Department of Applied Mathematics
Preliminary Examination in Numerical Analysis
August, 2013

January 29, 2014

Solutions:

1. **Root Finding.**
We want to find a function such that the Newton iterations $x_{n+1} = x_n - f(x_n)/f'(x_n)$ 'hop about' forever within a finite interval, without ever converging. The easiest example would seem to be if the iterates form some short cycle, the simplest of all such arising if $x_{n+1} = -x_n$, i.e. $-x_n = x_n - f(x_n)/f'(x_n)$. Simplifying the notation by writing x in place of x_n, this will be satisfied if $f'(x) = \frac{1}{2x}\, f(x)$. We can thus choose $f(x) = \left\{ \begin{array}{ll} c\sqrt{x} & \text{if} \quad x \geq 0 \\ -c\sqrt{-x} & \text{if} \quad x < 0 \end{array} \right.$, where c is an arbitrary constant.

2. **Numerical quadrature.**

   (a) Let h denote the length of a single subinterval before the extrapolation is done. Including also the subinterval midpoint, the trapezoidal rule over this subinterval would have the weights at its ends and midpoint: $T_0 = h\,[\frac{1}{2}\ 0\ \frac{1}{2}]$ and, when using also the midpoint $T_1 = h\,[\frac{1}{4}\ \frac{1}{2}\ \frac{1}{4}]$. Weighing these two results together according to Richardson procedure gives the extrapolation $S_0 = \frac{4\,T_1 - T_0}{3} = \frac{h}{6}\,[1\ 4\ 1]$, which we recognize as the Simpson weights.

   (b) The corresponding argument will for the extrapolated Simpson method start with $S_0 = \frac{h}{6}\,[1\ 0\ 4\ 0\ 1]$ and $S_1 = \frac{h}{12}\,[1\ 4\ 2\ 4\ 1]$, producing the extrapolation $C_0 = \frac{16\,S_1 - S_0}{15} = \frac{h}{90}\,[7\ 32\ 12\ 32\ 7]$. Next question is whether this agrees with the corresponding Newton-Cotes weights (which one is unlikely to recall by heart, or wish to re-derive). Hence, we note that C_0, by its Richarson construction must produce a sixth order accurate method. Its order of accuracy thus matches that of the equally wide Newton-Cotes approximation, for which the weights are uniquely obtained by polynomial interpolation. Hence, C_0 must indeed agree with the the corresponding Newton-Cotes formula. We will not obtain any further Newton-Cotes formulas in a similar manner, since extrapolation using a formula with more that three nodes will add three or more nodes for a gain of only two orders of accuracy. That is less than what the Newton-Cotes formulas achieve.

3. **Interpolation/Approximation.**
We start by multiplying the numerator and denominator of $p_n(x)$ by $\Psi_n(x)$, to obtain

$$p_n(x) = \frac{\sum_{j=0}^n w_j f(x_j)(x-x_0)\cdot\ldots\cdot(x-x_{j-1})(x-x_{j+1})\cdot\ldots\cdot(x-x_n)}{\sum_{j=0}^n w_j\,(x-x_0)\cdot\ldots\cdot(x-x_{j-1})(x-x_{j+1})\cdot\ldots\cdot(x-x_n)}.$$

Note next that $\Psi'(x)$ will become a sum of $n+1$ terms, all but one vanishing when substituting $x \to x_j$. Hence,

$$\Psi'(x_j) = (x_j-x_0)\cdot\ldots\cdot(x_j-x_{j-1})(x_j-x_{j+1})\cdot\ldots\cdot(x_j-x_n).$$

Substituting $w_j = 1/\Psi'(x_j)$ into the expression for $p_n(x)$ above thus gives

$$p_n(x) = \frac{\sum_{j=0}^n f(x_j)\frac{(x-x_0)\cdot\ldots\cdot(x-x_{j-1})(x-x_{j+1})\cdot\ldots\cdot(x-x_n)}{(x_j-x_0)\cdot\ldots\cdot(x_j-x_{j-1})(x_j-x_{j+1})\cdot\ldots\cdot(x_j-x_n)}}{\sum_{j=0}^n 1\cdot\frac{(x-x_0)\cdot\ldots\cdot(x-x_{j-1})(x-x_{j+1})\cdot\ldots\cdot(x-x_n)}{(x_j-x_0)\cdot\ldots\cdot(x_j-x_{j-1})(x_j-x_{j+1})\cdot\ldots\cdot(x_j-x_n)}}.$$

The denominator is identically one (being the Lagrange interpolation polynomial to data that is one at every node point), and it can therefore be omitted. The expression has then reduced to the standard form of the Lagrange interpolation polynomial.

4. **Linear algebra**

(a) Reduction to Hessenberg form can be performed using either Housholder reflections or Givens rotations. The Hessenberg form of a general matrix is of the form

$$\begin{pmatrix} x & x & \ldots & x & x \\ x & x & \ldots & x & x \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & x & x \\ 0 & 0 & \ldots & x & x \end{pmatrix}$$

and is tridiagonal for self-adjoint matrices. Its purpose is to reduce the computational burden of QR iteration.

(b) Initialize $\mathbf{A}_1 = \mathbf{A}$. QR iteration proceeds as

$$\begin{aligned} \mathbf{A}_n &= \mathbf{Q}_n\mathbf{R}_n \\ \mathbf{A}_{n+1} &= \mathbf{R}_n\mathbf{Q}_n, \;\; n = 1, 2, \ldots \end{aligned}$$

for $n = 1, 2, \ldots$. Here $\mathbf{Q}_n$ is unitary and $\mathbf{R}_n$ is upper triangular. These matrices are obtained via QR factorization. The second step is the product of these matrices in the reverse order.

