CP315 - Portfolio Part 2

Mason Cooper
Wilfrid Laurier University

March 31, 2017

Contents

6 Eigenvalue/Eigenvector Problems

The following is my code for Faddeev-Leverrier scheme:

$Faddeev := proc(A)$
$local(i, n, p, P, l);$
$n := RowDimension(A);$
$P := A;$
$p := Trace(A);$
$l := (-lambda)^n + p * lambda^(n-1);$
$forifrom2tondo:$
$P := Multiply(A, P - p * IdentityMatrix(n));$
$p := Trace(P)/i;$
$l := l + p * lambda^(n-i);$
$enddo;$
$return(-l);$
$endproc$

Results:
Faddeev(HilbertMatrix(5))

$$lambda^5 - (563/315) * lambda^4 + (735781/2116800) * lambda^3 - (852401/222264000) * lambda^2$$
$$+ (61501/53343360000) * lambda - 1/266716800000$$

Proof all eigenvalues are positive:
$v := evalf(solve(CharacteristicPolynomial(HilbertMatrix(5), lambda), lambda))$
$0.3287928772e{-5}, 0.3058980402e{-3}, 0.1140749162e{-1}, .2085342186, 1.567050691$

Sum of eigenvalues:
$evalf(Sum(v[i], i = 1..5))$
1.787301587

Product of eigenvalues:
$product(v[i], i = 1..5)$
$3.749295133 * 10^(-12)$

My code for Cholesky decomposition,

$cholesky := proc(A)$
$local(n, m, i, j, B);$
$n := RowDimension(A);$
$m := ColumnDimension(A);$
$B := Matrix(n, m);$
$forifrom1tomdo;$
$forjfrom1tondo :$
$ifj < ithen;$
$B[i, j] := (A[i, j] - add(B[i, x] * B[j, x], x = 1..j - 1))/B[j, j];$
$elifi = jthen;$
$B[i, j] := sqrt(A[j, j] - add(B[i, x]^2, x = 1..i - 1));$
$endif;$
$enddo;$
$enddo;$
$return(B);$
$endproc$

Results,
$cholesky(HilbertMatrix(5))$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/2 & (1/6)*\sqrt{3} & 0 & 0 & 0 \\ 1/3 & (1/6)*\sqrt{3} & (1/30)*\sqrt{5} & 0 & 0 \\ 1/4 & (3/20)*\sqrt{3} & (1/20)*\sqrt{5} & (1/140)*\sqrt{7} & 0 \\ 1/5 & (2/15)*\sqrt{3} & (2/35)*\sqrt{5} & (1/70)*\sqrt{7} & 1/210 \end{bmatrix}$$

$Multiply(cholesky(HilbertMatrix(5)), Transpose(cholesky(HilbertMatrix(5))))$

$$\begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}$$

7 Interpolation, Curve Fitting

7.1 Vandermonde-based collocation

My Vandermonde-based collocation code,

$vandermode := proc(xy)$
$local(n, y, a, V, inv, i, p);$
$n := numelems(xy);$
$V := VandermondeMatrix(Transpose(xy)[1]);$
$y := Column(VandermondeMatrix(Transpose(xy)[2], 7), 2);$
$inv := MatrixInverse(V);$
$a := Multiply(inv, y);$
$p := 0;$
$orifrom1tondo :$
$p := p + a[i] * x^i;$
$enddo;$
$return(p);$
$endproc$

