# PHY407 Final Cheatsheet

*Author: QiLin Xue*

## Integrals

Python rounds after 16 digits. The fractional error $C$ for number $x$ is defined such that $\sigma = C|x|$, where $\sigma$ is standard deviation of error. Typically, $C = O(10^{-16})$.

**Trapezoidal rule**:

$$I(a,b) \approx h\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N-1} f(a+kh)\right]$$

$$\epsilon = \frac{h^2[f'(a) - f'(b)]}{12}$$

**Simpson's rule**:

$$I(a,b) \approx \frac{h}{3}\left[f(a) + f(b) + 4\sum_{odd} f(a+kh) + 2\sum_{even} f(a+kh)\right]$$

$$\epsilon = \frac{h^4[f'''(a) - f'''(b)]}{180}$$

**Practical error estimate** (get $C$ by computing $I_N$ and $I_{2N}$:

$$I(a,b) = I_N(a,b) + Ch^n \quad (1)$$

$$I(a,b) = I_{2N}(a,b) + C(h/2)^n \quad (2)$$

Improve on Newton-Cotes using **Gaussian quadrature**, where $P_N(x)$ are the Legendre polynomials:

$$x_k \in \{x : P_N(x) = 0\}, w_k = \left[\frac{2}{1-x^2}\left(\frac{dP_n}{dx}\right)^{-2}\right]_{x=x_k}$$

$$I(a,b) = \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right)$$

Error improves by a factor of $c/N^2$ when number of sample points increases by 1. However requires function to be reasonably smooth and hard to get an accurate estimate of the error. $N$ points exact for $2N-1$ order polynomial.

## Derivatives

Forward difference 1st-order in $h$. Central difference 2nd-order in $h$. For forward difference, each term $f(x+h), f(x)$ has error $C|f(x)|$. The total error and the optimal $h$ to choose is (can be derived using taylor series)

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{2}h|f''(x)|, \qquad h = \sqrt{4C\left|\frac{f}{f''}\right|}$$

For central difference, second term is $\frac{1}{24}h^2|f'''(x)|$.

## Solving Linear systems

Can implement standard methods to solve $Ax = b$, but need to be careful of machine precision (avoid dividing small numbers!).

**Partial pivoting**: At $m$th row, check to see which of the rows below has the largest $m$th value. Swap this row with the current $m$th value and proceed.

**LU Decomposition**: Write $A = LU$. Then solve $Ly = b$ first, then $Ux = y$. Easy since triangular matrices. Gaussian elimination yields $U = L_n L_{n-1} \cdots L_0 A = L^{-1}A$. Doesn't work if matrix is close to singular.

**QR Decomposition** looks for eigenvectors for Hermitian matrices. Gram-Schmidt on columns of $A$ to get $Q$. Write $A = QR$ and compute $A_1 = RQ = Q^TAQ = Q_1R_1$, $A_2 = R_1Q_1 = Q_1^TQAQQ_1 = Q_2R_2, \ldots$. Then $A_k$ converges to a near-diagonal matrix $A_k = V^TAV$, where $V = QQ_1 \cdots Q_k$. Diagonal entries of $A_k$ are eigenvalues and eigenvectors are columns of $V$.

## Root Finding

**Relaxation:** Finds stable fixed points by iteratively solving $x = f(x)$. Error of $\frac{x - x'}{1 - 1/f'(x)}$

**Newton's Method:** Much faster than relaxation, but doesn't always converge: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. Error of $\epsilon = O(\epsilon_0^{2^N}), \epsilon = x - x'$

**Bisection:** Always able to find a root if it exists, but it's slower, can't find double roots, and only works for a single bracketed root. Error of $\Delta/2^N$

**Golden ratio search:** Find the minimum by shrinking intervals, Start with 2 points $x_1, x_4$ bracketing the interval. Choose 2 points $x_2, x_3$ inside the interval. Check which of $f(x_2)$ and $f(x_3)$ is lower to determine new brackets. If $f(x_2) < f(x_3)$, new interval is $[x_1, x_3]$. Use the golden ratio to determine the most optimal placement of the internal points, i.e. $x_4 - x_1 = (x_3 - x_1)z = (x_4 - x_2)z$

