# Ideal Optimal Test Functions for Simple Problems

Truman E. Ellis

April 28, 2015

**Abstract**

DPG minimizes the residual in a dual norm through the use of optimal test functions. We consider a couple special cases for simple problems.

## Abstract Derivation of a Generalized Minimum Residual Method

Let $U$ and $V$ be trial and test Hilbert spaces for a well-posed variational problem $b(u,v) = l(v)$. In operator form this is $Bu = l$, where $B : U \to V'$. We seek to minimize the residual for the discrete space $U_h \subset U$:

$$u_h = \arg\min_{w_h \in U_h} \frac{1}{2} \|Bw_h - l\|_{V'}^2$$

Use the Riesz inverse to minimize in the $V$-norm rather than its dual:

$$\begin{aligned} \frac{1}{2} \|Bu_h - l\|_{V'}^2 &= \frac{1}{2} \left\|R_V^{-1}(Bu_h - l)\right\|_V^2 \\ &= \frac{1}{2} \left(R_V^{-1}(Bu_h - l), R_V^{-1}(Bu_h - l)\right)_V . \end{aligned}$$

First order optimality requires the Gâteaux derivative to be zero in all directions $\delta u \in U_h$, i.e.,

$$\left(R_V^{-1}(Bu_h - l), R_V^{-1} B\delta u\right)_V = 0, \quad \forall \delta u \in U_h.$$

By definition of the Riesz operator, this is equivalent to

$$\left\langle Bu_h - l, R_V^{-1} B\delta u_h \right\rangle = 0 \quad \forall \delta u_h \in U_h .$$

Identify $v_{\delta u_h} \coloneqq R_V^{-1} B\delta u_h$ as the optimal test function for trial function $\delta u_h$. This gives us

$$b(u_h, v_{\delta u_h}) = l(v_{\delta u_h}).$$

This gives a simple bilinear form

$$b(u_h, v_{\delta u_h}) = l(v_{\delta u_h}),$$

with $v_{\delta u_h} \in V$ that solves the auxiliary problem

$$(v_{\delta u_h}, \delta v)_V = \langle R_V v_{\delta u_h}, \delta v \rangle = \langle B\delta u_h, \delta v \rangle = b(\delta u_h, \delta v) \quad \forall \delta v \in V.$$

## Convergence in Different Norms

Because DPG guarantees stability for any well-posed variational formulation, we are free to experiment with many different formulations of the same problem. In this research note, we will concern ourselves with Poisson, pure convection, and convection-diffusion. We illustrate the implied norms of convergence for several different formulations of these problems.

1 2 3
4 5 6
7 8 9

## Optimal Test Functions for Simple Problems

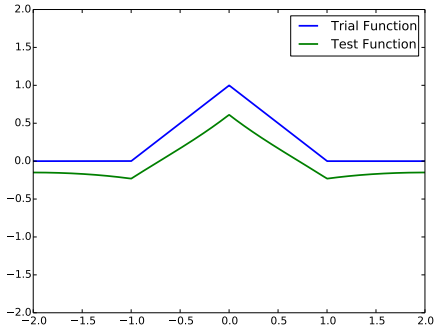
It can be educational to visualize these optimal test functions for simple cases. The precise shape depends very strongly on the chosen topology for $V$ (i.e. the choice of norm). In order to better draw analogy to classical finite element methods, we assume $C^0$ continuity of both the trial and test spaces. For simple problems such as Poisson and pure convection, we could probably derive the optimal test functions analytically, but to facilitate simple comparisons, we assembled a simple FEniCS script to compute the global optimal test functions on an interval mesh $[-2, 2]$ for a hat function defined on $[-1, 1]$. In order to accurately represent the infinitely dimensional *ideal* optimal test function, we solve with 10000 low order elements. The script we used is in the appendix.

We plot the computed optimal test functions for four different test norm topologies (norms without subscripts are $L^2$):

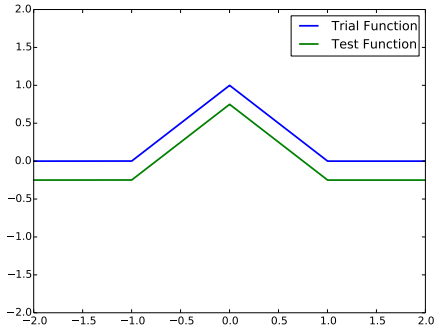
$$\begin{aligned}
\|v\|_V^2 &= \|v\|^2 + \|\nabla v\|^2 \\
\|v\|_V^2 &= 10^{-6} \|v\|^2 + \|\nabla v\|^2 \\
\|v\|_V^2 &= \|\nabla v\|^2 \\
\|v\|_V^2 &= \|v\|^2
\end{aligned}$$

Note that the third norm requires global boundary conditions in order to produce a unique optimal test function. We use the results from norm 2 to infer the correct boundary conditions.

