Lecture 7
Stiff Equations and Boundary Value Problems
Finite Differences

▶ Forward Differences

\[ f'(x) = \frac{f(x + h) - f(x)}{h} \]

▶ Backward difference

\[ f'(x) = \frac{f(x) - f(x - h)}{h} \]

▶ Forward-Backward difference

\[ f'(x) = \frac{f(x + h) - f(x - h)}{2h} \]
Stiff equations

- occurs for sets of ODE that have very different scales in $x$
- explicit (forward) methods:
- new value $y_{n+1}$ computed explicitly from old value $y_n$
- that is a real problem:
Stiff equations

- consider
  \[ y' = cy \]
- Euler formula for stepsize \( h \) 
  \[ y_{n+1} = y_n + hy'_n = (1 + ch)y_n \]
- if \( h > 2/c \) → disaster because \( y_n \rightarrow \infty \) for \( n \rightarrow \infty \)
- → method is unstable if \( h > 2/c \)
- explicit methods require stepsizes smaller than the smallest scale of the problem!
Figure 16.6.1. Example of an instability encountered in integrating a stiff equation (schematic). Here it is supposed that the equation has two solutions, shown as solid and dashed lines. Although the initial conditions are such as to give the solid solution, the stability of the integration (shown as the unstable dotted sequence of segments) is determined by the more rapidly varying dashed solution, even after that solution has effectively died away to zero. Implicit integration methods are the cure.
Stiff equations

- that can be bad:

\[ u' = 998u + 1998v \]  \hspace{1cm} (1)
\[ v' = -999u - 1999v \]  \hspace{1cm} (2)

- initial values:

\[ u(0) = 1 \]  \hspace{1cm} (3)
\[ v(0) = 0 \]  \hspace{1cm} (4)
Stiff equations

- analytic solution:
  \[ u = 2 \exp(-x) + \exp(-1000x) \]  
  \[ v = -\exp(-x) + \exp(-1000x) \]

- explicit methods

- stepsize \( h \approx 2/c = 1/1000 \) is required for stability
Implicit Methods

- evaluate RHS at the *new y* point!
- e.g., backward Euler scheme
- consider
  \[ y' = cy \]
- Euler formula for stepsize \( h \) →
  \[ y_{n+1} = y_n + hy'_{n+1} \]
- therefore
  \[ y_{n+1} = \frac{y_n}{1 - ch} \]
- → absolutely stable!
Implicit Methods

- can take arbitrary steps
- of course, accuracy would be lost if $h$ too large
- absolute stability only for linear ODEs
- but implicit methods are always more stable
Implicit Methods

- generalization to systems of ODEs
- example: linear, constant coefficients

\[ \vec{y}' = -C \cdot \vec{y} \]

with positive definite matrix \( C \)

- explicit differencing

\[ \vec{y}_{n+1} = (1 - Ch) \cdot \vec{y}_n \]
Implicit Methods

- $A^n \to 0$ for $n \to \infty$ only if the largest eigenvalue of $A$ is $< 1$
- explicit method stable only if

$$h < \frac{2}{\lambda_{\text{max}}}$$

where $\lambda_{\text{max}}$ is the largest eigenvalue of $C$
- implicit differencing

$$\vec{y}_{n+1} = (1 + C h)^{-1} \cdot \vec{y}$$
Implicit Methods

- $\lambda$: eigenvalues of $C \rightarrow$ eigenvalues of $(1 + Ch)^{-1}$ are $(1 + \lambda h)^{-1}$
- $\rightarrow < 1$ for all $h$
- $\rightarrow$ stable!
- price: must solve matrix equations at each step
Implicit Methods

- extension to nonlinear ODEs:

\[ \ddot{y} = \ddot{f}(\dot{y}) \]

- implicit differencing

\[ \ddot{y}_{n+1} = \ddot{y}_n + h\dot{f}(\ddot{y}_{n+1}) \]

- non-linear equations!
Implicit Methods

- use Newton’s method $\rightarrow$ linearization

$$\tilde{y}_{n+1} = \tilde{y}_n + h \left[ 1 - h \frac{\partial \tilde{f}}{\partial \tilde{y}} \right]^{-1} \cdot \tilde{f}(\tilde{y}_n)$$

- if $h$ is small $\rightarrow$ use only one Newton iteration

$\rightarrow$ semi-implicit method
Stiff Equations

- generalizations of RK \(\rightarrow\) Kaps-Rentrop methods
- generalizations of Bulirsch-Stoer \(\rightarrow\) Bader/Deuflhard
- Gear’s methods (frequently used!)
2 point BCs

- consider ODE where value(s) are given at begin $x_s$ and end $x_f$ of interval
- numerical solution has to fulfill solution at both points
- more complication variations exist
- many can be reduced to the above problem
2 point BCs

