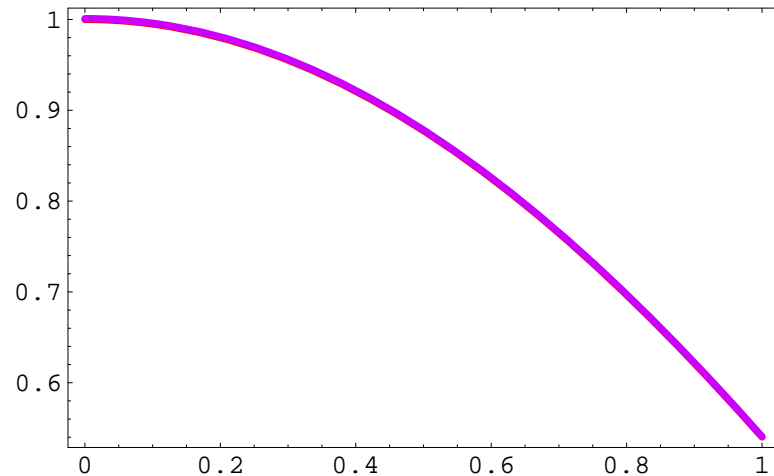
Show[ODESolnPlot, ExpansionODESolnPlot, DisplayFunction -> $DisplayFunction];
```



Regular Perturbation Analysis

Lets compare the expansion solution with the solution for the original ODE system:

```
In[358]:=
ODESoln = NDSolve[ODESys /. {ε -> 0.001}, x, {t, 0, 1}];
ODESolnPlot = Plot[x[t] /. ODESoln, {t, 0, 1}, PlotStyle -> {Hue[0.3], Thickness[0.01]},
  Frame -> True, PlotRange -> All, DisplayFunction -> Identity, Axes -> None];
ExpansionODESolnPlot = Plot[xtest[t] /. (Join[{ε -> 0.001}, Order0Soln, Order1Soln, Order2Soln] // Flatten), {t, 0, 1},
  PlotStyle -> {Hue[0.8], Thickness[0.01]}, Frame -> True, PlotRange -> All, DisplayFunction -> Identity, Axes -> None];
Show[{ODESolnPlot, ExpansionODESolnPlot}, DisplayFunction -> $DisplayFunction];
```



23 of 24

Michaelis–Menten Kinetics: Asymptotic Analysis

```
In[1073]:=
ODESys = Append[ParamEnzymeODE /. {k1b -> 1, k1f -> 1, k2f -> 1, STotal[0] -> 1}, {Ŝ[0] == 1, EnzS[0] == 0}] // Flatten
```

```
Out[1073]=
{EnzS[τ] (2 + Ŝ[τ]) + ε EnzS'[τ] == Ŝ[τ], EnzS[τ] (1 + Ŝ[τ]) == Ŝ[τ] + Ŝ'[τ], Ŝ[0] == 1, EnzS[0] == 0}
```

Read in the PerturbationODE package, that does the perturbation analysis automatically

```
In[1075]:=
  << PerturbationODE.m

In[1099]:=
  OrderList = PolyOrderList[ODESys, {Ŝ[τ], EnzS[τ]}, ε, 1];
  (* ε: zeroth order equation *)
  OrderList[[1]] // ColumnForm

Out[1100]=
  2 EnzS[0][τ] + EnzS[0][τ] Ŝ[0][τ] == Ŝ[0][τ]
  EnzS[0][τ] + EnzS[0][τ] Ŝ[0][τ] == Ŝ[0]'[τ] + Ŝ[0][τ]
  Ŝ[0][0] == 1
  EnzS[0][0] == 0

In[1101]:=
  (* ε: first order equation *)
  OrderList[[2]] // ColumnForm

Out[1101]=
  EnzS[0]'[τ] + 2 EnzS[1][τ] + EnzS[1][τ] Ŝ[0][τ] + EnzS[0][τ] Ŝ[1][τ] == Ŝ[1][τ]
  EnzS[1][τ] + EnzS[1][τ] Ŝ[0][τ] + EnzS[0][τ] Ŝ[1][τ] == Ŝ[1]'[τ] + Ŝ[1][τ]
  Ŝ[1][0] == 0
  EnzS[1][0] == 0
```

Conclusion: the expansion equations are not well-formed, ODE systems!
 In fact, need to do **asymptotic matching**



Singular Perturbation

- Construct **inner**, or **singular solution**: boundary layer
- Construct **outer**, or **quasi-steady state** solution
- Matching condition: solution is continuous going from **inner** to **outer** solutions
- Obtain the **composite** solution, which is **uniformly valid** approximation for all time