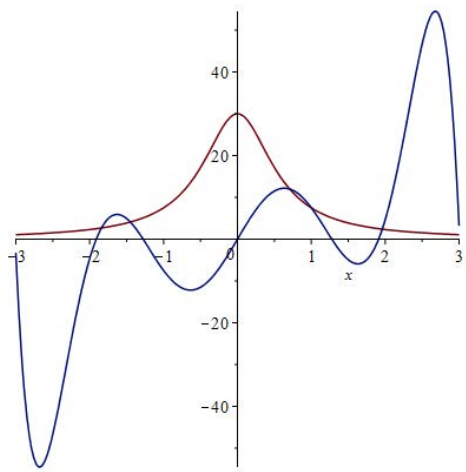
7.2 Lagrange interpolation

My Lagrange Interpolation code,

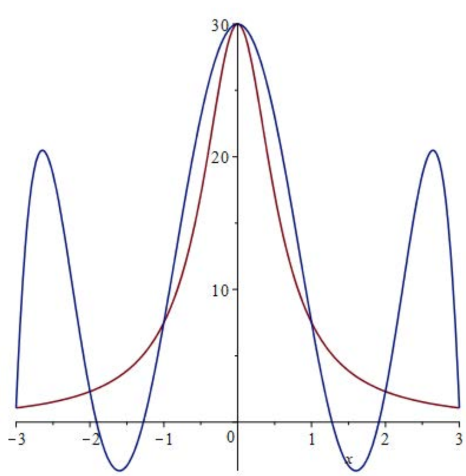
$lagrange := proc(xy)$
$local(j, k, n, X, Y);$
$n := numelems(xy) - 1;$
$X[k] := Transpose(xy)[[1]][1, k + 1];$
$Y[k] := Transpose(xy)[[2]][1, k + 1];$
$X[j] := Transpose(xy)[[1]][1, j + 1];$
$return(evalf(Sum(Y[k] * (product((x - X[j])/(X[k] - X[j]), j = 0..k - 1)) *$
$(product((x - X[j])/(X[k] - X[j]), j = k + 1..n)), k = 0..n)));$
$endproc$

7.3 Comparison

Vandermonde-Based Collocation



Lagrange Interpolation



9 Numerical Integration I, Newton-Cotes

My code for Trapezoidal Rule,

$TrapRule := proc(a, b, n, f)$
$local(h, k);$
$h := (b - a)/n;$
$return(evalf((1/2)*h*(f(a)+f(b)+2*(sum(f(h*k+a), k = 1..n-1)))));$
$endproc$

My code for Simpson's 1/3 Rule,

$simpsonsRule := proc(f, a, b, n)$
$local(h, i, q);$
$h := (b - a)/n;$
$q := evalf((1/3) * h * (f(a) + 4 * (Sum(f((2 * i - 1)/n), i = 1..(1/2) * n)) +$
$2 * (Sum(f(2 * i/n), i = 1..(1/2) * n)) + f(b)));$
$return(q);$
$endproc$

Results for Trapezoidal Rule, $TrapRule(0, 1, 100, (x^2 + x + 1)/(x^4 + x^3 + x^2 + x + 1))$
0.8648000930

$TrapRule(0, (1/2) * Pi, 1, sin(theta)^3/(sin(theta)^3 + cos(theta)^3))$
0.7853981635

$TrapRule(-Pi, Pi, 3000, cos(x^2))$
1.131389003

Results for Simpson's 1/3 Rule, $simpsonsRule((x^2+x+1)/(x^4+x^3+x^2+x+1), 0, 1, 100)$
0.8688062664

$simpsonsRule(sin(theta)^3/(sin(theta)^3+cos(theta)^3), 0, (1/2)*Pi, 1000)$
0.3842072117

$simpsonsRule(cos(x^2), -Pi, Pi, 3000)$
5.681712088

10 Numerical Integration II, Romberg

10.1 Romberg - code

My code for bisection method,

$romberg := proc(f, a, b, N)$
$local(R, h, k, row, col);$
$R := array(0..N, 0..N);$
$forrowfrom1toNdo;$
$while\frac{1}{10000} < (1/2)*b-(1/2)*ado$
$h := (1/2)*h;$
$R[row, 0] := evalf(.5*R[row-1, 0]+sum(h*f(a+(2*k-1)*h), k = 1..2^(row-1)));$
$forcolfrom1torowdo;$
$R[row, col] := (4^col*R[row, col-1]-R[row-1, col-1])/(4^col-1);$
$enddo;$
$enddo;$
$forrowfrom0toNdo$
$forcolfrom0torowdo$

$enddo;$

$enddo;$
$return(R[N, N])$
$endproc$

10.2 Romberg - Questions

For $1/(1+x)$,
$romberg(f, 0, 1, 7)$
.6931471806, 7 rows needed.

