Homework Assignment 10

Please submit the following files as indicated below:

Question 1 | 2 marks On the assignment page you can find videos of four animated solutions of the parabolic problem

$$\begin{aligned} \partial_t u(t) - a\Delta u(t) &= f(t) && \text{in } Q =]0,T[\times\Omega \\ u(0) &= u_0 && \text{in } \Omega \\ \frac{\partial u}{\partial n} &= 0 && \text{on } \Sigma =]0,T[\times\partial\Omega \end{aligned} \tag{H}$$

with the data from Assignment 9. However, the initial condition has been replaced with the function

$$u_0(x) = \begin{cases} 50 & \text{if } |x-(1,1)^\top| < 0.5 \\ 20 & \text{elsewhere} \end{cases}$$

Explain your observations.

Solution. First, we note that since our initial condition in Assignment 9 was $u_0 = 20$ in Ω, the only change in the initial condition is that now $u_0 = 20$ throughout Ω, except for those points contained in the region $|x-(1,1)^\top| < 0.5$ wherein $u_0 = 50$.

Now, some initial observations:

- Our initial data has a discontinuous "hot spot" in the north-eastern corner of the domain.
- The forcing term applies gaussianly distributed heat centred about the south-western corner of the domain for the first second of the five second simulation.
- The heat equation will smooth out the discontinuity immediately in the continuous case due to the sharp discontinuity containing high frequency eigenmodes which are damped exponentially fast. Ideal numerical schemes should mimic this effect.
- Failure of the numerical scheme in handling the discontinuity effectively will likely lead to ringing of the solution, possibly with extreme "wiggles" and other effects due to instabilities.

And, additionally, some notes on the three methods used for solving the wave equation:

- The Crank-Nicolson method is only A-stable, and not strictly or strongly A-stable. In fact, in the limit as the frequency of the eigenmode goes to infinity, the eigenmode is not damped at all.
- The backward Euler method is strongly A-stable, and in fact L-stable. This means that high frequency modes are damped immediately.
- The θ-method is strongly A-stable for $\theta > 1/2$. Like the backward Euler method, this means that high frequency modes will be exponentially damped, though since the θ-method is not L-stable, they likely will not be damped as quickly.

Crank-Nicolson (cn-50.webm): $h = 1/50$ In finite elements simulations using linear finite elements, discontinuities cannot be represented exactly, and in particular discontinous functions will be approximated by continuous piecewise linear functions. If the step size h is large enough, the discontinuity will actually not be all that discontinuous in the discrete approximation. As can be seen from the video, the high-frequency modes of the initial condition which theoretically should not be smoothed out my the Crank-Nicolson method do manage to smooth out as time moves on. This is likely due to the relative large spatial resolution. It should be noted, however, that this process does not at all happen quickly, and there is noticeable numerical ringing around the "discontinuity".

Crank-Nicolson (cn-200.webm): $h = 1/200$ In contrast with the Crank-Nicolson solution with $h = 1/50$ above, here the discontinuous initial is indeed approximated well enough to show large amounts of ringing and instability near the region of discontinuity. The discontinuity is in fact never smoothed out, as would be expected for the Crank-Nicolson method with discontinuous initial data.

Backward Euler (be-200.webm): $h = 1/200$ The backward Euler method, being L-stable, immediately damps out the discontinuity in the initial condition. The solution smooths out quickly and uniformly, and remains as such throughout the simulation. In fact, the disconinuity appears to be smoothed out on the very first frame, as far as I can tell, and this should be expected for an L-stable method.

The θ-Method (th-200.webm): $\theta = 0.55$, $h = 1/200$ The θ-method is strongly A-stable, and so the high-frequency discontinuity is damped out exponentially fast at the beginning of the simulation, and the temperature distribution is smooth throughout the domain. As expected, however, the rate at which the damping occurs is slower than that of the backward Euler method, in which the smoothing occurs effectively immediately.