## Fourier Transform

Fourier Transform

$$\hat{f}(\nu) = \int_{-\infty}^{\infty} f(x)e^{2\pi i\nu x}dx \approx \sum_{k=0}^{N-1} f(x_k)\exp(2\pi i\nu x_k)(\Delta x)$$

DFT: If function is real, coefficients from $N/2 \to N$ are complex conjugates of the first half. Runtime: $O(N^2)$. DFT and IDFT:

$$c_k = \sum_{n=0}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right), y_n = \frac{1}{N}\sum_{k=0}^{N-1} c_k \exp\left(i\frac{2\pi kn}{N}\right)$$

To recover frequencies, $f_i = \frac{i}{N}\frac{1}{\Delta t}$.

**FFT:** Divide and Conquer algorithm: $O(N\log(N))$.

## ODE

Euler: Error $O(h^2)$ each step, $O(h)$ globally.

$$x_k \to [x], \qquad x_k \mapsto x_k + f(x,t)$$

**RK2**: Derived by estimating the midpoint slope using Euler's method. Error $O(h^3)$ each step, $O(h^2)$ globally.

$$k_1 = hf(x,t), k_2 = hf\left(x + \frac{k_1}{2}, t + \frac{h}{2}\right), x(t+h) = x(t) + k_2.$$

**RK4**: Error $O(h^4)$ globally.

$$k_1 = hf(x,t), k_2 = hf\left(x + \frac{k_1}{2}, t + \frac{h}{2}\right)$$

$$k_3 = hf\left(x + \frac{k_2}{2}, t + \frac{h}{2}\right), k_4 = hf(x + k_3, t + h)$$

$$x(t+h) = x(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

**Leapfrog Method:** Use each point as a midpoint:

$$x(t+h) = x(t) + hf\left(x + \frac{h}{2}f(x,t), t + \frac{h}{2}\right)$$

$$x\left(t + \frac{3}{2}h\right) = x(t + h/2) + hf[x(t+h), t+h]$$

Also $O(h^2)$ error, but useful since time-reversible (RK2 is not since mid-point is lost and can't retrace steps), allows conservation of energy. To get backwards, set $h \mapsto -h, t \mapsto t + 3h/2$.

**Verlet:** Extends leapfrog to two coupled ODEs. Consider $\dot{x} = v, \dot{v} = f(x)$. Then:

$$x(t+h) = x(t) + hv\left(t + \frac{h}{2}\right)$$

$$v\left(t + \frac{3h}{2}\right) = v\left(t + \frac{h}{2}\right) + hf(x(t+h))$$

Also conserves energy. Often used in particle dynamics.

**Estimating Error:** Assume local error is $\epsilon = ch^5$. Error after 2 time steps is $\epsilon_1$. Error after 1 time step of $2h$ is $\epsilon_2$. Estimated error is $\epsilon = \frac{\epsilon_2 - \epsilon_1}{c}(x_1 - x_2)$, where $x_1, x_2$ are what you get for the two cases. For RK4, this is

$$\epsilon = ch^5 = \frac{1}{30}(x_1 - x_2).$$

**Adaptive Time Step:** Target error is $\delta$ per unit time (physical, not step). For RK4 (easily extended to other methods), if

$$\rho = \frac{h\delta}{\epsilon} = \frac{30h\delta}{|x_1 - x_2|} = \frac{30\delta}{ch^4} > 1$$

then $h$ is too small. Update values with the step size $= 2h$ result, then set $h' = h\rho^{1/4}$ to get $\rho' = 1$. If $\rho < 1$, time step is too large. Don't record values but still set $h' = h\rho^{1/4}$.

**Leapfrog Error:** Error is an odd function, $\epsilon(-h) = -\epsilon(h)$, so Taylor expansion gives $\epsilon(h) = c_3h^3 + c_5h^5 + \cdots$. Cumulative error is even in $h$, Each improvement gives us two orders of accuracy. We can do this via the following method.

