CHAPTER 8: *Finite Difference Methods For BVPs*

Use knowledge of IVPs, but no solving IVPs!

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(x, \mathbf{y}), \qquad a < x < b, \\ \mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) &= \mathbf{0}. \end{aligned}$$

Define a mesh π (partition)

$$\begin{aligned} \pi &= \{a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b\}, \\ \Delta x_n &= x_n - x_{n-1} = n^{th} \text{ stepsize (subinterval length)} \end{aligned}$$

We wish to solve for

$$\mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_N, \quad \text{where} \quad \mathbf{y}_j \approx \mathbf{y}(x_j).$$

Note 1. • *Solve for all* $\mathbf{y}_j$ *at once!*
$\rightarrow$ *No method for BVPs can be explicit.*

Implicit methods are no less convenient than explicit methods!

- *Multistep methods don't really make sense either. (What are "past" values ?)*

- *Symmetric implicit RK are natural.*
$\rightarrow$ *They treat both directions equally.*

8.1 Midpoint (and Trapezoidal) Method(s)

Recall the midpoint method

$$\frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{\Delta x_n} = \mathbf{f}\left(x_{n-1/2}, \frac{1}{2}(\mathbf{y}_n + \mathbf{y}_{n-1})\right), \quad n = 1, 2, \cdots, N.$$

$\rightarrow Nm$ equations.

We also must satisfy the BCs

$$\mathbf{g}(\mathbf{y}_0, \mathbf{y}_N) = \mathbf{0} \qquad \rightarrow m \text{ equations.}$$

$(N+1)m$ equations for $(N+1)m$ unknowns $\underbrace{\mathbf{y}_j.}_{m \text{ components}}$

$j = 0, 1, \cdots, N.$

Example 1. $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{q}(x)$

with $\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2\lambda^3 & \lambda^2 & 2\lambda \end{pmatrix}$

has exact solution

$$\mathbf{y} = \begin{pmatrix} u \\ u' \\ u'' \end{pmatrix} \quad \textit{with} \quad u = \frac{e^{\lambda(x-1)} + e^{2\lambda(x-1)} + e^{-\lambda x}}{2 + e^{-\lambda}} + \cos \pi x.$$

BCs: $u(0),\ u(1),\ u'(1)$ *prescribed.*

Solve problem for $\lambda = 1, 50, 500.$

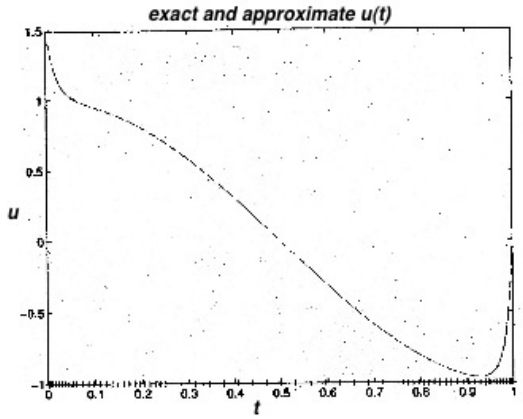

Figure 8.1: Example 1: Exact amd approximate solutions (indistinguishable) for lambda = 50, using the indicated mesh

Table 1: Maximum errors for Example 1 using the midpoint method: Uniform meshes.

N	λ	*Error*	*Rate*	λ	*Error*	*Rate*
10	*1*	*.60e-8*		*50*	*.57*	
20		*.15e-2*	*2.0*		*.32*	*.84*
40		*.38e-3*	*2.0*		*.14e-1*	*1.9*
80		*.94e-4*	*2.0*		*.34e-1*	*1.9*

N	λ	*Error*	*Rate*
10	*500*	*.96*	
20		*.90*	*.09*
40		*.79*	*.19*
80		*.62*	*.35*

Table 2: Maximum errors for Example 1 using the midpoint method: Nonuniform meshes.

