APSC 256 final exam cheat sheet by Michael Ma

Errors

$$E_t = \text{True value} - \text{Approximate value}$$

True relative error

$$\epsilon_t = \frac{\text{true value} - \text{approximation}}{\text{true value}} \times 100\%$$

Approximate relative error

$$\epsilon_a = \frac{\text{present approximation} - \text{previous approximation}}{\text{present approximation}} \times 100\%$$

Stopping criterion

$$|\epsilon_a| \le \epsilon_s$$
$$\epsilon_s = (0.5 \times 10^{2-n})\%$$

Digital representations

The base-10 digit representation

$$s_m d_{m0}.d_{m1}d_{m2}d_{m3}d_{m4}\ldots \times 10^{s_e d_{e0} d_{e1} d_{e2} d_{e3}}$$

Where s_m is mantissa sign digit, d_m are mantissa digits, s_e is exponent sign digit, d_e is exponent digit. Must be normalized, i.e. only one digit before the decimal point.

The base-10 floating point representation

$$= \pm m \times 10^{\pm e}$$

Overflow errors

Overflow errors are caused when a value exceeds the max positive value of a given digit representation.

Underflow errors

Underflow errors are just the opposite, when a value is smaller than the min value.

Roundoff errors

A resolution is one-tenth of the step size. The corresponding roundoff error is one-half of the resolution.

Taylor series

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n$$

a is centre point for approximated function and $f(x)$ is close to a. Truncation error:

$$R_n = \frac{f^{(n+1)}(c)}{(n+1)!}(x-a)^{(n+1)}$$

Where c is a number between x and a.

Simplified Taylor series

$$f(x_{i+1}) \approx f(x_i) + \frac{f'(x_i)}{1!}h + \frac{f''(x_i)}{2!}h^2 + \ldots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

where h is step size, $f(x_{i+1})$ is the approximation and $f(x_i)$ is the past approximation.

Derivative approximation

Low Accuracy Formulas

Forward finite difference

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

Backwards finite difference

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h)$$

Centred finite difference

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

High Accuracy Formulas

Forward finite difference of $O(h^2)$

$$f'(x) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h} + O(h^2)$$

Backward finite difference of $O(h^2)$

$$f'(x) = \frac{3f(x) - 4f(x-h) - f(x-2h)}{2h} + O(h^2)$$

Centred finite difference of $O(h^4)$

$$f'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} + O(h^4)$$

Bracketing methods for root finding

Bisection method

A root of a function is between upper and lower bounds if the sign changes between the lower bound and upper bound. Multiplying the values at the two bounds should yield a negative number. Now we can test with a bisection.

- Test A: If $f(x_l) \times f\left(\frac{x_l + x_u}{2}\right) > 0$, then there is no root in the bracket. Set $x_l = \frac{x_l + x_u}{2}$.
- Test B: If $f(x_l) \times f\left(\frac{x_l + x_u}{2}\right) < 0$, then there is a root in the bracket. Set $x_u = \frac{x_l + x_u}{2}$.

Run it a few times, root should approximately be $x_R \approx \frac{x_l + x_u}{2}$

False position method/linear interpolation

Ensure that the root is in between the lower and upper bounds my performing a multiplication test and see if the result is negative.

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

Open methods of root finding

Fixed point iteration

Isolate x and the function on the right is $g(x)$. Repeat till root is found.

$$x_{i+1} = g(x_i)$$

Newton-Raphson method

$$x_{i+1} \approx x_i - \frac{f(x_i)}{f'(x_i)}$$

Will have problems if $f'(x_i) = 0$.

Secant method

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

Linear algebra and matrices

Linear systems

Row vector

$$[b] = \begin{bmatrix} b_1 & b_2 & b_3 & \cdots \end{bmatrix}$$

Column vector

$$[c] = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \end{bmatrix}$$

Matrix

$$[A] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Identity matrix

$$[I] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Diagonal matrix

$$[A] = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{mn} \end{bmatrix}$$

Upper triangular matrix

$$[A] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{mn} \end{bmatrix}$$

Lower triangular matrix

$$[A] = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Linear operations

Matrix transposition

Flipping a matrix along a diagonal axis from top left to bottom right.

$$[A]^T = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix}$$

Matrix addition and subtraction

$$[A] + [B] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

Addition/subtraction is commutative

$$[A] + [B] = [B] + [A]$$

Addition/subtraction is associative

$$([A] + [B]) + [C] = [A] + ([B] + [C])$$

Matrix multiplication

Scalar multiplication

$$k[A] = k \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix}$$

Matrix multiplication

$$[A][B] = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} \end{bmatrix}$$

Matrix multiplication is associative

$$([A][B])[C] = [A]([B][C])$$

Matrix multiplication is distributive

$$[A]([B] + [C]) = [A][B] + [A][C]$$

$$([A] + [B])[C] = [A][C] + [B][C]$$

Matrix multiplication is NOT commutative

$$[A][B] \neq [B][A]$$

Element-by-element matrix multiplication (Hadamard product)

$$[A] \circ [B] = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

Matrix determinant calculation

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

Matrix division

$$[x] = [A^{-1}][b]$$

Gauss elimination

Kinda like REF and RREF, with some pivoting. Backwards substitution to find values.