For $sin(theta)^3/(sin(theta)^3 + cos(theta)^3)$,
0.7853981635, Only one row needed.

Romberg - For 1/(1+x)

0.7500000000							
0.7083333333	0.6944444443						
0.6970238095	0.6932539683	0.6931746033					
0.6941218504	0.6931545307	0.6931479013	0.6931474775				
0.6933912022	0.6931476530	0.6931471947	0.6931471835	0.6931471824			
0.6932082083	0.6931472103	0.6931471807	0.6931471805	0.6931471804	0.6931471804		
0.6931624389	0.6931471827	0.6931471807	0.6931471806	0.6931471804	0.6931471804	0.6931471805	
0.6931509952	0.6931471807	0.6931471807	0.6931471806	0.6931471804	0.6931471804	0.6931471805	0.6931471806

11 Numerical Solution of ODE's, IVP's

11.1 One Equation

My Euler's method,

$eulers := proc(f, a, b, Y, h)$
$local(pts, n, i, y, t)$
$Digits := 5;$
$y := Y;$
$pts := [[a, y]];$
$n := (b - a)/h;$
$forifroma + 1tondo;$
$t := h * i;$
$y := evalf(y + h * f(t, y));$
$pts := [op(pts), [t, y]];$
$enddo;$
$return(pts);$
$endproc$

My Heun's method,
$heuns := proc(f, a, b, Y, h)$
$local(y, pts, n, i, t);$
$Digits := 5;$
$y := Y;$
$pts := [[a, y]];$
$n := (b - a)/h;$
$forifroma + 1tondo$
$t := h * i;$
$y := evalf(y + (1/2) * h * (f(t, y) + f(t + 1, y + h * f(t, y))));$
$pts := [op(pts), [t, y]];$
$enddo;$
$return(pts)$
$endproc$

My Runge-Kutta,

$runge kutta := proc(f, a, b, Y, h)$
$local(y, pts, n, i, t, k1, k2, k3, k4);$
$Digits := 5;$
$y := Y;$
$pts := [[a, y]];$
$n := (b - a)/h;$
$forifroma + 1tondo$
$t := h * i;$
$k1 := h * f(t, y);$
$k2 := h * f(t + (1/2) * h, y + (1/2) * k1);$
$k3 := h * f(t + (1/2) * h, y + (1/2) * k2);$
$k4 := h * f(t + h, y + k3);$
$y := evalf(y + 1/6 * (k1 + 2 * k2 + 2 * k3 + k4));$
$pts := [op(pts), [t, y]];$
$enddo;$
$return(pts)$
$endproc$