N	λ	*Error*	*Rate*	λ	*Error*	*Rate*
10	*50*	*.14*		*500*	*	
20		*.53e-1*	*1.4*		*.26e-1*	
40		*.14e-1*	*1.9*		*.60e-2*	*2.1*
80		*.32e-2*	*2.2*		*.16e-2*	*1.9*

Note 2. • $\lambda = 1$ *with uniform mesh* $\rightarrow$ *good results, second-order convergence.*
(Because $\mathbf{y}(x)$ *is smooth, high-order methods could be used to obtain very accurate solutions.)*

- $\lambda = 50$ *or* 500 *with uniform mesh*
  $\rightarrow$ *accuracy, convergence rate deteriorate!*
  Errors generated from where mesh was not fine enough propagate and pollute solution!

- *Even with crude nonuniform mesh:*
  $\rightarrow$ *accurate results + second-order convergence.*
  It is possible to generate more sophisticated meshes to produce accurate solutions with N *dependent only on ETOL, NOT* λ*!*

  For multiple shooting, N grows linearly with λ*.*

8.1.1 Quasi-Linearization

Focus on Newton's method:

Newton's method applied to midpoint discretization of nonlinear BVP

is the same as

Midpoint discretization applied to Linearized BVP

$\updownarrow$

somehow apply Newton to the problem!

$\updownarrow$

quasi-linearization

Let $\mathbf{y}^{(0)}$ be an initial guess of the BVP solution and write

$$\mathbf{y}'^{(\nu+1)} = \mathbf{f}(x, \mathbf{y}^{(\nu)}) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(x, \mathbf{y}^{(\nu)})(\mathbf{y}^{(\nu+1)} - \mathbf{y}^{(\nu)})$$

$$\mathbf{0} = \mathbf{g} + \frac{\partial \mathbf{g}}{\partial \mathbf{y}(a)}\big(\mathbf{y}^{(\nu+1)}(a) - \mathbf{y}^{(\nu)}(a)\big) + \frac{\partial \mathbf{g}}{\partial \mathbf{y}(b)}\big(\mathbf{y}^{(\nu+1)}(b) - \mathbf{y}^{(\nu)}(b)\big)$$

$$\uparrow \qquad \uparrow \qquad \uparrow$$

$$\mathbf{B}_a \qquad \mathbf{B}_b$$

$\mathbf{g}, \mathbf{B}_a, \mathbf{B}_b$ all evaluated at $\mathbf{y}^{(\nu)}$.

Denoting $\mathbf{A}(x) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(x, \mathbf{y}^{(\nu)}(x))$,

we find $\mathbf{y}^{(\nu+1)}$ satisfies the linear BVP

$$\begin{aligned} \mathbf{y}' &= \mathbf{A}(x)\mathbf{y} + \mathbf{q}(x), \qquad a < x < b, \\ \mathbf{B}_a\mathbf{y}(a) + \mathbf{B}_b\mathbf{y}(b) &= \boldsymbol{\beta}, \\ \text{where} \quad \mathbf{q}(x) &= \mathbf{f}(x, \mathbf{y}^{(\nu)}(x)) - \mathbf{A}(x)\mathbf{y}^{(\nu)}(x), \\ \boldsymbol{\beta} &= -\mathbf{g}(\mathbf{y}^{(\nu)}(a), \mathbf{y}^{(\nu)}(b)) + \mathbf{B}_a\mathbf{y}^{(\nu)}(a) \\ &\quad + \mathbf{B}_b\mathbf{y}^{(\nu)}(b). \end{aligned}$$

$\rightarrow$ Defines a sequence of linear BVPs that (hopefully!) converges to the solution of the nonlinear BVP.

So if we can solve linear BVPs, we can solve nonlinear BVPs!

