Symbolic/Numeric Methods for BVPs

Ian Gladwell
Department of Mathematics
Southern Methodist University
Dallas, Texas 75275
gladwell@seas.smu.edu

Based on work work with my graduate student

Hilary Risser

Plan of Talk

Introduction

Plan of Talk

Introduction

Linear BVP analysis

Plan of Talk

Introduction

Linear BVP analysis

BVP software

Plan of Talk

Introduction

Linear BVP analysis

BVP software

Impact of analysis on use of software

Introduction

ODE BVP Software

- ODE BVP Software mainly designed for nonlinear problems
- Linear problems usually solved in nonlinear form

Introduction

ODE BVP Software

- ODE BVP Software mainly designed for nonlinear problems
- Linear problems usually solved in nonlinear form

Difficulties

- strong nonlinearity
- solution approximability
- mesh placement and error estimation
- Cannot handle difficult linear problems (e.g. singular perturbations) without assistance

Linear singular perturbation problems

Simple form $\epsilon y'' - a(x)y' + b(x)y = f(x)$ with Dirichlet BCs $y(x_L) = L$, $y(x_R) = R$

Linear singular perturbation problems

Simple form $\epsilon y'' - a(x)y' + b(x)y = f(x)$ with Dirichlet BCs $y(x_L) = L$, $y(x_R) = R$

If $a(x_L) > 0$ potential boundary layer at x_L; if $a(x_R) < 0$ potential boundary layer at x_R; for all x_0 such that $a(x_0) = 0$, $x_L < x_0 < x_R$ potential turning layer at x_0

Linear singular perturbation problems

Simple form $\epsilon y'' - a(x)y' + b(x)y = f(x)$ with Dirichlet BCs $y(x_L) = L$, $y(x_R) = R$

If $a(x_L) > 0$ potential boundary layer at x_L; if $a(x_R) < 0$ potential boundary layer at x_R; for all x_0 such that $a(x_0) = 0,\ x_L < x_0 < x_R$ potential turning layer at x_0

Example: $\epsilon y'' + y' = 0$, $y(0) = 1$, $y(1) = 2$

Linear singular perturbation problems

Simple form $\epsilon y'' - a(x)y' + b(x)y = f(x)$ with Dirichlet BCs $y(x_L) = L$, $y(x_R) = R$

If $a(x_L) > 0$ potential boundary layer at x_L; if $a(x_R) < 0$ potential boundary layer at x_R; for all x_0 such that $a(x_0) = 0, \; x_L < x_0 < x_R$ potential turning layer at x_0

Example: $\epsilon y'' + y' = 0$, $y(0) = 1$, $y(1) = 2$

Boundary layer at $x = 1$ of width $O(\epsilon)$. Inner solution near $x = 1$ behaves like $e^{(x-1)/\epsilon}$

Linear singular perturbation problems

Simple form $\epsilon y'' - a(x)y' + b(x)y = f(x)$ with Dirichlet BCs $y(x_L) = L$, $y(x_R) = R$

If $a(x_L) > 0$ potential boundary layer at x_L; if $a(x_R) < 0$ potential boundary layer at x_R; for all x_0 such that $a(x_0) = 0,\ x_L < x_0 < x_R$ potential turning layer at x_0

Example: $\epsilon y'' + y' = 0$, $y(0) = 1$, $y(1) = 2$

Boundary layer at $x = 1$ of width $O(\epsilon)$. Inner solution near $x = 1$ behaves like $e^{(x-1)/\epsilon}$

To $O(\epsilon)$ solution to BVP is $1 + e^{(x-1)/\epsilon}$ – found by matching inner and outer solutions

Linear singular perturbation problems

Example: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$, Kervorkian and Cole.

Linear singular perturbation problems

Example: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$, Kervorkian and Cole.

Potentially has boundary layer at each of $x = -1$ and $x = 1$ and turning layer at $x = 0$

Linear singular perturbation problems

Example: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$, Kervorkian and Cole.

Potentially has boundary layer at each of $x = -1$ and $x = 1$ and turning layer at $x = 0$

In fact no turning layer. Central part of solution always looks like straight line thru' $x = 0$ for $\epsilon > 0$ small enough

Linear singular perturbation problems

Example: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$, Kervorkian and Cole.