□

Question 2 | 1 mark We have seen that the homogeneous wave equation

$$\begin{aligned} \partial_t^2 u - c^2 \Delta u &= 0 \quad \text{in } Q =]0,T[\times \Omega \\ u(0) &= u_0 \quad \text{in } \Omega \\ \partial_t u(0) &= v_0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Sigma =]0,T[\times \partial\Omega \end{aligned} \tag{W}$$

with propagation speed $c > 0$ can equivalently be re-written as

$$\begin{aligned} \partial_t u - v &= 0 \quad \text{in } Q =]0,T[\times \Omega \\ \partial_t v - c^2 \Delta u &= 0 \quad \text{in } Q =]0,T[\times \Omega \\ u(0) &= u_0 \quad \text{in } \Omega \\ v(0) &= v_0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Sigma =]0,T[\times \partial\Omega \\ v &= 0 \quad \text{on } \Sigma =]0,T[\times \partial\Omega. \end{aligned} \tag{W'}$$

Discretising with the θ-method in time and linear finite elements in space leads to the coupled system for the vectors of nodal values $\vec{u}_+^h$ and $\vec{v}_+^h$

$$\begin{aligned} M^h \vec{u}_+^h - \theta \Delta t M^h \vec{v}_+^h &= M^h \vec{u}_\circ^h + (1-\theta)\Delta t M^h \vec{v}_\circ^h \\ \theta \Delta t c^2 K^h \vec{u}_+^h + M^h \vec{v}_+^h &= -(1-\theta)\Delta t c^2 K^h \vec{u}_\circ^h + M^h \vec{v}_\circ^h \end{aligned}$$

which has to be solved at every time step. Show that this is equivalent to the two smaller, successively solvable problems

$$\begin{aligned} \left(M^h + (\theta \Delta t c)^2 K^h\right) \vec{u}_+^h &= M^h \left(\vec{u}_\circ^h + \Delta t \vec{v}_\circ^h\right) - \left(\theta (1-\theta) (\Delta t c)^2\right) K^h \vec{u}_\circ^h \\ M^h \vec{v}_+^h &= M^h \vec{v}_\circ^h - \Delta t c^2 K^h \left(\theta \vec{u}_+^h + (1-\theta)\, \vec{u}_\circ^h\right). \end{aligned}$$

Solution. If we multiply the second equation of the coupled system by $\theta \Delta t$ and add it to the first equation of the coupled system and simply, we get

$$\begin{aligned} \left(M^h + (\theta \Delta t c)^2 K^h\right) \vec{u}_+^h &= M^h \left(\vec{u}_\circ^h + \Delta t \vec{v}_\circ^h\right) + \left(\theta^2 (\Delta t c)^2 - \theta (\Delta t c)^2\right) K^h \vec{u}_\circ^h \\ &= M^h \left(\vec{u}_\circ^h + \Delta t \vec{v}_\circ^h\right) - \left(\theta (1-\theta) (\Delta t c)^2\right) K^h \vec{u}_\circ^h, \end{aligned} \tag{1}$$

which is the first desired equation of the new uncoupled system.

Now, if we simply subtract $\theta \Delta t c^2 K^h \vec{u}_+^h$ from both sides of the second equation of the coupled system and simply, we get

$$\begin{aligned} M^h \vec{v}_+^h &= M^h \vec{v}_\circ^h + \left(\Delta t c^2 K^h \theta - \Delta t c^2 K^h\right) \vec{u}_\circ^h - \left(\theta \Delta t c^2 K^h\right) \vec{u}_+^h \\ &= M^h \vec{v}_\circ^h - \Delta t c^2 K^h \left(\theta \vec{u}_+^h + (1-\theta) \vec{u}_\circ^h\right), \end{aligned} \tag{2}$$

which is the second desired equation of the new uncoupled system.

□

Question 3 | 2 marks

(a) Download and complete the FEniCS script `hw10.py` to solve Problem (W) with the data provided.

(b) Solve the wave equation

- with the (symplectic) implicit midpoint rule
- with the backward EULER method
- with the forward EULER method

and look at the solutions in ParaView.

Hint: Use the 'Warp by Scalar' filter, re-scale the colour map to the range $[-1, 1]$ and tick the box 'enable opacity mapping for surfaces' in the colour map editor.

For each of the three time stepping schemes, create a graph with curves of the total energy $E(u(t), v(t)) = T(v(t)) + V(u(t))$, the kinetic energy $T(v(t)) = \frac{1}{2}\|v(t)\|_{L^2}^2$ and the potential energy $V(u(t)) = \frac{c^2}{2}\|\nabla u(t)\|_{L^2}^2$ as functions of time. Please submit these plots and interpret the results:

Solution. See Appendix A for the figures.

