**Numerical Methods II**, Courant Institute, Spring 2020

http://www.math.nyu.edu/faculty/goodman/teaching/HonorsAlgebraII2020/NumericalMethodsII2020.html

**Always** check the `classes` message board before doing any work on the assignment.

# Assignment 1, due February 24

**Corrections:** [none yet]

1. A representation

$$A = \sum A_k$$

is *efficient* if the sum converges rapidly. A sum converges *geometrically* (also called *exponentially*) if there is a $C$ and $\rho < 1$ so that

$$|A_k| \leq C\rho^{|n|} \; . \tag{1}$$ tec

This inequality says (sort of) that term $A_{k+1}$ is smaller than term $A_k$ by a factor of $\rho$. If the $A_n$ go to zero geometrically (tec (1)), then the partial sums converge to the full sum geometrically. Specifically, show that the inequality (tec (1)) implies that there is another constant $C'$ so that (with the same $\rho$)

$$\left| \sum A_k - \sum_{|k| \leq n} A_k \right| \leq C' \rho^n \; .$$

2. Let $f(x)$ is periodic with period $2\pi$, for simplicity in formulas. Then $f$ is *analytic* with radius of convergence $r$ if the derivatives of $f$ satisfy (with some constant $C$)

$$\left| f^{(n)}(x) \right| \leq \frac{Cn!}{r^n} \; , \quad \text{for all } x \; . \tag{2}$$ db

Here, $f^{(n)}$ is the $n-$th derivative of $f$, and the same constants $r$ and $C$ work for every $n$. The Fourier series representation of $f$ is

$$f(x) = \sum_{k=-\infty}^{\infty} e^{ikx} \widehat{f}_k \; .$$

The Fourier coefficients are given by

$$\widehat{f}_k = \frac{1}{2\pi} \int_0^{2\pi} e^{-ikx} f(x) \, dx \; .$$

(a) Show that if $f$ is analytic, then the Taylor series converges as long as $|y - x| < r$:

$$f(y) = \sum_{n=0}^{\infty} \frac{(y-x)^n}{n!} f^{(n)}(x) \; .$$

(This is from basic calculus, put here to remind you why the bound (db (2)) makes sense.)

(b) Show that if $k \neq 0$ and for any $n \geq 0$,

$$\widehat{f}_k = \frac{1}{2\pi} \frac{1}{(ik)^n} \int_0^{2\pi} e^{-ikx} f^{(n)}(x)\, dx \ .$$

(c) Show that if $f$ is analytic in the sense of (2), then, for any $k \neq 0$ and $n \geq 0$,

$$\left|\widehat{f}_k\right| \leq \frac{Cn!}{|rk|^n} \ .$$

(d) *Stirling's approximation* (also called Stirling's *formula*) is

$$n! \approx \sqrt{2\pi n}\, n^n e^{-n} \ .$$

This is accurate in the sense that as $n \to \infty$,

$$\frac{n! - \sqrt{2\pi n}\, n^n e^{-n}}{n!} \to 0 \ .$$

This is called *relative accuracy* in numerical computing. For now, ignore the "prefactor" $\sqrt{2\pi n}$ and use the simpler but less accurate approximation

$$n! \sim n^n e^{-n} \ .$$

The $\sim$ instead of $\approx$ indicates is for "is something like" instead of "is approximately equal to". Show that if the simpler "approximation" were an equality, then there is $\rho < 1$ and $C$ so that

$$\left|\widehat{f}_k\right| \leq C\rho^{|k|} \ .$$

*Hint*: choose $n$ proportional to $k$ in the part (c) inequality.

(e) Use the actual Stirling approximation (with relative accuracy) and the result of exercise (1) to show that the Fourier series representation of $f$ is geometrically efficient if $f$ is analytic.

3. The cubic $b-$spline basis is a family of cubic $b-$splines $b_j(x)$ with the property that $b_j(x_j) = 1$ and $b_j(x) = 0$ unless $x_{j-2} < x < x_{j+2}$. If $j = 0$ or $j = 1$ then $b_j(x) = 0$ if $x \geq x_{j+2}$. If $j = m$ or $j = m-1$, then $b_j(x) = 0$ if $x \leq x_{j-2}$.

(a) Show that any cubic $b-$spline may be written in the form

$$f(x) = \sum_{j=0}^{m} w_j b_j(x) \ .$$

(b) Show that the coefficients $w_j$ may be found by solving a tridiagonal system of linear equations of the form $Aw = F$, where the interpolation conditions are $f(x_j) = F_j$, for $j = 0, \ldots, m$.

(c) Show that if the knot points $x_j$ are evenly spaced then $A$ has 1 on the diagonal and $\frac{1}{4}$ on the subdiagonal and superdiagonal.