Note 3. *The iterates $\mathbf{y}^{(\nu)}(x)$ are never evaluated except at mesh points.*

Also, linearization and discretization commute (it doesn't matter what order you do them in). (easy to verify)

Example 2. $u'' + e^{u+1} = 0, \quad u(0) = u(1) = 0.$

Let $\mathbf{y} = \begin{pmatrix} u \\ u' \end{pmatrix}$ *then* $\mathbf{y}' = \underbrace{\begin{pmatrix} y_2 \\ -e^{y_1+1} \end{pmatrix}}_{\mathbf{f}}, \ 0 < x < 1.$

BCs (linear, homogeneous):

$$\mathbf{B}_a \mathbf{y}(0) + \mathbf{B}_b \mathbf{y}(1) = \mathbf{0},$$

where $\mathbf{B}_a = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \ \mathbf{B}_b = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$ *(verify!)*

Jacobian $\qquad \frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \begin{pmatrix} 0 & 1 \\ -e^{-y_1+1} & 0 \end{pmatrix}.$

Quasi-linearization at iteration ν *:*

$$\begin{aligned} \mathbf{y}' &= \mathbf{A}(x)\mathbf{y} + \mathbf{q}(x), \\ \textit{where} \quad \mathbf{A}(x) &= \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(x, \mathbf{y}^{(\nu)}(x)) \\ &= \begin{pmatrix} 0 & 1 \\ -e^{-y_1^{(\nu)}+1} & 0 \end{pmatrix}, \\ \mathbf{q}(x) &= \mathbf{f}^{(\nu)} - \mathbf{A}\mathbf{y}^{(\nu)}, \\ \mathbf{y} &= \mathbf{y}^{(\nu+1)}. \end{aligned}$$

Then solve linear BVP for sequence of ν's.

Use with initial guess $u^{(0)}(x) = c_2 x(1-x)$, $0 \le x \le 1$.

Obtain convergence with midpoint method and uniform mesh with $N = 10$ in 2 Newton iterations.
$c_2 = 0.5,\ 10$ gives lower, upper solution.

In practice, we solve for Newton correction at $\mathbf{y}^{(\nu)}$:

$$\begin{aligned} \boldsymbol{\eta}(x) &= \mathbf{y}^{(\nu+1)} - \mathbf{y}^{(\nu)}, \\ \text{then set} \quad \mathbf{y}^{(\nu+1)} &= \mathbf{y}^{(\nu)} + \boldsymbol{\eta}. \end{aligned}$$

$\boldsymbol{\eta}(x)$ solves the linear BVP

$$\begin{aligned} \boldsymbol{\eta}' &= \mathbf{A}(x)\boldsymbol{\eta} + \mathbf{q}(t), \quad a < x < b, \\ \mathbf{B}_a\boldsymbol{\eta}(a) + \mathbf{B}_b\boldsymbol{\eta}(b) &= \boldsymbol{\beta}, \end{aligned}$$

where $\mathbf{A}, \mathbf{B}_a, \mathbf{B}_b$ are as before.

But $\mathbf{q}, \boldsymbol{\beta}$ simplify to

$$\begin{aligned} \mathbf{q}(x) &= \mathbf{f}(x, \mathbf{y}^{(\nu)}) - \mathbf{y}'^{(\nu)} \\ \boldsymbol{\beta} &= -\mathbf{g}(\mathbf{y}^{(\nu)}(a), \mathbf{y}^{(\nu)}(b)) \end{aligned}$$

with $\mathbf{y}'^{(\nu)} = \dfrac{\mathbf{y}_n^{(\nu)} - \mathbf{y}_{n-1}^{(\nu)}}{\Delta x_n}$ for the midpoint

