Math 660: Problem Set 5

Matthew Grasinger

April 24, 2017

1 C1: ADI

The relative L_∞ errors for $k = h = \frac{1}{10}, \frac{1}{20}$ and $\frac{1}{40}$ were 0.00163, 0.000476, and 0.000161, respectively. This means that, roughly, each time k and h were halved the error decreased by a factor of four. This suggests that the approximation accuracy is second order, which agrees with the theory. In the three figures that follow, the ADI approximation is compared with the exact solution. The approximation agrees well with the exact solution in all cases.

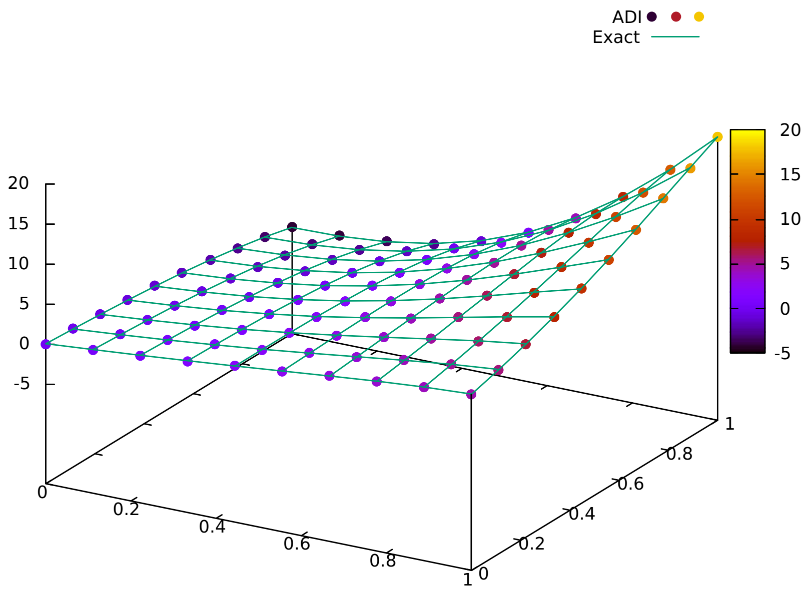

Figure 1: ADI approximation compared with exact solution. $h = \frac{1}{10}$.

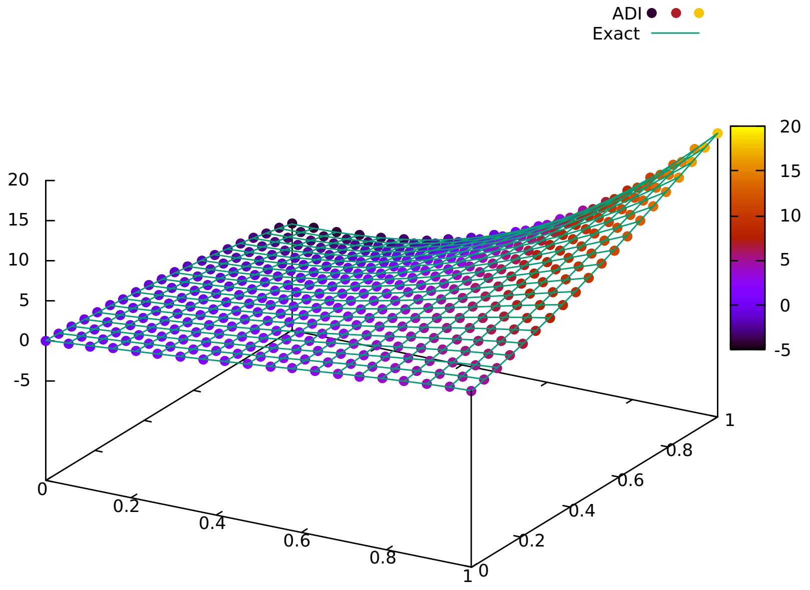

Figure 2: ADI approximation compared with exact solution. $h = \frac{1}{20}$.