Implicit Midpoint Rule As hinted at in the question, the implicit midpoint rule is a symplectic integrator for the wave equation. This means that, to numerical precision, the conserved quantity of interest for the wave equation (the energy, as we have defined it) is conserved for each time step. Indeed, we see that the total energy is conserved to within numerical precision, and it appears that the potential and kinetic energies in Figure 1 obey a sort of pseudo-periodic pattern, as might be expected given the initial condition.

Backward Euler Method The backward Euler method is strongly A-stable. However, the backward Euler method is dissipative - eigenmodes corresponding to imaginary eigenvalues are damped exponentially in time. Since the wave equation has imaginary eigenvalues, we expect to see damping of the solution in time. Indeed, all three of the total energy, potential energy, and kinetic energy are exponentially damped in Figure 2.

Forward Euler Method The forward Euler method is not A-stable. In general, for iterates to remain bounded, time steps must be suitable small due to the fact that the region of stability of the forward Euler method does not include the whole left half complex plane. In particular, we must have that (from Example 3.2.5 in the notes):

$$\Delta t \leq -2\frac{\operatorname{Re}\lambda}{|\lambda|^2}.$$

For the homogenous wave equation which has imaginary eigenvalues, this means that no positive time step can result in stable iterations: the forward Euler method is an unconditionally unstable time-stepping method. As can be seen from Figure 3, the solution does indeed grow unstably and it's magnitude grows unboundedly; all three of the total energy, potential energy, and kinetic energy grow exponentially fast with time.

□

Your Learning Progress What is the one most important thing that you have learnt from this assignment?

I'd say there were a fair number of important things; I think the splitting trick from the first question is extremely useful in practice for obtain speedups of large system solves. Additionally, gaining more insight into the benefits/detriments of different integration methods is extremely valuable.

Any new discoveries or achievements towards the objectives of your course project?

Yes, I have been using `Fenics` for my project now and I find these assignments great practice. I find myself stealing small snippets of code from these assignments to use in my project!

What is the most substantial new insight that you have gained from this course this week? Any *aha moment*?

This is a tough one. I don't think there was one particular thing, but it really all starting to fall into place. Love me some numerical analysis exercises, and finite element methods are some of the most interesting!

