10.34 Numerical Methods Applied to Chemical Engineering Fall 2015

Quiz #2 Review

Study guide based on notes developed by J.A. Paulson, modified by K. Severson

In this section of the course, we learned ordinary differential equations. Specifically we looked at IVPs, DAEs, and BVPs. We also spent a lecture on numerical integration.

Initial Value Problems (IVPs)

1. Solve 1st order ODE of the form

$$\begin{aligned} \frac{d}{dt}\mathbf{x}(t) &= \mathbf{f}(\mathbf{x}(t), t) \\ \mathbf{x}(t_0) &= \mathbf{x_0} \end{aligned}$$

2. Write differential equations as a system of first order ODEs

   (a) Ex. Rewrite the N-th order ODE system for $x(t)$ in terms of the new dependent variable $\mathbf{u}(t)$, where $\mathbf{u} = [u_1\ u_2\ u_3, \ldots,\ u_N]^T$ and $u_1 = x$, $u_2 = \frac{dx}{dt}$, $u_3 = \frac{d^2x}{dt^2}$, etc.

   (b) See HW6 Problem 2

3. Finite difference (FD) approximations for derivatives

   (a) Order of accuracy

   - Use Taylor expansion to evaluate the accuracy of FD approximations:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \frac{d}{dx}\mathbf{x}(t) + \frac{(\Delta t)^2}{2!}\frac{d^2}{dt^2}\mathbf{x}(t) + \ldots$$
$$\mathbf{x}(t - \Delta t) = \mathbf{x}(t) - \Delta t \frac{d}{dx}\mathbf{x}(t) + \frac{(\Delta t)^2}{2!}\frac{d^2}{dt^2}\mathbf{x}(t) - \ldots$$

   - Derivative $\mathbf{f}(\mathbf{x}, t)$ is said to be p-th order accurate if $Error = E(\Delta t) = \mathbf{f}(\mathbf{x}, t)_{actual} - \mathbf{f}(\mathbf{x}, t)_{predicted} \sim (\Delta t)^p$

   (b) Forward difference: 1st order accurate, i.e. $E(\Delta t) \sim \Delta t$

$$\frac{d}{dt}\mathbf{x}(t) \approx \frac{1}{\Delta t}\left(\mathbf{x}(t + \Delta t) - \mathbf{x}(t)\right)$$

   (c) Backward difference: 1st order accurate, i.e. $E(\Delta t) \sim \Delta t$

$$\frac{d}{dt}\mathbf{x}(t) \approx \frac{1}{\Delta t}\left(\mathbf{x}(t) - \mathbf{x}(t - \Delta t)\right)$$

   (d) Central difference: 2nd order accurate, i.e. $E(\Delta t) \sim (\Delta t)^2$

$$\frac{d}{dt}\mathbf{x}(t) \approx \frac{1}{2\Delta t}\left(\mathbf{x}(t + \Delta t) - \mathbf{x}(t - \Delta t)\right)$$

4. Local vs. global truncation error for numerical integration methods

   (a) Local truncation error (LTE): difference between the approximation ($\hat{\mathbf{x}}(t_k)$) and the exact solution started from the previous solution ($\mathbf{x}^*(t_k)$)
   $LTE = \|\hat{\mathbf{x}}(t_k) - \mathbf{x}^*(t_k)\|$
   where $\mathbf{x}^*(t_k)$ is the exact solution to $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t)$ with I.C. of $\mathbf{x}(t_{k-1}) = \hat{\mathbf{x}}(t_{k-1})$

   (b) Global truncation error (GTE): difference between the approximation ($\hat{\mathbf{x}}(t_k)$) and the exact solution ($\mathbf{x}(t_k)$)
   $GTE = \|\hat{\mathbf{x}}(t_k) - \mathbf{x}(t_k)\|$