For midpoint applied to linear problem,

$$\frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{\Delta x_n} = \mathbf{A}(x_{n-1/2})\left(\frac{\mathbf{y}_n + \mathbf{y}_{n-1}}{2}\right) + \mathbf{q}(x_{n-1/2}),$$

$$n = 1, 2, \cdots, N,$$

$$\mathbf{B}_a \mathbf{y}_0 + \mathbf{B}_b \mathbf{y}_N = \boldsymbol{\beta}.$$

This is a large sparse linear system of $m(N+1)$ equations:

$$\mathbf{L}\mathbf{y}_\pi = \mathbf{r},$$

where

$$\mathbf{L} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{R}_1 & & & \\ & \mathbf{S}_2 & \mathbf{R}_2 & & \\ & & \ddots & \ddots & \\ & & & \mathbf{S}_N & \mathbf{R}_N \\ \mathbf{B}_a & & & & \mathbf{B}_b \end{bmatrix}, \quad \begin{matrix} \}m \\ \}m \\ \\ \\ \end{matrix} \quad \Big\updownarrow N+1$$

$$\mathbf{y}_\pi = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{N-1} \\ \mathbf{y}_N \end{bmatrix}, \qquad \mathbf{r} = \begin{bmatrix} \mathbf{q}(x_{\frac{1}{2}}) \\ \mathbf{q}(x_{\frac{3}{2}}) \\ \vdots \\ \mathbf{q}(x_{N-\frac{1}{2}}) \\ \boldsymbol{\beta} \end{bmatrix},$$

and

$$\begin{aligned} \mathbf{S}_n &= -\left[\frac{1}{\Delta x_n}\mathbf{I} + \frac{1}{2}\mathbf{A}(x_{n-1/2})\right], \\ \mathbf{R}_n &= \left[\frac{1}{\Delta x_n}\mathbf{I} - \frac{1}{2}\mathbf{A}(x_{n-1/2})\right], \\ & \qquad n = 1, 2, \cdots, N. \end{aligned}$$

- See Algorithm 8.1, p200.

8.1.2 Consistency, 0-stability, Convergence

The concepts/definitions are very similar
(if not identical in some cases) as in the IVP case.

Rather than repeat them here, we refer to the corresponding section in the text.

8.2 Solving the Linear Equations

Having discretized a linear BVP, we end up with

$$\mathbf{L}\mathbf{y}_\pi = \mathbf{r},$$

where $\mathbf{L}$ is sparse: e.g., if $m = 3$ and $N = 100$, there are 1818 possibly non-zero entries out of 91,809 (i.e., only 2% of entries are non-zero).

We need algorithms that take advantage of sparsity!
i.e., no operations with zeros
(e.g., multiplying by 0, adding 0).

Also algorithms that do not 'fill in' zeros
(or else we destroy the sparsity!).

It is very nice if $\mathbf{L}$ is banded:

$\rightarrow$ All nonzero entries are near the main diagonal.

If so, LU decomposition can run with truncated loops in $\mathcal{O}(m^3 N)$ instead of $\mathcal{O}(m^3 N^3)$.

For separated BCs, we can do it!

$$\mathbf{B}_a = \begin{pmatrix} \mathbf{B}_{a1} \\ 0 \end{pmatrix} \begin{matrix} \} \, k \\ \end{matrix} \quad , \quad \mathbf{B}_b = \begin{pmatrix} 0 \\ \mathbf{B}_{b1} \end{pmatrix} \begin{matrix} \\ \} \, m-k \end{matrix}$$

$\rightarrow$ permute rows of $\mathbf{B}_a$ to top of $\mathbf{L}$!
(Don't forget to permute the corresponding rows of $\mathbf{r}$!)

