Exam_1

February 18, 2015

1 ChEn 541: Exam 1: Winter 2015

2 Due Friday October 20, BEFORE 9:00 AM

3 Problem 1

3.1 Closed Book, Notes, Internet.

3.1.1 Part a

If a stepsize of 0.25 seconds is needed to achieve a given integration accuracy using the Modified Euler method, what stepsize would be needed to get the same accuracy using the fourth order Runge Kutta method?

3.1.2 Part b

Write down the formula for the Modified Midpoint method.

3.1.3 Part c

For each of the following curves/methods, show graphically the progression of the solution for the root. Continue the labelling: a, b, c, d, e, etc. The starting points a (and optionally b) are given.

3.1.4 Part d

What is the reasonable range of ω for the SOR method?

3.1.5 Part e

How does the cost of Gauss elimination scale with the number of unknowns? What about the Thomas Algorithm?

3.1.6 Part f

When performing floating point operations, which of the following would you expect to present more roundoff error problems: (a) forming the product of some number of numbers; (b) forming the sum of some number of numbers.

3.1.7 Part g

Derive a second order accurate finite difference (FD) approximation of the derivative $f'(x)$, evaluated at point 0. The FD approximation should be written only in terms of points 0, 1, and 2. You will need to write Taylor series about points 1 and 2, evaluated at the wall.

3.1.8 Part h

What method would you use to solve each of the following linear systems: (a) A tridiagonal system of 1 million equations; (b) A heptadiagonal (7 diagonals of the matrix have elements, the rest are zero) system of 1 million equations; (c) a sparse system of 1 million equations; (d) a dense system (matrix) with 50 equations;

3.1.9 Part i

Which method is less expensive to compute: Gauss elimination, or LU decomposition? Then give a reason why you would ever use the more expensive one.

3.1.10 Part j

Why are stiff ODE systems hard to solve?

3.1.11 Part k

What kind of solution approach do we usually have to use to solve stiff systems?

3.1.12 Part l

Write the formula used for Numerically computing the $J_{i,j}$ element of the Jacobian matrix as relates to solving the ODE system: $dy/dt = f(y)$. Make sure you include how to compute the stepsize.

4 Problem 2

4.1 Open book, open notes. Closed homework, homework solutions, and closed code examples. You may use the internet for Matlab and Python help only.

4.1.1 Part a

A chemical reaction occurs in a continuous stirred tank reactor (CSTR). There is an inlet, and an outlet. The contents are perfectly mixed and the outlet composition is the same as the reactor composition.

The reactions are

A + B –> C (reaction 1)

B + C –> D (reaction 2)

The reaction rates for the species are given by

$$\begin{aligned} r_A &= -k_1[A][B] \\ r_B &= -k_1[A][B] - k_2[B][C] \\ r_C &= k_1[A][B] - k_2[B][C] \\ r_D &= k_2[B][C] \end{aligned}$$

The mass balance for each species in the CSTR is

$$\frac{d[Y]}{dt} = \frac{[Y]_{in} - [Y]}{\tau} + r,$$

where $[Y]$ is one of $[A]$, $[B]$, $[C]$, or $[D]$. Also, τ is the residence time in the reactor (volume divided by volumetric flow rate).

Solve for the concentrations $[Y]$ as functions of time to an end time of 5τ as described below. Also solve for the selectivity $S = C/(C + D)$, where C is a desired product and D is undesired. (This is initially undefined, so set the initial selectivity to 1.0.

The initial concentration in the tank is $[B]_0 = 1.0$ and all other species are zero.

The inlet concentrations are $[A]_{in} = 1.0$, and $[B]_{in} = 1.0$.

Take $\tau = 10.0$, $k_1 = 1.0$, $k_2 = 5.0$. Use an initial stepsize of $dt = 0.01$.

You should plot the concentrations and selectivity on the same plot. (Include a legend.) Use a linestyle that includes symbols so the step size is clear. On a separate plot, show the stepsize versus time. Use a log scale for the stepsize.

You should solve the problem using the Implicit Euler method. This is a nonlinear, multidimensional problem. You will need a nonlinear solver. Use Newton's method with an analytic Jacobian. You can use a built-in (Matlab or Python) linear solver. Set the step size using the step doubling technique. Use a relative tolerance of 0.005 for all error testing, and use the 2-norm. You may want to solve this in steps, making sure things work before adding complexity.

4.1.2 Part b

Write a function to solve the system $Ax = b$ using the Gauss Seidel method.

In Matlab fill in your function as:

```
function x = linear_gs(A, b)
   % write the solution here
end
```

In Python, fill in your function as:

```
def linear_gs(A,b) :
    # write the solution here
    return x
```