LU factorization

$$[A][x] = [b],$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$[L][d] = [b],$$

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$[U][x] = [d],$$

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Iterative methods

Linear systems

Gauss-Seidel method

$$x_1^{(i)} = \frac{b_1 - a_{12}x_2^{(i-1)} - a_{13}x_3^{(i-1)}}{a_{11}}$$

$$x_2^{(i)} = \frac{b_2 - a_{21}x_1^{(i)} - a_{23}x_3^{(i-1)}}{a_{22}}$$

$$x_3^{(i)} = \frac{b_3 - a_{31}x_1^{(i)} - a_{32}x_2^{(i)}}{a_{33}}$$

Nonlinear systems

Successive substitution method

$$\begin{aligned} x_{1,i+1} &= f_1(x_{1,i}, x_{2,i}, \ldots, x_{n,i}) \\ x_{2,i+1} &= f_1(x_{1,i+1}, x_{2,i}, \ldots, x_{n,i}) \\ &\vdots \\ x_{n,i+1} &= f_1(x_{1,i+1}, x_{2,i+1}, \ldots, x_{n,i}) \end{aligned}$$

Newton-Raphson method

$$[x_{i+1}] = [x_i] - [J]^{-1}[f]$$

Where the Jacobian matrix is

$$[J] = \begin{bmatrix} \frac{\partial f_{1,i}}{\partial x_1} & \frac{\partial f_{1,i}}{\partial x_2} & \cdots & \frac{\partial f_{1,i}}{\partial x_n} \\ \frac{\partial f_{2,i}}{\partial x_1} & \frac{\partial f_{2,i}}{\partial x_2} & \cdots & \frac{\partial f_{2,i}}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{n,i}}{\partial x_1} & \frac{\partial f_{n,i}}{\partial x_2} & \cdots & \frac{\partial f_{n,i}}{\partial x_n} \end{bmatrix}$$

Numerical integration methods

Analytical method

What you learned in Calculus 1.

$$\int_a^b f(x)dx = F(b) - F(a)$$

Newton-Cotes formulas

Replace the function with one that is easier to integrate.

$$\int_a^b f(x)dx \approx \int_a^b f_n(x)dx$$

Trapezoidal rule

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

Error for trapezoidal rule

$$E_t = -\frac{1}{12}f''(\xi)(b-a)^3$$

Where ξ is somewhere between a and b.

Composite trapezoidal rule

$$\int_a^b f(x)dx \approx \frac{h}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n)]$$

Simpson's rules

Simpson's 1/3 rule

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

Composite Simpson's 1/3 rule

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + f(x_4)]$$

Simpson's 3/8 rule

$$\int_a^b f(x)dx \approx \frac{3h}{8}[f(x_0)+3f(x_1)+3f(x_2)+f(x_3)]$$

Ordinary differential equations

Explicit Euler's method

$$y_{i+1} = y_i + f(t_i, y_i)h$$

Error of $E_t = O(h)$.

Implicit Euler's method

$$y_{i+1} = y_i + f(t_{i+1}, y_{i+1})h$$

Error of $E_t = O(h)$.

Heun's method

Predictor step

$$y^0_{i+1} = y_i + f(t_i, y_i)h$$

Corrector step

$$y_{i+1} = y_i + \frac{1}{2}\left(f(t_i, y_i) + f\left(t_{i+1}, y^0_{i+1}\right)\right)h$$

Error of $E_t = O(h^2)$.

Midpoint method

$$y_{i+1/2} = y_i + f(t_i, y_i)\frac{h}{2}$$
$$y_{i+1} = y_i + f(t_{i+1/2}, y_{i+1/2})h$$

where

$$t_{i+1/2} = t_i + \frac{1}{2}h$$

Error of $E_t = O(h^2)$.

Runge-Kutta 4th order method (RK4)

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

where

$$k_1 = f(t_0, y_0)$$
$$k_2 = f(t_i + \frac{h}{2}, y_i + k_1\frac{h}{2})$$
$$k_3 = f(t_i + \frac{h}{2}, y_i + k_2\frac{h}{2})$$
$$k_4 = f(t_i + h, y_i + k_3 h)$$

Error of $E_t = O(h^4)$

MATLAB functions

Basic shit

```
log()
```

Natural logarithm.

```
log10()
```

Base 10 logarithm.

```
linspace(x1,x2,n)
```

Creates a vector with `n` equally spaced points from `x1` to `x2`, inclusive.

```
logspace(x1,x2,n)
```

Creates a vector with `n` logarithmically spaced points from $10^{\texttt{x1}}$ to $10^{\texttt{x2}}$, inclusive.

```
fhandle = @(arglist) expression
```

Anonymous function

fzero (Captain Falcon!)

```
[x,fx] = fzero(function,x0,options,p1,p2,...)
```

Calculates roots

Matrix stuff

```
A[n,m]
```

Element at `n`th row, `m`th column

```
A[n,:]
```

Returns `n`th row

```
A[:,m]
```

Returns `m`th column

```
A*B
```

Multiplies matrices

```
A.*B
```

Multiplies matrices element by element

```
det(A)
```

Calculates determinant

```
inv(A)
```

Calculates inverse of matrix

```
A/b
```

Right division

```
A\b
```

Left division

LU factorization

```
[L,U] = lu(A)
```

LU factorization

Iterative methods

```
[x, fx] = fsolve(function, x0, options)
```

Solves nonlinear equations

MATLAB order of operations

1. Exponentiation: `^`
2. Negation: `-`
3. Multiplication/division: `*` and `/`
4. Left division: `\`
5. Addition/subtraction: `+` and `-`