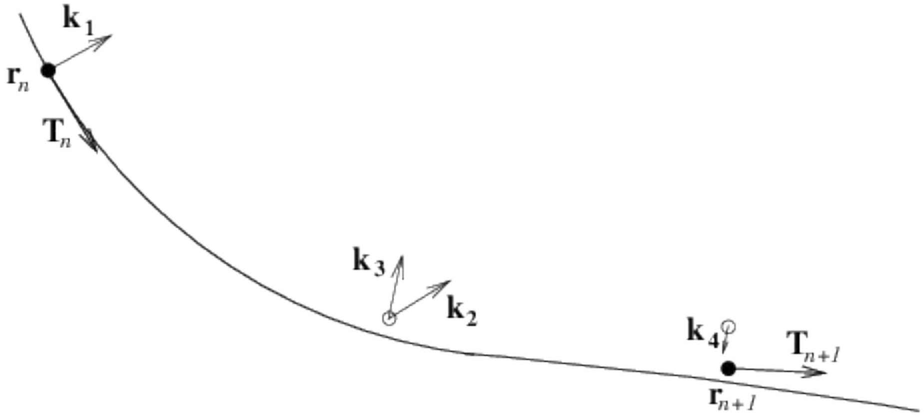
EigenStepper: current state

$$
\begin{aligned}
k_1 &= f(x_n, y_n, y'_n) \\
k_2 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}y'_n + \frac{h^2}{8}k_1, y'_n + \frac{h}{2}k_1) \\
k_3 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}y'_n + \frac{h^2}{8}k_1, y'_n + \frac{h}{2}k_2) \\
k_4 &= f(x_n + h, y_n + hy'_n + \frac{h^2}{2}k_3, y'_n + hk_3) \\
y'_{n+1} &= y'_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
y_{n+1} &= y_n + hy'_n + \frac{h^2}{6}(k_1 + k_2 + k_3)
\end{aligned}
$$



EigenStepper: current state

1. Evaluation of k1 (once)
2. Evaluation of k2 - k4 -> Error estimation
3. Change step size until error is small enough

$$\varepsilon = h^2(k_1 - k_2 - k_3 + k_4)$$

If step is accepted: update pos & dir

```
double
step(State& state) const
{
  // Charge-momentum ratio, in SI units
  const double qop = state.q / units::Nat2SI<units::MOMENTUM>(state.p);

  // Runge-Kutta integrator state
  double   h2, half_h;
  Vector3D B_middle, B_last, k2, k3, k4;

  // First Runge-Kutta point (at current position)
  const Vector3D B_first = getField(state, state.pos);
  const Vector3D k1      = qop * state.dir.cross(B_first);

  // The following functor starts to perform a Runge-Kutta step of a certain
  // size, going up to the point where it can return an estimate of the local
  // integration error. The results are stated in the local variables above,
  // allowing integration to continue once the error is deemed satisfactory
  const auto tryRungeKuttaStep = [&](const double h) -> double {
    // State the square and half of the step size
    h2     = h * h;
    half_h = h / 2;

    // Second Runge-Kutta point
    const Vector3D pos1 = state.pos + half_h * state.dir + h2 / 8 * k1;
    B_middle            = getField(state, pos1);
    k2                  = qop * (state.dir + half_h * k1).cross(B_middle);

    // Third Runge-Kutta point
    k3 = qop * (state.dir + half_h * k2).cross(B_middle);

    // Last Runge-Kutta point
    const Vector3D pos2 = state.pos + h * state.dir + h2 / 2 * k3;
    B_last              = getField(state, pos2);
    k4                  = qop * (state.dir + h * k3).cross(B_last);

    // Return an estimate of the local integration error
    return h * (k1 - k2 - k3 + k4).template lpNorm<1>();
  };

  // Select and adjust the appropriate Runge-Kutta step size
  // @todo remove magic numbers and implement better step estimation
  double error_estimate = tryRungeKuttaStep(state.stepSize);
  while (error_estimate > 0.0002) {
    state.stepSize = 0.5 * state.stepSize;
    error_estimate = tryRungeKuttaStep(state.stepSize);
  }
```

EigenStepper: covariance

$$\boldsymbol{u} = \begin{bmatrix} x \\ y \\ z \\ \Lambda \end{bmatrix}, \qquad \boldsymbol{u}' = \frac{d\boldsymbol{u}}{ds} = \begin{bmatrix} T^x \\ T^y \\ T^z \\ \lambda \end{bmatrix}$$