5. Accuracy (order of the method): order of accuracy for the derivative approximation $\mathbf{f}(\mathbf{x}(t), t)$

6. Stability

   (a) Test problem for 1st order ODE to determine the region of absolute stability of the method in the complex plane:

$$\frac{d}{dt}x(t) = \lambda x(t) = f(x, t)$$

   (b) $\left|\frac{\hat{x}(t_{k+1})}{\hat{x}(t_k)}\right| \leq 1$ for numerical stability

   (c) Stability of the ODE problem vs. stability of the method
   - Numerical method is stable if as $k \to \infty$, $|\hat{x}(t_k)|$ is bounded.
   - Original problem is stable if as $t \to \infty$, $|x(t)|$ is bounded.
   - Numerical method and the original system can have different stability region.
   - Stability and accuracy do not necessarily correlate with one another

7. Explicit vs. implicit methods: A method is said to be *explicit* if the next state $\hat{\mathbf{x}}(t_{k+1})$ is based only on past past and current values, i.e. values of $\hat{\mathbf{x}}(t)$ for $t < t_{k+1}$. Otherwise, the method is said to be *implicit*. We can think of implicit methods as "fsolve" type methods.

8. Explicit/Forward Euler method

$$\hat{\mathbf{x}}(t + \Delta t) = \hat{\mathbf{x}}(t) + (\Delta t)\mathbf{f}(\hat{\mathbf{x}}(t), t)$$

   (a) $LTE \sim (\Delta t)^2$

   (b) $GTE \sim \Delta t$

   (c) 1st order accurate method

   (d) Condition for numerical stability: $|1 + \lambda\Delta t| \leq 1$

9. Implicit/Backward Euler method

$$\hat{\mathbf{x}}(t + \Delta t) = \hat{\mathbf{x}}(t) + (\Delta t)\mathbf{f}(\hat{\mathbf{x}}(t + \Delta t), t + \Delta t)$$

   (a) $LTE \sim (\Delta t)^2$

   (b) $GTE \sim \Delta t$

   (c) 1st order accurate method

   (d) Condition for numerical stability: $|1 - \lambda\Delta t| \geq 1$

   (e) Need to solve system of non-linear equations

- Newton's method
- Predictor-corrector method
  Predictor step: Explicit solve gives a better initial guess
  Corrector step: Implicit solve

10. Example: trapezoidal method

    (a) Average of the forward and backward Euler methods

$$\frac{1}{\Delta t}\left(\hat{x}(t_{k+1}) - \hat{x}(t_k)\right) = \frac{1}{2}\left(f(\hat{x}(t_k), t_k) + f(\hat{x}(t_{k+1}), t_{k+1})\right)$$

    (b) Is this method explicit, implicit, neither?

    (c) Apply the method to the test equation $dx/dt = \lambda x$ with $\lambda \in \mathbb{C}$ to determine the region of absolute stability of the method in the complex plane. How does the stability of the test equation compare qualitatively with the stability of the method?

    (d) Show that the method is 2nd-order accurate.

11. Stiffness: ratio of eigenvalues,

$$\frac{\max\limits_i \{|\mathrm{Re}(\lambda_i)|\}}{\min\limits_i \{\mathrm{Re}(|\lambda_i|)\}}$$

    where λ_i's are eigenvalues of the Jacobian matrix, $\frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}, t)$. For stiff problems, use *adaptive time stepping*.

12. Numerical integration

    (a) Any set of definite integrals can be numerical integrated by rewriting as an IVP,

$$\mathbf{x}(t) = \int_{t_0}^{t} \mathbf{f}(\tau) d\tau$$

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(t), \ \mathbf{x}(t_0) = \mathbf{0}$$

    More efficient with adapative time stepping; integrate over continuous regions (skip discontinuities)

    (b) Polynomial interpolation
    Ex. midpoint rule, trapezoidal rule, Simpson's rule

    (c) Gaussian quadrature approximates the integral using a weighted sum of function values at specified points. This approach will yield an exact result for polynomials of degree $2n-1$ or less for particular choices of the points x_i and weights w_i for $i = 1 \ldots n$.