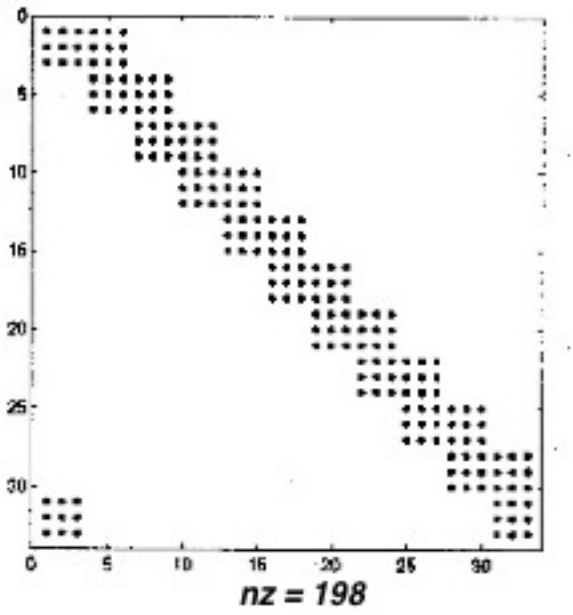

Figure 8.2: Zero-struture of the matrix L, m = 3, N = 10. The matrix size is m(N+1) = 33.

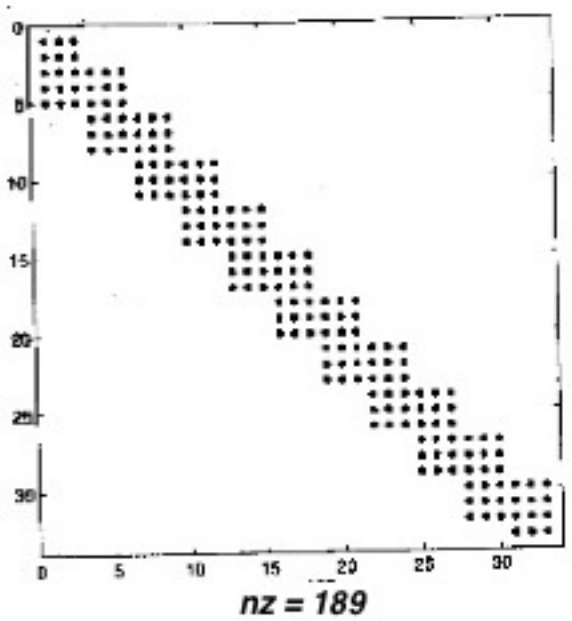

Figure 8.3: Zero-struture of the permuted matrix L with separated boundary conditions, m = 3, k = 2, N = 10.

8.3 Higher-Order Methods

Midpoint and trapezoidal methods are second order.

If you have a sufficiently smooth solution and you want high accuracy, then a higher-order method is more efficient.

Two ways to get higher order:
higher-order RK or acceleration techniques.

8.3.1 Collocation

We prefer symmetric methods for BVPs.
$\rightarrow$ collocation at Gauss or Lobatto points.

We studied these IRK methods in the context of IVPs.

For BVPs, the idea is to quasi-linearize, then write down the collocation equations for the linear system as a large sparse system for the unknown solution values at mesh points and internal stage values.

- This system can be solved all at once (but a new form of $\mathbf{L}$) or internal stage values can be eliminated locally leading to almost block-diagonal form (the old form of $\mathbf{L}$).

Example 3.

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{q}(x),$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2\lambda^3 & \lambda & 2\lambda^2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} u \\ u' \\ u'' \end{pmatrix},$$

$u(x)$ *unknown;*
$u(0), u(1), u'(1)$ *specified.*

$\lambda = 1$	*(easy) Order 6 convergence.*
$\lambda = 50$	*Good results even on uniform mesh. (Non-uniform mesh is better.)*
$\lambda = 500$	*Uniform mesh not very good; need a better non-uniform mesh to obtain full convergence order (of 4).*

Table 3: Maximum errors for Example 3 using collocation at three Gaussian points: Uniform meshes.

N	λ	*Error*	*Rate*	λ	*Error*	*Rate*
10	*1*	*.60e-8*		*50*	*.54e-1*	
20		*.94e-10*	*6.0*		*.66e-2*	*3.0*
40		*.15e-11*	*6.0*		*.32e-3*	*4.4*
80		*.24e-14*	*5.9*		*.73e-5*	*5.5*

N	λ	*Error*	*Rate*
10	*500*	*.71*	
20		*.50*	*.50*
40		*.27*	*.91*
80		*.89e-1*	*1.6*

Table 4: Maximum errors for Example 3 using collocation at three Gaussian points: Non-uniform meshes.