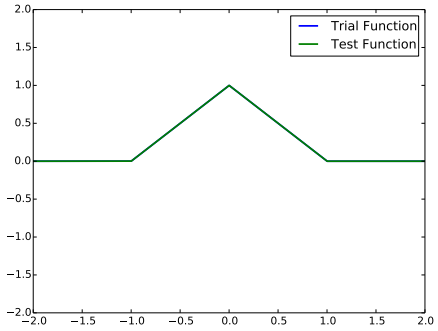
## Poisson



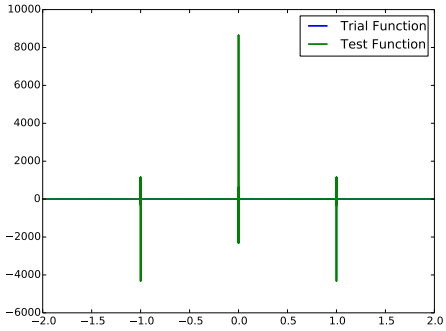

(a) $\|v\|^2 + \|\nabla v\|^2$




(b) $10^{-6}\|v\|^2 + \|\nabla v\|^2$



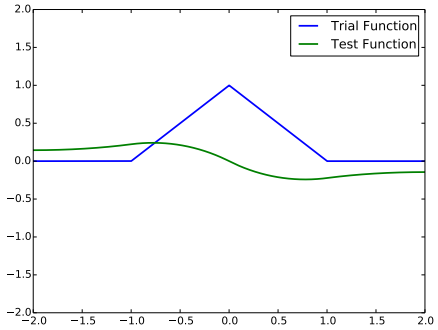

(c) $\|\nabla v\|^2$ with $v(-2) = 0$ and $v(2) = 0$



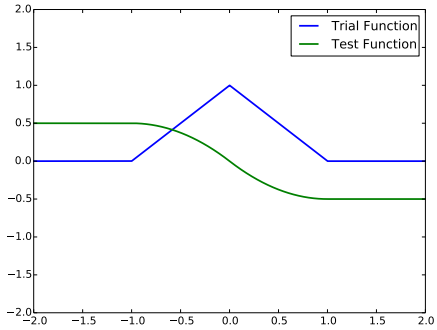

(d) $\|v\|^2$

Figure 1: Poisson ideal optimal test functions

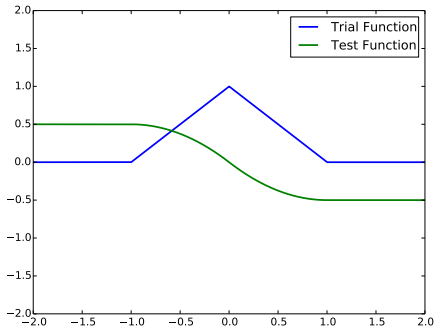
## Convection



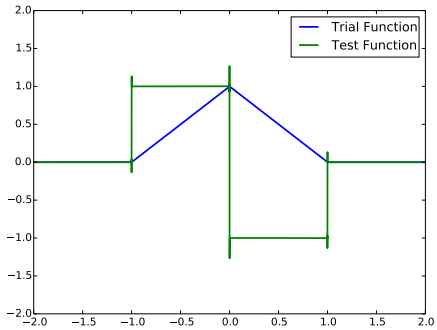

(a) $\|v\|^2 + \|\nabla v\|^2$




(b) $10^{-6} \|v\|^2 + \|\nabla v\|^2$



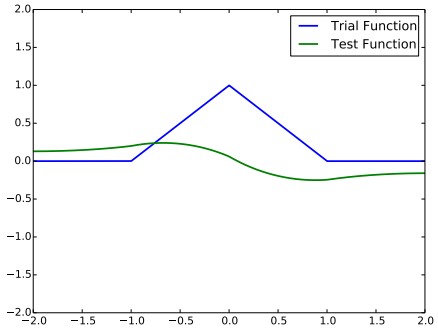

(c) $\|\nabla v\|^2$ with $v(-2) = 0.5$ and $v(2) = -0.5$



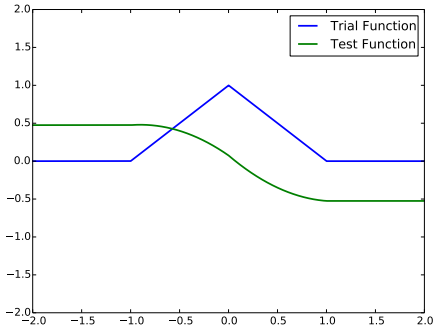

(d) $\|v\|^2$

Figure 2: Convection ideal optimal test functions

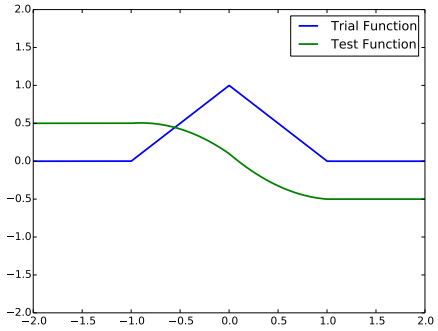
### Convection-Diffusion with $\epsilon = 10^{-1}$



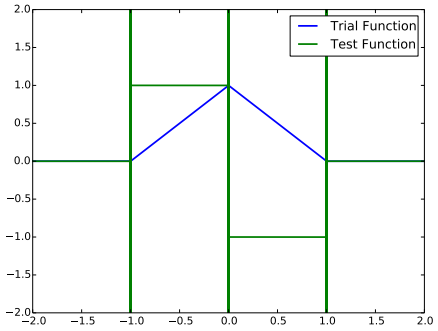

(a) $\|v\|^2 + \|\nabla v\|^2$




(b) $10^{-6}\|v\|^2 + \|\nabla v\|^2$




(c) $\|\nabla v\|^2$ with $v(-2) = 0.5$ and $v(2) = -0.5$




(d) $\|v\|^2$

Figure 3: Convection-diffusion ideal optimal test functions

## Analysis

From the results, it seems that the $L^2$ topology produces delta optimal test functions which may include delta functions at sharp kinks in the trial function. A more thorough mathematical analysis is warranted.

SpecialCaseOptimalTestFunctions.py

```
from dolfin import *
import numpy as np
import pylab as pl

# Create mesh and define function space
N = 10000;
p = 1;
pfine = 1;
r = 1
mesh = IntervalMesh(N, -2, 2)
finemesh = IntervalMesh(r*N, -2, 2)

# For Primal
Vp = FunctionSpace(mesh, "Lagrange", p)
Vpfine = FunctionSpace(finemesh, "Lagrange", pfine)