(c) We have

$$\mathbf{Q}_n^*\mathbf{A}_n = \mathbf{R}_n,$$

so that

$$\mathbf{A}_{n+1} = \mathbf{Q}_n^*\mathbf{A}_n\mathbf{Q}_n.$$

(d) QR iteration converges if the eigenvalues $|\lambda_1| > |\lambda_2| > \ldots|\lambda_n|$, i.e., absolute values of the eigenvalues are distinct.

5. **ODEs**

(a) We have

$$\mathbf{f}\left(t_n + h, \mathbf{y}_n + \mathbf{k}_1\right) = \mathbf{f}\left(t_n, \mathbf{y}_n\right) + h\left(\frac{\partial \mathbf{f}(t_{n,}\mathbf{y}_n)}{\partial t} + \frac{\partial \mathbf{f}(t_{n,}\mathbf{y}_n)}{\partial \mathbf{y}}\mathbf{f}(t_{n,}\mathbf{y}_n)\right) + \mathcal{O}(h^2),$$

so that

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n) + \frac{h^2}{2}\left(\frac{\partial \mathbf{f}(t_{n,}\mathbf{y}_n)}{\partial t} + \frac{\partial \mathbf{f}(t_{n,}\mathbf{y}_n)}{\partial \mathbf{y}}\mathbf{f}(t_{n,}\mathbf{y}_n)\right) + \mathcal{O}(h^3).$$

Comparing with the Taylor expansion of the exact solution at $\mathbf{f}(t_{n,}\mathbf{y}_n)$,

$$\mathbf{y}'' = \frac{\partial \mathbf{f}(t, \mathbf{y})}{\partial t} + \frac{\partial \mathbf{f}(t, \mathbf{y})}{\partial \mathbf{y}}\mathbf{f}(t,, \mathbf{y}),$$

we obtain

$$\mathbf{y}(t_{n+1}) = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n) + \frac{1}{2}h^2\left(\frac{\partial \mathbf{f}(t_n, \mathbf{y}_n)}{\partial t} + \frac{\partial \mathbf{f}(t_n, \mathbf{y}_n)}{\partial \mathbf{y}}\mathbf{f}(t_n, \mathbf{y}_n)\right) + \mathcal{O}(h^3)$$

so that the order of the method is $p = 2$.

(b) Applying Heun's method to the test problem

$$\begin{aligned} \mathbf{y}' &= \lambda \mathbf{y} \\ \mathbf{y}(0) &= \mathbf{y}_0, \end{aligned}$$

where $\lambda \in \mathbb{C}$, we have

$$\begin{aligned} \mathbf{k}_1 &= \lambda h \mathbf{y}_n, \\ \mathbf{k}_2 &= \lambda h\left(\mathbf{y}_n + \mathbf{k}_1\right), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{1}{2}\left(\mathbf{k}_1 + \mathbf{k}_2\right) \end{aligned}$$

so that

$$\mathbf{y}_{n+1} = \left(1 + \lambda h + \frac{(\lambda h)^2}{2}\right)\mathbf{y}_n = \left(1 + z + \frac{z^2}{2}\right)\mathbf{y}_n, \quad z = \lambda h.$$

Since when $z = 0$ the solution of this recurrence remains bounded, the method is stable.

(c) The region of absolute stability is the set of all $z \in \mathbb{Z}$ such that

$$\left|1 + z + \frac{z^2}{2}\right| \leq 1.$$

To sketch the shape of this region without a computer, one can find several points where $\left|1 + z + z^2/2\right| = 1$, e.g., $z = 0$, $z = -2$, etc.

6. **PDEs**
We look for the solution in the form

$$u(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}u_{mn}\sin\left(\pi\left(m+1\right)x\right)\sin\left(\pi\left(n+\frac{1}{2}\right)y\right)$$

so that

$$\frac{\partial u}{\partial y}(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}u_{mn}\pi\left(n+\frac{1}{2}\right)\sin\left(\pi\left(m+1\right)x\right)\cos\left(\pi\left(n+\frac{1}{2}\right)y\right)$$

satisfies the Neumann boundary at $y=1$. Computing

$$\Delta u(x,y)=-\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}u_{mn}\pi^2\left(\left(m+1\right)^2+\left(n+\frac{1}{2}\right)^2\right)\sin\left(\pi\left(m+1\right)x\right)\sin\left(\pi\left(n+\frac{1}{2}\right)y\right),$$

we seek an expansion of the right hand side,

$$f(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f_{mn}\sin\left(\pi\left(m+1\right)x\right)\sin\left(\pi\left(n+\frac{1}{2}\right)y\right)$$

so that we can set

$$u_{mn}=-\frac{f_{mn}}{\pi^2\left(\left(m+1\right)^2+\left(n+\frac{1}{2}\right)^2\right)}.$$

Consider $x_k=\left(k+1\right)/M$, $k=0,\dots M-1$ and $y_l=\left(l+1\right)/N$, $l=0,\dots N-1$ so that

$$f(x_k,y_l)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f_{mn}\sin\left(\pi\frac{\left(m+1\right)k}{M}\right)\sin\left(\pi\frac{\left(n+\frac{1}{2}\right)\left(l+1\right)}{N}\right).$$

Since sine transforms requires $\mathcal{O}\left(MN\log N\right)+\mathcal{O}\left(MN\log M\right)$ operations, using these formulas yields a fast algorithm.