N	λ	*Error*	*Rate*	λ	*Error*	*Rate*
10	*50*	*.25e-2*		*500*	*.54e-3*	
20		*.12e-3*	*4.4*		*.14e-3*	*1.9*
40		*.27e-7*	*5.5*		*.75e-4*	*.90*
80		*.40e-7*	*6.1*		*.33e-4*	*1.2*

8.3.2 Acceleration techniques

Idea: Stick with a low-order discretization method, but use it more than once!

- Extrapolation:
  Idea: Apply the same method on different meshes. Use to eliminate successive Taylor series terms.

  Suppose the global error on a given mesh π looks like $\mathbf{e}_n = \mathbf{y}(x_n) - \mathbf{y}_n = \mathbf{c}(\Delta x_n)^2 + \mathcal{O}((\Delta x)^4)$ where $\mathbf{c}$ varies slowly with x, independent of Δx.

  Dividing π in half and using the same method,

$$\mathbf{e}_n = \mathbf{y}(x_n) - \tilde{\mathbf{y}}_{2n} = \frac{1}{4}\mathbf{c}(\Delta x_n)^2 + \mathcal{O}((\Delta x)^4).$$

  $\Rightarrow \quad \mathbf{y}^*_n = \frac{1}{3}(4\tilde{\mathbf{y}}_{2n} - \mathbf{y}_n)$ is $\mathcal{O}((\Delta x)^4)$!

  $\rightarrow$ can repeat to obtain even higher order.

- Deferred correction:
  Idea: Apply same method to same mesh,
  but use solutions to update equations.

  ↓

  make more accurate

  Focus on expansion of local truncation error
  (not expansion of global error).

  e.g., applying trapezoidal rule to $\mathbf{y}' = \mathbf{f}(x, \mathbf{y})$ yields

$$\mathbf{d}_n = \sum_{j=1}^{r} (\Delta x_n)^{2j} \mathbf{T}_j[\mathbf{y}(x_{n-1/2})] + \mathcal{O}((\Delta x_n)^{2r+2}),$$

$$\text{where } \mathbf{T}_j[\mathbf{z}(x)] = \frac{-1}{2^{2j-1}(2j+1)!} \mathbf{f}^{(2j)}(x, \mathbf{z}(x)).$$

  Let $\mathbf{y}_\pi = \{\mathbf{y}_n\}_{n=0}^{N}$ be the numerical solution obtained by applying trapezoidal rule to BVP on mesh π, and let $\mathbf{f}_n = \mathbf{f}(x_n, \mathbf{y}_n)$.

Then, e.g.,

$$\begin{aligned}
\mathbf{T}_1\big[\mathbf{y}(t_{n-1/2})\big] &\approx \mathbf{T}_1\big[\mathbf{y}_{n-1/2}\big] \equiv \mathbf{T}_{1,n-1/2} \\
&= \frac{1}{24(\Delta x_n)^2}(-\mathbf{f}_{n-2} + \mathbf{f}_{n-1} + \mathbf{f}_n - \mathbf{f}_{n+1}), \\
&\qquad n = 2, 3, \cdots, N-1.
\end{aligned}$$

Now add this correction to the RHS of the trapezoidal discretization

$$\begin{aligned}
\frac{\tilde{\mathbf{y}}_n - \tilde{\mathbf{y}}_{n-1}}{\Delta x_n} &= \frac{1}{2}\big[\mathbf{f}(x_n, \tilde{\mathbf{y}}_n) + \mathbf{f}(x_{n-1}, \tilde{\mathbf{y}}_{n-1})\big] + (\Delta x_n)^2 \mathbf{T}_{1,n-\frac{1}{2}}, \\
&\qquad n = 1, 2, \cdots, N, \\
\mathbf{g}(\tilde{\mathbf{y}}_0, \tilde{\mathbf{y}}_N) &= \mathbf{0}.
\end{aligned}$$

Process can be repeated for higher orders, but the approximations can become cumbersome.