**Modified Midpoint (MMP) Method:** To eliminate even powers of $\epsilon$ during the first half step, integrate from $t$ to $t + H$ with $n+1$ time steps:

$$x_0 = x(t), x_{1/2} = x_0 + hf(x_0, t)/2$$

$$x_1 = x_0 + hf(x_{1/2}, t + h/2), x_{3/2} = x_{1/2} + hf(x_1, t+h) + \cdots.$$

Then do both the whole integer and the forward Euler half step:

$$x_n = x_{n-1} + hf(x_{n-1/2}, t + H - h/2) \approx x(t+H)$$

$$x_n' = x_{n-1/2} + hf(x_n, t+h) \approx x(t+H).$$

Then

$$x(t+H)_{final} = \frac{x_n + x_n'}{2}.$$

**Bulirsch-Stoer:** MMP rarely used by itself, but basis for this powerful method: Take 1 single MMP step of size $h_1 = H$ to get estimate of $x(t+H) = R_{1,1}$ where $R$ stands for Richardson extrapolation. Now take 2MMP steps of size $h_2 = H/2$ to get second estimate $x(t+H) = R_{2,1}$. Since MMP is 2nd order and even total error, both these estimates are

$$x(t+H) = R_{1,1} + c_1h_1^2 + O(h_1^4)$$

$$x(t+H) = R_{2,1} + c_1h_2^2 + O(h_2^4).$$

Using $h_1 = 2h_2$, we can equate to get

$$c_1h_2^2 = \frac{1}{3}(R_{2,1} - R_{1,1}) + O(h_2^4).$$

Plugging this back into $x(t+H)$ to get a new estimate $R_{2,2}$ :

$$x(t+H) \approx R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + O(h_2^4) \equiv R_{2,2} + O(h_2^4).$$

In general: $h = H$, set $n = 1$ and use MMP to find $x(t+H)$. Continue to refine grid to find new estimates and error estimates. When error is acceptable, stop. The iteration can be expressed by:

$$x(t+H) = R_{n,m+1} + O(h_n^{2m+2}), \text{ where}$$

$$R_{n,m+1} = R_{n,m} + \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1}, h_n = \left(\frac{n-1}{n}\right) h_{n-1}.$$

**Shooting Method:** Suppose we want the root to be at a particular point. We can use a root finding method by adjusting some parameter (i.e. initial velocity), integrate it, until root is found.
**Eigenvalue Problems:** To solve Schrodinger equation, do not vary the slope with the shooting method, but rather the eigenvalue (energy).

## PDEs

Consider the PDE

$$f = \alpha\phi_{xx} + \beta\phi_{xy} + \gamma\phi_{yy} + \delta\phi_x + \epsilon\phi_y$$

and define $\Delta = \beta^2 - 4\alpha\gamma$. Note that $\Delta = 0$ is parabolic, $\Delta < 0$ is elliptic, and $\Delta > 0$ is hyperbolic.
**Parabolic:** Use relaxation methods. Expand out partial derivatives, i.e.

$$\frac{\partial^2\phi}{\partial x^2} \approx \frac{\phi(x+a,y) - 2\phi(x,y) + \phi(x-a,y)}{a^2},$$

and solve for $\phi(x,y) = g(x,y)$. Then set

$$\phi_{new}(x,y) = (1+\omega)g(x,y) - \omega\phi(x,y).$$

where $\omega > 0$ for overrelaxation (which is usually faster but not always stable. $\omega = 0$ is always stable). The Gauss-Seidel method assumes that the value of each new computed cell is more accurate, so we only need to use one array.
**Parabolic:** We can use Forward Time Centered Space (FTCS) method: We can discretize space and time. for $\frac{\partial T}{\partial t} = \kappa\frac{\partial^2 T}{\partial x^2}$, we get

$$T_m^{n+1} = T_m^n + \frac{\kappa h}{a^2}(T_{m+1}^n - 2T_m^n + T_{m-1}^n),$$

where $m = 0, \ldots, M$, $a = L/m,$, and $T_m(t_n) \equiv T_m^n$.
**Von Neumann Stability Analysis:** Substitute $T_m^n = \hat{T}_k^n e^{ikam}$, and solve for the growth factor $|\hat{T}_k^{n+1}/\hat{T}_k^n|$. This is stable if it is smaller than 1. For heat equation, the condition is $h \leq \frac{a^2}{2\kappa}$. For hyperbolic equations however, FTCS is always unstable.
**Hyperbolic (Explicit):** Consider $\frac{\partial^2\phi}{\partial t^2} = c^2\frac{\partial^2\phi}{\partial x^2}$. We can approximate the LHS and write two 1st order ODEs,

$$\frac{d\phi_m}{dt} = \psi_m, \frac{d\psi_m}{dt} = \frac{c^2}{a^2}(\phi_{m+1} - 2\phi_m + \phi_{m-1}).$$