\( \frac{dy_i(x)}{dx} = g(x, y_1, y_2, \ldots, y_N) \)

- example 1: Eigenvalue problem
- consider

\( \frac{dy_i}{dx} = f_i(x, y_i, \lambda) \)

- \( N + 1 \) BCs \( \rightarrow \) solution only for specific values of \( \lambda \) (eigenvalues)
2 point BCs

- eigenvalue $\lambda$ is constant $\rightarrow$
- add equation of the form

$$y_{N+1} = \lambda$$

- with

$$\frac{dy_{N+1}}{dx} = 0$$
2 point BCs

- example 2: free boundary problem
- $x_s$ is given, but $x_f$ needs to be determined from given values of $y_i$
- $\rightarrow N + 1$ conditions
2 point BCs

- add a constant ‘independent’ variable

\[ y_{N+1} = x_f - x_s \]  \hspace{1cm} (7)

\[ \frac{dy_{N+1}}{dx} = 0 \]  \hspace{1cm} (8)

- and use transformed independent variable \( 0 \leq t \leq 1 \) with

\[ x - x_s = ty_{N+1} \]

- solve \( dy_i/dt \) in \([0, 1]\)
2 point BCs

- two classes of methods:
  1. shooting
  2. relaxation
Figure 17.0.1. Shooting method (schematic). Trial integrations that satisfy the boundary condition at one endpoint are “launched.” The discrepancies from the desired boundary condition at the other endpoint are used to adjust the starting conditions, until boundary conditions at both endpoints are ultimately satisfied.
Figure 17.0.2. Relaxation method (schematic). An initial solution is guessed that approximately satisfies the differential equation and boundary conditions. An iterative process adjusts the function to bring it into close agreement with the true solution.
Pure Shooting

- let $n_1, n_2$ be the number of conditions posed at $x_1 = x_s$ and $x_2 = x_f$
- at $x_1$ we can choose $n_2$ $y$'s at will
- consider this a vector $\vec{V}$ of dimension $n_2$ so that
  \[ y_i(x_1) = y_i(x_1; \vec{V}) \]
- we can then integrate the ODE to $x_2$
Pure Shooting

- define an error vector $\vec{F}$ that measures the deviation of the computed $n_2$ BCs.
- this vector is a function of $\vec{V}$!
- goal:
  $$\vec{F}(\vec{V}) = 0$$

- $\rightarrow$ system of (non-linear) equations!
- $\rightarrow$ e.g., use Newton’s method
Pure Shooting

- Jacobian for
  \[ \mathbf{J}_\delta \mathbf{V} = -\mathbf{F} \]
  can not be computed analytically (usually!)
- approximate by numerical Jacobian, e.g.,
  \[ \frac{\partial F_i}{\partial V_j} \approx \frac{F_i(V_j + \Delta V_j) - F_i(V_j)}{\Delta V_j} \]
Pure Shooting

- caution: $\Delta V_j$ determination!
- shooting requires $n_2 + 1$ integrations of $N$ ODEs per iteration
- linear ODEs $\rightarrow$ only one cycle required
Shooting to a fitting point

- pure shooting can cause trouble
- e.g., for some estimates the solution may be totally off
- or very sensitive to guesses
- Newton method may not converge etc.
- related problems: singular points at $x_1$ and/or $x_2$
- shooting to a fitting point:
- start from $x_1$ and $x_2$ and integrate to an intermediate point inside $[x_1, x_2]$
- $\rightarrow N$ independent variables
Shooting to a fitting point

- at the fitting point $x_m$ we have
  \[ y_i(x_m; \vec{V}_1) = y_i(x_m, \vec{V}_2) \]

- $\vec{V}_1$: the $n_2$ guesses at $x_1$
- $\vec{V}_2$: the $n_1$ guesses at $x_2$
- use $N$ dimensional Newton method to find solution
Relaxation methods

- basic idea:
- convert the ODE into an FDE by discretization
- FDE = finite difference equation
- needs a ’mesh’ of points over the integration interval
Relaxation methods

▶ example: write ODE

\[ \frac{dy}{dx} = g(x, y) \]

▶ possible FDE discretization:

\[ y_k - y_{k-1} - (x_k - x_{k-1})g(x_{k-1/2}, y_{k-1/2}) = 0 \]

▶ \( k \) mesh points \( x_k \) and results \( y_k \)