- Acceleration methods are faster for simple problems.

  High-order Gauss collocation does better for difficult (stiff) problems.

8.4 More on Solving Nonlinear BVPs

Newton's method has rapid (quadratic) convergence provided the initial guess is "sufficiently good"!

For IVPs, often $\mathbf{y}_{n-1}$ is sufficiently close to $\mathbf{y}_n$.

For BVPs, no such initial guess for $\mathbf{y}_\pi$ is available.

$\rightarrow$ One of the most important aspects of a general-purpose BVP solver is the Newton solver!

8.4.1 Damped Newton

For

$$\mathbf{h}(\mathbf{y}_\pi) = \mathbf{0},$$

at iteration ν, Newton's method is :

$$\text{Solve} \quad \left(\frac{\partial \mathbf{h}}{\partial \mathbf{y}}(\mathbf{y}_\pi^{(\nu)})\right)\boldsymbol{\eta}_\pi^{(\nu)} = -\mathbf{h}(\mathbf{y}_\pi^{(\nu)})$$

$$\text{Update} \quad \mathbf{y}_\pi^{(\nu+1)} = \mathbf{y}_\pi^{(\nu)} + \boldsymbol{\eta}_\pi^{(\nu)}$$

$\rightarrow$ Take a step of length 1 in direction $\boldsymbol{\eta}_\pi^{(\nu)}$.

If $\mathbf{y}_\pi^{(\nu)}$ is still "far" from the solution $\mathbf{y}_\pi$, the iteration may diverge!

Idea: (Damped Newton)

Update $\mathbf{y}_\pi(\nu+1) = \mathbf{y}_\pi^{(\nu)} + \gamma\boldsymbol{\eta}_\pi^{(\nu)}$

$\gamma \in (0, 1]$ is the damping parameter.

chosen to ensure some minimal decrease in $\|\mathbf{h}\|$.

e.g., $\|\mathbf{h}(\mathbf{y}_\pi^{(\nu+1)})\|_2 \leq (1-\delta)\|\mathbf{h}(\mathbf{y}_\pi^{(\nu)})\|_2$

where $\delta > 0$, but small; say $\delta = 0.01$.

Now we have theory that says under some conditions, a sequence $\{\gamma_\pi^{(\nu)}\}_{\nu=1}^{\infty}$ can be found such that Newton's method will converge globally,
i.e., from any initial guess!

Note 4. • *We have not said how to compute* $\gamma_\pi^{(\nu)}$.
- Although such a sequence may exist,
we may not be able to find it!

- *It may not make sense to take a step of any non-zero length in the Newton direction from a given (sufficiently poor) iterate.*

  $\rightarrow$ *There is no substitute for a good initial guess!*

- *When successful,* $\gamma_\pi^{(\nu)} \rightarrow 1$ *as* $\nu \rightarrow \infty$;
  i.e., the local convergence properties of Newton's method kick in.

8.4.2 Shooting for initial guesses

(If possible) guess only unknown components of $\mathbf{y}(a)$ (instead of entire $\mathbf{y}_\pi^{(0)}$), and let BVP solver find $\mathbf{y}_\pi^{(0)}$.

This approach to obtaining an initial guess has obvious limitations because it involves shooting.

But the advantage is that a lot less is being asked of shooting: the goal is only to find an initial guess for the solution rather than the solution itself.

In other words, for a given guess for $\mathbf{y}(a)$, the integration just needs to get to the end ($x = b$); it does not have to also satisfy $\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \mathbf{0}$.