Potentially has boundary layer at each of $x = -1$ and $x = 1$ and turning layer at $x = 0$

In fact no turning layer. Central part of solution always looks like straight line thru' $x = 0$ for $\epsilon > 0$ small enough

First order perturbation solution $y = e^{-(1+x)/\epsilon} + 2e^{-(1-x)/\epsilon}$

Linear singular perturbation problems

Example: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$, Kervorkian and Cole.

Potentially has boundary layer at each of $x = -1$ and $x = 1$ and turning layer at $x = 0$

In fact no turning layer. Central part of solution always looks like straight line thru' $x = 0$ for $\epsilon > 0$ small enough

First order perturbation solution $y = e^{-(1+x)/\epsilon} + 2e^{-(1-x)/\epsilon}$

Except possibly for missing turning layer no sign from singular perturbation analysis that this problem has any peculiarities

Linear singular perturbation problems

Barrier problem: $\epsilon y'' + (x^2 - 0.5^2)y = 0$, $y(-1) = 1$, $y(1) = 2$

Linear singular perturbation problems

Barrier problem: $\epsilon y'' + (x^2 - 0.5^2)y = 0$, $y(-1) = 1$, $y(1) = 2$

Not amenable to standard perturbation analysis (no y' term). Can use WKB analysis

Linear singular perturbation problems

Barrier problem: $\epsilon y'' + (x^2 - 0.5^2)y = 0$, $y(-1) = 1$, $y(1) = 2$

Not amenable to standard perturbation analysis (no y' term). Can use WKB analysis

Solution highly oscillatory in intervals $[-1, -0.5]$ and $[0.5, 1]$. Almost constant in interior of $[-0.5, 0.5]$ for $\epsilon > 0$ small enough

Linear singular perturbation problems

Barrier problem: $\epsilon y'' + (x^2 - 0.5^2)y = 0$, $y(-1) = 1$, $y(1) = 2$

Not amenable to standard perturbation analysis (no y' term). Can use WKB analysis

Solution highly oscillatory in intervals $[-1, -0.5]$ and $[0.5, 1]$. Almost constant in interior of $[-0.5, 0.5]$ for $\epsilon > 0$ small enough

Frequency of oscillation proportional to $1/\sqrt{\epsilon}$

Linear singular perturbation problems

More complex form $\epsilon y'' - a(x,\epsilon)y' + b(x,\epsilon)y = f(x)$ with Dirichlet BCs $y(x_L) = L(\epsilon)$, $y(x_R) = R(\epsilon)$ where at least one of $a(x,\epsilon)$ and $b(x,\epsilon)$ are $O(1)$ as $\epsilon \to 0$

Linear singular perturbation problems

More complex form $\epsilon y'' - a(x,\epsilon)y' + b(x,\epsilon)y = f(x)$ with Dirichlet BCs $y(x_L) = L(\epsilon)$, $y(x_R) = R(\epsilon)$ where at least one of $a(x,\epsilon)$ and $b(x,\epsilon)$ are $O(1)$ as $\epsilon \to 0$

Same rules as above

Linear singular perturbation problems

More complex form $\epsilon y'' - a(x,\epsilon)y' + b(x,\epsilon)y = f(x)$ with Dirichlet BCs $y(x_L) = L(\epsilon)$, $y(x_R) = R(\epsilon)$ where at least one of $a(x,\epsilon)$ and $b(x,\epsilon)$ are $O(1)$ as $\epsilon \to 0$

Same rules as above

Example: $\epsilon^3 y'' + x^3 y' + (x^3 - \epsilon)y = 0$, $y(0) = 1$, $y(1) = 2$

Linear singular perturbation problems

More complex form $\epsilon y'' - a(x,\epsilon)y' + b(x,\epsilon)y = f(x)$ with Dirichlet BCs $y(x_L) = L(\epsilon)$, $y(x_R) = R(\epsilon)$ where at least one of $a(x,\epsilon)$ and $b(x,\epsilon)$ are $O(1)$ as $\epsilon \to 0$

Same rules as above

Example: $\epsilon^3 y'' + x^3 y' + (x^3 - \epsilon)y = 0$, $y(0) = 1$, $y(1) = 2$

Rules for turning layer satisfied at boundary $x = 0$ so potential for boundary layer there

Linear singular perturbation problems

Two layers at $x = 0$ one of width $O(\epsilon)$ and one of width $O(\epsilon^{1/2})$

Linear singular perturbation problems

Two layers at $x = 0$ one of width $O(\epsilon)$ and one of width $O(\epsilon^{1/2})$