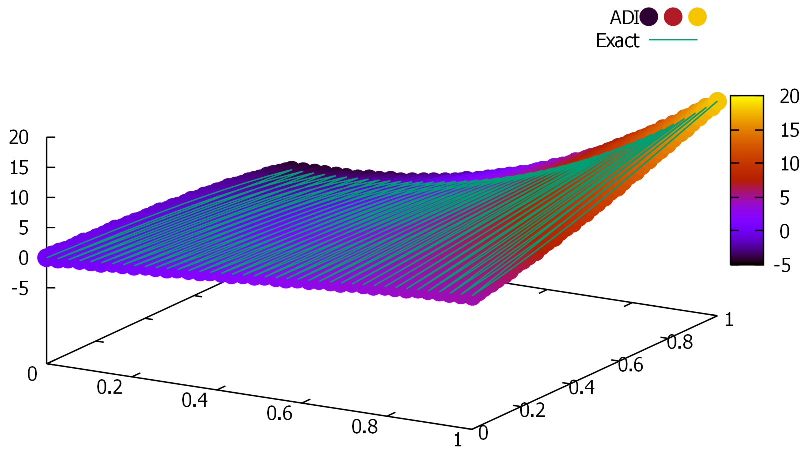

Figure 3: ADI approximation compared with exact solution. $h = \frac{1}{40}$.

1.1 Source Code

```julia
asoln(x, y, t) = exp(0.75 * t) * sin(2 * x - y) * cosh(1.5 * (x + y));

errs = [];
hs = [1/10; 1/20; 1/40];

for h in hs
  k = h;
  μ = k / (h*h);
  println("k = $k, h = $h");

  const aax = -μ / 2;
  const bbx = (μ + 1);
  const ccx = aax;

  const aay = -μ;
  const bby = (2 * μ + 1);
  const ccy = aay;

  xs = linspace(0.0, 1.0, Int(round(1.0 / h)));
  ys = copy(xs);
  ts = linspace(0.0, 1.0, Int(round(1.0 / k)));
  const M, L, K = length(xs), length(ys), length(ts);

  u = zeros(M, L, K);
  for (m, x) in zip(1:M, xs), (l, y) in zip(1:L, ys)
    u[m, l, 1] = asoln(x, y, 0);
  end

  for n in 1:K-1
    u_temp = zeros(M, L);
    thalf = (ts[n]+ts[n+1]) / 2;
    # calculate boundary terms
    for l in 1:L
      u_temp[1, l] = asoln(0.0, ys[l], thalf);
      u_temp[M, l] = asoln(1.0, ys[l], thalf);
      u[1, l, n+1] = asoln(0.0, ys[l], ts[n+1]);
      u[M, l, n+1] = asoln(1.0, ys[l], ts[n+1]);
    end
    for m in 2:M-1
      u_temp[m, 1] = asoln(xs[m], 0.0, thalf);
      u_temp[m, L] = asoln(xs[m], 1.0, thalf);
      u[m, 1, n+1] = asoln(xs[m], 0.0, ts[n+1]);
      u[m, L, n+1] = asoln(xs[m], 1.0, ts[n+1]);
    end

    for l in 2:L-1
      # calculate pi and qi for Thomas' algorithm
      p = zeros(L);
      q = zeros(L);
      p[2], q[2] = 0.0, asoln(0.0, ys[l], thalf);
      for m=2:M-1
        dd = u[m, l, n] + (μ *
                          (u[m, l+1, n] - 2 * u[m, l, n] + u[m, l-1, n]));
        denom = aax * p[m] + bbx;
        p[m+1] =  -ccx / denom;
        q[m+1] = (dd - aax * q[m]) / denom;
      end
      u_temp[M, l] = asoln(1.0, ys[l], thalf);
      for m=M-1:-1:2
        u_temp[m, l] = p[m+1] * u_temp[m+1, l] + q[m+1];
      end
      u_temp[1, l] = asoln(0.0, ys[l], thalf);
    end