Differential-Algebraic Equations (DAEs)

1. Fully-implicit form

    (a) The most general form of DAEs is the fully implicit form

$$\mathbf{F}(\mathbf{z}, \dot{\mathbf{z}}, t) = \mathbf{0}, \quad \mathbf{z}(0) = \mathbf{z}_0$$

    where $\dot{\mathbf{z}} := \frac{d}{dt}\mathbf{z} \in \mathbb{R}^{n_z}$.

(b) Note that only first-order derivatives are allowed in this definition. You can always rewrite higher order derivatives by defining new state variables.

(c) Generally, $\mathbf{z}$ can be composed of differential states $\mathbf{x}$ (i.e., states for which derivatives appear in $\mathbf{F}$) and algebraic states $\mathbf{y}$ (i.e., states for which derivatives do not appear in $\mathbf{F}$).

(d) A DAE is well posed when the dimension of $\mathbf{F}$ is equal to the dimension of the state vector $\mathbf{x}$.

2. Semi-explicit form

(a) Many chemical processes can be written in semi-explicit form

$$\mathbf{M}\frac{d}{dt}\mathbf{z}(t) = \mathbf{f}(\mathbf{z}(t), t), \quad \mathbf{z}(0) = \mathbf{z}_0$$

where $\mathbf{M}$ denotes the mass matrix.

(b) If $\mathbf{M}$ is full rank, then the system is an ODE as you can rewrite the system as

$$\frac{d}{dt}\mathbf{z}(t) = \mathbf{M}^{-1}\mathbf{f}(\mathbf{z}(t), t)$$

whereas it is a DAE (not an ODE) if $\mathbf{M}$ is singular.

(c) Commonly, chemical processes are naturally written in the form of a simplified version of the semi-explicit form

$$\begin{aligned}\frac{d}{dt}\mathbf{x}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{y}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{0} &= \mathbf{g}(\mathbf{x}(t), \mathbf{y}(t), t), \quad 0 = \mathbf{g}(\mathbf{x}_0, \mathbf{y}_0, t)\end{aligned}$$

where $\mathbf{x}(t)$ and $\mathbf{y}(t)$ denote the differential and algebraic states, respectively. This can be rewritten in the mass matrix form as follows

$$\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}(t), \mathbf{y}(t), t) \\ \mathbf{g}(\mathbf{x}(t), \mathbf{y}(t), t) \end{bmatrix}$$

(d) The semi-explicit form is index 1 if and only if $\frac{\partial \mathbf{g}}{\partial \mathbf{y}}$ is non-singular ($\det\left[\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right] \neq 0$). We can see this by differentiating the algebraic equations once

$$\mathbf{0} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\frac{d\mathbf{x}}{dt} + \frac{\partial \mathbf{g}}{\partial \mathbf{y}}\frac{d\mathbf{y}}{dt} + \frac{\partial \mathbf{g}}{\partial t}\frac{dt}{dt} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \mathbf{g}}{\partial \mathbf{y}}\dot{\mathbf{y}} + \frac{\partial \mathbf{g}}{\partial t}$$

Rearranging we write,

$$\frac{d\mathbf{y}}{dt} = -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^{-1}\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}, \mathbf{y}, t) + \frac{\partial \mathbf{g}}{dt}\right)$$

If we combine this we the set of differential equations above, we obtain

$$\begin{bmatrix} I & 0 \\ 0 & \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{x}(t), \mathbf{y}(t), t) \\ -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)\mathbf{f}(\mathbf{x}(t), \mathbf{y}(t), t) - \frac{\partial \mathbf{g}}{\partial t} \end{bmatrix}$$

We have an ODE set after one derivative (index 1) if the mass matrix is non-singular, which is only true when $\frac{\partial \mathbf{g}}{\partial \mathbf{y}}$ is non-singular. The DAE must be higher index when this condition is not met.

3. Numerical simulation

   (a) The derivation of numerical methods for DAEs is very similar to ODEs.

