MAT3360: Mandatory assignment #1, spring 2019
Suggested solution

Exercise 1.

a) If $v := u_t + cu_x$ then

$$v_t - cv_x = (u_t + cu_x)_t - c(u_t + cu_x)_x = u_{tt} + cu_{xt} - cu_{tx} - c^2 u_{xx} = u_{tt} - c^2 u_{xx} = 0$$

since $u_{tx} = u_{xt}$.

b) The solution of the homogeneous transport equation $v_t - cv_x = 0$ is found by the method of characteristics. Let $x(t)$ be the solution of $x'(t) = -c$, $x(0) = x_0$, that is, $x(t) = x_0 - ct$. Then

$$v(x(t), t) = v(x_0, 0) \qquad \Leftrightarrow \qquad v(x_0 - ct, t) = v(x_0, 0) \qquad \Leftrightarrow \qquad v(x, t) = v(x + ct, 0).$$

We have $v(x_0, 0) = (u_t + cu_x)(x_0, 0) = \psi(x_0) + c\phi'(x_0)$. Hence,

$$v(x, t) = \psi(x + ct) + c\phi'(x + ct).$$

c) We can interpret the identity $v = u_t + cu_x$ as "u solves the nonhomogeneous transport equation $u_t + cu_x = v$". Since we have an expression for v we can solve for u. To this end we use the method of characteristics: Letting $x(t) = x_0 + ct$, we know that

$$\frac{d}{dt} u(x(t), t) = v(x(t), t).$$

Integrating over $[0, t]$ gives

$$u(x(t), t) = u(x_0, 0) + \int_0^t v(x(s), s)\, ds = u(x_0, 0) + \int_0^t v(x_0 + cs, s)\, ds.$$

Inserting $u(x_0, 0) = \phi(x_0)$ and solving for $x = x(t) = x_0 + ct$, we obtain finally

$$u(x, t) = \phi(x - ct) + \int_0^t v(x - ct + cs, s)\, ds,$$

which is what we wanted.

d) Inserting our expression for v gives

$$\begin{aligned} u(x, t) &= \phi(x - ct) + \int_0^t \psi((x - ct + cs) + cs) + c\phi'((x - ct + cs) + cs)\, ds \\ &= \phi(x - ct) + \int_0^t \psi(x - ct + 2cs)\, ds + c\int_0^t \phi'(x - ct + 2cs)\, ds. \end{aligned}$$

In the first integral we make the change of variables $\theta = x - ct + 2cs$, yielding

$$\int_0^t \psi(x - ct + 2cs)\, ds = \frac{1}{2c} \int_{x-ct}^{x+ct} \psi(\theta)\, d\theta.$$

In the second integral we use the fact that $\frac{d}{ds}\phi(x - ct + 2cs) = 2c\phi'(x - ct + 2cs)$ to get

$$c\int_0^t \phi'(x - ct + 2cs)\, ds = \frac{c}{2c} \int_0^t \frac{d}{ds}\phi(x - ct + 2cs)\, ds = \frac{1}{2}\big(\phi(x + ct) - \phi(x - ct)\big),$$

where the last step follows from the fundamental theorem of calculus. All in all, this gives

$$u(x, t) = \frac{1}{2}\big(\phi(x + ct) + \phi(x - ct)\big) + \frac{1}{2c} \int_{x-ct}^{x+ct} \psi(\theta)\, d\theta.$$

Exercise 2.

a) We have

$$F(x) = F(a) + \int_a^x F'(y)\,dy,$$

which implies that

$$\int_a^b F(x)\,dx = F(a)(b-a) - \int_a^b (-1)\int_a^x F'(y)\,dydx = F(a)(b-a) + \int_a^b (b-y)F'(y)\,dy.$$

Similarly, using $F(x) = F(b) - \int_x^b F'(y)\,dy$,

$$\int_a^b F(x)\,dx = F(b)(b-a) - \int_a^b (y-a)F'(y)\,dy.$$

Hence

$$\begin{aligned}\int_a^b F(x)\,dx &= \frac{1}{2}(b-a)(F(a)+F(b)) + \int_a^b \left(y - \frac{a+b}{2}\right) F'(y)\,dy \\ &= \frac{1}{2}(b-a)(F(a)+F(b)) + \frac{1}{8}(b-a)^2(F'(b)-F'(a)) - \frac{1}{2}\int_a^b \left(y-\frac{a+b}{2}\right)^2 F''(y)\,dy \\ &= \frac{1}{2}(b-a)(F(a)+F(b)) + \frac{1}{2}\int_a^b (b-y)(y-a)F''(y)\,dy,\end{aligned}$$

which implies that

$$\begin{aligned}\left|\int_a^b F(x)\,dx - \frac{1}{2}(b-a)(F(a)+F(b))\right| &\le \frac{1}{2}\left\|F''\right\|_\infty \int_a^b (b-y)(y-a)\,dy \\ &= \frac{1}{12}\left\|F''\right\|_\infty (b-a)^3.\end{aligned}$$