```

```
    for m in 2:M-1
      # calculate pi and qi for Thomas' algorithm
      p = zeros(M);
      q = zeros(M);
      p[2], q[2] = 0.0, asoln(xs[m], 0.0, ts[n+1]);
      for l=2:L-1
        dd = u_temp[m, l] + (μ / 2 *
                         (u_temp[m+1, l] - 2 * u_temp[m, l] + u_temp[m-1, l]));
        denom = aay * p[l] + bby;
        p[l+1] =  -ccy / denom;
        q[l+1] = (dd - aay * q[l]) / denom;
      end
      u[m, L, n+1] = asoln(xs[m], 1.0, ts[n+1]);
      for l=L-1:-1:2
        u[m, l, n+1] = p[l+1] * u[m, l+1, n+1] + q[l+1];
      end
      u[m, 1, n+1] = asoln(xs[m], 0.0, ts[n+1]);
    end

    u_exact = zeros(M, L);
    for m in 1:M, l in 1:L
      u_exact[m, l] = asoln(xs[m], ys[l], ts[n+1]);
    end

    if n % 5 == 0
      println("t=$(ts[n+1]), relative L∞ error: ", norm(u[:, :, n+1] - u_exact, Inf) / norm(u_exact, Inf));
      println("t=$(ts[n+1]), relative L2 error: ", norm(u[:, :, n+1] - u_exact, 2) / norm(u_exact, 2));
      open(w -> begin
        for m in 1:M, l in 1:L
          write(w, "$(xs[m]),$(ys[l]),$(u[m, l, n+1]),$(asoln(xs[m], ys[l], ts[n+1]))\n");
        end
        end, "h-$(Int(round(h*100)))_t-$(Int(round(ts[n+1]*100))).csv", "w");
    end
  end

  u_exact = zeros(M, L);
  for m in 1:M, l in 1:L
    u_exact[m, l] = asoln(xs[m], ys[l], ts[K]);
  end
  push!(errs, maximum(map(x -> abs(x), u[:, :, K] - u_exact)));
  println("t=1.0, relative L∞ error: ", norm(u[:, :, K] - u_exact, Inf) / norm(u_exact, Inf));
  println("t=1.0, relative L2 error: ", norm(u[:, :, K] - u_exact, 2) / norm(u_exact, 2));
  println();

  open(w -> begin
    for m in 1:M, l in 1:L
      write(w, "$(xs[m]),$(ys[l]),$(u[m, l, K]),$(asoln(xs[m], ys[l], ts[K]))\n");
    end
    end, "h-$(Int(round(h*100)))_end.csv", "w");
end

println(@sprintf("%10s %10s %10s", "h", "max(|e|)", "ratio"));
println(@sprintf("%10.4lf %10.4lf %10s", hs[1], errs[1], "N/A"));
println(@sprintf("%10.4lf %10.4lf %10lf", hs[2], errs[2], errs[1]/errs[2]));
println(@sprintf("%10.4lf %10.4lf %10lf", hs[3], errs[3], errs[2]/errs[3]));
```

2 Douglas-Rachford (Optional)

For the Douglas-Rachford scheme the grid spacings were $h = \frac{1}{10}$, $\frac{1}{20}$ and $\frac{1}{40}$, but k was taken to be $k = h^2$ because the Douglas-Rachford scheme is second order accurate in space but only first order accurate in time. The relative L_∞ errors for $h = \frac{1}{10}$, $\frac{1}{20}$ and $\frac{1}{40}$ were 0.00114, 0.000393, and 0.000314, respectively. Notice that, roughly, when h was halved the first time (from $h = \frac{1}{10}$ to $h = \frac{1}{20}$ and $k = \frac{1}{100}$ to $k = \frac{1}{400}$) the error decreased by a factor of four, which agrees with the theory. However, when h was halved again there were diminishing returns on the accuracy as the accuracy only increased by less than 25%. In the three figures that follow, the ADI approximation is compared with the exact solution. The approximation agrees well with the exact solution in all cases.

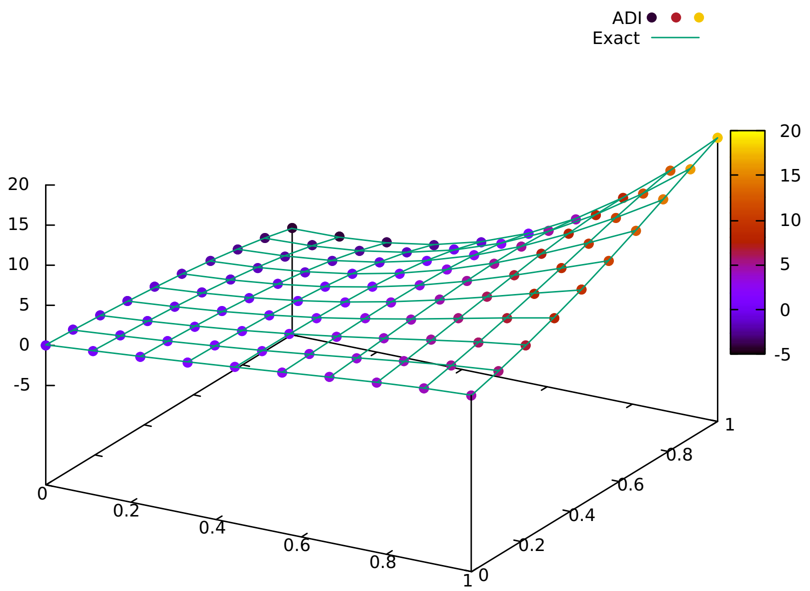

Figure 4: ADI approximation compared with exact solution. $h = \frac{1}{10}$, $k = \frac{1}{100}$.

