4a) Inputs: $x_0, \alpha, \Delta t$

We set $K = 0$

for $K = 0 \ldots N-1$, do this

while $\|r(K, x_K, u=0)\| > tol$, do

Evaluate:

$$r(K, Z, x_K) = \begin{bmatrix} F(K_1, x_K + \Delta t \sum_{i=1}^{\gamma} a_{1i} K_i Z_i \\ \vdots \\ F(K_\gamma, x_K + \Delta t \sum_{i=1}^{\gamma} a_{\gamma i} K_i Z_\gamma \end{bmatrix}$$

$$= 0$$

and:

$$M = \begin{bmatrix} \frac{\partial r(K, Z, x_K)}{\partial K} & \frac{\partial r(K, Z, x_K)}{\partial Z} \end{bmatrix}$$

Newton step:

$$\begin{bmatrix} K \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} K \\ Z \end{bmatrix} - \alpha M^{-1} r$$

Then integratorstep:

$$X_{K+1} = X_K + \Delta t \sum_{i=1}^{r} b_i k_i$$

return $X_{0,\ldots,N}$

end

4b) The maximum order "O" for given number of stages "s" for an IRK is given by:

$$O = 2s$$

Every $K_{1,\ldots,o}$ must be implicitly given.
or A must have values along the diagonal which are non-zero.
The coefficients can't take an arbitrary value if the RK-method is to achieve the highest order possible.

4c) Butcher tableau, c, A, b^T will provide the coefficients for calculating $K_{1...o}$ in A. The coefficients for $K_{1,...,o}$ when calculating the integrator step y_{n+1} in b^T. And information regarding the time-steps in C.

The tableau will not provide information on whether it is stable. To determine this, one must check the eigenvalues and the time-step with the stability function for the method.
Also: Tells if it is Explicit or Implicit

4d) They will generally require a lot of computations, since you must evaluate the jacobian of r() for each iteration. This increases complexity dramatically as the number of stages increase.
These are, however, very accurate, so they can be used for slow dynamic systems.

4e) The higher the order of the ERK-method, the lower the error. However, the decrease in error is asymptotically going towards zero, and the smaller error of the higher order methods are going to be less significant, but more computationally heavier. It is therefore not worth it, and one should use a lower order IRK instead.

4f) You can adjust the step-size to what is needed to meet the desired accuracy.

Fast dynamics → lower Δt

Slow dynamics → higher Δt

Then, the method won't be as computationally heavy as if we'd set the Δt to a constant tiny value.

It can compare the error to adjust Δt.

It can do this by computing x_{k+1} for two different Butcher tableaus

$$e = \|x_{k+1} - \tilde{x}_{k+1}\|$$

Reduce Δt if $e > tol$.
and vice versa.

Since you can adapt to fast dynamics, this should be more stable than doing ordinary RK.
It costs more, but you can decrease this cost by having two identical Butcher tableau's, but with different b^T.

e.g.

$$\begin{array}{c|c} \vdots & \ddots \\ c & A \\ \vdots & \ddots \\ \hline & \cdots b^T \cdots \\ & \cdots \hat{b}^T \cdots \end{array}$$

4g) Suppose you need to simulate a bouncing ball, then it is better to do an integrator-step as the ball changes direction after hitting the floor.
By not using event-based, we'd have to have very small time steps for normal numerical solvers.
They work by adding an event condition $e(x) = 0$ and doing something to handle the event.

This method should work for all Butcher tableaus.