   (b) Use a forward Euler for the differential variables in the semi-explicit form. However, you must solve the algebraic equations implicitly so that the overall method is implicit

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \big(\Delta t\big)\mathbf{f}(\mathbf{x}_k, \mathbf{y}_k, t_k)\\ \mathbf{0} &= \mathbf{g}(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}, t_{k+1})\end{aligned}$$

   where variables at k imply variables evaluated at $t = t_0 + k\Delta t$.

   (c) We might instead use backward Euler (as done on the homework)

$$\begin{aligned}\mathbf{x}_k &= \mathbf{x}_{k-1} + \Delta t\mathbf{f}(\mathbf{x}(t_k), \mathbf{y}_k, t_k)\\ 0 &= \mathbf{g}(\mathbf{x}(t_k), \mathbf{y}(t_k), t_k)\end{aligned}$$

   (d) As for ODEs, computational times are greatly reduced for stiff systems by using adaptive time-stepping.

   (e) DAEs that are index 1 are straightforward to simulate using generalizations of algorithms derived for ODEs (e.g. ode15s, ode23s) while higher index problems require a reformulation before the numerical methods can be applied.

4. Index

   (a) The differential index (or **index** for short) is the minimum number of derivatives of the original DAE system required to generate a complete system of first-order ODEs.

   (b) An example set of equations is

$$\begin{array}{l}\dot{x}_1 + x_2 = g_1(t)\\ x_1 = g_2(t)\end{array} \quad \xrightarrow{\frac{d}{dt}} \quad \begin{array}{l}\ddot{x}_1 + \dot{x}_2 = \dot{g}_1(t)\\ \boxed{\dot{x}_1 = \dot{g}_2(t)}\end{array} \quad \xrightarrow{\frac{d}{dt}} \quad \begin{array}{l}\dddot{x}_1 + \ddot{x}_2 = \ddot{g}_1(t)\\ \ddot{x}_1 = \ddot{g}_2(t)\end{array}$$

   After two derivatives, we can substitute $\ddot{x}_1 = \ddot{g}_2(t)$ into the first model equation to derive a complete set of first-order ODEs

$$\begin{aligned}\ddot{g}_2(t) + \dot{x}_2 &= \dot{g}_1(t)\\ \dot{x}_1 &= \dot{g}_2(t)\end{aligned}$$

   Therefore, the index of this problem is **two**.

5. Implicit constraints

   (a) The solution to a DAE system may be required to satisfy hidden/implicit constraints (as well as explicit constraints). Higher index DAEs (index > 1) include some hidden constraints. DAEs with index 1 very rarely have implicit constraints.

   (b) In the example above, there is an implicit constraint $\dot{x}_1 = \dot{g}_2(t)$ that must be satisfied.

6. Consistent initialization

(a) Consistent initialization for the implicit DAE $\mathbf{F}(\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{y}(t), \mathbf{u}(t), t) = \mathbf{0}$, where $\mathbf{u}(t)$ denotes inputs/forcing functions, must satisfy the original DAE

$$\mathbf{F}(\mathbf{x}(t_0), \dot{\mathbf{x}}(t_0), \mathbf{y}(t_0), \mathbf{u}(t_0), t_0) = \mathbf{0}$$

We denote the number of differential and algebraic variables as n and m, respectively. The **maximum** number of degrees of freedom (DOF) that can be specified is

$$\begin{array}{rl} (2n+m) & \text{variables } \{\mathbf{x}(t_0), \dot{\mathbf{x}}(t_0), \mathbf{y}(t_0)\} \\ \underline{-(n+m)} & \text{equations } \{\mathbf{F}(\mathbf{x}(t_0), \dot{\mathbf{x}}(t_0), \mathbf{y}(t_0), \mathbf{u}(t_0), t_0) = \mathbf{0}\} \\ n & \text{DOFs} \end{array}$$

The DOFs is less for systems with index > 1 due to implicit constraints discovered when determining the index.