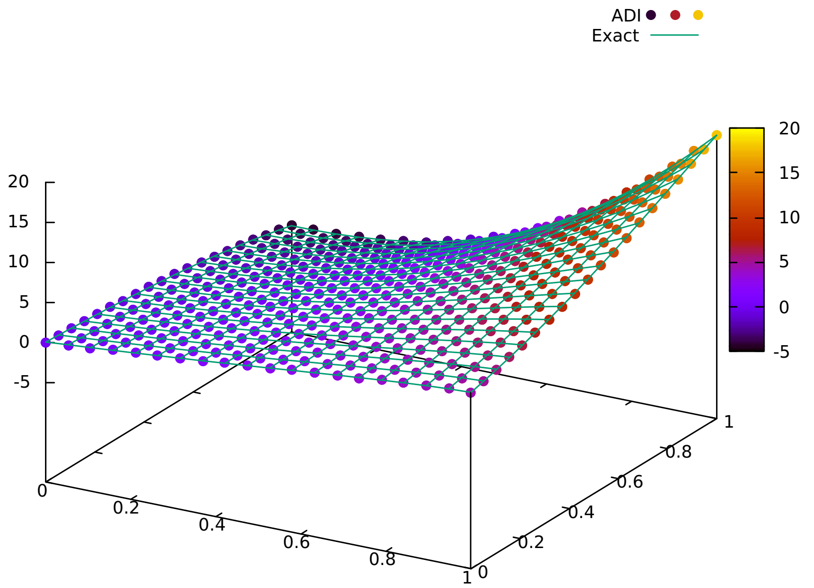

Figure 5: ADI approximation compared with exact solution. $h = \frac{1}{20}$, $k = \frac{1}{400}$.

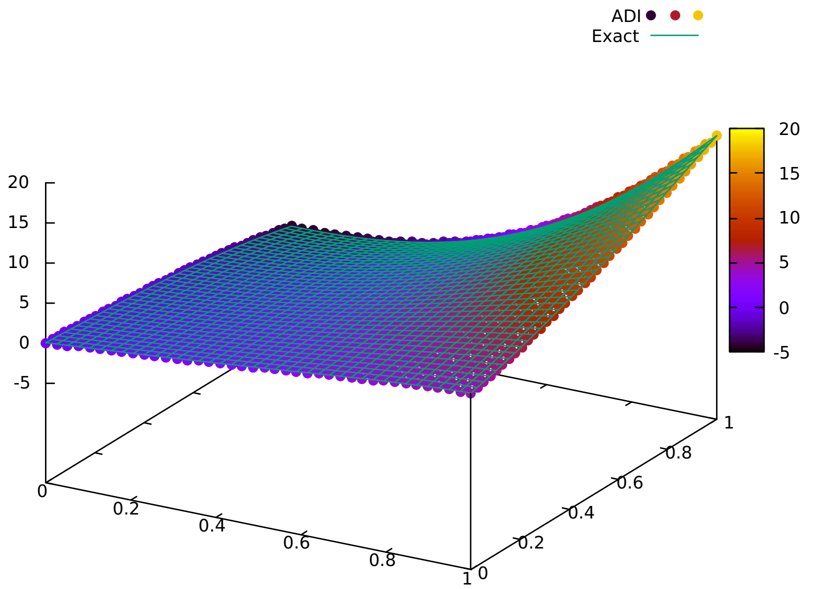

Figure 6: ADI approximation compared with exact solution. $h = \frac{1}{40}$, $k = \frac{1}{1600}$.