Forward Euler on each (time step of $h$) gives, in matrix form:

$$\begin{pmatrix}\phi_m^{n+1}\\ \psi_m^{n+1}\end{pmatrix} = \begin{pmatrix}1 & h\\ -\frac{2hc^2}{a^2} & 1\end{pmatrix}\begin{pmatrix}\phi_m^n\\ \psi_n^m\end{pmatrix} + \begin{pmatrix}0\\ \frac{c^2h}{a^2}(\phi_{m+1}^n + \phi_{m-1}^n)\end{pmatrix}.$$

Stability analysis on a 2d system is done using the same substitution, and getting it in the form:

$$\begin{pmatrix}\hat{\phi}_k^{n+1}\\ \hat{\psi}_k^{n+1}\end{pmatrix} = A\begin{pmatrix}\hat{\phi}_k^n\\ \hat{\psi}_k^n\end{pmatrix}.$$

The eigenvalues of $A$ are $\lambda_\pm$. If $|\lambda_\pm|^2 < 1$, then it is stable.
**Hyperbolic (Implicit):** Make the substitution $h \mapsto -h, n \mapsto n+1$ to get

$$\begin{pmatrix}\phi_m^n\\ \psi_m^n\end{pmatrix} = \begin{pmatrix}1 & -h\\ \frac{2hc^2}{a^2} & 1\end{pmatrix}\begin{pmatrix}\phi_m^{n+1}\\ \psi_m^{n+1}\end{pmatrix} - \begin{pmatrix}0\\ \frac{c^2h}{a^2}(\phi_{m+1}^n + \phi_{m-1}^n)\end{pmatrix}.$$

This is stable, but solutions will decay exponentially.
**Crank-Nicolson:** The average between both methods. Taking the average, we have:

$$\phi_m^{n+1} - \frac{h}{2}\psi_m^{n+1} = \phi_m^n + \frac{h}{2}\psi_m^n$$

$$\psi_m^{n+1} - \frac{h}{2}\frac{c^2}{a^2}(\phi_{m-1}^{n+1} + \phi_{m+1}^{n+1} - 2\phi_m^{n+1}) =$$
$$\psi_m^n + \frac{h}{2}\frac{c^2}{a^2}(\phi_{m-1}^n + \phi_{m+1}^n - 2\phi_m^n)$$

The Von Neumann substitution gives $|\lambda_\pm| = 1$, so solution neither grows nor decays. It is 2nd-order accurate in time, while implicit method is 1st-order accurate.
**Spectral Methods:** Suppose we are interested in finding $f$ such that

$$\nabla^2 f(x,y) = g(x,y).$$

We can write $f, g$ in Fourier series:

$$f, g := \sum a_{j,k}e^{i(jx+ky}, \sum b_{j,k}e^{i(jx+ky)},$$

substitute, in order to get

$$\sum -a_{j,k}(j^2+k^2)e^{i(jx+ky)} = \sum b_{j,k}e^{i(jx+ky)}.$$

By the uniqueness theorem for Fourier expansions, we can equate Fourier coefficients term by term, giving

$$a_{j,k} = -\frac{b_{j,k}}{j^2+k^2}.$$

With periodic boundary conditions, the Poisson equation possesses a solution only if $b_{0,0} = 0$. Therefore, we can freely choose a0,0 which will be equal to the mean of the resolution. This corresponds to choosing the integration constant. To turn this into an algorithm, only finitely many frequencies are solved for. This introduces an error which can be shown to be proportional to $h^n$, where $h := 1/n$ and $n$ is the hgighest frequency treated. In general: Compute the Fourier transform $(b_{j,k})$ of $g$. Compute the fFurier transform of $a_{j,k}$ of $f$ using the above formula (or equivalent). compute $f$ by taking an inverse Fourier transform of $(a_{j,k})$.
A direct implication is that we can compute derivatives by computing Fourier transforms. Let $f = \sum \hat{f}_n \exp(ik_nx)$. Then $\frac{\partial f}{\partial x} = \sum ik_n\hat{f}_n \exp(ik_nx)$.