Propagation (pos & dir):

$$\boldsymbol{u}_{n+1} = F(s_n, \boldsymbol{u}_n, \boldsymbol{u}'_n) = \boldsymbol{F}_n(\boldsymbol{u}_n, \boldsymbol{u}'_n) = \boldsymbol{u}_n + h\boldsymbol{u}'_n + \frac{h^2}{6}(\boldsymbol{u}''_1 + \boldsymbol{u}''_2 + \boldsymbol{u}''_3)$$

$$\boldsymbol{u}'_{n+1} = G(s_n, \boldsymbol{u}_n, \boldsymbol{u}'_n) = \boldsymbol{G}_n(\boldsymbol{u}_n, \boldsymbol{u}'_n) = \boldsymbol{u}'_n + \frac{h}{6}(\boldsymbol{u}''_1 + 2\boldsymbol{u}''_2 + 2\boldsymbol{u}''_3 + \boldsymbol{u}''_4)$$

Propagation (jacobian):

$$\mathbf{J}_{n+1} = \begin{bmatrix} \frac{\partial \boldsymbol{u}_{n+1}}{\partial \boldsymbol{\xi}^I} \\ \frac{\partial \boldsymbol{u}'_{n+1}}{\partial \boldsymbol{\xi}^I} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{\xi}^I} \\ \frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{\xi}^I} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n} & \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}'_n} \\ \frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n} & \frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}'_n} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{\xi}^I} \\ \frac{\partial \boldsymbol{u}'_n}{\partial \boldsymbol{\xi}^I} \end{bmatrix} = \mathbf{D}_n \cdot \mathbf{J}_n$$

-> requires the evaluation of D:

$$\mathbf{D}_n = \frac{\partial(\boldsymbol{F}_n, \boldsymbol{G}_n)}{\partial(\boldsymbol{u}_n, \boldsymbol{u}'_n)} = \begin{bmatrix} \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n} & \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}'_n} \\ \frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n} & \frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}'_n} \end{bmatrix}$$

$$\begin{aligned}
\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n} &= 1 + \frac{h^2}{6}\left(\frac{\partial \boldsymbol{u}''_1}{\partial \boldsymbol{u}_n} + \frac{\partial \boldsymbol{u}''_2}{\partial \boldsymbol{u}_n} + \frac{\partial \boldsymbol{u}''_3}{\partial \boldsymbol{u}_n}\right) \\
\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}'_n} &= h + \frac{h^2}{6}\left(\frac{\partial \boldsymbol{u}''_1}{\partial \boldsymbol{u}'_n} + \frac{\partial \boldsymbol{u}''_2}{\partial \boldsymbol{u}'_n} + \frac{\partial \boldsymbol{u}''_3}{\partial \boldsymbol{u}'_n}\right) \\
\frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n} &= \frac{h}{6}\left(\frac{\partial \boldsymbol{u}''_1}{\partial \boldsymbol{u}_n} + 2\frac{\partial \boldsymbol{u}''_2}{\partial \boldsymbol{u}_n} + 2\frac{\partial \boldsymbol{u}''_3}{\partial \boldsymbol{u}_n} + \frac{\partial \boldsymbol{u}''_4}{\partial \boldsymbol{u}_n}\right) \\
\frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}'_n} &= 1 + \frac{h}{6}\left(\frac{\partial \boldsymbol{u}''_1}{\partial \boldsymbol{u}'_n} + 2\frac{\partial \boldsymbol{u}''_2}{\partial \boldsymbol{u}'_n} + 2\frac{\partial \boldsymbol{u}''_3}{\partial \boldsymbol{u}'_n} + \frac{\partial \boldsymbol{u}''_4}{\partial \boldsymbol{u}'_n}\right)
\end{aligned}$$

EigenStepper: covariance

$$\begin{aligned}
\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n} &= 1+\frac{h^2}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n}+\frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n}+\frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n}\right)\\
\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n'} &= h+\frac{h^2}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n'}+\frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n'}+\frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n'}\right)\\
\frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n} &= \frac{h}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n}+2\frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n}+2\frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n}+\frac{\partial \boldsymbol{u}_4''}{\partial \boldsymbol{u}_n}\right)\\
\frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n'} &= 1+\frac{h}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n'}+2\frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n'}+2\frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n'}+\frac{\partial \boldsymbol{u}_4''}{\partial \boldsymbol{u}_n'}\right)
\end{aligned}$$