2.1 Source Code

```julia
dfilename(h, t) = @sprintf("h-%04d_t-%04d.csv", Int(round(h*100)), Int(round(t*100)));
dfilename(h) = @sprintf("h-%04d_end.csv", Int(round(h*100)));

asoln(x, y, t) = exp(0.75 * t) * sin(2 * x - y) * cosh(1.5 * (x + y));

errs = [];
hs = [1/10; 1/20; 1/40];

for h in hs
  k = h^2;
  output_step = Int(round(1 / k * 0.2));

  μ = k / (h*h);
  println("k = $k, h = $h");

  const aax = -μ;
  const bbx = (2 * μ + 1);
  const ccx = aax;

  const aay = -2 * μ;
  const bby = (4 * μ + 1);
  const ccy = aay;

  xs = linspace(0.0, 1.0, Int(round(1.0 / h)));
  ys = copy(xs);
  ts = linspace(0.0, 1.0, Int(round(1.0 / k)));
  const M, L, K = length(xs), length(ys), length(ts);

  u = zeros(M, L, K);
  for (m, x) in zip(1:M, xs), (l, y) in zip(1:L, ys)
    u[m, l, 1] = asoln(x, y, 0);
  end

  for n in 1:K-1
    u_temp = zeros(M, L);
    thalf = (ts[n]+ts[n+1]) / 2;
    # calculate boundary terms
    for l in 1:L
      u_temp[1, l] = asoln(0.0, ys[l], thalf);
      u_temp[M, l] = asoln(1.0, ys[l], thalf);
      u[1, l, n+1] = asoln(0.0, ys[l], ts[n+1]);
      u[M, l, n+1] = asoln(1.0, ys[l], ts[n+1]);
    end
    for m in 2:M-1
      u_temp[m, 1] = asoln(xs[m], 0.0, thalf);
      u_temp[m, L] = asoln(xs[m], 1.0, thalf);
      u[m, 1, n+1] = asoln(xs[m], 0.0, ts[n+1]);
      u[m, L, n+1] = asoln(xs[m], 1.0, ts[n+1]);
    end

    for l in 2:L-1
      # calculate pi and qi for Thomas' algorithm
      p = zeros(L);
      q = zeros(L);
      p[2], q[2] = 0.0, asoln(0.0, ys[l], thalf);
      for m=2:M-1
        dd = u[m, l, n] + (2 * μ *
                           (u[m, l+1, n] - 2 * u[m, l, n] + u[m, l-1, n]));
        denom = aax * p[m] + bbx;
        p[m+1] =  -ccx / denom;
        q[m+1] = (dd - aax * q[m]) / denom;
      end
      u_temp[M, l] = asoln(1.0, ys[l], thalf);
```

```julia
      for m=M-1:-1:2
        u_temp[m, l] = p[m+1] * u_temp[m+1, l] + q[m+1];
      end
      u_temp[1, l] = asoln(0.0, ys[l], thalf);
    end

    for m in 2:M-1
      # calculate pi and qi for Thomas' algorithm
      p = zeros(M);
      q = zeros(M);
      p[2], q[2] = 0.0, asoln(xs[m], 0.0, ts[n+1]);
      for l=2:L-1
        dd = u_temp[m, l] - (2 * μ *
                           (u[m, l+1, n] - 2 * u[m, l, n] + u[m, l-1, n]));
        denom = aay * p[l] + bby;
        p[l+1] =  -ccy / denom;
        q[l+1] = (dd - aay * q[l]) / denom;
      end
      u[m, L, n+1] = asoln(xs[m], 1.0, ts[n+1]);
      for l=L-1:-1:2
        u[m, l, n+1] = p[l+1] * u[m, l+1, n+1] + q[l+1];
      end
      u[m, 1, n+1] = asoln(xs[m], 0.0, ts[n+1]);
    end

    u_exact = zeros(M, L);
    for m in 1:M, l in 1:L
      u_exact[m, l] = asoln(xs[m], ys[l], ts[n+1]);
    end

    if n % output_step == 0
      println("t=$(ts[n+1]), relative L∞ error: ", norm(u[:, :, n+1] - u_exact, Inf) / norm(u_exact, Inf));
      println("t=$(ts[n+1]), relative L2 error: ", norm(u[:, :, n+1] - u_exact, 2) / norm(u_exact, 2));
      open(w -> begin
        for m in 1:M, l in 1:L
          write(w, "$(xs[m]),$(ys[l]),$(u[m, l, n+1]),$(asoln(xs[m], ys[l], ts[n+1]))\n");
        end
        end, dfilename(h, ts[n+1]), "w");
    end
  end

  u_exact = zeros(M, L);
  for m in 1:M, l in 1:L
    u_exact[m, l] = asoln(xs[m], ys[l], ts[K]);
  end
  push!(errs, maximum(map(x -> abs(x), u[:, :, K] - u_exact)));
  println("t=1.0, relative L∞ error: ", norm(u[:, :, K] - u_exact, Inf) / norm(u_exact, Inf));
  println("t=1.0, relative L2 error: ", norm(u[:, :, K] - u_exact, 2) / norm(u_exact, 2));
  println();

  open(w -> begin
    for m in 1:M, l in 1:L
      write(w, "$(xs[m]),$(ys[l]),$(u[m, l, K]),$(asoln(xs[m], ys[l], ts[K]))\n");
    end
    end, dfilename(h), "w");
end

println(@sprintf("%10s %10s %10s", "h", "max(|e|)", "ratio"));
```

```
println(@sprintf("%10.4lf %10.4lf %10s", hs[1], errs[1], "N/A"));
println(@sprintf("%10.4lf %10.4lf %10lf", hs[2], errs[2], errs[1]/errs[2]));
println(@sprintf("%10.4lf %10.4lf %10lf", hs[3], errs[3], errs[2]/errs[3]));
```