Appendix A

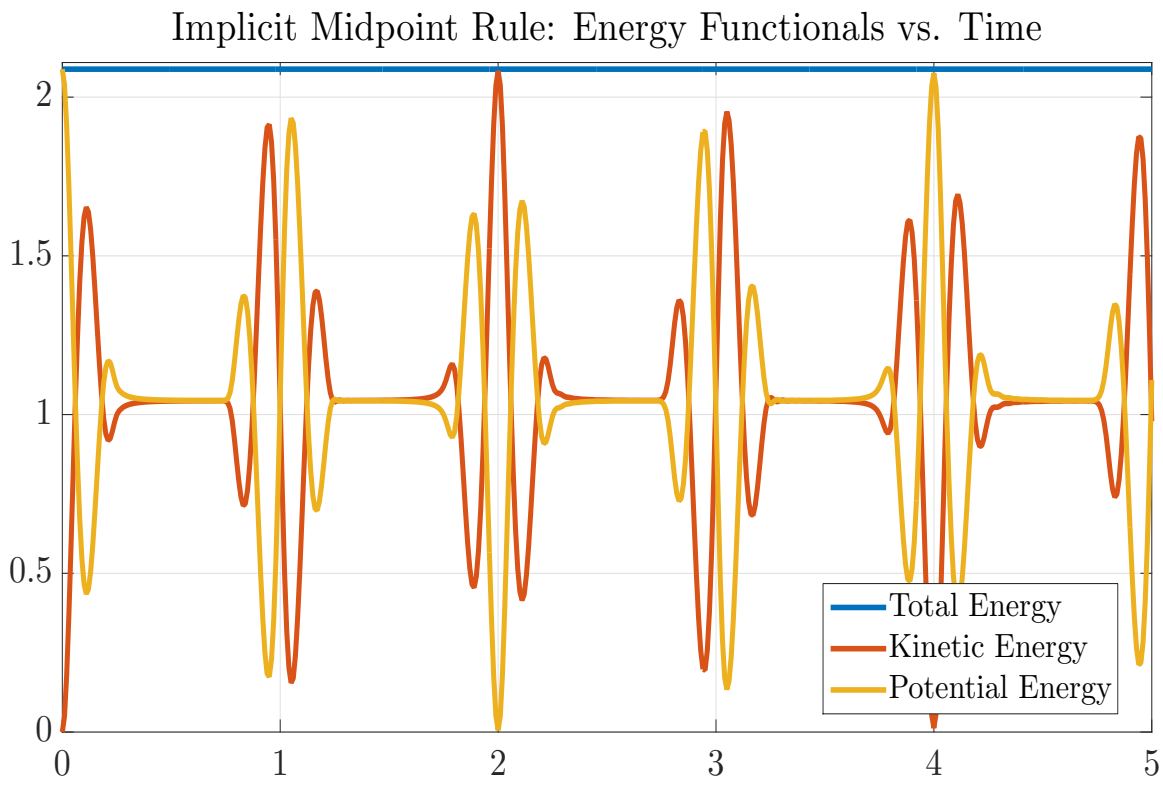

Figure 1: Plots of the total energy, kinetic energy, and potential energy obtained when solving the homogenous wave equation using the implicit midpoint rule. Note the conservation of energy throughout time.

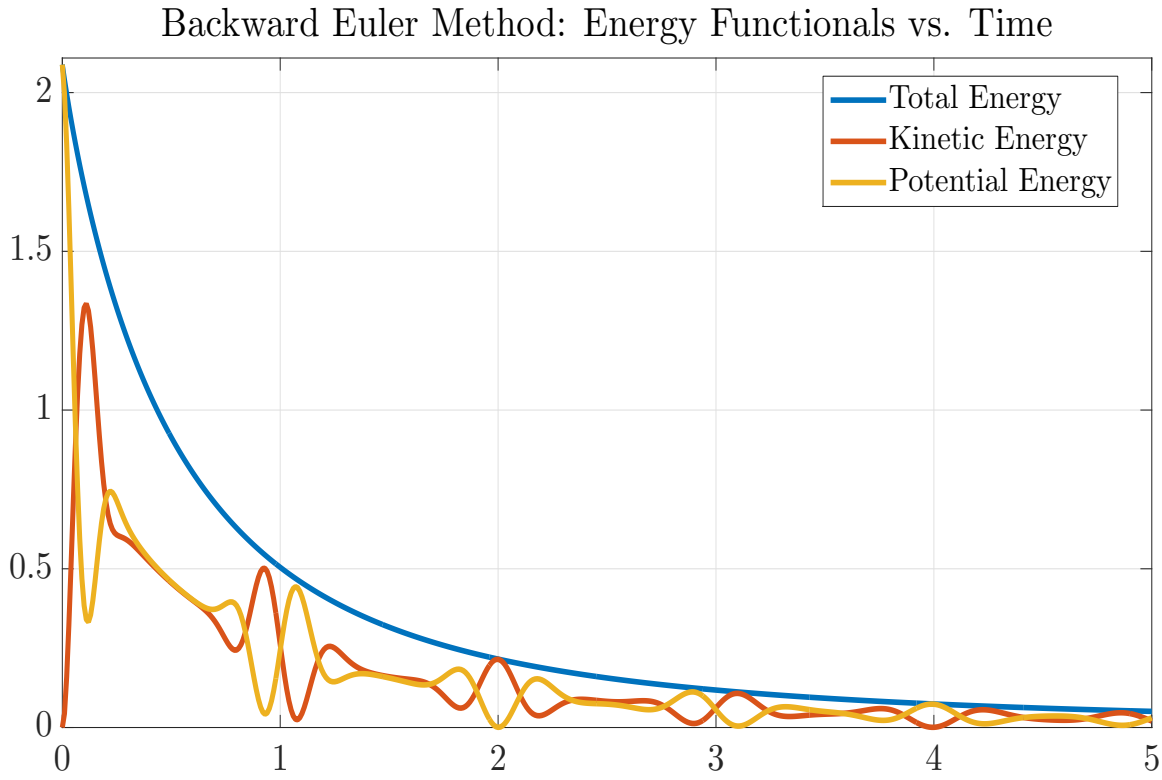

Figure 2: Plots of the total energy, kinetic energy, and potential energy obtained when solving the homogenous wave equation using the backward Euler method. Note the exponential decay of the total energy throughout time.