Equations of motions:

$$\begin{aligned}
x'' &= \lambda(T^y B_z - T^z B_y)\\
y'' &= \lambda(T^z B_x - T^x B_z)\\
z'' &= \lambda(T^x B_y - T^y B_x)\\
\Lambda'' &= -\frac{\lambda^3 g E}{q^2}
\end{aligned}$$

->Evaluation of A & C

$$\mathbf{A}_k = \frac{\partial \boldsymbol{u}_k''}{\partial \boldsymbol{u}_n'}, \qquad \mathbf{C}_k = \frac{\partial \boldsymbol{u}_k''}{\partial \boldsymbol{u}_n}$$

$$\mathbf{A} = \begin{bmatrix} \frac{\partial x''}{\partial T^x} & \cdots & \frac{\partial x''}{\partial \lambda} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Lambda''}{\partial T^x} & \cdots & \frac{\partial \Lambda''}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} 0 & \lambda B_z & -\lambda B_y & T^y B_z - T^z B_y \\ -\lambda B_z & 0 & \lambda B_x & T^z B_x - T^x B_z \\ \lambda B_y & -\lambda B_x & 0 & T^x B_y - T^y B_x \\ 0 & 0 & 0 & \left(\frac{1}{\lambda}\left(3-\frac{p^2}{E^2}\right)+\frac{1}{g}\frac{\partial g}{\partial \lambda}\right)\Lambda'' \end{bmatrix} \tag{17}$$

and

$$\mathbf{C} = \begin{bmatrix} \frac{\partial x''}{\partial x} & \cdots & \frac{\partial x''}{\partial \Lambda} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Lambda''}{\partial x} & \cdots & \frac{\partial \Lambda''}{\partial \Lambda} \end{bmatrix} = \begin{bmatrix} \lambda(T^y \frac{\partial B_z}{\partial x} - T^z \frac{\partial B_y}{\partial x}) & \lambda(T^y \frac{\partial B_z}{\partial y} - T^z \frac{\partial B_y}{\partial y}) & \lambda(T^y \frac{\partial B_z}{\partial z} - T^z \frac{\partial B_y}{\partial z}) & 0 \\ \lambda(T^z \frac{\partial B_x}{\partial x} - T^x \frac{\partial B_z}{\partial x}) & \lambda(T^z \frac{\partial B_x}{\partial y} - T^x \frac{\partial B_z}{\partial y}) & \lambda(T^z \frac{\partial B_x}{\partial z} - T^x \frac{\partial B_z}{\partial z}) & 0 \\ \lambda(T^x \frac{\partial B_y}{\partial x} - T^y \frac{\partial B_x}{\partial x}) & \lambda(T^x \frac{\partial B_y}{\partial y} - T^y \frac{\partial B_x}{\partial y}) & \lambda(T^x \frac{\partial B_y}{\partial z} - T^y \frac{\partial B_x}{\partial z}) & 0 \\ -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial x} & -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial y} & -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial z} & 0 \end{bmatrix} \tag{18}$$

EigenStepper: covariance

$$
\begin{aligned}
\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n} &= 1 + \frac{h^2}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n} + \frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n} + \frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n}\right) \\
\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}_n'} &= h + \frac{h^2}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n'} + \frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n'} + \frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n'}\right) \\
\frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n} &= \frac{h}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n} + 2\frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n} + 2\frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n} + \frac{\partial \boldsymbol{u}_4''}{\partial \boldsymbol{u}_n}\right) \\
\frac{\partial \boldsymbol{G}_n}{\partial \boldsymbol{u}_n'} &= 1 + \frac{h}{6}\left(\frac{\partial \boldsymbol{u}_1''}{\partial \boldsymbol{u}_n'} + 2\frac{\partial \boldsymbol{u}_2''}{\partial \boldsymbol{u}_n'} + 2\frac{\partial \boldsymbol{u}_3''}{\partial \boldsymbol{u}_n'} + \frac{\partial \boldsymbol{u}_4''}{\partial \boldsymbol{u}_n'}\right)
\end{aligned}
$$

Equations of motions:

$$
\begin{aligned}
x'' &= \lambda(T^y B_z - T^z B_y) \\
y'' &= \lambda(T^z B_x - T^x B_z) \\
z'' &= \lambda(T^x B_y - T^y B_x) \\
\Lambda'' &= -\frac{\lambda^3 g E}{q^2}
\end{aligned}
$$