## Random Numbers + Monte Carlo

**Correlation:** To test for randomness, we can compute

$$\epsilon(N,n) = \frac{1}{N}\sum_{i=1}^N x_ix_{i+n} - E[x^2],$$

where $N$ is the number of data points, $n$ is the correlation distance. We want to avoid correlations between pairs of numbers.
**Moments:** $k$th moment of $\mu(N,k) = E[x^k]$.
**Linear Congruential Generator:** Consider $x_{i+1} = (ax_i + c) \pmod m$. Let $x_0$ be the seed, and $m$ is a large integer which determines the period. For good results, $a - 1$ is a multiple of $p$ for every prime divisor $p$ of $m$ and $c$ is relatively prime to $m$.
**Non-uniform distribution:** Goal is to get a uniformly distributed random number, then apply transformation to make it seem like it came from a non-uniform distribution. To find $x(z)$ so that $x$ has the distribution we want, solve for $x(z)$ such that

$$z = \int_0^{x(z)} p(x')dx',$$

where $p(x)$ is the probability distribution we need.
**Monte Carlo Integration:** We want to use because: (1) Good for weird/fast-varying functions, (2) much faster for multi-dimensional integrals (standard way: each grid has side length $O(N^{1/d})$ so trapezoid integration has error $O(h^2) \propto 1/N^{2/d}$. But Monte Carlo has error $\epsilon \propto 1/N^{1/2}$, regardless of $d$), (3) Much easier to implement in complicated domains.
**Hit-or-miss MC:** If the function fits inside a box, the probability that a random point falls in the shaded area (under a curve) is $p = I/A$. Randomly pick $N$ locations in the box. count the number of locations that are in the shaded region (call the count $k$). The integral estimate is $I \approx \frac{kA}{N}$. The expected error is $\sigma = \sqrt{\frac{(A-I)I}{N}}$, which is very slow.
**Mean Value MC:** Estimate integral as $I = (b-a)\langle f\rangle$. Use random numbers for $x$ to estimate $\langle f\rangle$. We have $I \approx \frac{b-a}{N}\sum_{i=1}^N f(x_i)$. Error estimate is $\sigma = (b-a)\sqrt{\frac{\text{var } f}{N}}$, where var $f = \langle f^2\rangle - \langle f\rangle^2$. It also varies as $N^{-1/2}$, but the leading constant is smaller than the hit-or-miss method.
**Importance Sampling MC:** If integrand contains a divergence, we want to place more points in region where the integrand is large to better estimate the integral. We can estimate,

$$I = \langle\frac{f(x)}{w(x)}\rangle_w \int_a^b w(x)dx.$$

For example, if $w(x) = x^{-1/2}$ (i.e. this is the term that makes the integrand diverge), then

$$\langle\frac{f(x)}{w(x)}\rangle_w = \frac{1}{N}\sum_{i=1}^N \frac{f(x_i)}{w(x_i)} = \frac{1}{N}\sum_{i=1}^N \frac{f(x_i)}{w(x_i)}.$$

We need to sample points using $p(x) = \frac{w(x)}{\int_a^b w(x)dx}$. Error is $\sigma = \sqrt{\text{var}(f/w)/N}\int_a^b w(x)dx$.
**Markov:** Choose a random starting state $i$, compute the energy $E_i$. Choose a transition to a new allowed state $j$ uniformly. If $E_j \leq E_i$, accept it. If $E_j > E_i$, accept it with probability $\exp(-\Delta E/kT)$. Repeat.
**Simulated Annealing:** Start with high temperature and slowly lower it. Usually done with exponential decreasing.