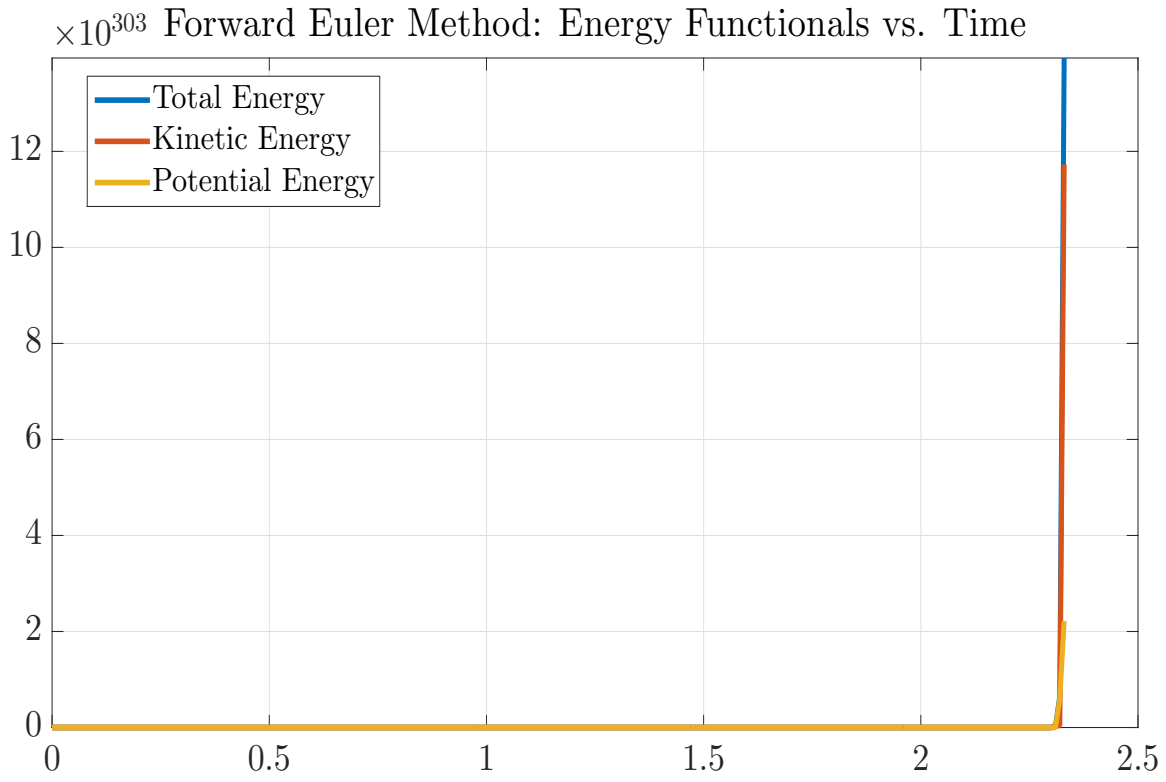

Figure 3: Plots of the total energy, kinetic energy, and potential energy obtained when solving the homogenous wave equation using the forward Euler method. Note the exponential blowup of the solution due to the instability of the method.