That of width $O(\epsilon^{1/2})$ is the inner layer and that of width $O(\epsilon)$ is the (narrower) inner-inner layer

Linear singular perturbation problems

Two layers at $x = 0$ one of width $O(\epsilon)$ and one of width $O(\epsilon^{1/2})$

That of width $O(\epsilon^{1/2})$ is the inner layer and that of width $O(\epsilon)$ is the (narrower) inner-inner layer

Solutions in these layers have the forms $e^{-x/\epsilon}$ and $e^{-\epsilon/(2x^2)}$ respectively

Linear singular perturbation problems

Two layers at $x = 0$ one of width $O(\epsilon)$ and one of width $O(\epsilon^{1/2})$

That of width $O(\epsilon^{1/2})$ is the inner layer and that of width $O(\epsilon)$ is the (narrower) inner-inner layer

Solutions in these layers have the forms $e^{-x/\epsilon}$ and $e^{-\epsilon/(2x^2)}$ respectively

Overall approximate solution by matching to $O(\epsilon)$ is $e^{-x/\epsilon} + 2e[e^{-\epsilon/(2x^2)} + e^{-x} - 1]$

Linear singular perturbation problems

Two layers at $x = 0$ one of width $O(\epsilon)$ and one of width $O(\epsilon^{1/2})$

That of width $O(\epsilon^{1/2})$ is the inner layer and that of width $O(\epsilon)$ is the (narrower) inner-inner layer

Solutions in these layers have the forms $e^{-x/\epsilon}$ and $e^{-\epsilon/(2x^2)}$ respectively

Overall approximate solution by matching to $O(\epsilon)$ is $e^{-x/\epsilon} + 2e[e^{-\epsilon/(2x^2)} + e^{-x} - 1]$

For moderately sized ϵ inner-inner layer quite visible

Linear singular perturbation problems

Position of boundary layers determined by sign of $a(x)$ at ends of interval

Linear singular perturbation problems

Position of boundary layers determined by sign of $a(x)$ at ends of interval

Positions of turning layers determined from changes of sign of $a(x)$. Note, positions can depend on ϵ in more general case

Linear singular perturbation problems

Position of boundary layers determined by sign of $a(x)$ at ends of interval

Positions of turning layers determined from changes of sign of $a(x)$. Note, positions can depend on ϵ in more general case

Inner-inner layers at same positions as inner(turning) or boundary layers – correspond to a second (faster changing) solution in a layer – can only occur in the more general case

Linear singular perturbation problems

Position of boundary layers determined by sign of $a(x)$ at ends of interval

Positions of turning layers determined from changes of sign of $a(x)$. Note, positions can depend on ϵ in more general case

Inner-inner layers at same positions as inner(turning) or boundary layers – correspond to a second (faster changing) solution in a layer – can only occur in the more general case

Width of layer – only known to be of width $O(g(\epsilon))$ where $g(s)$ is s or $\sqrt{s}$ in many applications – problem for meshing in numerical solutions – is $2g(\epsilon)$ safer choice computationally than $0.5g(\epsilon)$?

Analysis in Mathematica

Locate layers based on sign of $a(x)$ and by solving $a(x) = 0$ for all roots in interval

Analysis in Mathematica

Locate layers based on sign of $a(x)$ and by solving $a(x) = 0$ for all roots in interval

Compute form of outer solution over whole interval

Analysis in Mathematica

Locate layers based on sign of $a(x)$ and by solving $a(x) = 0$ for all roots in interval

Compute form of outer solution over whole interval

Compute form of inner solution(s) in each layer – more than one solution if there are inner-inner layers

Analysis in Mathematica

Locate layers based on sign of $a(x)$ and by solving $a(x) = 0$ for all roots in interval

Compute form of outer solution over whole interval

Compute form of inner solution(s) in each layer – more than one solution if there are inner-inner layers

Match each inner solution to outer solution at edge of its layer (asymptotically as $\epsilon \to 0$). Match inner-inner solutions to inner solution at edges of inner-inner layers (asymptotically as $\epsilon \to 0$)

Result of Analysis in Mathematica

Know locations and types of all layers

Result of Analysis in Mathematica

Know locations and types of all layers

Know form of solution in each layer which also reveals width of layers

Result of Analysis in Mathematica

Know locations and types of all layers

Know form of solution in each layer which also reveals width of layers

Often can compute global approximate solution. Do we need it? Can supply to nonlinear boundary value solvers but theoretically isn't needed for linear problems.

