DeriveEqn_Lect22-5

November 7, 2015

```
In [1]: from sympy import *
        from sympy.matrices import *
        import sympy.mpmath
        from sympy.utilities.lambdify import lambdify
        init_printing()

        import numpy as np00
        import scipy.linalg as LA

        %matplotlib inline
        import matplotlib.pyplot as plt
        #plt.style.available
        #plt.style.use('bmh')
```

1 Check Buck's Formulation

Derivation of a higher order approximation for flux boundary conditions. The initial 2-point stencil yielded an $O(h^1)$ order of accuracy. Buck claims that by modifying the forcing function a higher order of accuracy may be obtained. Here, values of η_1 and η_2, from lecture 22-5, are derived. Where the forcing function is modified by η_1 and η_2.

```
In [2]: h, hm, a, am, ap, eta1, eta2, k, phi, phim, phip = symbols('h_n h_{n-1} alpha alpha_{n-1} alpha
                                                                   eta_2 k phi_n^e phi_{n-1}^e phi
```

```
In [3]: phi1, phi2, phi3, phi4, phi5, phi6 = symbols('phi_{n\,x}^e phi_{n\,xx}^e phi_{n\,xxx}^e \
                                                      phi_{n\,xxxx}^e phi_{n\,xxxxx}^e phi_{n\,xxxxxx}^e')
```

```
In [4]: c0 = am + a; c0
```

Out[4]:

$$\alpha + \alpha_{n-1}$$

```
In [5]: c1 = -hm*am -1; c1
```

Out[5]:

$$-\alpha_{n-1}h_{n-1} - 1$$

Solve for α_n and α_{n-1} such that c_0 and c_1 are zero

```
In [6]: alpha_terms = solve([c0,c1],[am,a])
        a = alpha_terms[a]
        am = alpha_terms[am]
```

check

```
In [7]: c0 = am + a; c0
```

Out[7]:

$$0$$

```
In [8]: c1 = -hm*am -1; c1
```

Out[8]:

$$0$$

Solve for η_1 and η_2 such that c_2 and c_2 are zero

```
In [9]: c2 = hm**2*Rational(1,2) * am + eta1*k + eta2*k; c2
```

Out[9]:

$$\eta_1 k + \eta_2 k - \frac{h_{n-1}}{2}$$

```
In [10]: c3 = (hm**3*am)*Rational(1,6) + hm*eta1*k; c3
```

Out[10]:

$$\eta_1 h_{n-1} k - \frac{h_{n-1}^2}{6}$$

```
In [11]: eta_terms = solve([c2,c3],[eta1,eta2])
         eta1 = eta_terms[eta1]
         eta2 = eta_terms[eta2]
```

```
In [14]: eta1
```

Out[14]:

$$\frac{h_{n-1}}{6k}$$

```
In [15]: eta2
```

Out[15]:

$$\frac{h_{n-1}}{3k}$$

check

```
In [12]: c2 = hm**2*Rational(1,2) * am + eta1*k + eta2*k; c2
```

Out[12]:

$$0$$

```
In [13]: c3 = (hm**3*am)*Rational(1,6) + hm*eta1*k; c3
```

Out[13]:

$$0$$

calculate c_4

```
In [19]: c4 = (hm**4*am)*Rational(1,24) - (hm**2*eta1*k)*Rational(1,2); c4
```

Out[19]:

$$-\frac{h_{n-1}^{3}}{8}$$

calculate the local truncation error (τ)

```
In [20]: lte = c0*phi + c1*phi1 + c2*phi2 + c3*phi3 + c4*phi4; simplify(lte)
```

Out[20]:

$$-\frac{h_{n-1}^{3}\phi_{n,xxxx}^{e}}{8}$$

Therefore, the local truncation error is of $O(h^3)$