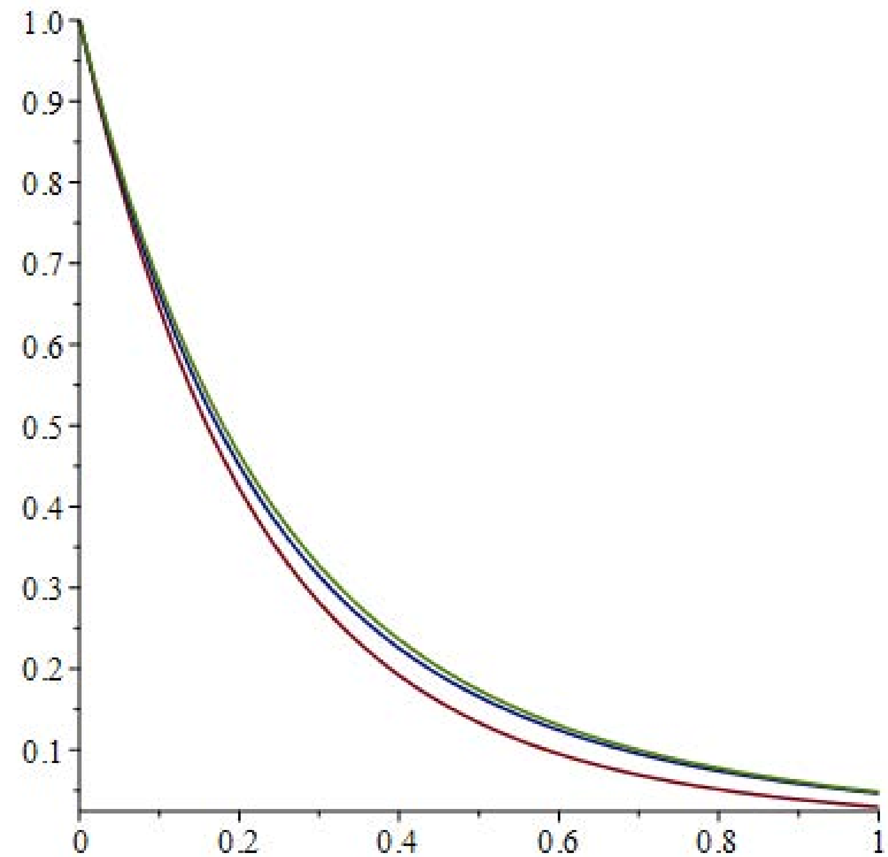
11.2 Results

Euler's Output,
$[[0, 1], [0.2e{-}1, .91922], [0.4e{-}1, .84576], [0.6e{-}1, .77892], [0.8e{-}1, .71807], [.10, .66264], [.12, .61211],$
$1], [.72, 0.90307e - 1], [.74, 0.85829e - 1], [.76, 0.81620e - 1], [.78, 0.77661e -$
$1], [.80, 0.73933e - 1], [.82, 0.70419e - 1], [.84, 0.67104e - 1], [.86, 0.63975e -$
$1], [.88, 0.61018e - 1], [.90, 0.58222e - 1], [.92, 0.55576e - 1], [.94, 0.53070e -$
$1], [.96, 0.50695e - 1], [.98, 0.48443e - 1], [1.00, 0.46305e - 1]]$

Heun's Output,
$[[0, 1], [0.2e{-}1, .91494], [0.4e{-}1, .83758], [0.6e{-}1, .76719], [0.8e{-}1, .70313], [.10, .64481], [.12, .59170],$
$1], [.62, 0.88818e - 1], [.64, 0.83259e - 1], [.66, 0.78115e - 1], [.68, 0.73351e -$
$1], [.70, 0.68936e - 1], [.72, 0.64840e - 1], [.74, 0.61037e - 1], [.76, 0.57503e -$
$1], [.78, 0.54216e - 1], [.80, 0.51156e - 1], [.82, 0.48304e - 1], [.84, 0.45645e -$
$1], [.86, 0.43163e - 1], [.88, 0.40844e - 1], [.90, 0.38675e - 1], [.92, 0.36645e -$
$1], [.94, 0.34744e{-}1], [.96, 0.32961e{-}1], [.98, 0.31288e{-}1], [1.00, 0.29717e{-}1]]$

Runge-Kutta Output,
$[[0, 1], [0.2e{-1}, .92276], [0.4e{-1}, .85216], [0.6e{-1}, .78759], [0.8e{-1}, .72853], [.10, .67448], [.12, .62496],$
$1], [.72, 0.94444e-1], [.74, 0.89702e-1], [.76, 0.85245e-1], [.78, 0.81052e-$
$1], [.80, 0.77104e-1], [.82, 0.73384e-1], [.84, 0.69876e-1], [.86, 0.66566e-$
$1], [.88, 0.63440e-1], [.90, 0.60486e-1], [.92, 0.57691e-1], [.94, 0.55046e-$
$1], [.96, 0.52542e-1], [.98, 0.50170e-1], [1.00, 0.47921e-1]]$

Euler: Red **Heuns:** Blue **Runge-Kutta**: Green



Exact Solution