▶ \( x_{k-1/2} = (x_k + x_{k-1})/2 \) etc.
Relaxation methods

- many possible ways to discretize!!
- $N$ equations with $M$ mesh points $\rightarrow N \times M$ unknowns
- start with a guess and iterate to improve
- $\rightarrow$ relax to the solution
- classical iteration scheme: Newton (did you guess?)
- produces matrix equation of the order $NM$
- than can be huge!
- but matrix has special form $\rightarrow$ solution easier
Relaxation methods

- consider problem from above (shooting)
- define mesh of $M \times$ points $x_1$ to $x_M$
- first and last correspond to interval
- $\vec{y}_k$: vector of the $N$ dependent vars at point $x_k$
- FDE may look like

$$
\vec{E}_k(\vec{y}_k, \vec{y}_{k-1}) = \vec{y}_k - \vec{y}_{k-1} - (x_k - x_{k-1})\vec{g}_k(x_{k-1/2}, \vec{y}_{k-1/2}) = 0
$$

- **backward difference**
- **forward differencing** also possible!
Relaxation methods

- $N$ equations for $2N$ variables (points $k$ and $k-1$)
- $M-1$ of those $\rightarrow (M-1)N$ equations
- remaining $N$ from the BCs at $x_1$ and $x_M$!
- first BC has $n_1$ non-zero components:
  \[
  \vec{E}_1 = \vec{B}(x_1, y_1)
  \]
- best for later: last $n_1$ are non-zero
- similar: at $x_M$ we have the first $n_2$ non-zeros:
  \[
  \vec{E}_{M+1} = \vec{C}(x_M, y_M)
  \]
Relaxation methods

- in general: non-linear system
- generate $MN$ guesses $y_{j,k}$ and use Newton’s method to compute corrections $\Delta y_{j,k}$ via
- interior points:

$$\vec{E}_k(\vec{y}_k, \vec{y}_{k-1}) + \sum_n \frac{\partial \vec{E}_k}{\partial y_{n,k-1}} \Delta y_{n,k-1} + \sum_n \frac{\partial \vec{E}_k}{\partial y_{n,k}} \Delta y_{n,k} = 0$$

- similar on the boundaries
- example: $N =$ variables, $M =$ 4 mesh points, 3 BCs at first point, 2 at the last point
Figure 17.3.1. Matrix structure of a set of linear finite-difference equations (FDEs) with boundary conditions imposed at both endpoints. Here $X$ represents a coefficient of the FDEs, $V$ represents a component of the unknown solution vector, and $B$ is a component of the known right-hand side. Empty spaces represent zeros. The matrix equation is to be solved by a special form of Gaussian elimination.
Relaxation methods

- *block* matrix (band structure)
- can be solved efficiently!
- iterate until corrections are small enough for your requirements
- use analytic change of variables to deal with difficult $y$’s
- example: implicit non-linear EOSs etc.
Mesh point allocation

- first idea: distribute points uniformly over interval
- then double number of points
- estimate truncation error
- dynamically allocate mesh points?
  Dorfi & Drury, *Simple Adaptive Grids for 1-D initial value problems*
- consider $x$ as a dependent variable
- define $q$ so that $q = 1$ is the first mesh point and $q = M$ the last
△q = 1 is the difference between two mesh points

transform to q →

\[ \frac{\dot{y}}{dq} = \vec{g} \frac{dx}{dq} \]

FDE version →

\[ y_k - y_{k-1} - \frac{1}{2} \left[ \left( \vec{g} \frac{dx}{dq} \right)_k + \left( \vec{g} \frac{dx}{dq} \right)_{k-1} \right] = 0 \]

important: \( dx/dq \) goes with \( \vec{g} \)
Mesh point allocation

- $dq/dx$ is mesh point density
- this density is related to the change in $y(x)$
- make up formula that is proportional to $dq/dx$
- guarantee that integral of mesh density is $M$?
Mesh point allocation

- use a linear function $Q(q)$ to adjust value of integral as required:

$$\frac{dQ}{dq} = \psi \quad \frac{d\psi}{dq} = 0$$

- $\psi$ is an intermediate variable that is there to allow easy prescription of actual mesh density
Mesh point allocation

- for example

\[
\frac{dQ}{dx} = \frac{dQ}{dq} \frac{dq}{dx} \equiv \phi(x)
\]

- write

\[
\frac{dx}{dq} = \frac{\Psi}{\phi(x)}
\]

- this adds 3 ODEs and we need to prescribe \( \phi(x) \)
Mesh point allocation

► example: uniform spacing

\[ \frac{dQ}{dx} = \phi(x) = \frac{1}{\Delta} \]

► more complicated:

\[ \frac{dQ}{dx} = \phi(x) = \frac{1}{\Delta} + \left| \frac{dy/dx}{y\delta} \right| \]

► \( \log \) change in \( y \) of \( \delta \) as ’attractive’ as a change in \( x \) of \( \Delta \)