Math 128A, Summer 2019
Lecture 30, Tuesday , 8/13/2019

1 Review: (Past) Final Exam Questions

Recall from 2019, we have the following problem:

$$\int_0^1 f(t)\ dt = af(1) + bf(0) + cf(-\theta),$$

for $\theta \in [-\frac{1}{2}, 2]$. The zeroth question is what is the highest degree polynomial f for which this integration rule will be exact? Turns out we can do so for degree up to 2.
Taking $f(t)$ to be $1, t, t^2$, we get the following system of equations (via the monomial basis):

$$\begin{aligned} 1 &= a + b + c \\ \frac{1}{2} &= a \cdot 1 + b \cdot 0 + c \cdot (-\theta) \\ \frac{1}{3} &= a \cdot 1^2 + b \cdot 0^2 + c(-\theta)^2, \end{aligned}$$

or equivalently we can look at the Newton basis (with malice of forethought to give a triangular system):

$$f(t) = f_0 + f_1(t) + f_2 t(t + \theta),$$

with

$$\begin{aligned} f_0 &= 1 \\ f_1 &= t \\ f_2 &= t(t + \theta) \end{aligned}$$

to give

$$\begin{aligned} \int_0^1 1dt = 1 &= a + b + c \\ \int_0^1 t\ dt = \frac{1}{2} &= a + b \cdot 0 + c(-\theta) \\ \int_0^1 t(t + \theta)\ dt = \frac{1}{3} + \frac{\theta}{2} &= a \cdot 1(1 + \theta) + b \cdot 0 \cdot (0 + \theta) + c(-\theta)(-\theta + \theta) \end{aligned}$$

or equivalently, we can use the Lagrange basis (diagonal linear system), although solving the integrals can be

1.1 3B

Suppose we have an **implicit variable-step Adams method**, where:

$$\begin{aligned} t_n - t_{n-1} &= \theta h \\ t_{n+1} - t_n &= h, \end{aligned}$$

we and an integration formula to take t_n to t_{n+1} into 0 to 1:

$$y(t_{n+1}) = y(t_n) + \underbrace{\int_{t_n}^{t_{n+1}} y'(s)\ ds} = f(s, y(s))\ ds,$$

and so transform this into

$$h\int_0^1 y'(t_n + th_1)\ dt = h[ay'(t_{n+1}) + by'(t_n) + cy'(t_{n-1})]$$

This means that our Adams method is

$$u_{n+1} = u_n + h[af_{n+1} + bf_n + cf_{n-1}]$$

Take the interpolating polynomial for y' :

$$p(t_{n+j}) = f_{n+j}, \qquad j = 1, 0, -1$$

Calculating the truncation error is simply:

$$h\tau_{n+1} := \int_{t_n}^{t_{n+1}} y'(s) - p(s)\ ds,$$

where we can use taylor series to expand; however in this case we know everything there is to know about polynomial interpolation, set $w := y'$, so:

$$\begin{aligned} w(s) - p(s) = y'(s) - p(s) &= \frac{w'''(\xi)}{3!}(s - s_{n+1})(s - s_n)(s - s_{n-1}) \\ &= \frac{y^{(4)}(\xi_s)}{3!}[(s - s_{n+1})(s - s_n)(s - s_{n-1})] \end{aligned}$$

(As an aside, the dimensions of local truncation error (in euler) indicate that it is a measure of error in y', not y. That is, $\tau_{n+1} = \frac{y_{n+1} - y_n}{h}$.)
We want to pull the derivative factor out, and this is legal precisely as long as $\omega(t)$ does not change sign on the domain of integration.
Thus, from above,

$$h\tau = y'(s) - p(s) = \frac{y^{(4)}(\xi)}{3!} \underbrace{\boxed{\int_{t_n}^{t_{n+1}} (s - t_{n+1})(s - t_n)(s - t_{n-1})\ ds}}_{O(h^4)}$$

So dividing across by h we get the desired result. We set $\theta \in [\frac{1}{2}, 2]$ to control the scaling of the step size so that our method is stable.

1.2 Computing the Cholesky Factorization

We first 'construct' a matrix that has a valid Cholesky Factorization:

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 0 & 3 & 5 \\ 0 & 0 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{bmatrix} =: A.$$

Now given the matrix A on the right, we want to factorize. We can write:

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ b & d & 0 \\ c & e & f \end{bmatrix} \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 + d^2 & \\ ac & & \end{bmatrix}$$

If we want to specify an algorithm, we could perform gaussian elimination to perform this operation.

1.3 Error: Hermite Interpolation

We have

$$f(t) - p(t) = \frac{f^k(\xi)}{k!}[(t-t_1)_1^m \cdots (t-t_n)_n^m],$$

so the degree is $k = m_1 + \cdots + m_n$. Check that each of these three factors is inevitable.
Suppose:

$$f(t) = \sum_{k=-17}^{17} f_k e^{ikx},$$

where in order to perform this summation because it takes 35 operations. Evaluating the derivative is still relatively cheap:

$$f'(t) = \sum_{k=-17}^{17} f_k (ik) e^{ikt}$$

Hermite interpolation can be especially cheap on a mesh with equispaced points. If we are doing this via an FFT (forward), then this is especially cheap.
Additionally, we can evaluate the function at particular points as well as its derivatives at those points. Storing these into a table, we can just do a table lookup to eliminate the computation cost for calculating the derivatives at desired Hermite interpolation points.