Appendix B

hw10.py

```
# coding=utf-8
"""
FEniCS program: Solution of the wave equation with homogeneous Dirichlet (reflective)
boundary conditions.
"""

from __future__ import print_function
from fenics import *
from mshr import *
import time
import csv

def wave_eqn(THETA = 0.5, foldname = 'cn'):
    # Create a mesh on the unit disk
    disk = Circle(Point(0., 0.), 1.)
    mesh = generate_mesh(disk, 100) # h ~ 1/50

    # Function space and boundary condition
    V = FunctionSpace(mesh, 'P', 1)

    def boundary(x, on_boundary):
        return on_boundary

    bc = DirichletBC(V, Constant(0.), boundary)

    # Problem data
    t0 = 0. # initial time
    T = 5. # final time
    t = t0 # current time
    c = Constant(1.) # propagation speed
    c2 = c**2 # square speed, for convenience

    # initial displacement and velocity
    u0 = interpolate(Expression("pow(x[0],2)+pow(x[1],2) < 1.0/16.0 ? "
        "pow(1.-16.*(pow(x[0],2)+pow(x[1],2)),2) : 0.0", degree=1), V)
    v0 = interpolate(Constant(0.0), V)

    # Parameters of the time-stepping scheme
    tsteps = 500 # number of time steps
    dt = T/tsteps # time step size
    dt2 = dt**2 # square time step, for convenience
    theta = Constant(THETA) # degree of implicitness (fenics constant object)

    # Define the variational problem
    w = TrialFunction(V) # w = u in the 1st equation and w = v in the 2nd equation
    z = TestFunction(V)
    B1 = (w*z + (theta*dt*c)**2 * dot(grad(w),grad(z)))*dx # LHS of the 1st equation
    B2 = w*z*dx # LHS of the 2nd equation

    # Set initial data
    u = Function(V, name='Displacement')
    v = Function(V, name='Velocity')
    u.assign(u0)
    v.assign(v0)
    T = assemble(0.5*v*v*dx)
    V = assemble(0.5*c2*dot(grad(u),grad(u))*dx)
    E = T + V

    # Write initial data to file
    displacement = File('wave/' + foldname + '/u.pvd')
    displacement << (u, t)
    energy = csv.writer(open('wave/' + foldname + '/energy.csv', 'w'))
    energy.writerow([t] + [E] + [T] + [V])
    # Import in Octave / MATLAB with command
    #   A = csvread('wave/energy.csv');
```

```
    # Or, if your FEniCS installation has graphics output enabled, you could also
    # plot directly from this file using Python commands

    # Time stepping
    for k in range(tsteps):

        # Current time
        t = t0 + (k+1)*dt
        print('Step = ', k+1, '/', tsteps , 'Time =', t)

        # System for the displacement
        L1 = (u0+dt*v0)*z*dx
        if THETA > 0.0 and THETA < 1.0:
            L1 -= theta*(1.0-theta)*dt2*c2 * dot(grad(u0),grad(z))*dx
        solve(B1 == L1, u, bc)

        # System for the velocity
        L2 = v0*z*dx
        if THETA > 0.0:
            L2 -= theta*dt*c2 * dot(grad(u),grad(z))*dx
        if THETA < 1.0:
            L2 -= (1.0-theta)*dt*c2 * dot(grad(u0),grad(z))*dx
        solve(B2 == L2, v, bc)

        # Compute energy
        T = assemble(0.5*v*v*dx)
        V = assemble(0.5*c2*dot(grad(u),grad(u))*dx)
        E = T + V

        # Write data to file
        displacement << (u, t)
        energy.writerow([t] + [E] + [T] + [V])

        # Update
        u0.assign(u)
        v0.assign(v)

if __name__ == "__main__":
    wave_eqn(THETA = 0.5, foldname = 'cn')
    wave_eqn(THETA = 1.0, foldname = 'be')
    wave_eqn(THETA = 0.0, foldname = 'fe')
```

energyplots.m

```
function energyplots
% Creates graphs with curves of the total energy E = T + V, the kinetic
% energy T, and the potential energy V as functions of time for the
% solution of the wave equation using three time stepping schemes, namely
% the implicit midpoint rule, the backward Euler method, and the forward
% Euler method.
%
% The energy, potential energy, and kinetic energy are defined as follows:
%    E = T + V
%    T = 1/2 * ||v||^2
%    V = c^2/2 ||grad(u)||^2

% load data
cn = load('cn.mat');
be = load('be.mat');
fe = load('fe.mat');

% create energy plots
plotenergies(cn, 'Implicit Midpoint Rule', 'cn');
plotenergies(be, 'Backward Euler Method', 'be');
plotenergies(fe, 'Forward Euler Method', 'fe');

end

function plotenergies(data, method, figname)

% utility functions
vec = @(x) x(:);

% plot data
figure('name', figname), grid on
ydata = [data.E, data.T, data.V];
h = plot(data.t, ydata);

% plot prettifying
ylim([0, 1.01*max(vec(ydata))]);
title([method, ': Energy Functionals vs. Time']);
leg = legend(h, 'Total Energy', 'Kinetic Energy', 'Potential Energy');
set(leg, 'location', 'best');

% Save figures. For the export_fig package, see the MATLAB file exchange:
%    https://www.mathworks.com/matlabcentral/fileexchange/23629-export-fig
export_fig(figname, '-pdf', '-transparent');

end
```