Result of Analysis in Mathematica

Know locations and types of all layers

Know form of solution in each layer which also reveals width of layers

Often can compute global approximate solution. Do we need it? Can supply to nonlinear boundary value solvers but theoretically isn't needed for linear problems.

Sometimes Mathematica's inability to take some 0/0 limits, and some other limits involving (its own) special functions restricts what us. Also Mathematica's inability to invert some inner solutions a difficulty. Some cases where even if we can complete the analysis have some unknown constants in the solution – not sure whether this always fixable

The linear BVP solver ODE_ADAP

Solves linear first order systems of BVP ODEs

The linear BVP solver ODE_ADAP

Solves linear first order systems of BVP ODEs

Written by June-Yub Lee, EWHA, Korea – result of Ph.D. studies with Leslie Greengard – development abandoned?

The linear BVP solver ODE_ADAP

Solves linear first order systems of BVP ODEs

Written by June-Yub Lee, EWHA, Korea – result of Ph.D. studies with Leslie Greengard – development abandoned?

Uses collocation based on Chebyshev polynomials

The linear BVP solver ODE_ADAP

Solves linear first order systems of BVP ODEs

Written by June-Yub Lee, EWHA, Korea – result of Ph.D. studies with Leslie Greengard – development abandoned?

Uses collocation based on Chebyshev polynomials

Adapts mesh based on measure of size of Chebyshev coefficients in each interval of mesh

The linear BVP solver ODE_ADAP

Solves linear first order systems of BVP ODEs

Written by June-Yub Lee, EWHA, Korea – result of Ph.D. studies with Leslie Greengard – development abandoned?

Uses collocation based on Chebyshev polynomials

Adapts mesh based on measure of size of Chebyshev coefficients in each interval of mesh

Permits user to specify initial mesh – does not remove “unneeded” points

COLMOD

Written by Jeff Cash and Ross Wright of Imperial College – built on underlying code COLNEW by Georg Bader and Uri Ascher

COLMOD

Written by Jeff Cash and Ross Wright of Imperial College – built on underlying code COLNEW by Georg Bader and Uri Ascher

Solves a sequence of nonlinear systems of BVP ODEs (continuation) – indicate a linear problem by setting flag – deals with systems of higher order equations directly

COLMOD

Written by Jeff Cash and Ross Wright of Imperial College – built on underlying code COLNEW by Georg Bader and Uri Ascher

Solves a sequence of nonlinear systems of BVP ODEs (continuation) – indicate a linear problem by setting flag – deals with systems of higher order equations directly

Uses collocation based on spline monomials

COLMOD

Written by Jeff Cash and Ross Wright of Imperial College – built on underlying code COLNEW by Georg Bader and Uri Ascher

Solves a sequence of nonlinear systems of BVP ODEs (continuation) – indicate a linear problem by setting flag – deals with systems of higher order equations directly

Uses collocation based on spline monomials

Adapts mesh based on measure of equidistribution of error – similar but different strategy to COLNEW – changed to work with continuation process

COLMOD

Written by Jeff Cash and Ross Wright of Imperial College – built on underlying code COLNEW by Georg Bader and Uri Ascher

Solves a sequence of nonlinear systems of BVP ODEs (continuation) – indicate a linear problem by setting flag – deals with systems of higher order equations directly

Uses collocation based on spline monomials

Adapts mesh based on measure of equidistribution of error – similar but different strategy to COLNEW – changed to work with continuation process

Uses continuation based on user-specified parameter – predicts new mesh using measures of change in previous mesh – removes unneeded mesh points

ACDC

Also written by Jeff Cash and Ross Wright – built on underlying BVP code TWPBVP by Jeff and Margaret Wright

ACDC

Also written by Jeff Cash and Ross Wright – built on underlying BVP code TWPBVP by Jeff and Margaret Wright

Solves a sequence of nonlinear systems of BVP ODEs – indicate a linear problem by setting flag – deals with first order systems only

ACDC

Also written by Jeff Cash and Ross Wright – built on underlying BVP code TWPBVP by Jeff and Margaret Wright

Solves a sequence of nonlinear systems of BVP ODEs – indicate a linear problem by setting flag – deals with first order systems only

Uses deferred correction but with monoimplicit RK formulas of TWPBVP replaced by Lobatto RK formulas