2 Iterative Improvement

With a quarter of lecture's time available left, recall that we found:

$$||x||_p = \begin{cases} \sum_j |x_j|, & p = 1 \\ \sqrt{\sum_j x_j^2}, & p = 2 \\ \max |x_j|, & p = \infty \end{cases}$$
$$= \left(\sum_{j=1}^{n} |x_j|^p \right)^{1/p}$$

and additionally, for the norm of a matrix:

$$||A||_p = \begin{cases} \max_j \sum_i |a_{ij}|, & p = 1 \\ \lambda \max(A^T A), & p = 2 \\ \max_i \sum_j |a_{ij}|, & p = \infty \end{cases}$$

where our memory trick (mnemonic?) is that 1 is standing up and hence is the column sums.
Also, we took:

$$||A|| = \max_{||x||=1} ||Ax|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$$

and this implies:

$$||Ax|| \leq ||A||\ ||x||$$
$$||AB|| \leq ||A||\ ||B||,$$

where

$$||A||\ ||A^{-1}|| \geq ||AA^{-1}|| = ||I|| = 1,$$

and we defined the condition number (not component-wise relative):

$$\kappa(A) = ||A||||A^{-1}|| \geq 1,$$

which Matlab recognizes as `rcond(A)`, where

$$\frac{1}{\text{rcond}(A)} = \infty$$

if A is not invertible. Also, we looked at the eigenvalues of A:

$$Ax = \lambda x$$
$$|\lambda|\ ||x|| = ||\lambda x|| = ||Ax|| \leq ||A||\ ||x||$$

which gives $|\lambda| \leq ||A||$. We proved that

$$||A|| < 1 \implies (I - A) \text{ invertible}$$

We did this via geometric series; however, we want to be able to readily prove this without the help of such. Suppose $(I - A)x = 0$ and $x = Ax$, and so

$$||x|| = ||Ax|| \leq ||A||\ ||x||$$

(Intuitively, if $||A||$ is less than 0, then $||x||$ has to be zero to make this greater than.)

Proof.

$$\underbrace{(1 - ||A||)}_{>0} \underbrace{||x||}_{\geq 0} \leq 0$$

implies

$$0 \leq ||x|| \leq 0$$

□

This proves that the space of invertible matrices is an open set (open ball) centered around the identity. We ponder: Is A^{-1} a continuous function of A?
We check:

$$A^{-1} - B^{-1} = O(A - B)$$
$$A^{-1}\left(I - AB^{-1}\right) = A^{-1}(B - A)B^{-1},$$

and another useful fact (relative perturbation in A^{-1}):

$$\frac{||A^{-1} - B^{-1}||}{||A^{-1}||} \leq ||A|| \frac{||B - A||\ ||B^{-1}||}{||A||} = ||A||\ ||B^{-1}|| = \underbrace{||A||\ ||B^{-1}||}\ \frac{||B - A||}{||A||}$$

The norm of the product is less than or equal to the product of the norms (Cauchy Schwarz), so we got this result (of course assuming A, B invertible):

$$\boxed{\frac{||A^{-1} - B^{-1}||}{||A^{-1}||} \leq ||A|| \frac{||B - A||}{||A||} ||B^{-1}||}$$

The reason we went through all this is that we want to derive a way of solving $Ax = b$, where we **don't know** how to, and it's too large and difficult to try doing so. Instead, we use

$$\hat{A}\hat{x} = b,$$

where:
(1) $\hat{A}^{-1}$ is close to A^{-1}, or
(2) $\hat{A}$ is close to A,
and most importantly: $\hat{A}\hat{x} = b$ is **easy to solve!**
We write:

$$A = \begin{bmatrix} a_{11} & & U \\ & \ddots & \\ L & & \end{bmatrix} = D + L + U,$$

where we separate the diagonal, lower, and upper triangular parts of A. We can take, for instance:

$$\hat{A} := D + L$$

or

$$\hat{A} := D + U$$

we suspect this would be a good choice is our matrix A is **diagonally dominant.**

2.1 Generalizing:

Suppose $Ax = b, \hat{A}\hat{x} = b$. Compute $\hat{e} : \hat{x} + \hat{e}$ closer to x. We look at the residual (how much our computed solution fails to solve the exact equation):

$$b - A\hat{x} =: r := \text{residual of } \hat{x} \text{ in } A\hat{x} = b,$$

and so we define the error:

$$e := x - \hat{x},$$

so that

$$Ae = Ax - A\hat{x} = b - Ax = r,$$

and the residual r is computable, and so we take:

$$\hat{x} + e = x.$$

We get the exact solution at one step, but we don't know how to solve $A\hat{e} = r$, so we solve:

$$\hat{A}\hat{e} = r,$$

which gets us **closer** to the exact solution. It's cheap and gets us something better (to iterate on). $\hat{x} + \hat{e}$ is closer to x.