8.4.3 Continuation

- A general and powerful approach

Idea: Embed the given problem in a family of problems

$$\boldsymbol{\phi}(\mathbf{y}_\pi; \mu) = \mathbf{0}, \qquad \mu_0 \leq \mu \leq \mu_1,$$

where $\boldsymbol{\phi}(\mathbf{y}_\pi; \mu_0) = \mathbf{0}$ is easy (solvable),

and $\boldsymbol{\phi}(\mathbf{y}_\pi; \mu_1) = \mathbf{0}$ is the problem you want to solve.

At each continuation step, solve

$$\boldsymbol{\phi}(\mathbf{y}_\pi; \mu + \Delta\mu) = \mathbf{0} \quad \text{for} \quad \mathbf{y}_\pi(t; \mu + \Delta\mu),$$

starting with initial guess $\mathbf{y}_\pi(t; \mu)$
(or something fancier based on $\mathbf{y}_\pi(t; \mu)$).

$\rightarrow$ Can be very successful in practice.
But it is hard to automate for difficult problems.

The biggest question is: how to do the embedding ?
e.g., simple interpolation between μ_0, μ_1 may not work!

- **Example 4.** *(Modified)*
Consider

$$\begin{aligned} u'' + ee^u &= 0, \\ u(0) &= u(1) = 0. \end{aligned}$$

Suppose you have no clue for an initial guess.

Re-write problem as

$$\begin{aligned} u'' + \mu e^u &= 0, \\ u(0) &= u(1) = 0. \end{aligned}$$

Start from $\mu = 0.$ *(solution is trivial!* $u(x) \equiv 0$*)*

$\downarrow$

kills off the nonlinearity e^u

Go to $\mu = e.$ *(the problem you want to solve)*

$\downarrow$

gradually introduce the nonlinearity

But it is not possible to obtain both solutions with this one embedding!

8.5 Error Estimation and Mesh Selection

IVP solvers control local error
(because this is the more convenient).

BVP solvers control global error. (Why not?)

- Process similar to extrapolation
  e.g., for midpoint or trapezoidal method on mesh π, re-solve problem with π halved.

  Then $$\underset{\uparrow}{\tilde{\mathbf{y}}_{2n}} - \underset{\uparrow}{\mathbf{y}_n} = \tfrac{3}{4}\mathbf{c}(\Delta x)^2 + \mathcal{O}((\Delta x)^4)$$

  solution on halved π — solution on π

  So global error satisfies

$$\mathbf{e}_n = \mathbf{y}(t_n) - \mathbf{y}_n \approx \frac{4}{3}(\tilde{\mathbf{y}}_{2n} - \mathbf{y}_n)$$

and $$\mathbf{e}_{2n} = \mathbf{y}(t_n) - \tilde{\mathbf{y}}_{2n} \approx \frac{1}{3}(\tilde{\mathbf{y}}_{2n} - \mathbf{y}_n).$$

In practice, we would like a cheaper estimate in order to adapt the mesh on the fly (dynamically).
e.g., use the local leading error term

$$\hat{\mathbf{e}}_n \approx (\Delta x_n)^k \|\mathbf{y}^{(k)}(x_n)\|, \text{ for some } 1 \leq k \leq p.$$

$\rightarrow$ Often sufficient to use when selecting a mesh despite the fact that $\hat{\mathbf{e}}_n$ itself is usually not that reliable.

The next mesh is chosen to equidistribute $\hat{\mathbf{e}}_n$.
i.e., pick the next mesh so that

$$\|\hat{\mathbf{e}}_i\| \approx \|\hat{\mathbf{e}}_j\| \quad \text{for all} \quad i, j = 1, 2, \cdots, N.$$

This does not have to be precise for it to work well!

We roughly minimize $\max_{1 \leq n \leq N} \|\hat{\mathbf{e}}_n\|$ for fixed N.

N is then chosen so that

$$\max_{1 \leq n \leq N} \|\hat{\mathbf{e}}_n\| \leq ETOL.$$