Using $a = x_i$, $b = x_i + h = x_{i+1}$, and summing, we get

$$\begin{aligned}\int_a^b F(x)\,dx = \sum_{i=0}^n \int_{x_i}^{x_{i+1}} F(x)\,dx &= \sum_{i=0}^n \left[\frac{h}{2}(F_i + F_{i+1}) + \mathcal{O}(h^3)\right] \\ &= h\frac{F_0}{2} + \sum_{i=1}^n hF_i + h\frac{F_{n+1}}{2} + \mathcal{O}(h^2).\end{aligned}$$

b) MATLAB code:

```
function u=trapez(a,b,f,n)
    h=(b-a)/(n+1);
    x=a+(0:n+1)*h;
    y=f(x);
    u=h*(y(1)+y(n+2))/2 + h*sum(y(2:n+1));
end
```

c) MATLAB example:

```
>> F=@(x) x.^5;
>> n=10; for i=1:5, approx(i)=trapez(0,1,F,n); err(i)=abs(approx(i)-1/6); n=2*n; end;
>> approx

approx =

    0.1701    0.1676    0.1669    0.1667    0.1667

>> rate=log(err(2:end)./err(1:end-1))/log(2)

rate =
```

```
   -1.8640   -1.9300   -1.9645   -1.9821
```

For the second example $G(x) = \sqrt{|x - 1/2|}$ we have

$$\int_0^1 G(x)\,dx = 2\int_0^{1/2} \sqrt{x}\,dx = \frac{\sqrt{2}}{3}.$$

Using this, we get

```
>> G=@(x) sqrt(abs(x-1/2));
>> n=10; for i=1:5, approx(i)=trapez(0,1,G,n); err(i)=abs(approx(i)-sqrt(2)/3); n=2*n; end;
>> approx

approx =

    0.4757    0.4729    0.4719    0.4716    0.4715

>> rate=log(err(2:end)./err(1:end-1))/log(2)

rate =

   -1.4922   -1.5212   -1.5292   -1.5279
```

We get a lower rate since G is not twice differentiable.

d) From (2.54) we immediately get

$$\begin{aligned} u(x) &= x\left(\int_0^1 f(y)\,dy - \int_0^1 yf(y)\,dy\right) + \int_0^x yf(y)\,dy - x\int_0^x f(y)\,dy \\ &= x(\alpha(1) - \beta(1)) + \beta(x) - x\alpha(x), \end{aligned}$$

which is what we wanted.

e) We see that $\alpha(x_{i+1}) - \alpha(x_i) = \int_{x_i}^{x_{i+1}} f(y)\,dy$. The expression has the integral replaced by the trapezoidal rule with $n = 2$, i.e., grid size $h/2$. Same for β_i.

f) MATLAB code:

```
function [u,x]=poissonsolve(f,n)
    a=zeros(1,n+1); b=zeros(1,n+1);
    h=1/(n+1);
    lasta=0;
    lastb=0;
    xip=0;
    fip=f(0);
    for i=1:n+1
        xi=xip; xim=xi+h/2; xip=xi+h;
        fi=fip; fim=f(xim); fip=f(xip);
        a(i)=lasta+0.25*h*(fi+2*fim+fip);
        b(i)=lastb+0.25*h*(xi*fi+2*xim*fim+xip*fip);
        lasta=a(i);
        lastb=b(i);
    end
    u=zeros(1,n);
    x=h*(1:n);
    for i=1:n
        u(i)=x(i)*(a(n+1)-b(n+1))+b(i)-x(i)*a(i);
    end
end
```

Example 2.1

```
>> f=@(x) ones(size(x));
>> [u,x]=poissonsolve(f,10);
>> max(abs(u-0.5*x.*(1-x)))

ans =

   1.1102e-16
```

Example 2.2

```
>> f=@(x) x;
>> [u,x]=poissonsolve(f,10);
>> max(abs(u-x.*(1-x.^2)/6))

ans =

   5.5511e-17
```

In both examples the error is roughly at the level of machine precision. This is to be expected since the trapezoidal rule is exact for first degree polynomials, and all integrands were of degree two or less.

With $f(x) = e^x$ we don't have exact integrations, so that we can expect larger errors. This is indeed what happens, and the method seems to be second-order accurate

```
>> f=@(x) exp(x); e=exp(1); ux=@(x) (e-1)*x - exp(x) + 1;
>> n=10; for i=1:5, [u,x]=poissonsolve(f,n); err(i)=max(abs(u-ux(x))); n=2*n; end;
>> err

err =

   1.0e-04 *

    0.3647    0.1000    0.0263    0.0067    0.0017

>> rate=log(err(2:end)./err(1:end-1))./log(2)

rate =

   -1.8673   -1.9288   -1.9645   -1.9821
```

g) The error in this method comes from using the trapezoidal rule instead of exact integration. If we want to increase the order of convergence, we could use a more accurate rule, such as Simpson's rule. This is reasonable if we have good reason to assume that f is sufficiently smooth so that Simpson's rule actually gives better numerical integration.