->Evaluation of A & C

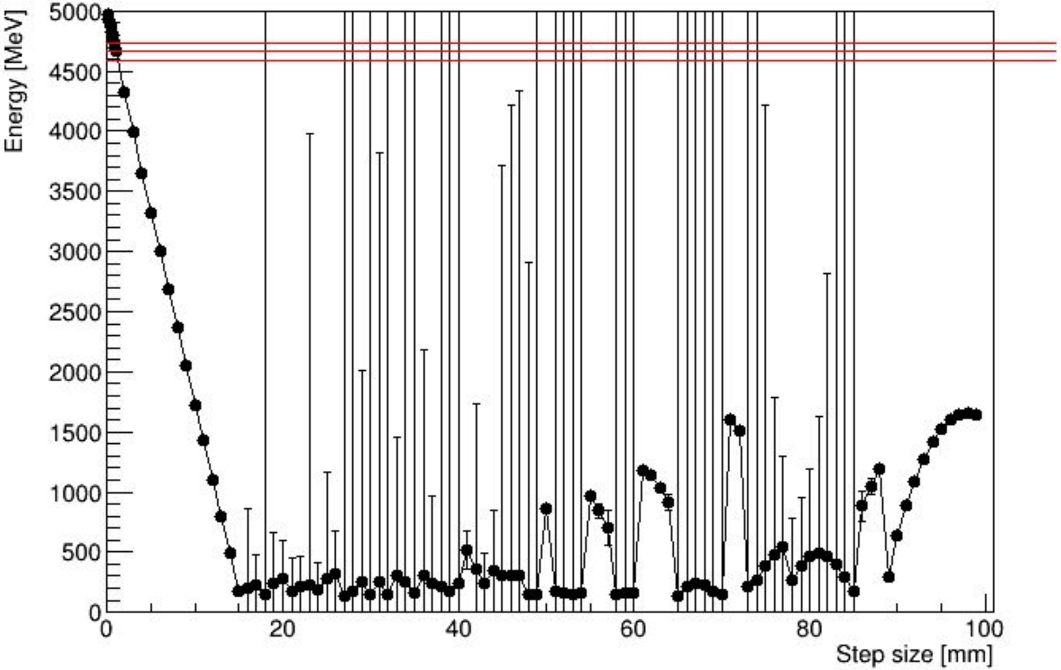
$$
\mathbf{A}_k = \frac{\partial \boldsymbol{u}_k''}{\partial \boldsymbol{u}_n'}, \qquad \mathbf{C}_k = \frac{\partial \boldsymbol{u}_k''}{\partial \boldsymbol{u}_n} \qquad
\mathbf{A} = \begin{bmatrix} \frac{\partial x''}{\partial T^x} & \cdots & \frac{\partial x''}{\partial \lambda} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Lambda''}{\partial T^x} & \cdots & \frac{\partial \Lambda''}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} 0 & \lambda B_z & -\lambda B_y & T^y B_z - T^z B_y \\ -\lambda B_z & 0 & \lambda B_x & T^z B_x - T^x B_z \\ \lambda B_y & -\lambda B_x & 0 & T^x B_y - T^y B_x \\ 0 & 0 & 0 & \cancel{\left(\frac{1}{\lambda}\left(3 - \frac{p^2}{E^2}\right) + \frac{1}{g}\frac{\partial g}{\partial \lambda}\right)\Lambda''} \end{bmatrix} \tag{17}
$$

and

$$
\mathbf{C} = \begin{bmatrix} \frac{\partial x''}{\partial x} & \cdots & \frac{\partial x''}{\partial \Lambda} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Lambda''}{\partial x} & \cdots & \frac{\partial \Lambda''}{\partial \Lambda} \end{bmatrix} = \cancel{\begin{bmatrix} \lambda(T^y \frac{\partial B_z}{\partial x} - T^z \frac{\partial B_y}{\partial x}) & \lambda(T^y \frac{\partial B_z}{\partial y} - T^z \frac{\partial B_y}{\partial y}) & \lambda(T^y \frac{\partial B_z}{\partial z} - T^z \frac{\partial B_y}{\partial z}) & 0 \\ \lambda(T^z \frac{\partial B_x}{\partial x} - T^x \frac{\partial B_z}{\partial x}) & \lambda(T^z \frac{\partial B_x}{\partial y} - T^x \frac{\partial B_z}{\partial y}) & \lambda(T^z \frac{\partial B_x}{\partial z} - T^x \frac{\partial B_z}{\partial z}) & 0 \\ \lambda(T^x \frac{\partial B_y}{\partial x} - T^y \frac{\partial B_x}{\partial x}) & \lambda(T^x \frac{\partial B_y}{\partial y} - T^y \frac{\partial B_x}{\partial y}) & \lambda(T^x \frac{\partial B_y}{\partial z} - T^y \frac{\partial B_x}{\partial z}) & 0 \\ -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial x} & -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial y} & -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial z} & 0 \end{bmatrix}} \tag{18}
$$