TAU = VectorFunctionSpace(mesh, "CG", p)
E = V*TAU
TAUfine = VectorFunctionSpace(finemesh, "CG", pfine)
Efine = Vfine*TAUfine

u0 = Constant(0)
# u0 = Expression('-x[0]/4')

def u0_boundary(x, on_boundary):
    return on_boundary

bc = DirichletBC(V, u0, u0_boundary)

epsilon = 1
beta = Constant((0,))

# Define variational problem
v = TrialFunction(V)
dv = TestFunction(V)

class HatExpression(Expression):
    def eval(self, value, x):
        if (x > -1 and x <= 0):
            value[0] = 1+x[0]
        elif (x > 0 and x < 1):
            value[0] = 1-x[0]
        else:
            value[0] = 0
u = HatExpression(domain=mesh)

# L2 Norm
# a = inner(v,dv)*dx

# Scaled H1 Norm
c = 1e-6;
a = c*inner(v,dv)*dx + inner(grad(v),grad(dv))*dx

# Grad Norm (requires boundary conditions for a unique solution)
# a = inner(grad(v),grad(dv))*dx

# Right hand side defined by the operator
# beta=0, epsilon=1 => Poisson
# beta=1, epsilon=0 => Convection
# beta=1, epsilon=1e-1 => Confusion

# Primal Formulation
L = -inner(beta*u,grad(dv))*dx + inner(epsilon*grad(u),grad(dv))*dx

# Ultra-Weak Formulation
L = -inner(beta*u,grad(dv))*dx + inner(epsilon*grad(u),grad(dv))*dx

# Compute solution
v = Function(V)

# With boundary conditions
# solve(a == L, v, bc)

# Without boundary conditions
solve(a == L, v)
```

```
vfine = project(v,Vfine)

vfine_vals = vfine.vector().array()

pl.figure(1)
x_vals = pl.linspace(-2,2,r*pfine*N+1)
u_vals = pl.zeros((r*pfine*N+1,))
for i in range(0,x_vals.size):
    u_vals[i] = u(x_vals[i])
pl.plot(x_vals,u_vals,linewidth=2,label='Trial Function')
pl.plot(x_vals,vfine_vals,'-',linewidth=2,label='Test Function')
pl.xlim((-2,2))
pl.ylim((-2,2))
pl.legend()

pl.show()
```