ACDC

Also written by Jeff Cash and Ross Wright – built on underlying BVP code TWPBVP by Jeff and Margaret Wright

Solves a sequence of nonlinear systems of BVP ODEs – indicate a linear problem by setting flag – deals with first order systems only

Uses deferred correction but with monoimplicit RK formulas of TWPBVP replaced by Lobatto RK formulas

Adapts mesh based on measure of equidistribution of error

ACDC

Also written by Jeff Cash and Ross Wright – built on underlying BVP code TWPBVP by Jeff and Margaret Wright

Solves a sequence of nonlinear systems of BVP ODEs – indicate a linear problem by setting flag – deals with first order systems only

Uses deferred correction but with monoimplicit RK formulas of TWPBVP replaced by Lobatto RK formulas

Adapts mesh based on measure of equidistribution of error

Uses continuation based on user-specified parameter – predicts new mesh using measures of change in previous mesh – removes unneeded mesh points

Possible use of Mathematica analysis

To provide suitable initial mesh for codes like ode_adap

Possible use of Mathematica analysis

To provide suitable initial mesh for codes like ode_adap

For codes like COLMOD or ACDC to jump start continuation process nearer end of path by providing good initial mesh in regions where asymptotics should apply

Possible use of Mathematica analysis

To provide suitable initial mesh for codes like ode_adap

For codes like COLMOD or ACDC to jump start continuation process nearer end of path by providing good initial mesh in regions where asymptotics should apply

As check to ensure computation kept on course, i.e. by making sure there are layers/oscillations where there should be

Choice of initial mesh

Choose total number of mesh points

Choice of initial mesh

Choose total number of mesh points

Assign mesh in each layer separately using Baylhalov formula – essentially proportionally to inverse function of inner solution – choose edge of layer as ϵ or $\sqrt{\epsilon}$ or whatever (plan to experiment with sensitivity later)

Choice of initial mesh

Choose total number of mesh points

Assign mesh in each layer separately using Baylhalov formula – essentially proportionally to inverse function of inner solution – choose edge of layer as ϵ or $\sqrt{\epsilon}$ or whatever (plan to experiment with sensitivity later)

Use remaining mesh points to create equispaced mesh between layers

Suitable initial mesh

Can supply an initial mesh for ode_adap

Suitable initial mesh

Can supply an initial mesh for ode_adap

Sometimes more efficient than ode_adap’s default initial mesh – never significantly less efficient

Suitable initial mesh

Can supply an initial mesh for ode_adap

Sometimes more efficient than ode_adap's default initial mesh – never significantly less efficient

When starting with small ϵ can be difference between success and failure

Reducing number of continuation steps

Aim to jump into continuation process closer to final value of parameter than may be possible without good initial mesh

Reducing number of continuation steps

Aim to jump into continuation process closer to final value of parameter than may be possible without good initial mesh

Hope to reduce total work by taking less continuation steps and by using better mesh in later stages

Reducing number of continuation steps

Aim to jump into continuation process closer to final value of parameter than may be possible without good initial mesh

Hope to reduce total work by taking less continuation steps and by using better mesh in later stages

Hope to extend range of ϵ over which solution may be obtained

Reducing number of continuation steps

Aim to jump into continuation process closer to final value of parameter than may be possible without good initial mesh

Hope to reduce total work by taking less continuation steps and by using better mesh in later stages

Hope to extend range of ϵ over which solution may be obtained

Experiments with COLMOD lead to no major improvement so far – probably need to hook into COLMOD's internal mesh placement strategy

Reducing number of continuation steps

Aim to jump into continuation process closer to final value of parameter than may be possible without good initial mesh

Hope to reduce total work by taking less continuation steps and by using better mesh in later stages

Hope to extend range of ϵ over which solution may be obtained

Experiments with COLMOD lead to no major improvement so far – probably need to hook into COLMOD's internal mesh placement strategy

Equidistribution really what is needed?