4. Here is a PDE model, where $u(x,t)$ models the temperature at location $x$ at time $t$.

$$\partial_t u = \partial_x^2 u + u^2 + A$$
$$u(0,t) = u(L,t) = 0 \; .$$

The first equation says that heat diffuses (the $\partial_x^2$ term) and is produced uniformly (the $A$ term) and that there is a temperature dependent chemical reaction (the $u^2$ term). We will take $A \geq 0$ (so heat is produced, not absorbed or lost) and $u(x,t) \geq 0$. The second line is *boundary conditions* giving the temperature at the ends of the computational domain. A steady state would have $\partial_t u = 0$. The equation or a steady state is

$$\partial_x^2 u + u^2 + A = 0 \; . \tag{3}$$ ss

The boundary conditions are the same. This exercise involves calculating steady states, if they exist. We consider a finite dimensional approximation space $S_h$ that consists of continuous and piecewise linear functions with knots at $0 = x_0 < x_1 < \cdots < x_{n+1} = L$. We will write $U \in S_h$ when $U$ is such a function. There is a *variational formulation* of the steady state PDE that uses the *functional*

$$F(u) = \frac{1}{2}\int_0^L (\partial_x u(x))^2 \, dx \; - \; \frac{1}{3}\int_0^L u(x)^3 \, dx \; - \; A\int_0^L u(x) \, dx \; .$$

(a) The "gradient" of $F$ (actually called the *first variation* and written $v = \delta F(u)$) is a function $v$ defined by

$$\int_0^L w(x)v(x) \, dx = \frac{d}{ds} F(u + sw)\bigg|_{s=0} \; .$$

Show that if $u$ and $w$ are twice differentiable and satisfy the boundary conditions, then

$$\delta F(u) = -\partial_x^2 u - u^3 - Au \; .$$

Conclude that if $\delta F(u) = 0$, then $u$ satisfies the steady state equation (3). This is a *variational formulation* of the steady state equation.

(b) A *Galerkin* approximation scheme (Boris Galerkin was a Russian computational scientist) is to look for $U \in S_h$ that satisfies the variational formulation as well as possible. The discrete functional is

$$F_h(U) = F(U) \; .$$

Since $S_h$ is an $n-$dimensional vector space, we can form the $n-$dimensional gradient

$$\nabla F_h(u) \; .$$

Suppose the nodal values are $U(x_k) = U_k$. Find explicit formulas for the gradient

$$G_k = \partial_{U_k} F_h(U) \ .$$

Find explicit formulas for the Hessian matrix elements

$$H_{jk} = \partial_{U_j} \partial_{U_k} F_h(U) \ .$$

The Hessian turns out to be tri-diagonal. The Galerkin approximation is $U$ with $\nabla F_h(U) = 0$, which is the sam as $G_k(U) = 0$ for all $k$.

(c) Show that if $U = 0$, and if the $x_k$ are uniformly spaced, then $H$ is the matrix of the second difference approximation to $\partial_x^2$.

5. Write code in Python to evaluate $G$ and $H$ and use this to look for solutions of the Galerkin approximations using Newton's method. For this you will have to form the tri-diagonal matrix $H$ and solve linear systems $HV = G$. You may use the appropriate Python linear algebra routines. Specifically,

   (a) Write code that takes an $n-$component vector $U$ (the nodal values) and calculates the number $F_h(U)$, The $n-$component vector $G(U)$, and the Hessian matrix $H(U)$. Write a separate routine that uses finite differences to check that $\nabla F_h = G$ and the derivatives of $G$ are the entries of $H$. For this, estimate $\partial_{U_j} F_h(U)$ by $\frac{1}{2\Delta U}(F_h(U + e_j \Delta U) - F_h(U - e_j \Delta U)$. Here, $e_j$ is the $n-$ component vector with all zeros except for 1 in component $j$. Use a similar formula to estimate the entries of $H$ from $G$. Check all the components of $G$ and all the entries of $H$ in this way. This is just a check of programming correctness, so you can run the test with modest value of $n$, but not too small.

   (b) Neglect the nonlinear term and compute the solution, and compare it to the exact solution, which has the form $u(x) = B(x - \frac{L}{2})^2)$, where $B$ is related to $A$. How accurate is the solution on a uniform mesh? Try to explain this accuracy.

   (c) Without neglecting the nonlinear term, explain why we might expect that $u(x) \approx u_0(x) = B(x - \frac{L}{2})^2)$ for small $A$.

   (d) Write the Newton solver using initial guess $u_0$ in part (c). Do not use safeguards such as line search. The Newton solver should converge rapidly if $A$ is small.

   (e) Write a *continuation* method, which uses an increasing sequence $0 < A_1 < A_2 < \cdots$ For each $A_k$, solve the equations using your Newton solver. Take the initial guess to be the $A_{k-1}$ converged solution. This will work if the $A_k$ increase slowly enough and if the $A_k$ are not too big.

   (f) The solution 'blows up" (goes to infinity) for some $\overline{A} < A^*$. Use your code to identify $\overline{A}$ as well as you can. What does the solution look like close to $\overline{A}$?

(g) Try to compute the solution more accurately near $\overline{A}$ by choosing the breakpoints $x_k$ in a way that clusters them.