StepStepper

Include energy loss via action -> post-step update of momentum



Mean + CI (0.68) from G4

-> This approach does not work
-> Need to become part of integrand in EigenStepper

Propagation modifications

dE/ds given by Bethe-Bloch & radiation (for muons: + photonuclear & pair creation)

-> dp/ds (const over h) -> k2-k4 receives the momentum loss

Additional criterion: p_final >= p0 + dp

Error estimation: $h_{n+1} = h_n \left(\frac{\tau}{|\varepsilon|} \right)^{\frac{1}{q+1}}$ $\quad \frac{1}{4} h_n \leq h_{n+1} \leq 4 h_n$

-> Step size may increase during the propagation

Covariance

$$\mathbf{A} = \begin{bmatrix} \frac{\partial x''}{\partial T^x} & \cdots & \frac{\partial x''}{\partial \lambda} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Lambda''}{\partial T^x} & \cdots & \frac{\partial \Lambda''}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} 0 & \lambda B_z & -\lambda B_y & T^y B_z - T^z B_y \\ -\lambda B_z & 0 & \lambda B_x & T^z B_x - T^x B_z \\ \lambda B_y & -\lambda B_x & 0 & T^x B_y - T^y B_x \\ 0 & 0 & 0 & \left(\frac{1}{\lambda}\left(3 - \frac{p^2}{E^2}\right) + \frac{1}{g}\frac{\partial g}{\partial \lambda} \right) \Lambda'' \end{bmatrix} \tag{17}$$

and

$$\mathbf{C} = \begin{bmatrix} \frac{\partial x''}{\partial x} & \cdots & \frac{\partial x''}{\partial \Lambda} \\ \vdots & \ddots & \vdots \\ \frac{\partial \Lambda''}{\partial x} & \cdots & \frac{\partial \Lambda''}{\partial \Lambda} \end{bmatrix} = \begin{bmatrix} \lambda\left(T^y \frac{\partial B_z}{\partial x} - T^z \frac{\partial B_y}{\partial x}\right) & \lambda\left(T^y \frac{\partial B_z}{\partial y} - T^z \frac{\partial B_y}{\partial y}\right) & \lambda\left(T^y \frac{\partial B_z}{\partial z} - T^z \frac{\partial B_y}{\partial z}\right) & 0 \\ \lambda\left(T^z \frac{\partial B_x}{\partial x} - T^x \frac{\partial B_z}{\partial x}\right) & \lambda\left(T^z \frac{\partial B_x}{\partial y} - T^x \frac{\partial B_z}{\partial y}\right) & \lambda\left(T^z \frac{\partial B_x}{\partial z} - T^x \frac{\partial B_z}{\partial z}\right) & 0 \\ \lambda\left(T^x \frac{\partial B_y}{\partial x} - T^y \frac{\partial B_x}{\partial x}\right) & \lambda\left(T^x \frac{\partial B_y}{\partial y} - T^y \frac{\partial B_x}{\partial y}\right) & \lambda\left(T^x \frac{\partial B_y}{\partial z} - T^y \frac{\partial B_x}{\partial z}\right) & 0 \\ -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial x} & -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial y} & -\frac{\lambda^3 E}{q^2}\frac{\partial g}{\partial z} & 0 \end{bmatrix} \tag{18}$$

C not included yet

Terms of A generalized by A(4,4) & d/dlambda parts in last column

StepperExtensionList

EigenStepper & StepStepper similar but latter is not “correction” of EigenStepper

-> 2 different steppers required?

Similar to Action-/AbortList: Combination in StepperExtensionList

```diff
- template <typename BField, typename corrector_t = VoidCorrector>
+ template <typename BField,
+           typename corrector_t     = VoidCorrector,
+           typename extensionlist_t = StepperExtensionList<DefaultExtension>,
+           typename auctioneer_t    = detail::VoidAuctioneer>
  class EigenStepper
  {
```

-> Calls set of functions along a step evaluation

StepperExtensionList / Auctioneer

Like Action-/AbortList: StepperExtensionList broadcasts function calls to all elements

-In vacuum: unnecessary function calls of dense environment

-In matter: overwriting of default EigenStepper evaluations

-> Require judgment who should perform the evaluation (during runtime)

Rule: Each extension makes 1 bid (based on the environment data) if responsible

-> Auctioneer decides who gets the job