Checking the solution

Two layer problem: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$

Checking the solution

Two layer problem: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$

Problem ill-conditioned for $\epsilon \approx 1/70$ but seems better conditioned for smaller values of ϵ

Checking the solution

Two layer problem: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$

Problem ill-conditioned for $\epsilon \approx 1/70$ but seems better conditioned for smaller values of ϵ

ode_adap can solve at just less than $\epsilon = 1/70$ with care

Checking the solution

Two layer problem: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$

Problem ill-conditioned for $\epsilon \approx 1/70$ but seems better conditioned for smaller values of ϵ

ode_adap can solve at just less than $\epsilon = 1/70$ with care

COLMOD and ACDC both fail if continue from $\epsilon \approx 1/2$ (default) or smaller value to $\epsilon = 1/70$ or even $\epsilon = 1/60$

Checking the solution

Two layer problem: $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$

Problem ill-conditioned for $\epsilon \approx 1/70$ but seems better conditioned for smaller values of ϵ

ode_adap can solve at just less than $\epsilon = 1/70$ with care

COLMOD and ACDC both fail if continue from $\epsilon \approx 1/2$ (default) or smaller value to $\epsilon = 1/70$ or even $\epsilon = 1/60$

Seen it suggested that ill-conditioning results corresponding eigenproblem having rounding error level eigenvalue when $\epsilon \approx 1/70$. But why does problem seem better conditioned for $\epsilon << 1/70$?

Checking the solution

If start at default $\epsilon = 1/2$ and aim for small ϵ continuation path for both COLMOD and ACDC seem to jump over difficult area then ”solve” the problem for all small value of ϵ attempted

Checking the solution

If start at default $\epsilon = 1/2$ and aim for small ϵ continuation path for both COLMOD and ACDC seem to jump over difficult area then ”solve” the problem for all small value of ϵ attempted

But often solution is not correct

Checking the solution

If start at default $\epsilon = 1/2$ and aim for small ϵ continuation path for both COLMOD and ACDC seem to jump over difficult area then ”solve” the problem for all small value of ϵ attempted

But often solution is not correct

Starting at $\epsilon = 1/2$ and aiming for $\epsilon = 10^{-i}$, $i = 1 \ldots, 8$ get correct behavior except for $\epsilon = 10^{-2}, 10^{-4}$ where right boundary layer is lost

Checking the solution

If start at default $\epsilon = 1/2$ and aim for small ϵ continuation path for both COLMOD and ACDC seem to jump over difficult area then ”solve” the problem for all small value of ϵ attempted

But often solution is not correct

Starting at $\epsilon = 1/2$ and aiming for $\epsilon = 10^{-i}, i = 1 \ldots, 8$ get correct behavior except for $\epsilon = 10^{-2}, 10^{-4}$ where right boundary layer is lost

Starting at $\epsilon = 10^{-2}$ and aiming for $\epsilon = 10^{-i}, i = 2 \ldots, 8$ lose left boundary layer for $\epsilon = 10^{-2}, 10^{-6}, 10^{-7}$ and right boundary layer for $\epsilon = 10^{-3}, 10^{-4}$

Checking the solution

If start at default $\epsilon = 1/2$ and aim for small ϵ continuation path for both COLMOD and ACDC seem to jump over difficult area then ”solve” the problem for all small value of ϵ attempted

But often solution is not correct

Starting at $\epsilon = 1/2$ and aiming for $\epsilon = 10^{-i}$, $i = 1 \ldots, 8$ get correct behavior except for $\epsilon = 10^{-2}, 10^{-4}$ where right boundary layer is lost

Starting at $\epsilon = 10^{-2}$ and aiming for $\epsilon = 10^{-i}$, $i = 2 \ldots, 8$ lose left boundary layer for $\epsilon = 10^{-2}, 10^{-6}, 10^{-7}$ and right boundary layer for $\epsilon = 10^{-3}, 10^{-4}$

ACDC behaves similarly unpredictably (but not exactly same

Checking the solution

Tried changing problem for COLMOD

Checking the solution

Tried changing problem for COLMOD

Tried changing problem $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$ to $\epsilon y'' + xy' - 2y = 0$, $y(-1) = 1$, $y(1) = 2$. Very similar results including near $\epsilon = 1/70$

Checking the solution

Tried changing problem for COLMOD

Tried changing problem $\epsilon y'' + xy' - y = 0$, $y(-1) = 1$, $y(1) = 2$ to $\epsilon y'' + xy' - 2y = 0$, $y(-1) = 1$, $y(1) = 2$. Very similar results including near $\epsilon = 1/70$

And to $\epsilon y'' + xy' - y = 0$, $y(-1) = -1$, $y(1) = 2$. Changes shape of solution but not behavior of code

Approximate initial solution

Not experimented sufficiently to draw conclusions

Approximate initial solution

Not experimented sufficiently to draw conclusions

Should only help where initial guess is used to start iteration or used as a check on computed solution