3 Office Hours

Let's do the local truncation error of 2019, 4A:

$$\begin{aligned} u_{n+1} &= u_n + \frac{h}{2} f\left(\frac{1}{3}u_n + \frac{2}{3}u_{n+1}\right) + \frac{h}{2} f\left(\frac{2}{3}u_n + \frac{1}{3}u_{n+1}\right) \\ &= u_n + \frac{h}{2}k_1 + \frac{h}{2}k_2, \end{aligned}$$

and so we have:

$$\begin{aligned} k_1 &= f(\frac{1}{3}u_n + \frac{2}{3}\left(u_n + \frac{h}{2}k_1 + \frac{h}{2}k_2)\right) \\ &= f\left(u_n + \frac{1}{3}hk_1 + h\frac{1}{3}k_2\right) \end{aligned}$$

Because $b_1 + b_2 = 1$, so $O(h)$ is satisfied. The second order condition was

$$b^T Ae = \frac{1}{2},$$

so we check :

$$\frac{1}{2}e^T Ae = \frac{1}{2},$$

so it checks out, and we have second-order accuracy. The left entries c are the row sums of the matrix A.
Now to do the local truncation error, we had:

$$u_{n+1} = u_n + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right),$$

so we have the local truncation error:

$$\begin{aligned}\tau &= \frac{y_{n+1}-y_n}{h} - \frac{1}{2}k_1 - \frac{1}{2}k_2 \\ &= \frac{\cancel{y} + hy' + \frac{1}{2}h^2y'' - \cancel{y}}{h} + O(h^2) - \frac{1}{2}k_1 - \frac{1}{2}k_2 \\ &= y' + \frac{1}{2}hy'' - \frac{1}{2}\left(k_1+k_2\right) + O(h^2) \\ &= f + \frac{1}{2}hf'(y)f(y) - (\cdots),\end{aligned}$$

where

$$y' = f(y), \qquad y'' = f'(y)f(y)$$

and taking $g = k_1 + k_2 = f(y) + O(h)$,

$$\begin{aligned}g = k_1 + k_2 &= f\left(y + \frac{h}{3}(k_1+k_2)\right) + f\left(y + \frac{h}{6}(k_1+k_2)\right) \\ &= f\left(y + \frac{h}{3}g\right) + f\left(y + \frac{h}{6}g\right) \\ &= f(y) + f'(y)\frac{h}{3}g + f(y) + \frac{h}{6}g + O(h^2) \\ &= 2f(y) + f'(y)\frac{h}{2}g + O(h^2),\end{aligned}$$

but Strain gets stuck here and says this may not be second-order accurate (as the problem would suggest), so we made a mistake (the problem is correct)! We look back into our notes, and Strain newly derives:

Second-order (accuracy) condition:

$$\sum b_i \sum a_{ij} = \frac{1}{2}.$$

Strain goes back to notice that as $h \to 0$, we have $g \to 2f$, so we found (and fixed) our issue.
Now for the second part of the problem,

$$\begin{aligned}f(y) &= \lambda y \\ u_{n+1} &= u_n + \frac{h}{2}f\left(\frac{1}{3}u_n + \frac{2}{3}u_{n+1}\right) + \frac{h}{2}f\left(\frac{2}{3}u_n + \frac{1}{3}u_{n+1}\right) \\ &= u_n + \frac{h\lambda}{6}u_n + \frac{h\lambda}{3}u_n + \frac{h\lambda}{3}u_{n+1} + \frac{h\lambda}{6}u_{n+1} \\ &= \left(1 + \frac{h\lambda}{2}\right)u_n + \frac{h\lambda}{2}u_{n+1} \\ \left(1 - \frac{h\lambda}{2}\right)u_{n+1} &= \left(1 + \frac{h\lambda}{2}\right)u_n,\end{aligned}$$

which gives the desired result:

$$u_{n+1} = \frac{1 + \lambda h/2}{1 - \lambda h/2}u_n$$

The other way that we found in lecture to do this is via the Butcher array:

$$k = \lambda(u_n e + hAk)$$

so we have:

$$\begin{aligned}
(I - h\lambda A)\, k &= \lambda u_n e \\
k &= (I - h\lambda A)^{-1}\, \lambda u_n e \\
u_{n+1} &= u_n + hb^T k \\
&= u_n + hb^T (I - h\lambda A)^{-1} \lambda u_k e \\
&= \left(1 + h\lambda b^T (I - h\lambda A)^{-1} e\right) u_n
\end{aligned}$$

and hence we write:

$$R(z) = 1 + z \underbrace{b^T (I - zA)^{-1} e}_{b^T e + z b^T A e + z^2 b^T A^2 e} .$$

But we conclude that it's better to do it the first way, algebrically on a timed exam.