(b) For the example above, we must specify $\{x_1(t_0), \dot{x}_1(t_0), x_2(t_0)\}$. We must satisfy the original equations and the implicit constraint found previously

$$\begin{array}{ll} \dot{x}_1(t_0) + x_2(t_0) = g_1(t_0) & \\ x_1(t_0) = g_2(t_0) & \longrightarrow \quad x_1(t_0) = g_2(t_0) \\ \dot{x}_1(t_0) = \dot{g}_2(t_0) & \end{array} \qquad \begin{array}{l} x_2(t_0) = g_1(t_0) - \dot{g}_2(t_0) \\ \\ \dot{x}_1(t_0) = \dot{g}_2(t_0) \end{array}$$

7. Pros and cons of DAEs vs. ODEs

(a) ODEs are easier to initialize and more solvers are readily available. However, many index-1 DAE solvers exist nowadays that this is not a huge issue. Higher index problems are difficult to handle such that they must be reformulated to an equivalent index-1 problem using the method of auxiliary variables.

(b) Fast explicit methods are available for non-stiff ODEs while DAE solvers must be implicit. However, note that implicit solvers have favorable stability properties.

(c) DAEs exactly satisfy algebraic constraints, which may be important for certain applications. For example, overall mass or energy balances can be exactly satisfied (within double precision) when using a DAE solver. These constraints are not explicitly satisfied when numerically solving the ODE version.

Boundary Value Problems (BVPs)

1. Problem formulation: 1st order ODEs

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

B.C. defined for sub-vectors of $\mathbf{x}$: $\mathbf{x_1}(0) = \mathbf{x_{1,0}}$, $\mathbf{x_2}(T) = \mathbf{x_{2,0}}$

2. Boundary condition formulations:

(a) In general, $\mathbf{g}(\mathbf{x}(0), \mathbf{x}(T)) = \mathbf{0}$.

(b) Linear case: $\mathbf{g} = \mathbf{B_0}\mathbf{x}(0) + \mathbf{B_T}\mathbf{x}(T) + \mathbf{b} = \mathbf{0}$

3. Shooting method - Prof. Green liked to remind us to "shoot first, then relax"

(a) Converts the BVP into an IVP where some of the initial conditions are unknown

(b) Guess initial conditions, solve the IVP, find the residual of the boundary points, update the initial conditions and repeat until convergence

(c) Look for the initial conditions $\mathbf{x}(0) = \mathbf{c}^*$ such that

$$\mathbf{G}(\mathbf{c}^*) = \mathbf{g}(\mathbf{c}^*, \mathbf{x}(T; \mathbf{c}^*)) = \mathbf{0}$$

(d) Newton's method: need to evaluate Jacobian $\mathbf{J} = \frac{\partial \mathbf{G}}{\partial \mathbf{c}}$

(e) May have no solutions, one solution or many solutions

4. Relaxation methods (aka method of weighted residuals)

(a) In general, relaxation methods attempt to solve a problem that is "close" to the one we have

i. $y_n(t) = \sum_{k=1}^{K} d_{nk}\phi_k(t)$

ii. Need to solve for the coefficients d_k $\forall k$

(b) Basis functions

i. The basis functions $\phi_k(t)$ must be linearly independent

ii. It is often nice to choose basis functions that "automatically" satisfy the boundary conditions but it is not a requirement.

iii. Two types of basis functions: global and local
Global examples: $\sin(k\pi x)$, e^{kix}, etc.
Local examples: "tent functions"

(c) Collocation

i. Choose a set of points $\{t_m\}$ such that $\mathbf{g}(t_m) = 0$

ii. Solve for the coefficients using Newton's method

iii. Generally have N boundary conditions and MN equations therefore $K = M + 1$ (Note: that this changes when BC's are satisfied automatically)

(d) Galerkin's method

$$\int_{t_0}^{t_f} \phi_k(t) g_n(t) dt = 0$$

i. Forces the error of the approximation to be orthogonal to the basis functions, averaged over the domains

ii. Collocation is a special case of the Galerkin method where the basis functions are delta functions

*Disclaimer: This document in not the gospel of numerical. There may be typos and you should always review your notes! Good luck with the exam!

10.34 Numerical Methods Applied to Chemical Engineering

Fall 2015